دانشکده مهندسی برق

طراحی، شبیه‌سازی و ساخت خودروی خودران با قابلیت درک محیط، برنامه ریزی و کنترل

Design, Simulation, And Construction of An Autonomous Vehicle with Environment Perception, Planning, and Control Capabilities

پایان‌نامه برای دریافت درجه کارشناسی

در رشته مهندسی برق گرایش کنترل

عباس امیدی

امیرحسین حیدریان اردکانی

میلاد سلطانی کادرویش

امیرحسین کازرونی

آیدا محمدشاهی

اساتید راهنما:

دکتر شمقدری

دکتر عباداللهی

شهریور ماه ۱۴۰۰


بسم الله الرحمن الرحیم

تشکر و قدردانی:

در ابتدا لازم است تا از زحمات خانواده‌هایمان و تمام کسانی که ما را در به ثمر رساندن این پروژه یاری نمودند، از جمله اساتید دلسوزمان دکتر عباداللهی، دکتر شمقدری و دکتر اسماعیل‌زاده که با راهنمایی‌ها و کمک‌های سخاوتمندانه‌ی خود نقش به‌سزایی در پیشبرد این پژوهش ایفا نمودند، مهندس امین‌حاتمی و مهندس مرادی که برادرانه ما را تا رسیدن به نتایج مطلوب همراهی کردند و همچنین مهندسین جوان امیرمهدی و امیرمحمد ظریف که شبیه‌سازی گرافیکی پروژه بدون یاری ایشان میسر نمی‌شد، تشکر و قدردانی نموده و از خداوند متعال برای ایشان، موفقیت و سربلندی آرزومندیم.

چکیده

در پروژه طراحی، شبیه‌سازی و ساخت خودروی خودران با قابلیت درک محیط، برنامه ریزی و کنترل سعی شده است با رویکرد کنترل، هوش مصنوعی و پردازش تصویر، سیستم خودرو خودرانی از ابتدا تا انتها طراحی، شبیه-سازی و ساخته شود. در بخش ادراک محیط خودرو از ترکیب دو رویکرد پردازش تصویر کلاسیک و یادگیری عمیق استفاده شده است. این بخش شامل تشخیص اشیا (انسان، خودرو سواری، اتوبوس و کامیون)، تشخیص تابلوهای راهنمایی رانندگی، تشخیص خطوط افقی (به طور مثال خطوط عابر پیاده) و عمودی (خطوط جاده) خیابان، اندازه گیری فاصله خودرو نسبت به خودرو های دیگر و تشخیص پیاده رو می‌باشد. ابتدا اشیا موجود در تصویر دید خودرو توسط آشکارساز[1] Yolov5 تشخیص داده می‌شوند و اگر تابلویی نیز در خیابان دیده شد، توسط آشکارساز Yolov5دیگری که برای تابلوهای راهنمایی رانندگی آموزش داده شده مشخص می‌شوند. حال با استفاده از شبکه عصبی PINet خطوط عمودی جاده ترسیم می‌شوند و در ادامه با استفاده از الگوریتم‌های پردازش تصویر کلاسیک با تعیین منطقه محدود کننده مورد نظر خطوط افقی که شامل خطوط عابر پیاده است به نمایش در می آیند. برای تعیین فاصله خودرو تشخیص داده شده نسبت به خودرو های از تحلیل اطلاعات سه بعدی صورت می‌گیرد که این کار توسط مفهوم تخمین عمق [2] انجام می‌شود. لازمه تخمین عمق وجود تصویر چپ و راست از یک دوربین است ولی با استفاده از شبکه عصبی SGDepth تخمین قابل قبولی نسبت به عمق با استفاده از یک تصویر حاصل می‌شود. همچنین در شبکه ذکر شده از بخش بندی[3] در فرآیند آموزش خود به منظور افزایش دقت تخمین عمق خود استفاده می‌کند به همین دلیل از مدل بخش بندی که همزمان در فرآیند تشخیص عمق آموزش داده شده به منظور تشخیص پیاده رو استفاده می‌گردد. بخش شبیه سازی شامل سه بخش می‌باشد. در بخش اول سعی شده است ابتدا محیط شبیه سازی توسط زبان پایتون ساخته شود و سپس با استفاده از الگوریتم های برنامه ریزی مسیر، مسیر یابی و کنترل کننده تطبیقی[4] سیستم پارک خودکار طراحی شود. بخش دوم شبیه سازی در شبیه ساز AVISEngine که در بستر یونیتی[5] می‌باشد انجام گرفته که در آن خودرو در دو محیط شهری و بزرگراه توسط الگوریتم‌های پردازش تصویر کلاسیک و یادگیری عمیق و سنوسور های موجود بر روی خودرو ادراک محیط را انجام دهد و سپس با استفاده از کنترل کننده PID خودرو در جاده کنترل گردد و تصمیم

[1] Object Detection
[2] Depth Estimation
[3] Segmentation
[4] MPC
[5] Unity

های مورد نیاز در هنگام رسیدن به چهار راه، تابلوهای راهنمایی و رسیدن به موانع اتخاذ کند. در نهایت سیستم خودرو خودرانی به صورت واقعی ساخته شده است که در آن سنسورهای مختلفی بر روی یک خودرو نصب شده و همچنین نقشه ای برای محیط شهری، بزرگراه و پارک خودکار ایجاد شده تا عملکرد خودرو در واقعیت بررسی شود.

واژه‌های کلیدی: پردازش تصویر، هوش مصنوعی، کنترل کننده تطبیقی، کنترل کننده PID، پارک خودکار، شبیه‌ساز، ادراک محیط

فهرست مطالب

مقدمه ۱

۱- خودروی خودران ۷

۱-۱- تعریف خودروی خودران ۷

۱-۱-۱- ادراک ۷

۱-۱-۲- تصمیم گیری ۸

۱-۱-۳- برنامه ریزی حرکت ۸

۱-۱-۴- کنترل ۹

۱-۲- سطوح خودمختاری خودرو ۱۰

۱-۲-۱- سطح صفر: بدون اتوماسیون رانندگی ۱۱

۱-۲-۲- سطح یکم : کمک راننده ۱۱

۱-۲-۳- سطح دوم : اتوماسیون رانندگی جزئی ۱۱

۱-۲-۴- سطح سوم : اتوماسیون رانندگی مشروط ۱۱

۱-۲-۵- سطح چهارم : اتوماسیون رانندگی بالا ۱۱

۱-۲-۶- سطح پنجم : اتوماسیون کامل رانندگی ۱۲

۱-۳- صنایع بالادستی و پایین‌دستی ۱۲

۱-۳-۱- صنایع بالادستی و پایین‌دستی صنعت خودرو ۱۲

۱-۳-۲- صنایع بالادستی و پایین‌دستی سامانه راننده خودکار ۱۳

۱-۴- تاریخچه خودروهای خودران ۱۶

۱-۵- چالش‌های خودروهای خودران ۲۲

۱-۵-۱- چالش‌های فنی ۲۳

۱-۵-۲- چالش‌های اجتماعی ۲۳

۱-۵-۳- چالش‌های قانونی ۲۴

۱-۶- فناوری‌های خودروهای خودران ۲۸

۱-۶-۱- فناوری‌های اصلی ۲۹

۱-۶-۲- فناوری‌های جانبی ۳۷

۱-۲-۶-۱- سیستم نظارت بر راننده ۳۷
۲-۲-۶-۱- سیستم کنترل کروز تطبیقی ۳۷
۳-۲-۶-۱- فناوری ۵جی ۳۷
۷-۱- مسئله‌ها و زیر مسئله‌ها ۳۹
۱-۷-۱- ادراک محیط ۴۰
۲-۷-۱- سیستم پردازشی ۴۲
۳-۷-۱- عملگرها ۴۵
۸-۱- ویژگی‌ها و قابلیت‌های سامانه راننده خودکار ۴۷
۱-۸-۱- ترمز هوشمند ۴۷
۲-۸-۱- حفظ خودرو در بین خطوط ۴۸
۳-۸-۱- پارک خودکار ۴۸
۴-۸-۱- تشخیص تابلوهای ایمنی ۴۹
۵-۸-۱- چراغ هوشمند ۵۰
۶-۸-۱- هشدار تغییر خط ۵۱
۷-۸-۱- تشخیص عابرین پیاده ۵۲
۹-۱- بازار هدف ۵۳
۱-۹-۱- مشتریان بالقوه و بالفعل : ۵۳
۲-۹-۱- بازار هدف داخلی و خارجی سامانه راننده خودکار: ۵۳
۱۰-۱- بازیگران صنعت: ۵۴
۱-۱۰-۱- بازیگران مرحله زیرساخت ۵۵
۲-۱۰-۱- تولید کنندگان ۵۶
۳-۱۰-۱- خدمات پس از فروش ۵۶
۱۱-۱- آمار واردات، صادرات و ظرفیت تولید ۵۶
۱-۱۱-۱- خودروهای هیبریدی وارداتی ۵۶
۲-۱۱-۱- ظرفیت تولید ۵۷
۱۲-۱- پیامد های ساخت خودروی خودران ۶۴
۱-۱۲-۱- اخلاقی و اجتماعی ۶۴

۲-۱۲-۱- اقتصادی ۶۶
۲-۱۲-۳- محیط زیست ۶۸
۲-۱۲-۴- سیاست ۶۹
۲-۱۳-۱- نمونه‌های مشابه ۶۹
۲-۱۳-۱-۱- محصولات تسلا ۶۹
۲-۱۳-۱-۲- ویمو ۷۰
۲-۱۳-۱-۳- لوسید ۷۱
۲-۱۳-۱-۴- خلیج فارس ۷۳
۲- روش‌های علمی ۷۴
۲-۱- جمع آوری و تحلیل اطلاعات علمی ۷۴
۲-۱-۱-۲- ساختار کلی ارتباط مقالات ۷۴
۲-۱-۲-۲- درک محیط توسط خودرو خودران ۷۵
۲-۱-۳-۲- تصمیم گیری در خودروهای خودران ۸۲
۲-۲- یافتن راه حل زیر مسائل ۹۶
۲-۲-۱-۲- پارک خودکار ۹۶
۲-۲-۲-۲- ادراک محیط ۱۰۸
۲-۳- تحلیل راه حل های زیر مسئله ها ۱۳۵
۲-۳-۱-۲- پارک خودکار ۱۳۵
۲-۳-۲-۲- ادراک محیط ۱۵۲
۳- ادراک ۱۶۸
۳-۱- شناسایی عابر، وسایل نقلیه و چراغ راهنمایی و رانندگی ۱۶۸
۳-۱-۱-۳- تبیین روش مورد نظر ۱۷۰
۳-۱-۲-۳- چگونگی آموزش شبکه **YOLOv5** ۱۷۷
۳-۱-۳-۳- نتیجه آموزش و خروجی شبکه ۱۸۴
۳-۱-۳-۳- ساختار کد ۱۸۷
۳-۲- تشخیص علائم راهنمایی و رانندگی ۱۸۸
۳-۲-۱-۳- تبیین روش مورد نظر ۱۸۸

۳-۲-۲- دیتاست‌های مورد استفاده ۱۸۹
۳-۲-۲-۱- **Swedish Dataset** ۱۹۰
۳-۲-۲-۲- **German Traffic Sign Detection Benchmark GTSDB** ۱۹۲
۳-۲-۲-۳- **MakeML Cars and Traffic Signs Dataset** ۱۹۴
۳-۲-۲-۴- **MakeML Road Signs Dataset** ۱۹۶
۳-۲-۲-۵- **Vicos/DFG** ۱۹۷
۳-۲-۲-۶- **Tsinghua Traffic Sign Detection and Classification in the Wild** ۲۰۲
۳-۲-۳- چگونگی آموزش و صحه‌گذاری ۲۰۶
۳-۲-۴- نتیجه خروجی و صحه‌گذاری ۲۰۷
۳-۲-۵- تشریح نحوه پیاده سازی الگوریتم و ساختار کد ۲۱۰
۳-۳- تشخیص خطوط جاده ۲۱۲
۳-۳-۱- تشخیص خطوط افقی ۲۱۲
۳-۳-۱-۱- تبیین روش مورد نظر به همراه ریاضیات کامل ۲۱۲
۳-۳-۱-۲- تشریح نحوه پیاده سازی الگوریتم، ساختار کد و ارائه ی نتایج ۲۱۷
۳-۳-۲- تشخیص خطوط عمودی ۲۲۵
۳-۳-۲-۱- تبیین شبکه **PINet** [۲۶۵] ۲۲۵
۳-۳-۲-۲- چگونگی آموزش شبکه **PINet** ۲۳۲
۳-۳-۲-۴- نتیجه آموزش و خروجی شبکه ۲۴۲
۳-۳-۲-۵- تشریح نحوه پیاده سازی الگوریتم و ساختار کد ۲۵۰
۳-۴- تعیین فاصله نسبت به خودروهای اطراف ۲۵۶
۳-۴-۱- فاصله سنج کلاسیک ۲۵۶
۳-۴-۱-۱- تبیین روش مورد نظر به همراه ریاضیات کامل ۲۵۶
۳-۴-۱-۲- تشریح نحوه پیاده سازی الگوریتم، ساختار کد و ارائه ی نتایج ۲۶۱
۳-۴-۲- فاصله سنج با رویکرد شبکه‌های عصبی ۲۶۶
۳-۴-۲-۱- تبیین روش مورد نظر ۲۶۶
۳-۴-۲-۲- چگونگی آموزش ۲۷۲
۳-۴-۲-۳- نتیجه آموزش و خروجی شبکه ۲۷۴

۴-۲-۴-۳- تشریح نحوه پیاده سازی الگوریتم و ساختار کد ۲۸۲
۵-۲-۴-۳- ایده ۲۸۵
۵-۳- تشخیص پیاده‌رو ۲۸۶
۱-۵-۳- تبیین روش مورد نظر ۲۸۶
۲-۵-۳- چگونگی آموزش و صحه‌گذاری ۲۸۶
۳-۵-۳- نتیجه آموزش، تشریح نحوه پیاده سازی الگوریتم و ساختار کد ۲۸۶
۶-۳- تشخیص خطوط عابر پیاده ۲۹۰
۱-۱-۶-۳- تبیین روش مورد نظر به همراه ریاضیات کامل ۲۹۰
۲-۱-۶-۳- تشریح نحوه پیاده سازی الگوریتم، ساختار کد و ارائه ی نتایج ۲۹۳
۷-۳- ارائه یک مثال حل شده ۲۹۶
۱-۷-۳- تشریح بستر نرم افزاری (کد اصلی) ۲۹۷
۲-۷-۳- تشریح آرگمان‌ها ۳۱۰
۳-۷-۳- خروجی نرم افزار برای ویدئو تست اول ۳۱۴
۴-۷-۳- خروجی نرم افزار برای ویدئو تست دوم ۳۱۷
۵-۷-۳- خروجی نرم افزار برای ویدئو تست سوم ۳۱۸
۴- پارک خودکار ۳۱۹
۱-۴ دینامیک و سینماتیک خودرو ۳۱۹
۱-۱-۴- سینماتیک خودرو ۳۱۹
۱-۱-۱-۴- دستگاه مختصات ثابت به زمین ۳۱۹
۲-۱-۱-۴- دستگاه مختصات بدنی ۳۲۰
۳-۱-۱-۴- تبدیل دستگاههای مختصات ۳۲۰
۴-۱-۱-۴- معادلات سینماتیک خودرو ۳۲۰
۲-۱-۴- نیروهای خارجی وارد بر خودرو ۳۲۱
۱-۲-۱-۴- نیروهای وارد بر چرخ ۳۲۱
۱-۱-۲-۱-۴- شعاع دوران ۳۲۲
۲-۱-۲-۱-۴- نیروی مقاوم دوران ۳۲۳
۳-۱-۲-۱-۴- نیروی مجانبی تایرها ۳۲۴

۲-۲-۱-۴- نیروی مقاومت هوا ۳۲۵
۳-۱-۴- مدل‌سازی دینامیک در راستای طولی ۳۲۶
۱-۳-۱-۴- معادلات دینامیکی حرکت در راستای طولی ۳۲۶
۲-۳-۱-۴- مدل‌سازی سیستم تولید و انتقال قدرت ۳۲۹
۱-۲-۳-۱-۴- مدل‌سازی موتور خودرو ۳۳۰
۲-۲-۳-۱-۴- مدل‌سازی رفتار راننده ۳۳۲
۳-۲-۳-۱-۴- مدل‌سازی سیستم انتقال قدرت ۳۳۲
۴-۲-۳-۱-۴- سیستم دینامیکی معادل ۳۳۴
۱-۴-۲-۳-۱-۴- گشتاورهای معادل کاهیده وسیله نقلیه ۳۳۶
۲-۴-۲-۳-۱-۴- ممان اینرسی‌های معادل کاهیده اجزای انتقال قدرت ۳۳۶
۴-۱-۴- مدل‌سازی دینامیک عرضی ۳۳۷
۱-۴-۱-۴- استخراج زوایای چرخش تایرها ۳۴۰
۲-۴-۱-۴- معادلات دینامیکی در راستای عرضی ۳۴۱
۵-۱-۴- پیاده‌سازی در بستر نرم‌افزاری ۳۴۳
۲-۴- توسعه قانون هدایت ۳۵۴
۱-۲-۴- الگوریتم A^* ۳۵۴
۲-۲-۴- درون یابی اسپلاین (**spline**) ۳۵۸
۳-۲-۴- طراحی مسیر پارک دوبل ۳۶۴
۳-۴- توسعه و بیان قانون کنترل ۳۶۹
۱-۳-۴- مقدمه‌ای بر کنترل پیشبین مدل ۳۶۹
۱-۱-۳-۴- مقایسه کنترل تناسبی- انتگرالی- مشتقی (**PID**) و کنترل پیش بین مدل ۳۷۰
۲-۱-۳-۴- استراتژی کنترل پیشبین مدل ۳۷۱
۳-۱-۳-۴- اصل افق کاهنده ۳۷۶
۴-۱-۳-۴- مزایای کنترل پیش‌بین مدل ۳۷۸
۵-۱-۳-۴- معایب کنترل پیش‌بین مدل ۳۷۹
۲-۳-۴- مدل سیستم ۳۷۹
۳-۳-۴- مدل اغتشاشات ۳۸۲

۴-۳-۴- پیاده سازی در بستر نرم افزاری ۳۸۳
۵- شبیه سازی پارک خودکار ۳۸۸
۵-۱- زبان برنامه نویسی ۳۸۹
۵-۱-۱- مفهوم شیءگرایی ۳۸۹
۵-۱-۲- معرفی کتابخانه ها ۳۸۹
۵-۲- ورود اطلاعات ۳۹۰
۵-۲-۱- پارکینگ ۱ ..... ۳۹۱
۵-۲-۲- ورود اطلاعات به صورت دستی ۳۹۲
۵-۳- نمایش اطلاعات ۳۹۳
۵-۳-۱- محیط شبیه سازی ۳۹۳
۵-۳-۲- ثبت اطلاعات ۳۹۷
۵-۴- برنامه ریزی مسیر ۴۰۰
۵-۴-۱- برنامه ریزی مسیر بخش اول ۴۰۰
۵-۴-۲- برنامه ریزی مسیر بخش دوم ۴۰۲
۵-۵- مدل سینماتیکی خودرو ۴۰۶
۵-۶- کنترل کننده پیش بین ۴۰۷
۵-۷- ساختار اصلی کد ۴۰۹
۵-۸- ارائه یک مثال از شبیه سازی ۴۱۲
۶- شبیه سازی شهری و بین شهری ۴۲۰
۶-۱- محیط شبیه سازی ۴۲۰
۶-۲- دینامیک شبیه سازی خودرو و محیط ۴۲۳
۶-۲-۱- فیزیک خودرو ۴۲۳
۶-۲-۲- تنظیمات برخورد ۴۲۵
۶-۲-۳- تنظیمات اصطکاک و شتاب گرانشی ۴۲۸
۶-۳- قابلیت‌ها ۴۲۹
۶-۳-۱- محیط‌ها ۴۲۹
۶-۳-۲- داشبورد و پنل نمایش ۴۳۳

۳-۳-۶- موانع ۴۳۴
۴-۳-۶- سنسور های اولتراسونیک ۴۳۵
۵-۳-۶- تابلو ها و علائم راهنمایی و رانندگی ۴۳۷
۷- شبیه سازی شهری ۴۴۰
۱-۷- تشخیص خطوط ۴۴۰
۱-۱-۷- تشخیص خطوط عمودی ۴۴۰
۲-۱-۷- تشخیص خطوط افقی ۴۴۵
۲-۷- سنسورهای مورد استفاده ۴۴۷
۱-۲-۷- سنسور انکودر ۴۴۷
۱-۱-۲-۷- سنسور افزایشی (**Incremental**) ۴۴۷
۲-۱-۲-۷- سنسور مطلق (**Absolute**) ۴۵۰
۳-۷- سیستم دور زدن در محیط شبیه ساز شهری ۴۵۲
۴-۷- سنسور آلتراسونیک ۴۵۵
۱-۴-۷- معرفی سنسور آلتراسونیک ۴۵۶
۲-۴-۷- سنسورهای التراسونیک در شبیه سازی: ۴۶۱
۵-۷- سیستم عدم برخورد با مانع و ترمز اضطراری ۴۶۵
۱-۵-۷- گام اول: مقادیر سنسور آلتراسونیک ۴۶۶
۲-۵-۷- گام دوم: ایست پشت مانع ۴۶۶
۳-۵-۷- گام سوم: شرط **side_pix** را بررسی کن ۴۶۶
۴-۵-۷- گام چهارم: سیستم دور زدن مانع ۴۶۶
۶-۷- شناسایی تابلوهای راهنمایی و رانندگی در محیط شبیه ساز ۴۶۷
۱-۶-۷- تابلوهای راهنمایی و رانندگی موجود در شبیه ساز ۴۶۷
۲-۶-۷- نحوه‌ی انجام شناسایی ۴۶۹
۳-۶-۷- نحوه‌ی پیدا کردن ماسک ۴۷۲
۴-۶-۷- سیستم تصمیمگیری در محیط شهری ۴۷۶
۵-۶-۷- سیستم کنترل موقعیت خودرو در حالت شهری ۴۸۲
۸- شبیه سازی بین شهری ۴۸۴

۸-۱- تشخیص لاین در بخش بزرگراه ۴۸۴
۸-۲- تشخیص خط زرد و تعیین محل خودرو ۴۸۹
۸-۳- سنسورهای مورد استفاده ۴۹۰
۸-۴- سیستم عدم برخورد با مانع در بزرگراه ۴۹۱
۸-۴-۱- گام اول: تبدیل تصویر از **BGR** به **HSV** ۴۹۲
۸-۴-۲ گام دوم: بدست آوردن ماسک مانع ۴۹۳
۸-۴-۳ گام سوم: حذف نویز با عملیات مورفولوژی ۴۹۴
۸-۴-۴ گام چهارم: اعمال منطقه مورد نظر به خروجی گام سوم ۴۹۸
۸-۴-۵ گام پنجم: استخراج کانتور (خطوط مرزی) ۴۹۹
۸-۵- سیستم تصمیم گیری در محیط بین شهری ۵۰۲
۸-۶- سیستم کنترل موقعیت خودرو ۵۰۷
۹- توصیف سیستم ۵۱۲
۹-۱- شماتیک سخت افزاری ۵۱۲
۹-۲- عملکرد سیستم در محیط پارک ۵۱۳
۹-۳- عملکرد سیستم در محیط شهری و بین شهری ۵۱۴
۱۰ - سخت افزار و پیاده سازی ۵۱۵
۱۰-۱- اجزای خودرو ۵۱۵
۱۰-۱-۱- سروو موتور ۵۱۵
۱۰-۱-۲- تعریف سروو موتور ۵۱۶
۱۰-۱-۳- مکانیزم سروو موتور ۵۱۶
۱۰-۱-۴- کاربرد سروو موتور ۵۱۶
۱۰-۱-۵- موتورگیربکس ۵۱۷
۱۰-۱-۶- موتورهای الکتریکی ۵۱۷
۱۰-۱-۷- موتور الکتریکی **DC** ۵۱۷
۱۰-۱-۸- موتور الکتریکی**AC** ۵۱۷
۱۰-۱-۹- گیربکس ۵۱۸
۱۰-۱-۱۰- درایو موتور ۵۲۲

۲-۱۰- آردوینو ۵۲۵
۱-۲-۱۰- مشخصات آردوینو اونو ۵۲۵
۲-۲-۱۰- نحوهی کار با آردوینو ۵۲۷
۳-۲-۱۰- پروتکلهای ارتباطی پشتیبانیشده در آردوینو اونو ۵۲۹
۳-۱۰- جتسون نانو ۵۳۰
۱-۳-۱۰- مشخصات ۵۳۱
۲-۳-۱۰- سیستم عامل ۵۳۴
۳-۳-۱۰- دوربین ۵۳۵
۴-۱۰- محیط تست فیزیکی ۵۳۸
۱-۴-۱۰- محیط بین شهری ۵۳۹
۲-۴-۱۰- محیط پارک خودکار ۵۴۲
۳-۴-۱۰- محیط شهری ۵۴۵
۵-۱۰- ساخت خودرو ۵۵۰
۱-۵-۱۰- اتصالات موتور ۵۵۰
۲-۵-۱۰- اتصال سنسورهای آلتراسونیک ۵۵۱
۳-۵-۱۰- اتصال برد جتسون نانو و دوربین ۵۵۲
۴-۵-۱۰- اتصال برد آردوییِنو ۵۵۳
۵-۵-۱۰- تغذیه خودرو ۵۵۴
۶-۵-۱۰- نتیجه ساخت ۵۵۵
۱۱- نتایج ۵۵۷
۱-۱۱- برنامه ریزی مسیر در نقشه پارک ۵۵۷
۱-۱-۱۱- پردازش تصویر نقشه ۵۵۸
۲-۱-۱۱- مسیریابی ۵۶۰
۳-۱-۱۱- بررسی سناریوهای مختلف پارک ۵۶۳
۲-۱۱- الگوریتم‌های پردازش تصویر در نقشه های شهری و بین شهری ۵۶۶
۱-۲-۱۱- تشخیص خطوط جاده ۵۶۶
۲-۲-۱۱- تشخیص تابلوهای راهنمایی و رانندگی ۵۶۹

۳-۲-۱۱- تشخیص موانع قرمز رنگ ۵۷۰
۳-۱۱- کنترل خودرو ۵۷۳
۱-۳-۱۱- ترم تناسبی (**Proportional**) ۵۷۳
۲-۳-۱۱- ترم مشتقی (**Derivative**) ۵۷۴
۳-۳-۱۱- ترم انتگرالی (**Integral**) ۵۷۴
۴-۳-۱۱- کنترلر نهایی طراحی شده ۵۷۵
۵-۳-۱۱- مقایسه ی انواع کنترلر های قابل پیاده سازی ۵۷۷
۴-۱۱- عملکرد خودرو در محیط واقعی ۵۸۱
۱-۴-۱۱- نقشه پارکینگ ۵۸۱
۲-۴-۱۱- نقشه شهری ۵۸۲
۳-۴-۱۱- نقشه بین شهری ۵۸۴
۵-۱۱- نتیجه گیری ۵۸۷
۶-۱۱- پیشنهادات ۵۸۸
مراجع ۵۸۹

فهرست اشکال

شکل الف محیط های شبیه‌سازی و پیاده‌سازی درنظر گرفته شده برای توسعه و بررسی الگوریتم‌های خودروی خودران ۵
شکل ب نگاه کلی به فعالیت های صورت گرفته در این پروژه ۶
شکل ۱-۱ نمای کلی از مراحل سیستم خودروی خودران ۷
شکل ۱-۲ نمای کلی از سیستم کنترلی به همراه اجزا ۱۰
شکل ۱-۳ نمای کلی لز سطوح مختلف خودمختاری سامانه‌ها ۱۰
شکل ۱-۴ صنایع مرتبط با صنعت خودرو ۱۳
شکل ۱-۵ صنایع مرتبط با سامانه‌های خودکار ۱۳
شکل ۱-۶ اولین خودرو کنترل از راه دور با نام شگفتی آمریکایی (سمت راست) و فرانسیس اودینا (سمت چپ) ۱۷
شکل ۱-۷ نمای از بیرون و داخل ون رباتیک ساخته دانشگاه مونیخ ۱۷
شکل ۱-۸ نمای داخلی و بیرونی خودروی دیملر بنز ۱۸
شکل ۱-۹ خودروی Navlab1 (سمت راست) و Navlab2 (سمت چپ) ۱۸
شکل ۱-۱۰ شرکت کنندگان در مسابقات دارپا ۲۰
شکل ۱-۱۱ نمایی از خودروی فایرفلای گوگل ۲۱
شکل ۱-۱۲ نمایی از سیستم راننده خودکار شرکت تسلا ۲۲
شکل ۱-۱۳ نمای کلی از فناوری‌های به کار گرفته شده در سامانه راننده خودکار ۲۸
شکل ۱-۱۴ نمونه‌ای از سنسورهای لیدار مورد استفاده در سامانه‌های خودران ۲۹
شکل۱۵-۱ نمایی از شبیه‌سازی سه‌بعدی محیط اطراف به وسیله فناوری لیدار ۳۰
شکل ۱-۱۶ نمونه‌ای از سنسورهای اولتراسونیک مورد استفاده در سامانه‌های راننده خودکار ۳۰
شکل ۱-۱۷ نمونه‌ای از دوربین‌های مورد استفاده در سامانه‌های راننده خودکار ۳۲
شکل ۱-۱۸ نمونه‌ای از رادارهای مورد استفاده در سامانه‌های راننده خودران ۳۳
شکل ۱-۱۹ نمونه‌ای از بینایی ماشین ۳۴
شکل ۱-۲۰ شمای کلی الگوریتم PID ۳۶
شکل ۱-۲۱ تقسیم‌بندی مسئله‌ها و زیرمسئله‌ها ۳۹
شکل ۱-۲۲ نقشه ابر نقاط تهیه شده توسط لیدار ۴۱
شکل ۱-۲۳ مسائل، زیر مسائل و چالش‌های سامانه راننده خودکار ۴۶
شکل ۱-۲۴ ویژگی ترمز هوشمند ۴۷
شکل ۱-۲۵ قابلیت حفظ خودرو در بین خطوط ۴۸
شکل ۱-۲۶ نمونه پارک زاویه ای خودکار ۴۹
شکل ۱-۲۷ قابلیت خواندن علائم رانندگی ۵۰
شکل ۱-۲۸ نمونه ای از چراغ هوشمند ۵۱
شکل ۱-۲۹ قابلیت هشدار تغییرخط ۵۲
شکل ۱-۳۰ ویژگی تشخیص عابرپیاده ۵۲
شکل ۱-۳۱ آمار تولید سالیانه کشور ایران ۵۷
شکل ۱-۳۲ روند تعداد خودروهای سواری تولید شده در جهان ۶۰

شکل ۳۳-۱ روند تعداد خودروهای سواری تولید شده در ایران ۶۰
شکل ۳۴-۱ روند تعداد خودروهای سواری مورد استفاده در ایران ۶۰
شکل ۳۵-۱ پیامدهای اخلاقی و اجتماعی خودروهای خودران ۶۴
شکل ۳۶-۱ پیامدهای اقتصادی خودروهای خودران ۶۶
شکل ۳۷-۱ درآمد خدمات جهانی حاصل از رانندگی خودران در سال ۲۰۵۰ (بر حسب میلیون دلار آمریکا) ۶۶
شکل ۱-۲ نقشه ی مبتنی بر لیدار ۷۶
شکل ۲-۲ منحنی spline ۸۴
شکل ۳-۲ روش انتخاب رفتار در مواجهه با عابران پیاده در معابر ۸۵
شکل ۴-۲ POMDP به عنوان یک شبکه تصمیم گیری پویا ۸۷
شکل ۵-۲ مثالی از یک مسیر تولید شده با روش RRT ۸۹
شکل ۶-۲ برنامه ریزی حرکت در خودرو خودران ۹۱
شکل ۷-۲ نمودار بلوکی از سیستم جلوگیری از مانع که توسط Guidolini ارائه شده است ۹۲
شکل ۸-۲ نمودار بلوکی زیرسیستم کنترل ۹۳
شکل ۹-۲ دیاگرام بلوکی کنترل کننده PID ۹۴
شکل ۱۰-۲ هندسه روش پیگیری خالص ۹۵
شکل ۱۱-۲ نمای کلی از ساختار سیستم پارک خودکار ۹۷
شکل ۱۲-۲ نمای کلی از سیستم استریو ۹۸
شکل ۱۳-۲ ابر نقاط تولید شده با استفاده از دو دوربین ۹۹
شکل ۱۴-۲ راه حل‌های ارائه شده برای زیر مسئله پارک خودکار به تفکیک چالش ۱۰۷
شکل ۱۵-۲ تشخیص تابلو STOP با استفاده از Haar-Cascade ۱۰۹
شکل ۱۶-۲ تعیین کلاس تابلو با استفاده از SVM ۱۰۹
شکل ۱۷-۲ نمونه ای از نتایج مدل‌های آشکارساز علائم راهنمایی و رانندگی ۱۱۲
شکل ۱۸-۲ پیدا کردن اشیا (خودرو ها) در تصویر یا استفاده از Haar-Cascade ۱۱۳
شکل ۱۹-۲ نحوه عملکرد یولو در تشخیص اشیا موجود در تصویر ۱۱۴
شکل ۲۰-۲ ساختار Yolov3 ۱۱۴
شکل ۲۱-۲ نحوه ی استخراج یک کلاس شی از یک تصویر با یولو ۱۱۵
شکل ۲۲-۲ مشاهده کلاس های مختلف اشیا از جمله عابر دوچرخه سوار با شبکه ی یولو ۱۱۵
شکل ۲۳-۲ معماری EfficientDet ۱۱۶
شکل ۲۴-۲ تصویر اصلی جهت لبه یابی ۱۱۶
شکل ۲۵-۲ لبه یابی با استفاده از الگوریتم Canny ۱۱۶
شکل ۲۶-۲ بهبود خطوط پیدا شده با استفاده از Hough Transform ۱۱۶
شکل ۲۷-۲ بنچمارک شبکه های موجود در تشخیص خطوط جاده ۱۱۷
شکل ۲۸-۲ نتایج PINet بر روی دیتاست TuSimple ۱۱۸
شکل ۲۹-۲ نتایج PINet بر روی دیتاست CULane ۱۱۸
شکل ۳۰-۲ نتایج کیفی LaneATT در TuSimple، CULane، و LLAMAS ۱۱۹
شکل ۳۱-۲ نمونه هایی از نتایج مشخص شده از خط و جاده توسط VPGNet5 ۱۲۰

شکل ۲-۳۲ (الف) مثال‌هایی از مجموعه داده برای سناریوهای مختلف. (ب) نسب سناریو های مختلف در آموزش این شبکه ۱۲۱
شکل ۲-۳۳ مجموعه ای از روش های کلاسیک برای ردیابی اجسام متحرک ۱۲۲
شکل ۲-۳۴ ردیابی شی مورد نظر در فریم های مختلف ۱۲۳
شکل ۲-۳۵ ساختار شبکه ROLO ۱۲۳
شکل ۲-۳۶ مشخص کردن باکس اطراف شی ۱۲۴
شکل ۲-۳۷ بدست آوردن ماسک و ردیابی آن توسط شبکه ۱۲۴
شکل ۲-۳۸ ساختار ردیابی اجسام توسط Tractor++ ۱۲۵
شکل ۲-۳۹ بدست آورن Perspective Transform جاده و محاسبه فاصله پیکسلی نسبت به خودرو ۱۲۶
شکل ۲-۴۰ تعیین فاصله نسبت به خودرو های دیگر با آموزش شبکه های عصبی بر روی داده های رادار و لیدار ۱۲۶
شکل۲-۴۱ نحوه ی انجام تقلید رفتاری ۱۲۷
شکل ۲-۴۲ دیاگرام آموزش شبکه ۱۲۸
شکل ۲-۴۳ تشخیص اشیا و تقسیم بندی موردی در یک تصویر ۱۳۰
شکل ۲-۴۴ مقایسه‌ی شناسایی اشیا و تقسیم بندی معنایی ۱۳۱
شکل ۲-۴۵ مقایسه‌ی تقسیم بندی اشیا با قدم های به ترتیب از چپ به راست ۸،۱۶،۳۲ پیکسلی و مقایسه با برچسب دستی ۱۳۲
شکل ۲-۴۶ نقشه‌ی بازگشتی مادون قرمز ۱۳۳
شکل ۲-۴۷ نقشه با خروجی کالیبره نشده ۱۳۴
شکل ۲-۴۸ نقشه با خروجی کالیبره شده ۱۳۴
شکل ۲-۴۹ استفاده از سنسور اولتراسونیک در خودرو ۱۳۶
شکل ۲-۵۰ موقعیت ماشین نسبت به ماشین‌های پارک شده ۱۳۶
شکل ۲-۵۱ مستطیل¬های جایگزین شده ۱۳۶
شکل ۲-۵۲ داده-های به دست آمده از لیدار ۱۳۶
نمودار۲-۵۳ الگوریتم‌های مختلف بینایی ماشین ۱۳۷
شکل ۲-۵۴ انواع روش‌های پارک ۱۳۹
شکل ۲-۵۵ پارک عمودی از عقب ۱۴۰
شکل ۲-۵۶ پارک عمودی از جلو ۱۴۰
شکل ۲-۵۷ خط سیر حالت ایده آل ۱۴۱
شکل ۲-۵۸ زاویه‌ی ماشین نسبت به ماشین پارک شده ۱۴۲
شکل ۲-۵۹ مسیر طی شده در حالت غیر ایده آل ۱۴۲
شکل ۲-۶۰ حالت‌های برخورد وسیله نقلیه با مانع ۱۴۳
شکل ۲-۶۱ نمای کلی از وضعیت ماشین ۱۴۴
شکل ۲-۶۲ قوانین فازی ۱۴۴
شکل ۲-۶۳ توابع عضویت ۱۴۴
شکل ۲-۶۴ بلوک دیاگرام روش کنترلی فازی ترکیبی با PID ۱۴۵
شکل ۲-۶۶ مقایسه‌ی دو کنترلر فازی پیشرفته و PID ۱۴۶

شکل ۲-۶۵ مسیر مرجع و مسیر دنبال شده ۱۴۶
شکل ۲-۶۷ بلوک دیاگرام کنترلر SMC پیشرفته ۱۴۷
شکل ۲-۶۸ مقایسه‌ی کنترلرهای پیاده سازی شده ۱۴۸
شکل ۲-۶۹ مدل کنترلی MPC ۱۴۸
شکل ۲-۷۰ مقایسه‌ی عملکردی روش‌های کنترلی ۱۴۹
شکل ۲-۷۱ مقایسه‌ی عملکردی روش‌های کنترلی ۱۴۹
شکل ۲-۷۲ مقایسه‌ی روش‌های کنترلی ۱۵۱
شکل ۲-۷۳ نمونه‌ای از موزاییک آگمنتیشن ۱۵۳
شکل ۲-۷۴ نمونه‌ای از موزاییک آگمنتیشن ۱۵۴
شکل ۲-۷۵ مقایسه EfficientNet از لحاظ دقت بر حسب تعداد پارامترها با شبکه های دیگر ۱۵۵
شکل ۲-۷۶ مقایسه عملکرد EfficientDet و YOLOv5 از لحاظ دقت بر حسب سرعت ۱۵۷
شکل ۲-۷۷ مقایسه روش های مختلف از لحاظ FLOPS بر زمان اجرا ۱۵۹
شکل ۲-۷۸ مقایسه روش های مختلف از لحاظ تعداد پارامترها بر زمان اجرا ۱۶۰
شکل ۲-۷۹ مقایسه روش های مختلف از لحاظ مصرف حافظه بر زمان اجرا ۱۶۰
شکل ۲-۸۰ بررسی کلی مدلهای آشکارساز علائم راهنمایی و رانندگی ۱۶۲
شکل ۲-۸۱ نمونه ای از عملکرد روش مبتنی بر پردازش تصویر کلاسیک ۱۶۵
شکل ۲-۸۲ نمونه ای از عملکرد روش مبتنی بر شبکه های عصبی و سنسور های رادار و لیدار ۱۶۶
شکل ۲-۸۳ ساختار عملکردی انتها به انتها یک خودرو خودران ۱۶۷
شکل ۳-۱ نمایی از مراحل شناسایی اشیا توسط YOLO ۱۶۹
شکل ۳-۲ نمونه‌ای از داده‌افزایی با استفاده از روش تبدیل مقیاس تصویر ۱۷۱
شکل ۳-۳ نمونه‌ای از داده‌افزایی با استفاده از روش تبدیل تنظیم فضای رنگ ۱۷۲
شکل ۳-۴ نمونه‌ای از روش داده‌افزایی موزاییکی ۱۷۳
شکل ۳-۵ ساختار شبکه CSPNet ۱۷۵
شکل ۳-۶ توزیع کلاس‌های مختلف دیتاست COCO ۱۷۹
شکل ۳-۷ نمونه‌ای از تصویر لیبل زده شده موجود در دیتاست COCO ۱۸۱
شکل ۳-۸ ساختار پوشه‌بندی دیتاست ۱۸۲
شکل ۳-۹ مدل‌های مختلف YOLOv5 و مشخصات آن‌ها ۱۸۳
شکل ۳-۱۰ نتایج آموزش شبکه YOlOv5s ۱۸۴
شکل ۳-۱۱ نتیجه شبکه YOLOv5 بر روی داده‌های تست ۱۸۶
شکل ۳-۱۲ نمونه‌ای از دیتاست Swedish ۱۹۱
شکل ۳-۱۳ نمونه‌ای دوم از دیتاست Swedish ۱۹۱
شکل ۳-۱۴ نمودار شمارش هر کدام از کلاس‌ها در دیتاست GDSDB ۱۹۳
شکل ۳-۱۵ تعداد اشیا هر کلاس در دیتاست MakeML ۱۹۴
شکل ۳-۱۶ نمونه‌ای از تصاویر دیتاست MakeML Cars and Traffic Signs Dataset ۱۹۵
شکل ۳-۱۷ نمونه‌ای دیگر از تصاویر دیتاست MakeML Cars and Traffic Signs Dataset ۱۹۵
شکل ۳-۱۸ نمونه‌ای از دیتاست MakeML Road Signs Dataset ۱۹۶

شکل ۳-۱۹ توزیع تعداد تابلوهای مربوط به هر کلاس در دیتاست DFG ۱۹۸
شکل ۳-۲۰ دیتاست Vicos یا DFG ۱۹۸
شکل ۳-۲۱ نمونه‌ی شماره ۱ از دیتاست DFG ۱۹۹
شکل ۳-۲۲ نمونه‌ی شماره ۲ از دیتاست DFG ۲۰۰
شکل ۳-۲۳ نمونه‌ی ای از دیتاست مصنوعی دیتاست DFG ۲۰۰
شکل ۳-۲۴ توزیع تابلوها در دیتاست DFG پس از استخراج تابلوهای مورد نظر ۲۰۱
شکل ۳-۲۵ کلاس‌های موجود در دیتاست Tsinghua ۲۰۲
شکل ۳-۲۶ نمونه‌ای از دیتاست Tsinghua ۲۰۴
شکل ۳-۲۷ توزیع تعداد و ابعاد داده ها در دیتاست نهایی ۲۰۵
شکل ۳-۲۸ نتیجه‌ی معیارهای مختلف شبکه‌ی آموزش داده شده Yolov5s بر روی دیتاست صحه‌گذاری ۲۰۷
شکل ۳-۲۹ تابلو دور برگردان ممنوع شناسایی شده در ویدئو شماره ۱ ..... ۲۰۸
شکل ۳-۳۰ تابلو ورود ممنوع شناسایی شده در ویدئو شماره ۲ ..... ۲۰۸
شکل ۳-۳۱ تابلوی رعایت حق تقدم شناسایی‌شده در یکی از خیابان‌های تهران ۲۰۹
شکل ۳-۳۲ تابلوهای متفاوت محدودیت سرعت که با برچسب‌های مختلف مشخص شده‌اند ۲۰۹
شکل ۳-۳۳ نمایش یک خط در فضای دو بعدی با معادله ی $\rho = xcos(\theta) + ysin(\theta)$ ۲۱۲
شکل ۳-۳۴ نمایش یک خط بالاتر از مبدا با نمایش قطبی ۲۱۳
شکل ۳-۳۵ دریافت نقاط ابتدایی و انتهایی خط تشخیص داده شده توسط الگوریتم هاف ۲۱۴
شکل ۳-۳۶ محاسبه ی شیب خطوط با نقاط دریافتی از الگوریتم هاف ۲۱۵
شکل ۳-۳۷ خط آبی رنگ، خط میانگین ترسیم شده از تمام خطوط افقی موجود در صفحه ۲۱۵
شکل ۳-۳۸ تعیین ناحیه ی دید مناسب برای سامانه، جهت میانگین گیری از خطوط افقی در آن ۲۱۶
شکل ۳-۳۹ تصویر داده شده به تابع به عنوان ورودی ۲۱۷
شکل ۳-۴۰ تصویر داده شده به تابع به عنوان ورودی جهت دریافت ناحیه ی دید راننده ۲۱۹
شکل ۳-۴۱ ناحیه ی دید راننده که توسط تابع برگردانده شده است ۲۲۰
شکل ۳-۴۲ تصویر داده شده به قطعه کد مربوط به انتخاب گستره ی رنگ خطوط جاده ۲۲۲
شکل ۳-۴۳ خروجی مربوط به قطعه کد مربوط به انتخاب گستره ی رنگ خطوط جاده ۲۲۲
شکل ۳-۴۴ تصویر ورودی اول به قطعه کد تشخیص خطوط افقی ۲۲۳
شکل ۳-۴۵ خروجی اول قطعه کد تشخیص خطوط افقی ۲۲۳
شکل ۳-۴۶ تصویر ورودی دوم به قطعه کد تشخیص خطوط افقی ۲۲۴
شکل ۳-۴۷ خروجی دوم قطعه کد تشخیص خطوط افقی ۲۲۴
شکل ۳-۴۸ ساختار شبکه PINet ۲۲۵
شکل ۳-۴۹ جزئیات بلوک ساعت شنی ۲۲۷
شکل ۳-۵۰ ماژول BottleNeck ۲۲۸
شکل ۳-۵۱ سورس کد شبکه PINet ۲۳۲
شکل ۳-۵۲ مثال‌هایی از مجموعه داده برای سناریوهای مختلف. (ب) درصد داده سناریو های مختلف در این دیتاست.. ۲۳۳
شکل ۳-۵۳ تقسیم‌بندی مجموعه داده CurveLanes ۲۳۵
شکل ۳-۵۴ نمونه‌هایی از مجموعه‌داده CurveLanes ۲۳۶

شکل ۳-۵۵ نمونه‌هایی از مجموعه‌داده Tusimple ۲۳۹
شکل ۳-۵۶ نتیجه تست توسط مدل TuSimple ۲۴۴
شکل ۳-۵۷ نتیجه تست دو توسط مدل TuSimple ۲۴۴
شکل ۳-۵۸ نتیجه تست یک توسط مدل CurveLanes ۲۴۵
شکل ۳-۵۹ نتیجه تست دو توسط مدل CurveLanes ۲۴۵
شکل ۳-۶۰ نتیجه تست سه توسط مدل CurveLanes ۲۴۶
شکل ۳-۶۱ نتیجه ویدئو یک ۲۴۷
شکل ۳-۶۲ نتیجه ویدئو دو ۲۴۷
شکل ۳-۶۳ نتیجه ویدئو سه ۲۴۸
شکل ۳-۶۴ نتیجه ویدئو چهار ۲۴۸
شکل ۳-۶۵ نتیجه ویدئو پنج ۲۴۹
شکل ۳-۶۶ اثر نقاشی نقاش ایتالیایی و مشاهده ی پرسپکتیو در آن ۲۵۷
شکل ۳-۶۷ انتخاب ذوزنقه ی مناسب جهت اعمال تبدیل پرسپکتیو ۲۵۸
شکل ۳-۶۸ نحوه تبدیل پرسپکتیو ۲۵۹
شکل ۳-۶۹ نحوه محاسبه فاصله ۲۶۰
شکل ۳-۷۰ تصویر ورودی قطعه کد مربوط به بخش به دست آوردن تصویر از بالا ۲۶۲
شکل ۳-۷۱ ذوزنقه‌ی انتخاب شده بر اساس خروجی بخش شناسایی خودرو ۲۶۲
شکل ۳-۷۲ تصویر ترنسفورم شده به دید از بالا ۲۶۳
شکل ۳-۷۳ خروجی اول قطعه کد محاسبه فاصله به روش کلاسیک ۲۶۴
شکل ۳-۷۴ خروجی دوم قطعه کد محاسبه فاصله به روش کلاسیک ۲۶۵
شکل ۳-۷۵ خروجی سوم قطعه کد محاسبه فاصله به روش کلاسیک ۲۶۵
شکل ۳-۷۶ نمای کلی از سیستم استریو ۲۶۶
شکل ۳-۷۷ دوربین استریو Zed ۲۶۷
شکل ۳-۷۸ تصاویر چپ و راست حاصل شده از دوربین Zed ۲۶۷
شکل ۳-۷۹ دیسپریتی حاصل شده از دو تصویر چپ و راست دوربین استریو Zed با الگوریتم SGBM ۲۶۸
شکل ۳-۸۰ ساختار شبکه SGDepth ۲۶۹
شکل ۳-۸۱ تقسیم‌بندی معنایی تصویر به منظور تشخیص اشیا در حال حرکت ۲۷۰
شکل ۳-۸۲ (a) تصویر ورودی، (b) نتیجه تصویر دیسپریتی به تنهایی، (c) تصویر دیسپریتی بعد از انجام آموزش کامل، (d) نتیجه تصویر تقسیم‌بندی شده به تنهایی، (e) تصویر تقسیم‌بندی شده بعد از انجام آموزش کامل ۲۷۱
شکل ۳-۸۳ نتایج شبکه SGDepth آموزش داده شده بر روی دو دیتاست Kitti و Cityscapes ۲۷۴
شکل ۳-۸۴ نتایج خروجی دیسپریتی و دسته بندی معنایی ویدئو تست ۱ ..... ۲۷۵
شکل ۳-۸۵ نتایج خروجی دیسپریتی و دسته بندی معنایی ویدئو تست ۲ ..... ۲۷۶
شکل ۳-۸۶ نتایج خروجی دیسپریتی و دسته بندی معنایی ویدئو تست۳ ..... ۲۷۷
شکل ۳-۸۷ هندسه دوربین استریو ۲۷۸
شکل ۳-۸۸ نتیجه خروجی اول اندازه گیری فاصله نسبت به خودروهای دیگر ۲۸۰
شکل ۳-۸۹ نتیجه خروجی دوم اندازه گیری فاصله نسبت به خودروهای دیگر ۲۸۰

شکل ۳-۹۰ نتیجه خروجی سوم اندازه گیری فاصله نسبت به خودروهای دیگر ۲۸۱
شکل ۳-۹۱ جداسازی خودرو از روی تصویر دیسپریتی ۲۸۵
شکل ۳-۹۲ ابرنقاط خودرو بدست آورده شده از تصویر ۳-۸۷ ..... ۲۸۵
شکل ۳-۹۳ نتیجه خروجی اول دسته‌بندی معنایی کلاس پیاده‌رو ۲۸۷
شکل ۳-۹۴ نتیجه خروجی دوم دسته‌بندی معنایی کلاس پیاده‌رو ۲۸۷
شکل ۳-۹۵ نمایش مفهوم تشخیص نقاط به هم پیوسته و یک رنگ در تصویر توسط کانتور ها ۲۹۰
شکل ۳-۹۶ فضای HSV ۲۹۱
شکل ۳-۹۷ تصویر داده شده به عنوان ورودی به قطعه کد تشخیص عابر پیاده ۲۹۴
شکل ۳-۹۸ خروجی اول قطعه کد تشخیص عابر پیاده ۲۹۴
شکل ۳-۹۹ خروجی قطعه کد تشخیص عابر پیاده بدون شرط مساحت ۲۹۵
شکل ۳-۱۰۰ تصویر داده شده به عنوان ورودی به قطعه کد تشخیص عابر پیاده ۲۹۵
شکل ۳-۱۰۱ خروجی دوم قطعه کد تشخیص عابر پیاده ۲۹۶
شکل ۳-۱۰۲ پوشه‌ی اصلی نرم‌افزار ۲۹۷
شکل ۳-۱۰۳ پوشه‌ی اصلی نرم‌افزار پس از اضافه کردن ویدئو ۳۰۵
شکل ۳-۱۰۴ تایپ cmd بر بخش آدرس پوشه‌ی اصلی ۳۰۶
شکل ۳-۱۰۵ محیط خط فرمان باز شده در پوشه‌ی اصلی ۳۰۶
شکل ۳-۱۰۶ عبارت مورد نیاز در خط فرمان ویندوز ۳۰۷
شکل ۳-۱۰۷ بارگذاری مدل‌های مختلف ۳۰۷
شکل ۳-۱۰۸ خروجی ویدئو در حال اجرای برنامه ۳۰۸
شکل ۳-۱۰۹ جزئیات پردازش ویدئو ۳۰۸
شکل ۳-۱۱۰ پوشه‌ی خروجی ویدئوی پردازش شده ۳۰۹
شکل ۳-۱۱۱ نمونه‌ی شماره‌ی اول از خروجی ویدئوی اول ۳۱۴
شکل ۳-۱۱۲ نمونه‌ی شماره دوم از خروجی ویدئوی اول ۳۱۴
شکل ۳-۱۱۳ نمونه‌ی شماره سوم از خروجی ویدئوی اول ۳۱۵
شکل ۳-۱۱۴ نمونه‌ی شماره چهارم از خروجی ویدئوی اول ۳۱۵
شکل ۳-۱۱۵ نمونه‌ی شماره پنجم از خروجی ویدئوی اول ۳۱۶
شکل ۳-۱۱۶ نمونه‌ی شماره اول از خروجی ویدئوی دوم ۳۱۷
شکل ۳-۱۱۷ نمونه‌ی شماره دوم از خروجی ویدئوی دوم ۳۱۷
شکل ۳-۱۱۸ نمونه‌ی شماره اول از خروجی ویدئوی سوم ۳۱۸
شکل ۳-۱۱۹ نمونه‌ی شماره دوم از خروجی ویدئوی سوم ۳۱۸
شکل ۴-۱ صفحه افقی دستگاه مختصات زمین ۳۱۹
شکل ۴-۲ صفحه افقی دستگاه مختصات زمین ۳۲۰
شکل ۴-۳ نیروها و گشتاورهای وارد بر تایر ۳۲۲
شکل ۴-۴ مرکز دوران تایر ۳۲۲
شکل ۴-۵ توزیع بار در تایر ۳۲۳
شکل ۴-۶ نیروها و گشتاورهای وارد بر خودرو در راستای طولی ۳۲۶

شکل ۴-۷ دیاگرام آزاد نیروها در چرخ جلوی خودرو ۳۲۸
شکل ۴-۸ نمای کلی از سیستم قدرت خودرو ۳۲۹
شکل ۴-۹ اجزای سیستم قدرت خودرو ۳۳۰
شکل ۴-۱۰ نمونه‌ای از منحنی‌های مشخصه موتور ۳۳۱
شکل ۴-۱۱ شماتیک سیستم انتقال قدرت خودرو در هنگام درگیری چرخ‌دنده‌ها ۳۳۲
شکل ۴-۱۲ سیستم معادل کاهیده انتقال قدرت ۳۳۳
شکل ۴-۱۳ راد اتصال در موتور احتراق داخلی ۳۳۷
شکل ۴-۱۴ دستگاه مختصات بدنی با مبدا مرکز جرم ۳۳۸
شکل ۴-۱۵ زاویه فرمان چرخ‌های محور جلو ۳۳۹
شکل ۴-۱۶ مدل دوچرخه از خودرو ۳۴۰
شکل ۴-۱۷ سرعت عرضی چرخ‌های محور جلو ۳۴۰
شکل ۴-۱۸ سرعت عرضی چرخ‌های محور عقب ۳۴۰
شکل ۴-۱۹ نیروهای خارجی وارد بر خودرو از دیدگاه مجانبی ۳۴۲
شکل ۴-۲۰ دیاگرام کلی مدل‌سازی دینامیکی ۳۴۳
شکل ۴-۲۱ مدل ماشین مورد استفاده در شبیه‌سازی دینامیکی ۳۴۴
شکل ۴-۲۲ مولفه‌های موقعیت بر حسب زمان برای ورودی‌های معادله ۶۶-۴ ۳۴۸
شکل ۴-۲۳ موقعیت خودرو در صفحه زمین برای ورودی‌های معادله ۶۶-۴ ۳۴۹
شکل ۴-۲۴ مولفه‌های سرعت خودرو بر حسب زمان برای ورودی‌های معادله ۶۶-۴ ۳۴۹
شکل ۴-۲۵ مولفه‌های موقعیت بر حسب زمان برای ورودی‌های معادله ۶۷-۴ ۳۵۰
شکل ۴-۲۶ موقعیت خودرو در صفحه زمین برای ورودی‌های معادله ۶۷-۴ ۳۵۰
شکل ۴-۲۷ مولفه‌های سرعت خودرو بر حسب زمان برای ورودی‌های معادله ۶۷-۴ ۳۵۱
شکل ۴-۲۸ مولفه‌های موقعیت بر حسب زمان برای ورودی‌های معادله ۶۹-۴ ۳۵۱
شکل ۴-۲۹ موقعیت خودرو در صفحه زمین برای ورودی‌های معادله ۶۹-۴ ۳۵۲
شکل ۴-۳۰ مولفه‌های سرعت خودرو بر حسب زمان برای ورودی‌های معادله ۶۹-۴ ۳۵۲
شکل ۴-۳۱ نمونه‌ی الگوریتم A^* ۳۵۵
شکل ۴-۳۲ یک نمونه از اعمال spline ۳۶۲
شک ۴-۳۳ طراحی مسیر پارک دوبل ۳۶۴
شکل ۴-۳۴ کمان های به دست آمده مسیر طراحی شده درجهت پارک دوبل ۳۶۵
شکل ۴-۳۵ شباهت رانندگی اتومبیل با استراتژی کنترل پیش‌بین مدل ۳۷۲
شکل ۴-۳۶ اساس کار کنترلر MPC ۳۷۳
شکل ۴-۳۷ ساختار اساسی کنترل پیش بین مدل ۳۷۷
شکل ۴-۳۸ ساختار کامل کنترل پیش بین مدل ۳۷۸
شکل۴-۳۹ نمودار پاسخ سیستم فرضی به ازای ورودی ضربه واحد و مقادیر پاسخ سیستم در زمان ۳۸۱
شکل ۴-۴۰ نمودار پاسخ سیستم فرضی به ازای ورودی پله واحد و مقادیر پاسخ سیستم در زمان ۳۸۲
شکل ۴-۴۱ نتیجه شبیه‌سازی برای مسیر مرجع معادله ۱۴۰-۴ ۳۸۷
شکل ۴-۴۲ نتیجه شبیه‌سازی برای مسیر مرجع معادله ۱۴۱-۴ ۳۸۷

شکل ۱-۵ دیاگرام بلوکی نرم‌افزار درحلقه پارک خودرکار ۳۸۹
شکل ۲-۵ محل‌های قابل پارک در پارکینگ ۳۹۳
شکل ۳-۵ محیط شبیه‌سازی توسعه داده شده ۳۹۵
شکل ۴-۵ رندر اطلاعات گرافیکی در محیط شبیه سازی توسعه داده شده ۳۹۷
شکل ۵-۵ نمودارهای ذخیره شده توسط ثبت کننده اطلاعات ۴۰۰
شکل۶-۵ مسیر با شکستگی های زیاد ۴۰۲
شکل ۷-۵ مسیر بدون شکستگی و به صورت منحنی ۴۰۲
شکل ۸-۵ چهار حالت ممکن برای ورود به نقطه پارک ۴۰۴
شکل ۹-۵ طراحی مسیر پارک براساس حالت چهارم ۴۰۵
شکل ۱۰-۵ ورود اطلاعات در شبیه ساز پارک خودکار ۴۱۳
شکل ۱۱-۵ رندر اولیه شبیه ساز مطابق با اطلاعات وارد شده ۴۱۳
شکل ۱۲-۵ برنامه ریزی مسیر ۴۱۴
شکل ۱۳-۵ کنترل و هدایت خودرو در مسیر برنامه ریزی شده ۴۱۵
شکل ۱۴-۵ پارک موازی در محل تعیین شده ۴۱۶
شکل ۱۵-۵ نمودار موقعیت افقی برحسب زمان ۴۱۷
شکل ۱۶-۵ نمودار موقعیت عمودی برحسب زمان ۴۱۷
شکل ۱۷-۵ نمودار سرعت خودرو برحسب زمان ۴۱۸
شکل ۱۸-۵ نمودار زاویه فرمان خودرو برحسب زمان ۴۱۸
شکل ۱۹-۵ نمودار موقعیت عمودی خودرو برحسب موقعیت افقی آن ۴۱۹
شکل ۲۰-۵ نمودار شتاب خودرو برحسب زمان ۴۲۰
شکل ۲۱-۵ نمودار زاویه خودرو (psi) برحسب زمان ۴۲۰
شکل ۱-۶ تصویری از طراحی گرافیکی اولیه از محیط شبیه سازی ۴۲۱
شکل ۲-۶ تصویری از نقشه ی استراتژیک اولیه از محیط شبیه سازی ۴۲۲
شکل ۳-۶ تصویری از طراحی گرافیکی پیاده سازی سایه و آفتاب از محیط شبیه سازی ۴۲۲
شکل ۴-۶ تصویری از یکی از نقشه های شهری مورد استفاده برای شبیه سازی ۴۲۳
شکل ۵-۶ تصویری از یکی از نقشه های شهری مورد استفاده برای شبیه سازی ۴۲۳
شکل ۶-۶ تصویری از یکی از نقشه های بین شهری مورد استفاده برای شبیه سازی ۴۲۴
شکل ۷-۶ مراحل تنظیم مشخصات فیزیکی و جرم خودرو ۴۲۵
شکل ۸-۶ طراحی نهایی خودروی مورد استفاده شده در شبیه سازی ۴۲۵
شکل ۹-۶ تصویری از برخورد خودرو با مانع در شبیه ساز و صفر شدن آنی سرعت ۴۳۰
شکل ۱۰-۶ تنظیمات شتاب گرانشی در شبیه‌ساز ۴۲۹
شکل ۱۱-۶ تنظیمات ضرایب اصطکاک در محیط شبیه‌ساز ۴۲۹
شکل ۱۲-۶ تصویری از یک محیط شهری در شبیه ساز ۴۳۰
شکل ۱۳-۶ تصویری از یک بن بست در محیط شهری در شبیه ساز ۴۳۱
شکل ۱۴-۶ تصویری از یک چهار راه در محیط شهری در شبیه ساز ۴۳۱
شکل ۱۵-۶ تصویری از یک محیط بین شهری در شبیه ساز ۴۳۲

شکل ۶-۱۶ تصویری از یک پیچ جاده ای تعبیه شده در محیط بین شهری در شبیه ساز ۴۳۲
شکل ۶-۱۷ تصویری از یک پیچ جاده ای تعبیه شده در محیط بین شهری در شبیه ساز ۴۳۳
شکل ۶-۱۸ تصویر داشبورد شبیه ساز ۴۳۳
شکل ۶-۱۹ تصویر تغییر وضعیت دیتای ارسالی از سنسور ها هنگام تشخیص مانع ۴۳۴
شکل ۶-۲۰ تصویری از پنل نمایش سرعت و زاویه ی چرخ ها به صورت بدون وقفه ۴۳۴
شکل ۶-۲۱ تصویری از پنل تنظیم حداکثر سرعت و زاویه ی سنسور ها ۴۳۴
شکل ۶-۲۲ تصویری از قرار گیری موانع در محیط بین شهری ۴۳۵
شکل ۶-۲۳ تصویری از قرار گیری موانع در محیط شهری ۴۳۵
شکل ۶-۲۴ تصویری از عملکرد سنسور های اولتراسونیک با زاویه ی ۳۹ درجه ۴۳۶
شکل ۶-۲۵ تصویری از عملکرد سنسور های اولتراسونیک با زاویه ی ۱۲ درجه ۴۳۶
شکل ۶-۲۶ تصویری از عملکرد سنسور های اولتراسونیک با زاویه ی ۴۰ درجه ۴۳۷
شکل ۶-۲۷ نمونه ای از تابلوی عبور مستقیم در محیط شبیه ساز ۴۳۷
شکل ۶-۲۸ نمونه ای از تابلوی گردش به راست در محیط شبیه ساز ۴۳۸
شکل ۶-۲۹ نمونه ای از تابلوی ایست در محیط شبیه ساز ۴۳۸
شکل ۷-۱ نمایی از جاده ۴۴۰
شکل ۷-۲ ماسک خودرو ۴۴۱
شکل ۷-۳ ماسک محیط اطراف خودرو ۴۴۲
شکل ۷-۴ ماسک خطوط جاده ۴۴۲
شکل ۷-۵ ماسک خطوط جاده بعد از اعمال منطقه مورد نظر ۴۴۳
شکل ۷-۶ مشخص کردن منطقه مورد نظر برای تشخیص خطوط افقی ۴۴۵
شکل ۷-۷ ساختمان یک نمونه انکودر ۴۴۷
شکل ۷-۸ ساختمان و نحوه کار انکودر دوار افزایشی ۴۴۸
شکل ۷-۹ ساختمان و نحوه کار انکودر دوار افزایشی ۴۴۸
شکل ۷-۱۰ ساختمان و نحوه کار انکودر دوار افزایشی و سیگنال های خروجی آن ۴۴۸
شکل ۷-۱۱ ساختمان و نحوه کار انکودر دوار مطلق ۴۵۰
شکل ۷-۱۲ مقایسه صفحه دوار انکودر دوار افزایشی و مطلق ۴۵۱
شکل ۷-۱۳ مشاهده‌ی مانع توسط خودرو ۴۵۳
شکل ۷-۱۴ دور زدن مانع توسط خودرو ۴۵۴
شکل ۷-۱۵ بازگشت خودرو به لاین خود ۴۵۴
شکل ۷-۱۶ زاویه‌ی چرخش منفی برای بازگشت به خط مربوطه ۴۵۵
شکل ۷-۱۷ سنسور التراسونیک ۴۵۶
شکل ۷-۱۸ فرستادن امواج صوتی و دریافت امواج اکو توسط سنسور التراسونیک ۴۵۷
شکل ۷-۱۹ نحوه عملکرد سنسور التراسونیک ۴۵۸
شکل ۷-۲۰ اجزا و پین‌های سنسور آلتراسونیک ۴۵۹
شکل ۷-۲۱ پین های سنسور التراسونیک و توضیحات هر یک ۴۵۹
شکل ۷-۲۲ ناحیه عملکرد و طرز عملکرد سنسور التراسونیک ۴۶۰

شکل ۷-۲۳ زاویه های متفاوت قرارگیری سنسور التراسونیک در برابر جسم و تفاوت عملکرد ۴۶۰
شکل ۷-۲۴ محیط شبیه سازی و بخش سنسورهای التراسونیک ۴۶۲
شکل ۷-۲۵ عملکرد سنسور التراسونیک وسط در محیط شبیه سازی ۴۶۲
شکل ۷-۲۶ عملکرد سنسور التراسونیک وسط و چپ در محیط شبیه سازی ۴۶۳
شکل ۷-۲۷ عملکرد سنسور التراسونیک وسط و راست در محیط شبیه سازی ۴۶۳
شکل ۷-۲۷ نمایی از جاده ۴۶۵
شکل ۷-۲۸ دیده شدن تابلوی گردش به راست و چپ توسط خودرو در محیط شبیه ساز ۴۶۷
شکل ۷-۲۹ تابلوی گردش به راست ۴۶۸
شکل ۷-۳۰ تابلوی گردش به چپ ۴۶۸
شکل ۷-۳۱ تابلوی ادامه به صورت مستقیم ۴۶۸
شکل ۷-۳۲ تابلوی توقف ۴۶۹
شکل ۷-۳۳ تشخیص محیط اصلی تابلو و ماسک به دست آمده ۴۷۰
شکل ۷-۳۴ ساختار شبکه‌ی عصبی استفاده شده برای یافتن کلاس هر تابلو ۴۷۲
شکل ۷-۳۵ خروجی اسکریپت hsvshow.py ۴۷۳
شکل ۷-۳۶ دکمه‌های کشویی موجود در برنامه color_detection.py ۴۷۵
شکل ۷-۳۷ محیط شهری شبیه ساز Avis شامل موانع، چهارراه، پیچ و انواع تابلو های گردش به چپ، گردش به راست و مستقیم ۴۷۶
شکل ۷-۳۸ سناریو حرکت در حالت شهری ۴۷۹
شکل ۷-۳۹ بینایی خودرو را در محیط شهری ۴۸۰
شکل ۷-۴۰ عملکرد کنترل‌کننده در حالت شهری ۴۸۳
شکل ۸-۱ محیط بزرگراه شبیه ساز، شامل موانع در جاده، پیچ های تند و اهمیت حرکت در لاین راست ۴۸۴
شکل ۸-۲ تصویر خام دریافتی از دوربین با فرمت رنگی RGB در سمت راست و تصویر تبدیل شده به فرمت HSV در سمت چپ ۴۸۵
شکل ۸-۳ تصویر محیط با فرمت رنگی HSV در سمت راست و ماسک باینری جاده در سمت ۴۸۶
شکل ۸-۴ تصویر (الف) ماسک خام جاده را نشان می دهد. همان طور که مشاهده می شود نویز کوچکی در حاشیه جاده قرار دارد. (ب) خروجی ماسک را پس از عمل Erosion نشان می‌دهد. (ج) خروجی نهایی ماسک را پس از Dilation نشان می دهد. ۴۸۷
شکل ۸-۵ ماسک لاین اصلی با بزرگترین مساحت در سمت راست و ماسک لاین دیگر در سمت چپ ۴۸۸
شکل ۸-۶ خط زرد وسط خیابان در سمت چپ ۴۸۹
شکل ۸-۷ خط زرد وسط خیابان در سمت راست ۴۸۹
شکل ۸-۸ جاده بزرگراه به همراه مانع در مقابل آن ۴۹۱
شکل ۸-۹ جداسازی رنگ سبز دلخواه با استفاده از فضای رنگی HSV ۴۹۲
شکل ۸-۱۰ نمایی از جاده در فضای HSV ۴۹۳
شکل ۸-۱۱ ماسک خطوط جاده ۴۹۴
شکل ۸-۱۲ عملیات مورفولوژی سایش Erosion با کرنل ۳×۳ ۴۹۵
شکل ۸-۱۳ ماسک مانع بعد از اعمال عملگر سایش ۴۹۶

شکل ۸-۱۴ ماسک مانع بعد از اعمال عملگر سایش ۴۹۷
شکل ۸-۱۵ ماسک مانع بعد از اعمال عملگر گسترش ۴۹۸
شکل ۸-۱۶ ماسک مانع و منطقه مورد نظر ۴۹۹
شکل ۸-۱۷ نمونه‌ای از وضعیت خودرو ۵۰۳
شکل ۸-۱۸ ماسک های مرتبط با حرکت ۵۰۴
شکل ۸-۱۹ سناریو های حرکت بین‌شهری ۵۰۵
شکل ۸-۲۰ مقدار حداکثر چرخش چرخ‌ها (۳۰+ درجه و ورودی ۱۰۰ به عملگر) ۵۰۷
شکل ۸-۲۱ پنجره‌ی کشویی برای تعیین ضرایب کنترل‌کننده ۵۰۹
شکل ۸-۲۲ نمونه‌ی اول از کنترل موقعیت خودرو ۵۱۰
شکل ۸-۲۳ نمونه‌ی دوم از کنترل موقعیت خودرو ۵۱۱
شکل ۸-۲۴ نمونه‌ی سوم از کنترل موقعیت خودرو ۵۱۱
شکل ۹-۱ شماتیک سخت افزاری خودرو ۵۱۲
شکل ۹-۲ دیاگرام عملکردی خودرو در محیط پارک ۵۱۳
شکل ۹-۳ دیاگرام عملکردی خودرو در محیط شهری و بین شهری ۵۱۴
شکل ۱۰-۱ یک نوع سروو موتور DC ۵۱۵
شکل ۱۰-۲ یک نوع سروو موتور AC ۵۱۵
شکل ۱۰-۳ نمودار پارامترهای تاثیرگذار در انتخاب الکتروگیربکس ۵۲۰
شکل ۱۰-۴ الکتروگیربکس کاهنده ۵۲۱
شکل ۱۰-۵ شماتیک مدار پل اچ ۵۲۲
شکل ۱۰-۶ تراشه L293D ۵۲۴
شکل ۱۰-۷ اتصال شیلد درایور موتور به آردوینو ۵۲۴
شکل ۱۰-۸ برد آردوینو اونو ۵۲۶
شکل ۱۰-۹ چراغ LED سرخود بورد آردوینو اونو ۵۲۷
شکل ۱۰-۱۰ نمایی از محیط Arduino IDE ۵۲۸
شکل ۱۰-۱۱ برد توسعه جتسون نانو ۵۳۰
شکل ۱۰-۱۲ برد توسعه جتسون نانو ۵۳۱
شکل ۱۰-۱۳ برد توسعه جتسون نانو ۵۳۳
شکل ۱۰-۱۴ سیستم عامل برد جتسون نانو ۵۳۴
شکل ۱۰-۱۵ برد جتسون نانو در کنار دوربین مورد استفاده ۵۳۵
شکل ۱۰-۱۶ دوربین ۵۳۶
شکل ۱۰-۱۷ ابعاد دوربین در مقایسه با برد ۵۳۶
شکل ۱۰-۱۸ تنظیمات مناسب برای انتقال تصاویر ۵۳۷
شکل ۱۰-۱۹ نمونه ای از موانع سه بعدی ساخته شده ۵۳۸
شکل ۱۰-۲۰ نقشه ی ساخته شده برای محیط بین شهری ۵۳۹
شکل ۱۰-۲۱ نمونه ای از شیب های تعبیه شده درون نقشه ی بین شهری ۵۴۰
شکل ۱۰-۲۲ نمونه ای از شیب های تعبیه شده درون نقشه ی بین شهری ۵۴۰

شکل ۱۰-۲۳ نمونه ای موانع سه بعدی قرار داده شده در طول مسیر ۵۴۱
شکل ۱۰-۲۴ نمونه ای موانع سه بعدی قرار داده شده در طول مسیر ۵۴۱
شکل ۱۰-۲۵ نقشه ی ساخته شده برای محیط پارک خودکار ۵۴۲
شکل ۱۰-۲۶ نقشه ی استراتژیک محیط پارک خودکار ۵۴۳
شکل ۱۰-۲۷ مقایسه ی مسیر های موجود برای مسیریابی ۵۴۴
شکل ۱۰-۲۸ ورودی نقشه پارکینگ ۵۴۴
شکل ۱۰-۲۹ نقشه ی ساخته شده برای محیط شهری ۵۴۵
شکل ۱۰-۳۰ نقشه ی استراتژیک محیط شهری ۵۴۶
شکل ۱۰-۳۱ نمونه ای از تصاویر طراحی و چاپ شده برای تابلوهای راهنمایی و رانندگی ۵۴۶
شکل ۱۰-۳۲ تعدادی از تابلوهای ساخته شده ۵۴۷
شکل ۱۰-۳۳ نمونه ای از تابلوهای راهنمایی و رانندگی در کنار خطوط جاده ۵۴۸
شکل ۱۰-۳۴ نمونه ای از تابلوهای راهنمایی و رانندگی در کنار خطوط جاده ۵۴۸
شکل ۱۰-۳۵ نمونه ای موانع سه بعدی قرار داده شده در طول مسیر ۵۴۹
شکل ۱۰-۳۶ نمونه ای از موانع سه بعدی و تابلوهای قرار داده شده در طول مسیر ۵۴۹
شکل ۱۰-۳۸ محل قرارگیری سنسورهای آلتراسونیک ۵۵۱
شکل ۱۰-۳۹ اتصال سنسورهای آلتراسونیک به برد برد ۵۵۱
شکل ۱۰-۴۰ قرار گیری برد جتسون نانو به همراه دوربین ۵۵۲
شکل ۱۰-۴۱ خروجی پین‌های برد جتسون نانو و آردوینو Uno ۵۵۳
شکل ۱۰-۴۲ اتصال سریال (RX/TX) بردهای جتسون نانو و آردوینو Uno ۵۵۳
شکل ۱۰-۴۳ تغذیه اجزای مختلف خودرو ۵۵۴
شکل ۱۰-۴۴ نتیجه نهایی ساخت خودرو ۵۵۵
شکل ۱۰-۴۵ محل و مقیاس قرارگیری خودروی ساخته شده بر روی نقشه ۵۵۶
شکل ۱-۱۱ مدل سه بعدی ستاپ پارک خودکار در محیط واقعی ۵۵۷
شکل ۲-۱۱ تغییر رزولوشن تصویر دریافتی از دوربین به ۱۰۰ پیکسل در ۱۰۰ پیکسل ۵۵۸
شکل ۳-۱۱ ماسک رنگی محل نهایی پارک در سمت راست و ماسک رنگی موانع در سمت چپ ۵۵۸
شکل ۴-۱۱ خروجی پردازش تصویر نقشه پارکینگ، تصویر چپ تصویر نقشه ثبت شده توسط دوربین، تصویر وسط ماسک باینری محل پارک و تصویر راست ماسک باینری ۵۵۹
شکل ۵-۱۱ خروجی نقشه پارکینگ واقعی در شبیه ساز پایتون ۵۶۰
شکل ۶-۱۱ برنامه زیری مسیر و درون یابی آن در شبیه ساز ۵۶۰
شکل ۷-۱۱ خروجی زیرسیستم برنامه‌ریزی مسیر در مقیاس اصلی تصویر نقشه ۵۶۱
شکل ۸-۱۱ نمونه اول پردازش تصویر و مسیریابی در نقشه واقعی ۵۶۳
شکل ۹-۱۱ نمونه دوم پردازش تصویر و مسیریابی در نقشه واقعی ۵۶۴
شکل ۱۰-۱۱ نمونه سوم پردازش تصویر و مسیریابی در نقشه واقعی ۵۶۵
شکل ۱۱-۱۱ نمایی از مسیر طراحی‌شده ۵۶۶
شکل ۱۲-۱۱ ماسک جاده برای تشخیص خطوط موجود در شکل ۱۱-۱۰ ۵۶۷
شکل ۱۳-۱۱ انتقال کانتورهای یافت‌شده از خطوط مسیر به تصویر اصلی ۵۶۷

شکل ۱۴-۱۱ شناسایی تابلوی ورود ممنوع توسط الگوریتم یولو ۵۶۹
شکل ۱۵-۱۱ شناسایی تابلوی حرکت رو به جلو توسط الگوریتم یولو ۵۶۹
شکل ۱۶-۱۱ وجود یک مانع در ادامه‌ی مسیر خودرو ۵۷۰
شکل ۱۷-۱۱ تشخیص ماسک مانع ۵۷۱
شکل ۱۸-۱۱ ماسک باینری و تصویر ماسک‌شده‌ی شامل مانع ۵۷۱
شکل ۱۹-۱۱ مانع در مسیر بین شهری ۵۷۲
شکل ۲۰-۱۱ ماسک و شکل ماسک شده ی مانع در محیط بین‌شهری ۵۷۲
شکل ۲۱-۱۱ مقایسه ی عملکرد کنترلر P در مقایسه با رفرنس کنترلی ۵۷۷
شکل ۲۲-۱۱ مقایسه ی عملکرد کنترلر PD در مقایسه با رفرنس کنترلی ۵۷۸
شکل ۲۳-۱۱ مقایسه ی عملکرد PID در مقایسه با رفرنس کنترلی ۵۷۹
شکل ۲۴-۱۱ مقایسه ی عملکرد همه ی کنترلر های طراحی شده در مقایسه با رفرنس کنترلی ۵۸۰
شکل ۲۵-۱۱ مراحل حرکت خودرو در نقشه پارک و انجام پارک موازی ۵۸۱
شکل ۲۶-۱۱ نتایج پردازش محیط از دید خودرو (on board) در محیط شهری ۵۸۲
شکل ۲۷-۱۱ نتایج عملکرد خودرو در نقشه (off board) در محیط شهری ۵۸۳
شکل ۲۸-۱۱ نتایج پردازش محیط از دید خودرو (on board) در محیط بین شهری ۵۸۴
شکل ۲۹-۱۱ نتایج عملکرد خودرو در نقشه (off board) در محیط بین شهری ۵۸۵
شکل ۳۰-۱۱ نتایج عملکرد خودرو در نقشه (off board) در محیط بین شهری ۵۸۶

فهرست جداول

جدول ۱-۱ چالش‌ها و راه حل‌های ممکن سامانه راننده خودکار ۲۵
جدول ۱-۲ مزایا و معایب سنسورهای اولتراسونیک ۳۱
جدول ۱-۳ مدل های معروف یادگیری عمیق مورد استفاده در فناوری اتومبیل های خودمختار ۳۵
جدول ۱-۴ تعداد اشتغال و ظرفیت واحدهای فعال تولید خودرو سواری به تفکیک استان تا پایان مهر ۱۳۹۶ ..... ۵۸
جدول ۱-۵ روند تولید و ظرفیت خودرو سواری در کشور(هزار دستگاه) ۵۸
جدول ۱-۶ روند تولید جهانی خودرو طی سال های ۲۰۱۲ تا ۲۰۱۶ (میلیون دستگاه) ۵۹
جدول ۱-۷ روند ارزش صادرات خودرو سواری - (میلیارد دلار) ۶۱
جدول ۱-۸ روند ارزش واردات خودرو سواری - (میلیارد دلار) ۶۱
جدول ۱-۹ ارزش واردات خودرو سواری طی سال های ۱۳۸۹ تا ۱۳۹۶ (میلیون دلار) ۶۱
جدول ۱-۱۰ ارزش صادرات خودرو سواری طی سال های ۱۳۸۹ تا ۱۳۹۶(میلیون دلار) ۶۲
جدول ۱-۱۱ ارزش صادرات خودروهای سواری طی سال های ۱۳۸۹ تا ۱۳۹۶ به تفکیک کشور (میلیون اختیار) ۶۲
جدول ۱-۱۲ واردات خودرو سواری طی سال های ۱۳۸۹ الی ۱۳۹۶ به تفکیک کشور (میلیون دلار) ۶۳
جدول ۱-۱۳ خلاصه ای از تأثیرات اقتصادی (در کل صنعت و اقتصاد) ۶۷
جدول ۱-۱۴ معرفی محصولات شرکت تسلا ۷۰
جدول ۱-۱۵ مشخصات خودروی خودران ویمو ۷۱
جدول ۱-۱۶ مشخصات خودرو لوسید ۷۲
جدول ۱-۱۷ شرکت‌های دیگر حاضر در صنعت خودروهای خودران ۷۲
جدول ۱-۱۸ مشخصات سامانه راننده خودکار خلیج فارس ۷۳
جدول ۱-۲ نتایج دقت بر روی مجموعه داده GTSDB (بر حسب درصد) که توسط هر مدل آشکارساز علائم راهنمایی و رانندگی بدست آمده ۱۱۲
جدول ۲-۲ مزایا و معایب روش‌های برنامه‌ریزی مسیر ۱۳۸
جدول۲-۳ مقایسه ای از مدل های مختلف YOLOV5 و YOLOV3 بر اساس سرعت، FLOPS، دقت و تعداد پارمترها ۱۵۴
جدول ۲-۴ مقایسه EfficientDet با دیگر آشکارسازها از لحاظ mAP، تعداد پارمترها، تاخیر و سرعت ۱۵۶
جدول ۲-۵ مقایسه دو روش تشخیص اشیا و تقسیم بندی موردی ۱۵۸
جدول ۲-۶ مقایسه روش های ردیابی اجسام متحرک ۱۶۴
جدول ۱-۳ مقایسه ای از مدل های مختلف YOLOV5 بر اساس سرعت، FLOPS، دقت و تعداد پارمترها ۱۷۶
جدول ۲-۳ مشخصات دیتاست COCO ۱۷۹
جدول ۳-۳ تابلوهای مورد بررسی در پروژه ۱۸۸
جدول ۴-۳ دیتاست Swedish ۱۹۲
جدول ۵-۳ دیتاست GTSDB ۱۹۳
جدول ۶-۳ دیتاست MakeML Cars and Traffic Signs Dataset ۱۹۶
جدول ۷-۳ دیتاست MakeML Road Signs Dataset ۱۹۷

جدول ۸-۳ دیتاست Vicos/DFG Dataset ۱۹۹
جدول ۹-۳ دیتاست Tsinghua ۲۰۳
جدول ۱۰-۳ نتایج صحه‌گذاری شبکه Yolov5s ۲۰۷
جدول ۱۱-۳ جزئیات شبکه Resizing ۲۲۶
جدول ۱۲-۳ جزئیات شبکه پیش بین ۲۲۹
جدول ۱۳-۳ نتایج شبکه PINet بر روی دیتاست TuSimple ۲۴۲
جدول ۱۴-۳ نتایج شبکه PINet بر روی دیتاست CULane ۲۴۳
جدول ۱۵-۳ مشخصات دوربین Kitti و Cityscapes ۲۷۹
جدول ۱۶-۳ زمان مورد نیاز پردازش هر فریم توسط هر بخش از نرم افزار ۳۱۰
جدول ۱-۶ تابلوهای راهنمایی و رانندگی موجود در شبیه‌ساز ۴۳۹
جدول ۱-۸ ضرایب کنترل‌کننده‌ی PID کنترل موقعیت ۵۱۰
جدول ۱-۱۰ حالت های مختلف مدار ۵۲۳
جدول ۲-۱۰ مشخصات برد آردوینو اونو ۵۲۶
جدول ۳-۱۰ مشخصات برد جتسون نانو ۵۳۲
جدول ۱-۱۱ مقادیر پارامترهای HSV برای تشخیص خطوط ۵۶۸
جدول ۲-۱۱ مقادیر پارامترهای HSV برای تشخیص مانع ۵۷۰
جدول ۳-۱۱ روش زیگلر نیکولز برای طراحی کنترل کننده ۵۷۵
جدول ۴-۱۱ آزمایشات صحیح و خطا برای یافتن مقادیر کنترلی مطلوب ۵۷۵
جدول ۵-۱۱ ضرایب نهایی طراحی شده برای خودروی خودران ۵۷۶

مقدمه

مردم سراسر جهان، روزانه بخش بزرگی از زمان خود را صرف جابه‌جایی از یک مکان به مکان دیگری می‌کنند. آیا نباید راحت‌تر و ایمن‌تر از هر جا که بخواهیم به جایی دیگر برویم؟ در ده سال گذشته گذشته یا قبل‌تر، محققان، دانشمندان و شرکت‌های فن‌آوری تلاش زیادی برای توسعه تکنولوژی خودروهای کاملا خودران انجام داده‌اند. همچنین این تکنولوژی در خیابان‌های واقعی شهرهای واقعی آزمایش شده است.

اتومبیل‌های خودران دیگر تنها در فیلم‌های علمی‌تخیلی نیستند. شرکت‌هایی مانند تویوتا و فورد، میلیاردها دلار صرف تحقیق و توسعه این تکنولوژی کرده‌اند. سرویس‌هایی مانند uber وLyft ، که در حال حاضر به رانندگان در قبال رانندگی دستمزد پرداخت می‌کنند، به زودی کل ناوگان خود را با ماشین‌های خودران تجهیز خواهند کرد. در چند سال، خواهیم دید که خودروهای خودران به مشتریان عادی فروخته خواهند شد. اما هنوز هم ترس و وحشتی پیرامون آن وجود دارد. شاید دلیل این مسئله این باشد که اکثر مردم نمی‌دانند که این ماشین‌ها چگونه کار می‌کنند.

هنگام نشستن انسان‌ها روی صندلی راننده، هم‌زمان با مشاهده محیط اطراف اطلاعاتی را از محیط پیرامون به عنوان ورودی دریافت و آن‌ها را به منظور تصمیم‌گیری در مورد جهت چرخش فرمان و زمان ترمز گرفتن، پردازش می‌کنیم. یک ماشین خودران معمولا با دستگاهGPS ، یک سیستم ناوبری اینرسی و طیف وسیعی از حسگرها تجهیز شده است. این ماشین‌ها اطلاعات مربوط به موقعیت جغرافیایی را از GPS دریافت می‌کنند و برای جانمایی خود از سیستم ناوبری و از اطلاعات دریافتی از حسگرها برای ساختن نقشهای داخلی از محیط اطراف، استفاده می‌کنند. به محض دریافت اطلاعات مربوط به موقعیت خودش بر روی نقشه داخلی، می‌تواند از آن نقشه برای یافتن مسیر بهینه، ضمن دوری از هرگونه مانعی، برای رسیدن به مقصدش استفاده کند. آنچه ما در اینجا توضیح دادیم، تنها یک توصیف بسیار سطح بالا در مورد نحوه کار کردن خودروهای خودران است.

دلایل متعددی برای تولید خودروهای خودران وجود دارد که در ادامه آورده شده است.

✓ دلایل اقتصادی

بهره‌برداری: به راستی که از ماشین‌ها به طور کامل استفاده نمی‌شود. در بیشتر مواقع، تنها از ٪۴ زمان در دسترس بودن ماشین استفاده می‌شود، در حالی که بقیه ٪۹۶، بیشتر آن‌ها در پارکینگ پارک می‌شوند. ماشین یکی از بزرگترین سرمایه‌گذاری‌های مردم است اما با این وجود به طور کامل از آن استفاده نمی‌شود. بنابراین، یک وسیله بسیار گران قیمت در دسترس داریم که در اغلب موارد بلا استفاده است.

هزینه: اگر به خدمات ارائه دهنده تقاضاهای جابه‌جایی مانندUber ، Lyftو غیره نگاه کنیم و هزینه به ازای هر کیلومتر را تحلیل کنیم، راننده ٪۵۰ هزینه را شامل می‌شود. اگر راننده را از این چرخه خارج کنیم و ماشین نیز با سوخت الکتریکی حرکت کند، هزینه به ازای هر کیلومتر به طور قابل توجهی کاهش می‌یابد.

✓ ضمیمه کردن خدمات

موضوع دیگری نیز هست که شرکت‌های فن‌آوری را به آن سوق می‌دهد. از آنجایی که اکثر اتومبیل‌ها سرنشینانی دارند که رانندگی نمی‌کنند، شما می‌توانید در مورد ارسال و یادآوری اطلاعات به آن‌ها فکر کنید. شما در ماشین حسگرهایی دارید، که می‌تواند موضوع صبحبت‌های سرنشینان خودرو را دنبال کند و همچنین با شما صحبت کند. ماشین از طریق این حسگرها می‌تواند پیشنهاداتی را برای شما داشته باشد، مانند جایی از مسیر که یک قهوه یا غذای خوب سرو می‌کنند یا می‌تواند به شما آیتم‌های شامل تخفیف در فروشگاه‌های موجود در مسیر و غیره را نشان دهد. شرکت‌های فن‌آوری می‌توانند ضمن جمع‌آوری اطلاعات در مورد مسافران، سرویس‌های مشابه چندگانه و یک مقدار اقتصادی برگرفته از آن را ارائه دهند.

✓ هوش مصنوعی

بسیاری از تکنولوژیست‌ها در جهان، در مورد تأثیر هوش مصنوعی (Artificial Intelligence) بر اقتصاد اتفاق نظر دارند آن‌ها معتقدند که هوش مصنوعی، در آینده‌ای نزدیک، حدود ۵ تا ۱۰ سال آینده، تاثیر عمده‌ای بر اقتصاد خواهد گذاشت.

اولین مدل کسب و کار هوش مصنوعی در رباتیک (Robotics) است ، اما بازار آن به اندازه‌ای بزرگ نیست که بتواند چنین سرمایه‌گذاری بزرگی را توجیه کند. ربات چت‌ها (Chatbots) ممکن است بازار دیگری باشند، اما این بازار نیز به اندازه کافی نمی‌تواند چنین سرمایه‌گذاری بزرگی (که شاید برای توسعه آن به میلیاردها دلار نیاز باشد) را توجیه کند. ما هنگام فکر کردن به اتومبیل، قضیه متفاوت است بازار خودرو، بازار بسیار بزرگی است و تقریبا هر فردی به آن احتیاج خواهد داشت. این یک پلت‌فرم ایده‌آل برای هوش مصنوعی است، زیرا اگر شما بخواهید یک ماشین خودران داشته باشید، به سنسورهایی نیاز خواهید داشت که جهان را به صورتی که واقعا هست بشناسد و خود را با ترافیک آن وفق دهد. به طور خلاصه، ما ماشینی نیاز داریم که بتواند رقیبی برای هوش انسان باشد. بنابراین، ما در اینجا، به طور جدی در پی کشف موارد استفاده از هوش مصنوعی همراه با مدل کسب و کار هستیم. از آنجا که ما اکنون یک مدل تجاری داریم که ممکن است سالانه میلیاردها دلار را از سراسر جهان جذب کند، این امر توسعه هوش مصنوعی را به طور کامل معنی می‌کند.

همین دلیل، ماشین خودران را به یک صنعت بسیار بزرگ تبدیل کرده است. این امر باعث جذابیت بسیار زیاد تقاضای جابه‌جایی شده است. هر سازمانی که در زمینه توسعه یا راه‌اندازی نرم‌افزار کار می‌کند، استارت‌آپ است، شرکت‌هایی نظیر اوبر(Uber) ، گوگل(Google) ، تسلا(Tesla) ، اپل (Apple) یا هر شرکتی در صنعت خودروسازی، که در بخش خودروهای خودران شروع به کار کرده است.

یک سیستم نرم‌افزاری خودکار خودرو را می‌توان به طور گسترده به سه دسته تقسیم کرد یعنی

۱. ادراک (Perception)

ادراک به طور کلی ادراک به توانایی یک سیستم خودکار در جمع آوری اطلاعات و استخراج دانش مربوطه از محیط اشاره دارد که این خود با استفاده از تجهیزات مختلفی از جمله رادار، لیدار، جی‌پی‌اس و بینایی ماشین و با استفاده از تکنیک‌های همچون پردازش تصویر، شبکه‌های عصبی مصنوعی، الگوریتم‌های فرا ابتکاری و داده‌کاوی با جهان بیرون انجام می شود.

۲. برنامه ریزی یا تصمیم گیری (Planning)

برنامه‌ریزی به فرآیند تصمیم‌گیری هدفمند برای رسیدن به اهداف مرتبه بالاتر خودرو که به طور معمول شامل آوردن وسیله نقلیه از محل شروع به مکان هدف و اجتناب از موانع و بهینه‌سازی طراحی شده اشاره دارد. برنامه ریزی خود شامل سه مجموعه زیر است.

- برنامه‌ریزی ماموریت[۶]

برنامه‌ریزی ماموریت به طور کلی از طریق جستجوی گراف روی یک گراف جهت‌دار صورت می‌گیرد که نشان‌دهنده اتصال شبکه یا مسیر است که اطلاعاتی شامل محل علامت توقف، عرض جاده و محل پارک را دارد.

[6] Mission planning

- برنامه ریزی رفتاری[7]

برنامه ریز رفتاری مسئولیت تصمیم گیری را بر عهده دارد تا اطمینان حاصل شود که وسیله نقلیه از قوانین مقرر شده در جاده پیروی می کند و ضمن پیشرفت فزاینده در مسیر تعیین شده برنامه ریز مأموریت، با روشی متعارف و ایمن با سایر عوامل تعامل می کند.

- برنامه ریزی حرکت[8]

برنامه ریزی حرکت به فرآیند تصمیم گیری در مورد توالی اقدامات برای رسیدن به یک هدف مشخص اشاره دارد، به طور معمول در حالی که از برخورد با موانع جلوگیری می کند.

۳. کنترل (Control)

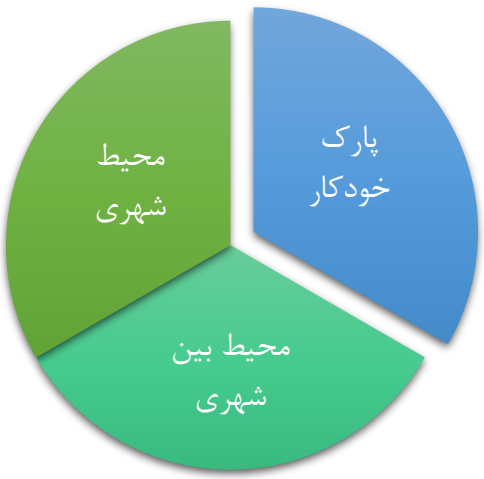
صلاحیت اجرای یک سیستم خودکار، که اغلب تحت عنوان کنترل حرکت نیز شناخته می شود، فرآیند تبدیل مقاصد به اعمال است. هدف اصلی آن اجرای اهداف برنامه ریزی شده است. با ارائه ورودی های لازم به سطح سخت افزار که حرکات مورد نظر را ایجاد می کند. کنترل کننده ها از نظر نیروها و انرژی، تعامل در دنیای واقعی را ترسیم می کنند، در حالی که الگوریتم های ناوبری و برنامه ریزی شناختی در یک سیستم خودکار معمولاً مربوط به سرعت و موقعیت خودرو نسبت به محیط آن هستند. از اندازه گیریهای داخل سیستم کنترل می توان برای تعیین میزان رفتار سیستم استفاده کرد و بنابراین کنترل کننده می تواند واکنش نشان دهد تا اختلالات را رد کند و دینامیک سیستم را به حالت مطلوب تغییر دهد. از مدلهای سیستم می توان برای توصیف حرکت مورد نظر با جزئیات بیشتر استفاده کرد که برای اجرای رضایت بخش حرکت ضروری است.

[7] Behavioral Planning
[8] Motion Planning

هدف از این پروژه طراحی، شبیه سازی و ساخت یک خودروی خودران در مقیاس کوچک است که قادر به درک محیط ساخته شده، برنامه ریزی حرکت و در نهایت کنترل خود تا رسیدن به هدف تعیین شده می‌باشد. در دو فصل ابتدایی این رساله به بررسی مفاهیم پایه ای خودروی خودران، تاریخچه و بازار داخلی و خارجی پرداخته شده است. انواع الگوریتم‌ها و روش های مختلف این حوزه، بررسی گردیده و مزایا و معایب هرکدام نیز نام برده شده است. تمامی اطلاعات به طور مستند از مقالات و کتب مختلف جمع آوری و ترجمه شده و درک کاملی از ابعاد این صنعت ارائه می‌دهد.

در فصل های سوم و چهارم با نگاهی تئوری محور به بررسی دو عملکرد مهم یک خودروی خودران، درک محیط و پارک خودکار، پرداخته شده است. در این فصل‌ها، روش های انتخابی به طور کامل از لحاظ مختلف سرعت، دقت، و ریاضیاتی بررسی می‌شوند.

فصل‌های پنجم تا هشتم مربوط به شبیه سازی این دو عملکرد در بسترهای گرافیکی مختلف می باشد. الگوریتم پارک خودکار که شامل زیر بخش‌های مدل‌سازی، برنامه‌ریزی و کنترل می‌باشد در بستر یک شبیه‌ساز پیاده سازی شده و عملکرد آن بررسی می‌گردد. سپس الگوریتم های درک محیط و کنترل یک خودروی خودران، در محیط های شهری و بین‌شهری شبیه ساز Avis، آزمایش شده و عملکرد یک خودروی خودران را در یک محیط گرافیکی سه بعدی نشان می‌دهد.



شکل الف محیط های شبیه‌سازی و پیاده‌سازی درنظر گرفته شده برای توسعه و بررسی الگوریتم‌های خودروی خودران

درنهایت در فصل نهم، اجزای مکانیکی و الکتریکی لازم برای یک خودرو آورده شده است. خودروی ساخته شده به همراه نقشه محیط های پارک، شهری و بین‌شهری ارائه داده شده اند. فصل دهم که ثمره نهایی این رساله می باشد، شامل گردهمایی تمامی این شبیه سازی ها در بستر سخت افزار و ارائه نتایج آن در دنیای واقعی است. در این فصل نتایج خروجی الگوریتم‌های پردازش تصویر، درک محیط، برنامه ریزی مسیر، تصمیم گیری و کنترل ارائه شده است.

دیاگرام زیر مسیر کلی این پروژه را نشان می‌دهد.

شکل ب نگاه کلی به فعالیت های صورت گرفته در این پروژه

۱- خودروی خودران

۱-۱- تعریف خودروی خودران

خودروی خودران (که بعضا اتومبیل خودمختار یا اتومبیل بدون راننده نامیده می شود) وسیله ای است که با استفاده از ترکیبی از سنسورها، دوربین ها، رادار و هوش مصنوعی (AI) می تواند بدون اپراتور انسانی بین مبدا و مقصد حرکت کند. در تحقیقات کنونی در زمینه رانندگی خودران، متداول ترین روش، استفاده از چارچوبی اساسی برای پردازش اطلاعات سریال و محاسبات است که از چهار ماژول تشکیل شده است: ادراک ، برنامه ریزی ، تصمیم گیری و کنترل. در ادامه به تشریح هر یک از این موارد پرداخته شده است[۱].

در شکل ۱-۱ مراحل کلی از سیستم خودروی خودران آورده شده است.

شکل ۱-۱، نمای کلی از مراحل سیستم خودروی خودران

۱-۱-۱- ادراک[9]

فناوری های AI در اتومبیل های خودران نقش اساسی دارند. توسعه دهندگان اتومبیل های خودران از مقادیر زیادی از داده های سیستم های شناسایی تصویر ، همراه با یادگیری ماشین و شبکه های عصبی ، برای ساخت سیستم هایی استفاده می کنند که می توانند به طور مستقل رانندگی کنند.

شبکه های عصبی الگوهایی را در داده ها شناسایی می کنند که به الگوریتم های یادگیری ماشین منتقل می شوند. این داده ها شامل تصاویری از دوربین های موجود در اتومبیل های خودران است که از طریق آنها شبکه

[9] Perception

عصبی می تواند چراغ های راهنمایی ، درختان ، حاشیه ها ، عابران پیاده ، علائم خیابان و سایر قسمت های هر محیط رانندگی را شناسایی کند..

به عنوان مثال ، پروژه اتومبیل رانندگی گوگل ، به نام ویمو[10] ، از ترکیبی از سنسورها ، لیدار[11] (تشخیص و سنجش با نور - فناوری مشابه رادار) و دوربین ها استفاده می کند و تمام داده های تولید شده توسط این سیستم ها را برای شناسایی همه چیز در اطراف خودرو ترکیب می کند. و پیش بینی می کند که هر شیئ ممکن است بعدا چه کاری انجام دهند. این فرآیند در کسری از ثانیه اتفاق می افتد. بلوغ برای این سیستم ها مهم است. هرچه سیستم بیشتر رانندگی کند ، داده های بیشتری می تواند در الگوریتم های یادگیری عمیق خود بگنجاند ، و آن را قادر می سازد تا انتخاب های رانندگی را با ظرافت بیشتری انجام دهد.

۲-۱-۱- تصمیم گیری

یک خودروی خودران باید بتواند با توجه به ادراک محیط و برنامه ریزی حرکت که از قبل انجام شده است، در هر لحظه تصمیم درست را انتخاب کند که این مبحث در قسمت تصمیم گیری قرار می گیرد. سیستم پیشرفته تصمیم‌گیری با ارائه الگوریتمی بر اساس خروجی داده‌کاوی سیستم ادراک خودرو تعیین می‌کند که خودرو در کدام مانور، خط و ...، و با چه سرعتی حرکت کند. تصمیم‌گیری و الگوریتم تعیین مسیر مانور با استفاده از شبیه‌سازی بهینه کنترل سیگنال‌های طولی و عرضی خودرو امکان‌پذیر خواهد بود. سناریوی‌های مختلف تصمیم‌گیری دارای برخی قیود مختلف است. از جمله ی آن ها می توان به موارد زیر اشاره کرد.

- نگهداری خودرو روی خط مرزی بین خطوط
- نگهداشتن خودرو در سرعت مطلوب
- جلوگیری کردن از تصادف با خودروهای اطراف و سایر موانع
- حفظ خودرو در مرزهای جاده
- احترام به طراحی فیزیکی و محدودیت های خودرو

۳-۱-۱- برنامه ریزی حرکت۱۲

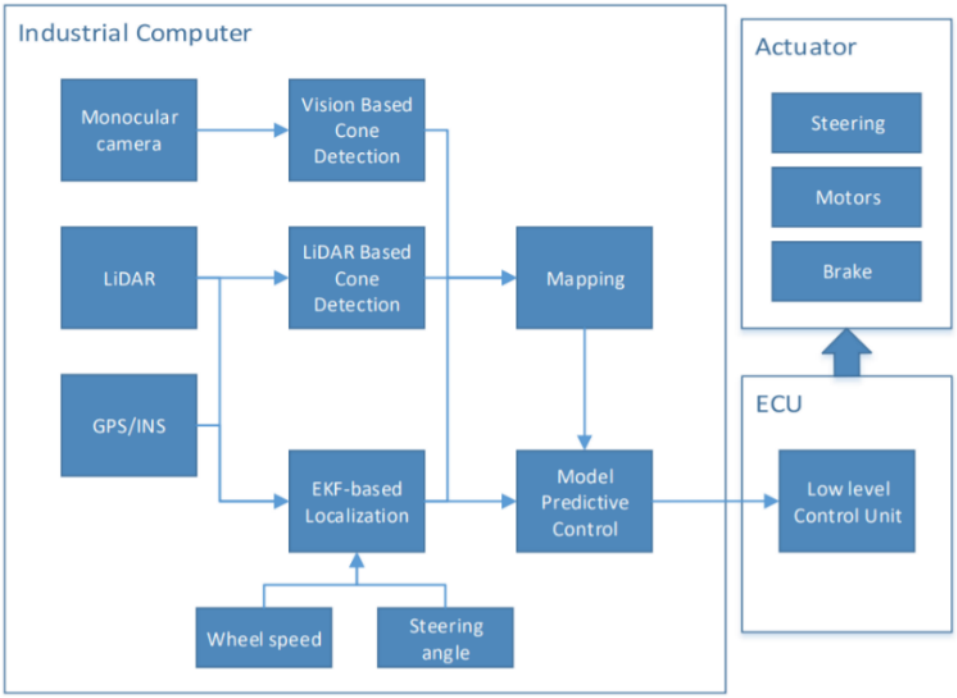
[10] WayMo
[11] LiDAR
[12] Motion Planning

برنامه‌ریزی حرکت اصطلاحی است که در رباتیک به فرایند جزئی‌سازی کاری، درقالب حرکت‌های جداگانه و گسسته اطلاق می‌شود. هدایت خودرو خودران را از یک مبدا به سمت مقصد را در نظر بگیرید. الگوریتم برنامه‌ریزی حرکت وظایف جداگانه را به عنوان ورودی می‌گیرد، فرمان سرعت و چرخش چرخ‌ها را تولید می‌کند و به خودرو ارسال می‌کند. برنامه‌ریزی حرکت دارای کاربردهای رباتیکی زیادی از قبیل خودمختاری، خودکارسازی، نرم‌افزار طراحی به وسیله کامپیوتر و ... می‌باشد. در واقع الگوریتم تعیین مسیر باید قادر به طراحی دوباره مسیری جدید باشد به طوری که اگر شرایط محیط عوض شد و قسمت تصمیم‌گیری تصمیم جدیدی اتخاذ کرد قادر به تولید مسیری جدید بر اساس پارامترهای خودرو و جاده باشد. در واقع الگوریتم مناسب برای طراحی مسیر باید به گونهای باشد که نسبت به شرایط ترافیک جاده، نوع مسیر را دوباره طراحی و تعیین نماید. در تعیین این مسیر از روشهای مختلف استفاده میشود و همچنین مسیر شامل توابع چند جملهای، مثلثاتی و .. می‌باشد. روش‌های استفاده شده در این بخش در اکثر موارد شامل الگوریتم‌های بهینه‌سازی، زنجیره مارکوف، شبکه‌های عصبی مصنوعی، نقشه ریسک و همچنین استفاده از روش‌های مبتنی بر تقابل راننده یا مدل راننده و روش‌های دیگر از این قبیل است که بتواند بر اساس تابعی ریاضی مبنا، مسیر مورد نظر را طراحی نماید. این مسیر به عنوان ورودی کنترلر برای بخش هدایت فرمان خودرو مورد استفاده قرار می‌گیرد. همچنین با استفاده از سیستم کروز کنترل پیشرفته که بر اساس میزان اصطکاک سطح و همچنین فواصل مختلف طولی و عرضی میزان سرعت را تعیین می‌نماید، سرعت خودرو متناسب با مسیر مانور تعیین می‌شود.

۴-۱-۱- کنترل

قسمت کنترل نیز شامل شناسایی فضای حالت[۱۳] ، مکان یابی و طراحی کنترل کننده می باشد که با توجه به نتیجه ای که از بخش های ادراک، برنامه ریزی حرکت و تصمیم گیری به دست آمده است، دستور کنترلی به عملگر ها ارسال می گردد. در این بخش مسیر و سرعت مطلوب وارد قانون کنترل می‌شود. خروجی قانون کنترلی ترمز، گاز و فرمان می‌باشد و در واقع این سیستم نقش عملی راننده را دارد. به عبارت دیگر بخش اول شامل چشم راننده است، بخش دوم و سوم شامل ذهن راننده است و بخش کنترلر تعقیب از مغز با اعمال ورودی‌های مورد نظر کنترلر است. به عبارت دیگر، چیزی که به چشم می‌آید همین بخش است و بخش‌های قبلی به عنوان سیستماتیک اعمال می‌شود و از دیدگاه افراد غیر متخصص پنهان است. کنترلر طراحی شده در این بخش باید قادر باشد اتومبیل را در مسیر طراحی شده، در شرایط مختلف جاده و سرعت‌های متفاوت و شرایط نامعین دیگر کنترل نماید.

[13] state estimation

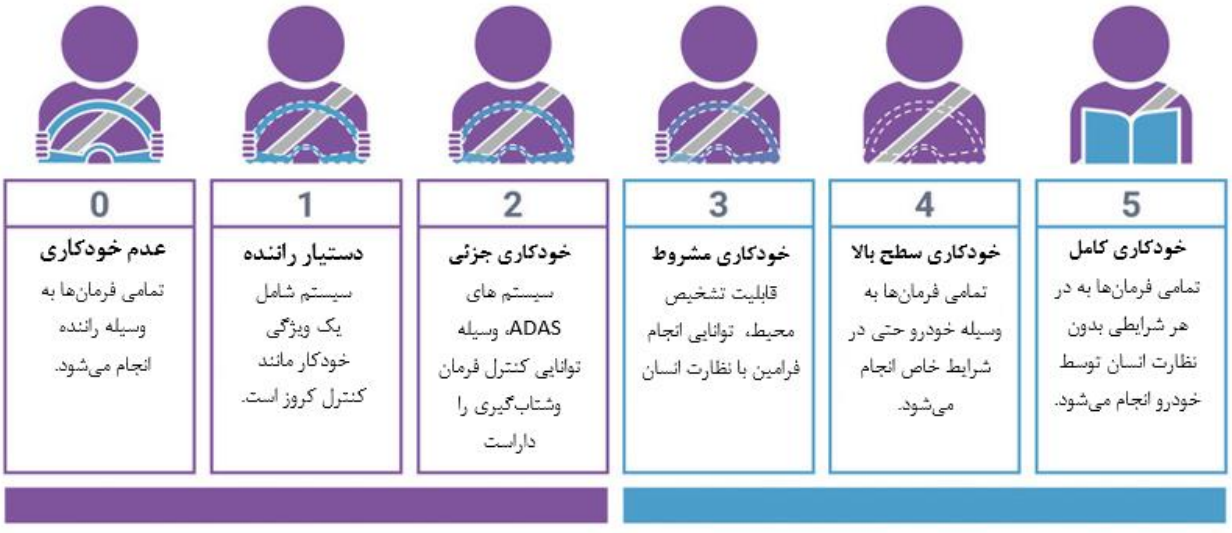


شکل ۲-۱، نمای کلی از سیستم کنترلی به همراه اجزا[۱].

در شکل ۲-۱ نمای کلی از سیستم کنترلی آورده شده است.

۲-۱- سطوح خودمختاری خودرو

سامانه‌های راننده خودکار شامل ۵ سطح از خودمختاری هستند. دسته‌بندی سامانه‌ها بر اساس معیارهای مختلف انجام می‌شود. در شکل ۳-۱ نمای کلی از سطوح مختلف خودمختاری سامانه‌ها آورده شده است.



شکل ۳-۱، نمای کلی از سطوح مختلف خودمختاری سامانه‌ها[۲].

در ادامه به توصیف این سطوح از خودمختاری سامانه‌ها پرداخته شده است[۲].

۱-۲-۱- سطح صفر: بدون اتوماسیون رانندگی

امروزه بیشتر وسایل نقلیه موجود در جاده سطح صفر هستند: به صورت دستی کنترل می شوند. انسان "وظیفه رانندگی پویا" را فراهم می کند ، اگرچه ممکن است سیستم هایی برای کمک به راننده وجود داشته باشد. به عنوان مثال می توان سیستم ترمز اضطراری را نام برد. از آنجا که از نظر فنی وسیله نقلیه را "رانندگی" نمی کند ، واجد شرایط اتوماسیون نیست.

۱-۲-۲- سطح یکم : کمک راننده

این پایین ترین سطح اتوماسیون است. این خودرو از یک سیستم خودکار برای کمک به راننده مانند فرمان یا شتاب (کنترل سرعت) بهره می برد. کروز کنترل تطبیقی ، جایی که وسیله نقلیه را می توان در فاصله ایمنی پشت ماشین بعدی نگه می دارد، به عنوان سطح ۱ شناخته می شود زیرا راننده انسان سایر جنبه‌های رانندگی مانند فرمان و ترمز را کنترل می کند.

۱-۲-۳- سطح دوم : اتوماسیون رانندگی جزئی

این به معنی سیستم های پیشرفته کمک راننده (ADAS) است. این وسیله نقلیه می تواند هم فرمان و هم شتاب را کنترل کند. در اینجا اتوماسیون رانندگی کوتاه است زیرا یک انسان در صندلی راننده می نشیند و می تواند ماشین را در هر زمان کنترل کند. سیستم های Super Cruise Tesla Autopilot و Cadillac (General Motors) هر دو واجد شرایط سطح ۲ هستند.

۱-۲-۴- سطح سوم : اتوماسیون رانندگی مشروط

پرش از سطح دوم به سطح سوم از منظر فناوری قابل توجه است. وسایل نقلیه سطح سوم از قابلیت "تشخیص محیط" برخوردار هستند و می توانند برای خود تصمیمات آگاهانه‌ای مانند عبور از یک خودروی کند بگیرند. درحالی که، آن‌ها هنوز هم به نظارت انسان نیاز دارند. اگر سیستم قادر به اجرای وظیفه نباشد، راننده باید هوشیار و آماده کنترل باشد.

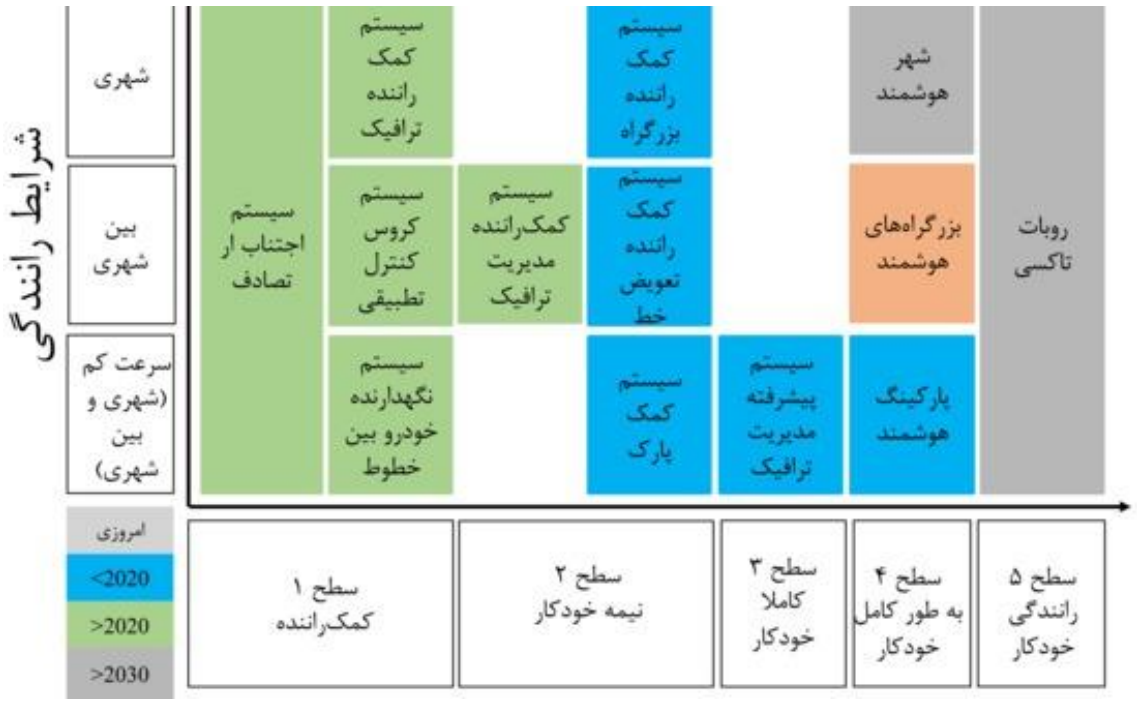
۱-۲-۵- سطح چهارم : اتوماسیون رانندگی بالا

تفاوت کلیدی بین اتوماسیون سطح سوم و سطح چهارم این است که در صورت وجود مشکلات یا خرابی سیستم، وسایل نقلیه سطح چهارم می توانند مداخله کنند. از این نظر ، این اتومبیل ها در بیشتر شرایط به تعامل انسانی احتیاج ندارند. با این حال، یک انسان هنوز این گزینه را دارد که به صورت دستی کنترل خودرو را به دست بگیرد.

۶-۲-۱- سطح پنجم : اتوماسیون کامل رانندگی

وسایل نقلیه سطح پنج نیازی به توجه انسان ندارند. اتومبیل های سطح پنجم حتی فرمان یا پدال شتاب و ترمز نخواهند داشت. آن ها قادر به رفتن به هر مکانی و انجام هر کاری هستند که یک راننده با تجربه انسانی می تواند انجام دهد. اتومبیل های کاملاً مستقل در حال آزمایش در چندین نقطه دنیا هستند ، اما هنوز هیچ کدام در دسترس عموم نیستند.

در شکل ۴-۱ نقشه ای از سطوح مختلف خودمختاری خودروهای خودران آورده شده است.



شکل ۴-۱، نقشه سطوح خودمختاری خودروهای خودران

۳-۱- صنایع بالادستی و پایین‌دستی

در این بخش به صنایع بالادستی و پایین‌دستی صنعت خودرو و سامانه‌های خودکار رانندگی پرداخته شده است.

۱-۳-۱- صنایع بالادستی و پایین‌دستی صنعت خودرو

به طور خلاصه، صنایع بالادستی و پایین‌دستی صنعت خودرو در شکل ۵-۱ (بالا) آورده شده است.

۲-۳-۱- صنایع بالادستی و پایین‌دستی سامانه راننده خودکار

با ورود سامانه‌های خودران به صنعت اتومبیل، ارتباط صنایع وابسته به خودروسازی به شکل پیچیده‌تری درآمده است. در شکل ۵-۱ (پایین)، نمای کلی از ارتباط صنایع مرتبط با سیستم‌های خودران آورده شده است.

شکل ۵-۱، صنایع مرتبط با صنعت خودرو[۳] (بالا) و صنایع مرتبط با سامانه‌های خودکار[۴] (پایین)

۱-۳-۲-۱- صنایع بالا دستی

صنایع بالادستی سامانه‌های راننده خودکار به شرح زیر است:

- **صنعت سنسورها:**

سنسورها در درک محیط به خودرو و در نتیجه به ناوبری صحیح آن کمک می‌کنند. سامانه‌های راننده خودکار به وسیله مجموعه‌ای از سنسورها شامل لیدار، رادار، سنسورهای اولتراسونیک و ... مجهز می‌شوند[۴]. با توجه به آنکه بسیاری از سنسورها از خارج از کشور تامین می‌شود، باید به واردات این محصولات توجه کرد. در حالی که با افزایش حجم فروش خودروهای خودران، می‌توان به مرور زمان از ظرفیت‌های داخلی برای تامیناین قطعات استفاده نمود.

- **توسعه دهندگان بینایی ماشین، هوش مصنوعی و یادگیری ماشین:**

استفاده از سنسورهای مختلف به منظور جمع آوری محیط باعث باعث به وجود آمدن همگام سازی داده‌ها می‌شود. به همین منظور از فناوری‌های ترکیب حسگرها[14] استفاده می‌شود. هم‌چنین، استفاده از دوربین‌ها برای شناسایی اجسام و موانع، نیازمند بهره‌مندی از بینایی ماشین است. به منظور پردازش اطلاعات و تولید فرامین کنترلی نیز از هوش مصنوعیو یادگیری ماشین استفاده می‌شود[۴]. در این زمینه با توجه به ظرفیت دانشگاه‌ها وشرکت‌های داخلی می‌توان بر توانمندی بومی کشور تکیه کرد.

- **تامین‌کنندگان اینترنت خودرو:**

به عنوان یکی از اعضای کلیدی، اینترنت خودرو نقش مهمی در برقراری ارتباط بین خودرو و دیگر وسایل نقلیه، زیرساخت‌ها و... ایفا می‌کند. استفاده از فناوری ۵جی برای انتقال سریع داده بین خودروها با توجه به نیاز سرعت عملکرد مناسب، از مولفه‌های ضروری در سامانه‌های راننده خودکار است[۴]. با توجه به راه‌اندازی نسل پنجم اینترنت در کشور، تامین نیاز صنایع اینترنت خودرو به خوبی انجام می‌گردد. از طرفی ظرفیت مناسبی برای شرکت‌های اینترنت اشیا به منظور ایجاد محصولات مرتبط با اینترنت خودرو فراهم است.

- **صنعت ناوبری و مکان یابی:**

نقشه‌ها با دقت بالا جز حیاتی از سامانه‌های راننده خودکار است. مکان‌یابی با دقت بالا یکی از نیازهای حیاتی خودروهای خودران است.. نقشه‌ها دارای اطلاعات اضافی مانند اندازه خطوط ، محل تقاطع ها و چهارراه ها و ... نیز هستند[۴]. با توجه به ظرفیت‌های موجود در داخل کشور مانند شرکت‌های سازنده نرم افزارهای " بلد" و "نشان" می‌توان از ظرفیت‌های داخلی برای رفع نیاز استفاده کرد.

[14] Sensor Fusion

- **صنعت پردازنده‌ها، پلفت‌فرم‌ها و بردها:**

قدرت پردازشی مورد نیاز برای سامانه‌های راننده خودکار، نیاز به طراحی پلتفرم‌های اختصاصی با عملکرد مطلوب است. طراحی و ساخت خودروی خودران با توجه به نوع عملکرد آن ، نیازمند قدرت پردازش گرافیکی بالا، توانایی انجام محاسبات سنگین و پیچیده در زمان کم ، ترکیب داده‌ها و ... است. به همین دلیل شرکت‌های تولید کننده سخت افزار مانند اینتل[۱۵] و انویدیا[۱۶] به دنبال طراحی وساخت پلتفرم‌های مخصوص سامانه‌های راننده خودکار هستند[۴]. با توجه به عدم ظرفیت بومی در این بخش، نیاز به تامین این قطعات به وسیله واردات وجود دارد.

- **ارتباطات (V2X,V2I,V2V):**

برای شناسایی اشیا و اجسامی که توسط سنسورها رصد شده‌اند، فناوری‌های پیشرفته ارتباطی نیاز است. فناوری ارتباط بین خودرو و همه‌چیز که به اختصار V2X نامیده می‌شود، قابلیت برقراری ارتباط با دیگر وسایل با نام اختصاری V2V و برقراری ارتباط با زیرساخت‌ها مانند چراغ‌های ترافیکی با نام اختصاریV2I را فراهم می‌کند[۴]. این فناوری در حال حاضر در کشور ظرفیت‌های بالایی برای توسعه دارد و تقریبا در این زمینه کاری انجام نشده است.

۲-۲-۳-۱- صنایع پایین دستی

با ورود سامانه‌های راننده خودکار، صنایع پایین‌دستی نیز دچار تغییر می‌شوند. شرکت‌های بیمه نیاز به ایجاد روش‌های جدید بیمه می‌گردند. خدمات پس از فروش این سامانه‌ها نیز از صنایع پایین‌دستی این سامانه‌ها است. در ادامه با توجه به کمبود فضا و تعداد کلمات به بررسی دو مورد مهم از صنایع پایین‌دستی سامانه‌های راننده خودکار پرداخته شده است:

- **امنیت سایبری:**

وجود ارتباطات گسترده خودروی خودران و هم‌چنین بهره گیری آن از فناوری‌های دیجیتالی، باعث به وجود آمدن چالش امنیت سایبری آن‌ها می‌شود. جلوگیری از هک شدن سامانه‌های راننده خودکار، درج اطلاعات غلط و هم‌چنین امنیت مسافران از مولفه‌های مهم در ایمنی خودروهای خودران است[۴]. به همین منظور، می‌توان از ظرفیت داخلی برای ایمن سازی ارتباطات سامانه‌های راننده خودکار استفاده نمود.

[15] Intel
[16] Nvidia

- **خدمات پویا**[17]:

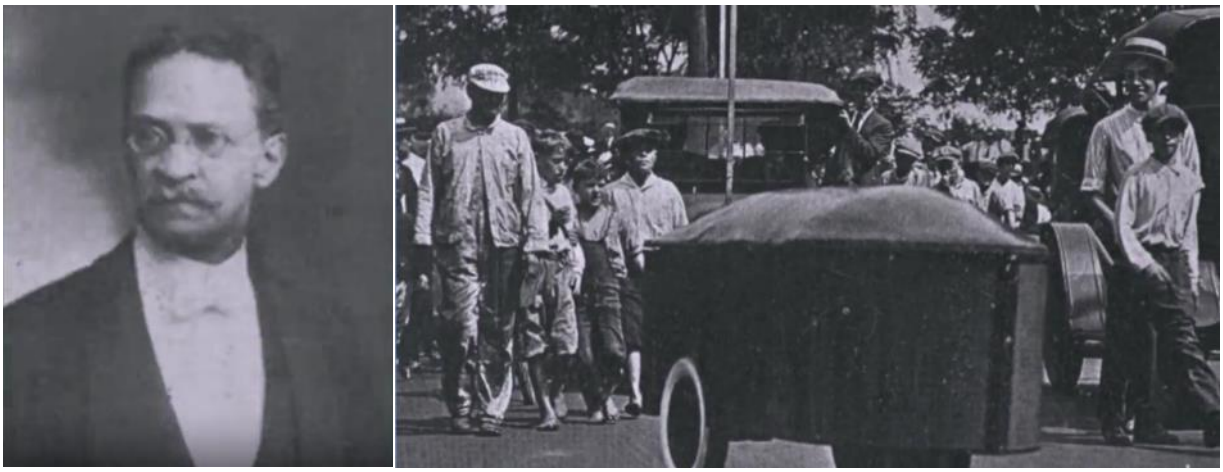
خدمات پویا به پلتفرم‌هایی گفته می‌شود که امکان ارائه خدمات متنوع در قالب یک سرویس را ارائه می‌دهند. از جمله این خدمات می‌توان به سفارش غذا، رزرو هتل، خرید بلیط و ... اشاره کرد[۴]. خودروهای خودران باعث ایجاد تغییرات در ارائه خدمت تاکسی‌های آنلاین و اجاره ماشین گردد. شرکت‌هایی مانند اوبر و اسنپ این خدمات را به مشتریان خود ارائه می‌دهند. در صورت استفاده از سامانه‌های راننده خودکار، هزینه سفرها برای شرکت‌ها به دلیل حذف راننده کاهش می‌یابد. به همین دلیل، این شرکت‌ها را می‌توان از سرمایه‌گذاران برای توسعه سامانه‌های راننده خودکار قلمداد کرد.

با توجه به مطالب گفته شده در بالا، استفاده از سامانه‌های راننده خودکار، باعث ورود بخش جدید از صنایع به صنعت خودروسازی می‌شود. مهم‌ترین این صنایع، صنایع الکترونیکی می‌باشد. با توجه به آنکه تولید بسیاری از قطعات سخت‌افزاری مانند سنسورها، بردها و ... در داخل کشور انجام نمی‌شود، این موضوع به یکی از چالش‌های سامانه‌های راننده خودکار تبدیل می‌گردد. تامین سخت‌افزار با کیفیت با توجه به تحریم‌های بین‌المللی در حجم زیاد باعث خروج ارز از کشور می‌گردد. از طرفی برای تولید داخلی این قطعات نیاز به طی مراحل تحقیق و توسعه است که از لحاظ زمانی چالش برانگیز می‌باشد. از طرف دیگر، صنایع بیمه نیز دچار دگرگونی خواهند شد. استفاده از راننده‌های خودکار، نیازمند تدوین قوانین جدید در سیستم حمل و نقل و به طبع آن شیوه‌های بروز شده برای کارگزارن بیمه خواهد بود. علاوه بر این، نیاز به توسعه نرم‌افزارها و الگوریتم‌های جدید برای به کارگیری در سامانه‌ها می‌باشد. هم‌چنین وجود قوانین و استانداردهای مختلف در زمینه حمل و نقل در کشورهای متفاوت، باعث عدم امکان استفاده از یک سیستم یکسان در کشورهای مختلف می‌شود.

۴-۱- تاریخچه خودروهای خودران

رویای اتومبیل های خودران تقریباً به قدمت اتومبیل است. در اوایل سال ۱۹۲۵، فرانسیس اودینا[18] یک اتومبیل کنترل از راه دور به نام شگفتی آمریکایی را به نمایش گذاشت که با چرخیدن در خیابان های منهتن با صندلی راننده خالی، عابران را می ترساند. در شکل ۶-۱ اولین خودرو کنترل از راه دور و فرانسیس اودینا نشان داده شده است.

[17] Mobility Services
[18] Francis Udina



شکل۶-۱ اولین خودرو کنترل از راه دور با نام شگفتی آمریکایی(سمت راست) و فرانسیس اودینا(سمت چپ)[۵].

رویای اتومبیل رانندگی در سال ۱۹۵۶ جان تازه ای گرفت. جنرال موتورز[۱۹] یک ویدیوی تبلیغاتی به نام "کلید آینده" منتشر کرد که ادعا می کرد اتومبیل های خودران تا سال ۱۹۷۶ وارد بازار می شوند. سرنشینان از طریق ویژگی های کنترل صوتی با ماشین ارتباط برقرار می کنند. کمی خوش بینانه است ، اما از نظر زمان‌بندی ، شگفت آور است که چگونه بسیاری از این پیش بینی های ویدیوها اکنون در بازار در دسترس هستند. در سال ۱۹۸۶ ، ارنست دیکنز[۲۰] و تیمش از دانشگاه مونیخ پیشگام در ساخت یک ون رباتیک شدند که می توانست کاملاً مستقل رانندگی کند و در سال ۱۹۸۷ توانستند سرعت آن را تا ۶۰ کیلومتر در ساعت برسانند. در شکل ۷-۱ نمایی بیرونی و داخلی این خودرو نشان داده شده است.

شکل ۷-۱، نمای از بیرون و داخل ون رباتیک ساخته دانشگاه مونیخ[۵].

[19] General Motors
[20] Ernst Dickens

در اوایل دهه ۱۹۹۰ ، تیم وی به طور قابل توجهی به پروژه Eureka Prometheus کمک کرد و توانست یک دیملر بنز خودران با استفاده از بینایی رایانه ایجاد کند. آنها آخرین ریزپردازنده های روز و همان رویکردهای احتمالی که در رباتیک استفاده می شود را به کار گرفتند تا خودرو به موقع به موقعیت های جاده واکنش نشان دهد. در نهایت ، خودروی آن ها ۱۶۰۰ کیلومتر را از مونیخ تا کپنهاگ با مداخله ۹ کیلومتری انسان، رانندگی کرد. در شکل ۸-۱ نمایی از داخل و بیرون این خودرو نشان داده شده است.

شکل ۸-۱ نمای داخلی و بیرونی خودروی دیملر بنز[۵].

در همان زمان در آمریکا ، آزمایشگاهی در دانشگاه کارنگی ملون مشغول ساخت یک سری پروتوتایپ از خودرو خودران بود. در سال ۱۹۸۶ ، اولین وسیله نقلیه خودران با نام Navlab1 و با استفاده از چندین ایستگاه خورشیدی، ۳۰ کیلومتر در ساعت در جاده را مدیریت کرد. سپس در سال ۱۹۹۰ خودروی Navlab2 که یک خودروی هوموی[۲۱] بهبود یافته که می توانست به صورت خودران و در جاده های مختلف حرکت کند، به سرعت ۱۱۰ کیلومتر در ساعت در جاده رسید. در شکل ۹-۱ خودروهای ساخته دانشگاه کارنگی ملون نشان داده شده است. این پیشرفت ها تیم را وادار به تلاش بسیار پروازانه تر کرد، و در سال ۱۹۹۶ ، آنها سفر بدون دست خودشان را در امریکا به پایان رساندند، مسیری ۴۸۰۰ کیلومتری در قاره با ۹۸.۲ درصد رانندگی خودران.

شکل ۹-۱، خودروی Navlab1 (سمت راست) و Navlab2 (سمت چپ) [۵].

[21] Humvee

اندکی پس از آن، پروژه موفقیت آمیز دانشگاه برکلی با نمایش موفقیت آمیز دسته های خودران وسایل نقلیه ای که در خطوط اختصاصی حرکت می کردند ، برجسته شد. در سال ۱۹۹۲ ، چهار وسیله نقلیه با تکیه بر نشانگرهای مغناطیسی موجود در جاده برای تعیین موقعیت نسبی دقیق به صورت کاروانی حرکت کردند و نشان دادند که استفاده از این نوع کاروان باعث صرفه جویی در هزینه های سوخت و کاهش کشش باد می‌شود. این پروژه همچنین زمینه استفاده از سیستم های رادار و ارتباطات خودرو به خودرو را در صنعت خودروهای خودران را ایجاد کرد، که منجر به پیشرفت هایی مانند کنترل کروز تطبیقی[۲۲] و ترمز اضطراری شد.

در سال ۲۰۰۲ دارپا ، آژانس پروژه های تحقیقات پیشرفته دفاعی اولین چالش های بزرگ خود را اعلام کرد. این باعث شد که برای همیشه درک جهان در مورد آنچه که ربات های خودمختار می توانند انجام دهند، تغییر یابد. اولین رویداد در سال ۲۰۰۴ برگزار شد و دارپا به برندگان یک جایزه یک میلیون دلاری پیشنهاد داد در صورتی که آن‌ها بتوانند یک وسیله نقلیه مستقل بسازند که بتواند ۱۴۲ مایل در صحرای موهاوی حرکت کند. اگرچه در اولین رویداد فقط چند تیم از خط شروع خارج شدند و تیمی که مقام اول را کسب کرده است در نهایت هفت مایل رانندگی کرده بود. اما واضح بود که وظیفه رانندگی ۱۴۰ مایل در صحرا بدون هیچ گونه کمک انسانی به طور قطع ممکن بود. در شکل ۱۰-۱ نمایی از شرکت کنندگان در این مسابقات نشان داده شده است.

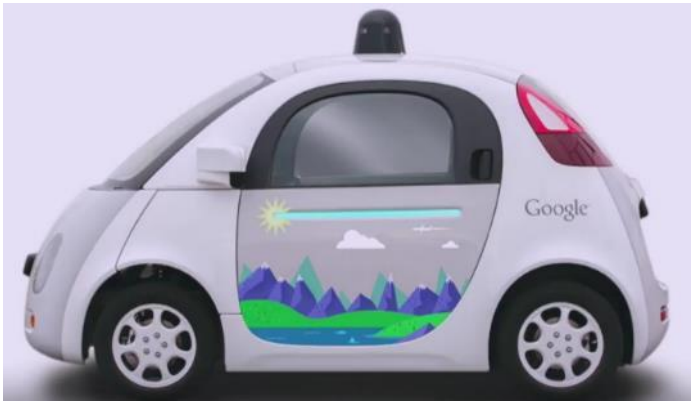
[22] Adaptive Cruise Control

شکل ۱۰-۱ شرکت کنندگان در مسابقات دارپا[۵].

در دومین چالش بزرگ DARPA در سال بعد ، پنج تیم از ۲۳ تیم انتظارات را شکستند و مسیر را بدون هیچ گونه مداخله انسانی با موفقیت به پایان رساندند. دوران اتومبیل های بدون راننده فرا رسیده بود. سپس ، بخش شهری مسابقات دارپا در سال ۲۰۰۷ به‌نام DARPA Urban Challenge ، از دانشگاه ها دعوت کرد تا وسایل نقلیه خودران خود را در جاده های شهری با نظارت رانندگان بدلکار حرفه ای به نمایش بگذارند.

گوگل بلافاصله تیم های کارنگی ملون و استنفورد ، کریس تامپسون و مایک مونت-کارلو را به شرکت خود جذب کرد تا طرح های خود را به جاده های عمومی هدایت کنند. تا سال ۲۰۱۰ ، ماشین گوگل بیش از ۱۴۰ هزار مایل در کالیفرنیا رانندگی کرده بود و بعداً در یک پست وبلاگ نوشته شد که خودرو های خودران می‌توانند تعداد مرگ

و میرهای ناشی از رانندگی را به حداقل نصف کاهش دهند. در سال ۲۰۱۲، اداره وسایل نقلیه موتوری نوادا اولین مجوز آزمایش وسایل نقلیه خودران را به گوگل صادر کرد؛ به این معنی که آن‌ها اکنون می‌توانند به صورت خودران در جاده های عمومی آزمایش کنند. حدود یک سال بعد ، گوگل اتومبیل فایرفلای[۲۳] خود را به نمایش گذاشت ، وسیله نقلیه ای که با سرعت نهایتا ۴۰ کیلومتر در ساعت به صورت خودران طراحی شده است. بدون داشتن فرمان یا پدال، مسافران می توانند در جای خود نشسته و در صورت احساس ناراحتی فقط یک دکمه توقف را فشار دهند. با این کار ، گوگل به ما نشان داد که آینده خودروهای خودران می تواند چگونه باشد. در شکل ۱۱-۱ نمایی از خودروی فایرفلای نشان داده شده است.



شکل ۱۱-۱ ، نمایی از خودروی فایرفلای گوگل[۵].

شرکت تسلا اواخر سال ۲۰۱۵ ، مجموعه ای از ویژگی های خودران خود را به نرم‌افزاری به نام Autopilot وارد کردند. تا سال ۲۰۱۶ ، آنها نشان دادند که چگونه Autopilot می تواند خود پارک کند و حتی تغییر خط دهد. این بزرگ‌ترین عرضه ویژگی های جزئی خودران در آن زمان بود و بنابراین تسلا میلیون‌ها کیلومتر را با خلبان اتوماتیک خود طی کرد. هم‌چنین، تسلا اخیرا از نسخه نرم افزاری کاملا خودران خود رونمایی کرده است که به صورت بتا در دسترس کابران قرار دارد. در شکل ۱۲-۱ نمایی از این سیستم نشان داده شده است. با افزایش تعداد مایل های مستقل رانده شده، اولین تصادف مرگبار با یک تسلا مدل S در فلوریدا در سال ۲۰۱۶ اتفاق افتاد. این امر به دلیل خرابی همزمان حسگرهای دوربین و رادار در شناسایی صحیح یک کامیون حمل و نقل در حال چرخش به چپ و همچنین بی توجهی راننده رخ داد.

[23] FireFly

شکل ۱۲-۱، نمایی از سیستم راننده خودکار شرکت تسلا[۵].

با وجود این عقب نشینی ، پیشرفت بی وقفه ادامه یافت. در آن زمان بود که مفهوم ناوگان تاکسی های خودران در حال محبوب شدن بود و استارت آپ هایی مانند Zoox و NuTonomy پیشگام هستند. در سال ۲۰۱۸ تصادف خودروی خودران اوبر[۲۴] جان عابر پیاده در آریزونا را گرفت. این حادثه موج شوک را به جامعه آزمایش اتومبیل های خودران فرستاد و اوبر مجبور شد آزمایش را متوقف کند تا زمانی که جزئیات از طریق تحقیقات مستقل توسط اداره ایمنی ترافیک بزرگراه ملی مشخص شود.

بنابراین، تاریخچه توسعه وسایل نقلیه خودران، داستان موفقیت های عظیم و همچنین شکست‌هایی است که به نظر می رسد اخیراً بیشتر اتفاق می‌افتد. درحالی که این فناوری هنوز بسیار گران است ، اما با گسترده شدن مقیاس آن، قیمت ها به سرعت کاهش می یابد. هنوز کارهای زیادی باید انجام شود تا تغییرات تحول آفرینی که انتظار داریم خودروهای بدون راننده در جامعه ما ایجاد کنند ، ایجاد شود. این صنعت همچنان به مهندسان برجسته ای برای تولید راه حل های ابتکاری برای رانندگی نیاز خواهد داشت[۵].

۵-۱- چالش‌های خودروهای خودران

چالش‌های خودروهای خودران به سه گروه مختلف تقسیم‌بندی شده است. این گروه‌ها در ادامه آورده شده است[۶]:

[24] Uber

۱-۵-۱- چالش‌های فنی

چالش‌های فنی خودروهای بدون راننده عبارتند از:

- **یکپارچه‌سازی نرم‌افزار:**

به دلیل وجود تعداد زیادی از سنسورها و الگوریتم‌های مورد نیاز در سامانه‌های خودکار، اطمینان از یکپارچه-سازی مناسب بخش‌های سخت‌افزاری و نرم‌افزاری این سامانه‌ها به عنوان یک چالش مطرح است.

- **پیش‌بینی و اعتماد در میان دیگر خودروها:**

پیش‌بینی رفتار و حرکات دیگر خودروها یکی از چالش‌های سامانه‌های خودران است. رانندگان انسانی با وجود اطلاعات کم و تنها از طریق مشاهده قادر به پیش‌بینی رفتار دیگر رانندگان هستند. در حالی که، پیش‌بینی رفتار دیگر رانندگان برای سامانه‌های خودران دارای عدم قطعیت‌های فراوانی است. علاوه‌بر این، وجود قوانین ترافیکی و رعایت حق تقدم بین خودروها از دیگر مسائلی است که سامانه خودران باید درک درستی از آن داشته باشد. وجود محیط شامل رانندگان خودکار و انسان‌های راننده به پیچیدگی این چالش می‌افزاید.

- **افزایش مقیاس:**

سامانه‌های راننده خودکار باید از عملکرد مناسبی در موقعیت‌های شلوغ مانند ترافیک‌های سنگین برخوردار باشند. سرعت مناسب در واکنش نشان دادن در مواجه با اشیایی که به صورت ناگهانی طاهر می‌شوند، از دیگر معیارهای اعتبارسنجی عملکرد مطلوب راننده‌های خودکار است.

- **شرایط جوی:**

سامانه‌های رانندگی خودکار، عموما در محیط‌های با شرایط غیرواقعی مورد تست و ارزیابی قرار می‌گیرند. یکی از چاش‌های بزرگ در این زمینه، عملکرد مطلوب سامانه در شرایط غیرطبیعی همانند شرایط جوی مختلف یا رانندگی در شب است. در این شرایط جمع‌آوری داده از محیط مختل شده و به تبع آن عملکرد مطلوب سامانه دچار مشکل می‌گردد.

۱-۵-۲- چالش‌های اجتماعی

یکی از گام‌های مهم در مسیر دستیابی به استفاده از خودروهای خودران ، پذیرفته شدن توسط عموم جامعه است. مطالعات نشان دهنده آن است که عموم مردم باور دارند که استفاده از خودروهای خودران ایمن‌تر است به شرطی که شرکت‌های خودروسازی، ملزم به ایمن‌سازی استفاده از این خودروها شوند. عوامل تاثیر گذاردر پذیرفته شدن این نوع از خودروها در بین عموم مردم عبارتند از:

- **کاربردی بودن:**

استفاده از سامانه‌های خودران باید موجب صرفه‌جویی در وقت مشتریان و راحت‌تر شدن زندگی برای آن‌ها شود.

- **کاربری راحت:**

طراحی سامانه‌های راننده خودکار باید به صورتی باشد که استفاده از آن به سادگی برای مشتریان امکان پذیر باشد. مطالعات نشان دهنده توجه بیشتر مردم به این عامل در مقایسه با ایمنی سامانه‌های خودران است.

- **قابل اعتماد بودن:**

قابل اعتماد بودن در واقع به معنای ایمنی و محافظت از استفاده کنندگان است. سیستمی که بیشتر مورد اعتماد کاربران باشد، از شانس بیشتری در تصمیم‌گیری آن‌ها برای استفاده برخوردار است.

۳-۵-۱- چالش‌های قانونی

ارزیابی عملکرد خودروهای خودران در محیط واقعی بخش مهمی از فرآیند توسعه سامانه‌های راننده خودکار است. در حالی که، قانون‌گذاران با چالش‌هایی در مواجه با امنیت عمومی مردم و در عین حال صدور مجوز شرکت‌های خودروسازی جهت تست و ارزیابی محصولات‌شان هستند. هم‌چنین، تدوین قوانین مناسب برای استفاده از سامانه‌های خودران و بروزرسانی آن از دیگر چالش‌های قانون گذاران است. در حالی که قوانین سخت-گیرانه‌ای که برای محافظت از جان مردم وضع می‌شود، موجب سخت‌تر شدن طراحی و ارزیابی عملکرد سامانه‌های خودران می‌شود. علاوه‌براین، وجود قوانین مختلف در کشورهای متفاوت نیز یکی دیگر از چالش‌های طراحی سامانه‌های خودران است. تطبیق سامانه‌های خودران با قوانین مختلف موجب محدودتر شدن فضای طراحی آن‌ها خواهد شد.

همچنین چالش‌های سامانه‌های راننده خودکار به طور جامع در جدول ۱-۱ آورده شده است و راه حل‌های ممکن برای این چالش‌ها نیز ذکر شده است[۶].

جدول ۱-۱، چالش‌ها و راه حل‌های ممکن سامانه راننده خودکار[۱۰].

طبقه بندی	چالش های اصلی		راه حل های امکان پذیر	
چالش های اجتماعی :S	چالش های سیاست :P		چالش های غیر فنی :NT	چالش های فنی:T
:T1 اعتبار سنجی / آزمایش	• مجموعه کاملی از نیازها نیست • عملیات پویا و غیر قطعی • پیچیدگی عملیات • ماهیت انتقادی		• تقسیم عملکرد اجزای نرم افزار / سخت افزار • مفاهیم عملیاتی محدود • روش های ارجاع استقرایی و یادگیری ماشین • تزریق عیب	
:T2 ایمنی و قابلیت اطمینان	• فاصله رانده شده در درایوهای آزمایشی قابلیت اطمینان را تعیین نمی کند • شباهت به اعتماد به نفس در سطح انسانی در مسئولیت به منابع زیادی نیاز دارد • قانون مبهم است • حذف عملکرد جدا کردن خطری است • چرخه اعتبار نامشخص است		• تعریف مأموریت های ایمن کوتاه مدت • تعریف بیشتر سیستم های ایمن در برابرخرابی • توسعه الگوریتم های پیچیده برای ماموریت های کوچک • به کارگیری یادگیری ماشین، یادگیری عمیق و هوش مصنوعی	
:T3 کیفیت نرم افزار	• الزامات بودجه هنگفت • سناریوهای پیش بینی نشده بسیار زیاد است • اتومبیل مستقل و نرم افزار آن سیستم پیچیده ای را نشان می دهد		• نتایج ایمن از شکست به جای غیر قابل پیش بینی بودن • تخریب عملکرد در صورت لزوم • تست کیفیت نرم افزار هدف گرا	
:T4 منابع محاسباتی	• اتومبیل خودمختار میزبان سنسورهای ناهمگن است • مقادیر زیادی از داده های تولید شده • افزایش در هزینه • پردازش داده در زمان واقعی • افزونگی قابلیت اطمینان و همچنین هزینه ها را افزایش می دهد		• واحد پردازش گرافیک (GPU) • سیستم بهینه شده روی تراشه • توافق در مورد استانداردها برای بازتر کردن آن در تحقیقات	

T5: **تهدیدات امنیتی و هک**	• **اتومبیل مستقل در محیط شبکه کار می کند و مستعد حملات شبکه است** • **تهدیدات مخرب تزریق ، مسدود کردن ، فازی و هک کردن**	• جداسازی امنیت داده از امنیت ارتباط • احراز هویت کارآمد و موثر • رویکردهای امنیتی مبتنی بر هوش مصنوعی • امنیت توسط طراحی
T6: حریم خصوصی	• چه کسی داده ها را ذخیره می کند؟ • به اشتراک گذاری داده های شخصی و موقعیتی دارای مفاد حریم خصوصی است • متقاعد کردن مصرف کنندگان برای به اشتراک گذاشتن داده های شخصی • تعارض بین حریم خصوصی و کیفیت خدمات	• آگاهی مصرف کننده • آیین نامه عمومی حفاظت از داده ها (GDPR) • سبک و سنگین کردن قابل قبول بین ناشناس بودن و کیفیت اطلاعات
T7: دقت و کارایی تشخیص اشیا	• داده های بسیاری از حسگرها باید در زمان واقعی پردازش شوند • الگوریتم های یادگیری عمیق فضای ذخیره سازی و محاسبه زیادی دارند • افزونگی داده ها ، پرت ها و دانه بودن داده های حسگرها • اصالت داده های سنجش شده	• افزایش منابع محاسباتی و ارتباطی • به اشتراک گذاری داده های حسگرها در بین گره های شبکه • سبک و سنگین کردن تعداد حسگرها و کارایی پردازش داده ها
T9: روشهای تصمیم گیری	• محیط غیرقابل پیش بینی • رفتار انسان از طریق ماشین دشوار است • تصمیم گیری بهینه چالش برانگیز است • شناسایی عیب و عملکرد نادرست سیستم دشوار است	• کشف و درک شی-آگاه از زمینه • آگاهی از وضعیت • الگوریتم های تصمیم گیری و کنترل متناسب با شرایط
T10: عملگر و تحریک	• سازگاری با محیط ناشناخته • اشباع محرک • ورودی اشتباه می تواند منجر به عواقب شدید شود	• مدل فازی و Takagi-Sugeno برای تحریک • اعتبار سنجی ورودی برای عملگر ها
NT1: اعتماد مصرف کننده	• مصرف کنندگان ممکن است تمایلی به اعتماد به اتومبیل های خودمختار نداشته باشند • آزمایش به تمام سوالات / نگرانی های مصرف کنندگان پاسخ نمی دهد • فقدان قانون کافی جهانی • امنیت و حریم خصوصی مصرف کنندگان • تقلید از رفتار رانندگی انسان دشوار است	• ابتکار قانون گذاری به رهبری ایالات متحده برای خودروهای خودمختار • استارتاپ هایی با هدف افزایش اعتماد مشتری • ارتقای آگاهی و انگیزه ها • تبلیغ داستانهای موفقیت • تضمین ایمنی در برابر خرابی • تعداد عملکردها را محدود کنید

:NT2 تنوع	• کنار آمدن با وسایل نقلیه متصل و غیر متصل • عامل انسانی در رانندگی ضروری است • عدم اطمینان در رانندگی انسان • رفتار غیرقابل پیش بینی خودروی خودمختار نسبت به رانندگان انسانی • تنوع محیطی	• اجرای قوانین سختگیرانه راهنمایی و رانندگی • تشخیص و جدا کردن رفتارهای سو رانندگی • آموزش اجتماعی
:NT3 هزینه نامشخص	• هزینه نرم افزار خیلی زیاد است • نگهداری و آزمایش گران استپ • سخت افزار در حال حاضر بیش از حد گران است • اشتراک خدمات ، به روزرسانی های پیشرفته نقشه و سایر هزینه ها • بازگشت سرمایه برای ارائه دهندگان خدمات	• ویژگی های محدود • هزینه های تراز شده • استفاده از اتومبیل خودمختار به عنوان یک ابزار • مدل تجاری مقرون به صرفه • تمرکز بر رضایت مصرف کننده
:NT4 استحکام عملیاتی	• تصمیم گیری در زمان واقعی در سناریوهای غیر قابل پیش بینی • مدیریت جمعیت	• ایمنی در برابر شکست • نمایه رانندگی • تشخیص شی در زمان واقعی • یادگیری از محیط اطراف • مدیریت هوشمند چراغ راهنمایی
:NT5 تعهدات	• چه کسی مسئول حوادث خواهد بود؟ • چه کسی بیمه خواهد شد؟ صاحب ماشین ، سرنشین یا سازنده؟	• مدل تجاری جدید و مقررات / قوانین جدید • بازنگری در تجارت بیمه • راه حل های تولید کننده محور • راه حل های کارآمد پزشکی قانونی
:S1 رفتار انسانی	• انسان ها به طور کلی تمایلی به تغییر رفتارهای خود ندارند • رفتار رانندگان نسبت به اتومبیل های خودمختار می تواند تهاجمی باشد • اتومبیل های خودمختار ممکن است رفتار رانندگی و تفکر انسان را کاملاً تقلید نکنند	• پاسخ سریع هوشمند و در زمان واقعی با نرم افزار کارآمد ، این موارد را کاهش می دهد • اجرایی نظریه های اخلاقی متعدد و آزمون • پیاده سازی رفتارهای مختلف رانندگی • مشاغل جایگزین برای افرادی که شغل خود را از دست می دهند
:S2 پیامدهای اخلاقی	• حق تقدم در مورد اتومبیل خودمختار پیچیده تر است • همدلی انسان را نمی توان در اتومبیل خودمختار اجرا کرد • تصمیم گیری بهینه سخت است • نگرانی های عدالت اجتماعی	• پاسخ سریع هوشمند و در زمان واقعی با نرم افزار کارآمد ، این موارد را کاهش می دهد • اجرایی نظریه های اخلاقی متعدد و آزمون • پیاده سازی رفتارهای مختلف رانندگی • مشاغل جایگزین برای افرادی که شغل خود را از دست می دهند

:P1 چالش های سیاست	• بررسی مجدد بسیاری از سیاست ها ممکن است چالش های بیشتری برای سیاست ها ایجاد کند • هیچ سیاست روشنی وجود ندارد زیرا اتومبیل های خودمختار هنوز تجاری نشده اند • ایمنی و سودمندی با عکس هم متناسب هستند • سیاست مشخصی برای صدور گواهینامه اتومبیل خودمختار وجود ندارد	• سیاست ها در حال حاضر در حال اجرا هستند • توصیه هایی که نگرانی های همه ذینفعان را در بر می گیرد • گروه ویژه بین المللی متشکل از کلیه ذینفعان برای تدوین سیاست های صحیح است

۶-۱- فناوری‌های خودروهای خودران

برای طراحی و ساخت سامانه‌های خودران، از فناوری‌های مختلفی استفاده می‌شود. در شکل ۱۳-۱ نمای کلی از این فناوری‌ها آورده شده است:

شکل ۱۳-۱، نمای کلی از فناوری‌های به کار گرفته شده در سامانه راننده خودکار[۷].

در ادامه به بررسی هر یک از فناوری‌های به کارگفته شده در طراحی سامانه راننده خودکار پرداخته می‌شود. این فناوری‌ها در دو دسته فناوری‌های اصلی و فناوری‌های جانبی مورد بررسی قرار گرفته است.

۱-۶-۱- فناوری‌های اصلی

فناوری‌های اصلی شامل فناوری‌هایی است که به صورت مستقیم در طراحی و ساخت سامانه راننده خودکار دخیل هستند. در ادامه به بررسی بخشی از این فناوری‌ها پرداخته شده است.

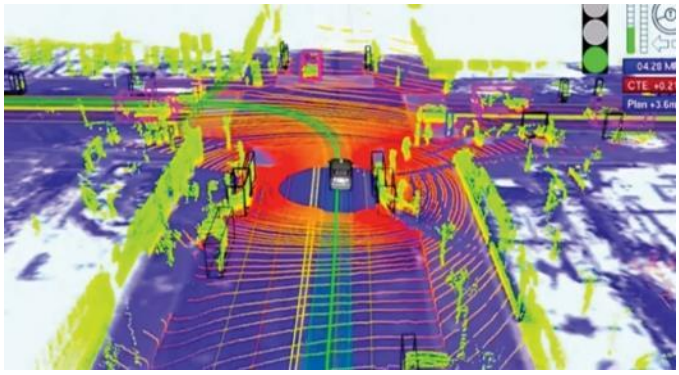
۱-۶-۱-۱- لیدار[۲۵]

آشکارساز و مسافت‌یاب لیزری[۲۶] که با اختصار لیدار نامیده می‌شود، به عنوان چشم سامانه خودران اتومبیل عمل می‌کند. این فناوری به عنوان یک فناوری برد بلند مورد استفاده قرار می‌گیرد. فناوری لیدار که همانند یک آژیر گردان دارای چرخش است، دید ۳۶۰ درجه‌ای از اطراف برای خودرو را تامین می‌کند[۸]. در شکل ۱۴-۱، نمونه‌ای از لیدار مورد استفاده در خودروهای خودران آورده شده است.

شکل ۱۴-۱، نمونه‌ای از سنسورهای لیدار مورد استفاده در سامانه‌های خودران[۹].

این فناوری از ارسال هزاران پالس لیزری در هر ثانیه، برخورد امواج لیزر با اجسام اطراف و سپس بازگشت آن به سوی سنسور استفاده می‌کند. یک رایانه با استفاده از پالس‌های بازگشتی، به سرعت یک مدل ابر نقاط سه‌بعدی از محیط اطراف را شبیه‌سازی کرده و در هر ثانیه آن را بروز رسانی می‌کند. در شکل ۱۵-۱، نمونه‌ای از شبیه‌سازی سه‌بعدی محیط آورده شده است. مدل سه‌بعدی با استفاه از اندازه‌گیری سرعت نور و میزان فاصله‌ای که برد لیزر تحت پوشش قرار می‌دهد، فاصله و موقعیت اجسام را تعیین می‌کند. هم‌چنین مدل سه‌بعدی، بر فاصله اجسام اطراف نظارت داشته که باعث می‌شود در صورت لزوم، سامانه اقدام به ترمزگیری یا افزایش سرعت نماید. تفاوت این فناوری با رادار در استفاده از امواج لیزری به جای امواج رادیویی است[۸].

[25] LiDar
[26] Light Detection and Ranging



شکل۱۵-۱، نمایی از شبیه‌سازی سه‌بعدی محیط اطراف به وسیله فناوری لیدار[۱۰].

۲-۱-۶-۱- اولتراسونیک[۲۷]

سنسورهای اولتراسونیک جز سنسورهای فعال و برد کوتاه هستند. همانند رادار و سونار، سنسورهای اولتراسونیک از اندازه‌گیری امواج بازگشتی برای رسیدن به هدف خود استفاده می‌کنند. این نوع از سنسورها با استفاده از امواج صوتی و اندازه‌گیری زمان ارسال و بازگشت آن، فاصله میان اجسام را مورد ارزیابی قرار می‌دهند. با توجه به نوع عملکرد این نوع از سنسورها، استفاده از آن‌ها برای شناسایی موانع در فواصل کوتاه به عنوان یکی از گزینه‌های مطلوب است. به همین دلیل اغلب از سنسورهای اولتراسونیک برای پاک خودرو استفاده می‌شود[۱۱]. در شکل ۱۶-۱ نمونه‌ای از این سنسورها آورده شده است.

شکل ۱۶-۱، نمونه‌ای از سنسورهای اولتراسونیک مورد استفاده در سامانه‌های راننده خودکار[۱۱].

[27] Ultrasonic

در این نوع از سنسورها، ابتدا انرژی الکتریکی به امواج صوتی تبدیل می‌گردد. پس از تاباندن امواج صوتی به اجسام، این اجسام امواج صوتی را به شکل‌های مختلف بازتاب می‌کنند. اجسام جامد از جنس فلزات، پلاستیک و شیشه امواج صوتی را بازتاب می‌کنند. در حالی‌که، اجسام نرم همانند پارچه‌ها این امواج را جذب می‌کنند. به همین دلیل از این سنسورها برای شناسایی موانع جامد و سخت استفاده می‌شود. مزایا و معایب این نوع از سنسورها در جدول ۲-۱ آورده شده است[۱۱].

جدول ۲-۱، مزایا و معایب سنسورهای اولتراسونیک[۱۱].

مزایا	معایب
جمع و جور بودن	تاثیرپذیری عملکرد از جنس موانع و شکل آن‌ها
عدم وجود اجزای متحرک	تاثیرپذیری شدید عملکرد از دما
عدم تاثیر پذیری عملکرد از رنگ یا شفافیت اجسام	طیف عملکردکوتاه – حداکثر ۸ متر
عدم تاثیر پذیری عملکرد از شرایط جوی	
عدم تاثیر پذیری عملکرد از شرایط نوری	
عملکرد مناسب برای اندازه‌گیری فاصله سطوح موازی	

۳-۱-۶-۱- دوربین

اتومبیل‌های خودران، به طور معمول دارای دوربین‌های ویدئویی برای دیدن و تفسیر اجسام موجود در جاده‌ها هستند. با مجهز کردن اتومبیل به دوربین‌ها در هر زاویه‌ای، سامانه راننده خودکار، توانایی داشتن دید ۳۶۰ درجه‌ای از محیط اطراف را داراست. هم‌چنین، این دوربین‌ها تصویری گسترده از شرایط ترافیکی اطراف خودرو را ارائه می‌دهند. امروزه از دوربین‌های سه‌بعدی برای گرفتن تصاویر با کیفیت، واقعی و با جزئیات استفاده می‌شود. این نوع از دوربین‌ها، به طور خودکار اجسام را شناسایی و دسته‌بندی می‌کنند. هم‌چنین، توانایی تعیین فاصله اجسام را نیز دارا هستند. به عنوان مثال، این دوربین‌ها توانایی تشخیص خودروها، عابران پیاده، دوچرخه‌سوارها، علائم ترافیکی، خط‌کشی خیابان‌ها و... را دارا هستند. دوربین‌های مورد استفاده در سامانه‌های راننده خودکار، شناسایی

و دسته‌بندی اجسام و افراد رابه وسیله فناوری پردازش تصویر انجام می‌دهند[۱۲]. در شکل ۱۷-۱ نمونه‌ای از این دوربین‌ها نشان داده شده است.

شکل ۱۷-۱ ، نمونه‌ای از دوربین‌های مورد استفاده در سامانه‌های راننده خودکار[۱۳].

استفاده از دوربین‌ها به تنهایی در سامانه‌های راننده خودکار انجام نمی‌شود. دلیل این امر، عملکرد نامطلوب دوربین‌ها در شرایط آب و هوایی مختلف مانند مه، باران و... است. هم‌چنین، در شرایطی که رنگ موانع و وسایل نقلیه دیگر با رنگ محیط یا پس‌زمینه مشابه باشد، دوربین عملکرد مطلوبی در تفکیک اجسام و موانع نخواهد داشت. علاوه بر این، وجود کنتراست نامناسب در تصویر ارسالی به مرکز پردازش سیستم نیز عملکرد نامطلوب سامانه راننده خودکار را در بر خواهد داشت[۱۲].

۴-۱-۶-۱- رادار[۲۸]

آشکارساز و مسافت‌یاب رادیویی[۲۹] که به اختصار رادار نامیده می‌شود، یکی از فناوری‌های مرسوم مورد استفاده در سامانه‌های راننده خودکار است. این فناوری با استفاده از تابش امواج الکترومغناطیس به اجسام و بازتاب آن برای شناسایی موانع کار می‌کند. استفاده از فناوری رادار به دلیل وجود نویزهای محیطی به تنهایی در سامانه‌های راننده خودکار مورد استفاده قرار نمی‌گیرد. از این فناوری در کنترل کروز وسایل نقلیه نیز استفاده می‌شود. با توجه به استفاده از طول موج رادویی در رادار و ساز و کار آن، این فناوری توانایی محاسبه سرعت اجسام به صورت مستقیم و عملکرد مطلوب در هر شرایط آب و هوایی را داراست. از دیگر مزایای آن می‌توان به قیمت کمتر آن نسبت با دیگر سنسورها اشاره کرد. رادارها دارای انواع مختلفی هستند و یکی از انواعی که در سامانه‌های راننده خودران بیشتر مورد استفاده قرار می‌گیرد، با نام امواج ممتد ماژول فرکانسی[۳۰] یا به اختصار FMCW است. این نوع از

[28] RADAR
[29] Radio Detection and Ranging
[30] Frequency-Modulated Continuous Wave

رادارها، امواج را به صورت ممتد به اطراف می‌تاباند و توانایی شناسایی اجسام خیلی کوچک و اندازه‌گیری سرعت را دارا می‌باشد. هم‌چنین، توانایی شناسایی محیط زیر وسایل نقلیه و عمق زمین تا حدود ۸ متر از دیگر مزیت‌های رادارهای مورد استفاده در سامانه‌های راننده خودران است[۱۴]. در شکل ۶-۱ نمونه‌ای از رادارهای مورد استفاده در سامانه‌های راننده خودران آورده شده است.

شکل۱۸-۱، نمونه‌ای از رادارهای مورد استفاده در سامانه‌های راننده خودران[۱۴].

۵-۱-۶-۱- بینایی ماشین

تشخیص شی[۳۱] و بینایی ماشین[۳۲] از مهمترین و اساسی ترین ویژگی های اتومبیل های خودمختار هستند. برای تقلید از رفتار راننده انسان ، اتومبیل های خودمختار باید "جاده" را ببینند و هر مانعی را در مقابل و اطراف آن تشخیص دهند، چه اتومبیل دیگری باشد، چه عابر پیاده، پوشش گیاهی و یا هر نوع مانع دیگر. این دو فناوری کلیدی به همراه سایر ماژول ها، ماشین مستقل را قادر می سازند تا در جاده رانندگی کند و به هر گونه وضعیت ناخواسته، به صورت ایمن عکس العمل نشان دهد، به عنوان مثال توقف در یک سیگنال ترافیک، کاهش سرعت در صورت کم شدن سرعت ماشین جلویی، جلوگیری از برخورد به عابر پیاده و...[۱۵].

اگرچه بینایی ماشین یک حوزه وسیع است که جنبه های مختلفی از کسب تصویر گرفته تا تقسیم بندی و طبقه بندی را پوشش می دهد، با این حال، در این کار ما فقط بر روی تشخیص شی، کالیبراسیون و برآورد حرکت در ارتباط با اتومبیل های خودمختار تمرکز می کنیم. تشخیص شی یک نیاز اساسی برای اتومبیل های خودمختار است. اتومبیل مستقل برای حفظ مانورهای مختلف باید اشیای ساکن و پویا را تشخیص دهد. با این وجود، به دلیل چندین دلیل مانند سایه ها، اجسام یکسان، شرایط نوری و غیره، تشخیص شی در اتومبیل های خودران

[31] Object Detection
[32] Computer Vision

چالش برانگیز است. بنابراین، الگوریتم های اساسی باید این عوامل را در نظر بگیرند. نمونه ای از بینایی ماشین را در شکل۷-نشان داده شده است[۱۵].

شکل ۱۹-۱ ، نمونه‌ای از بینایی ماشین[۱۵].

تشخیص اشیا با کمک حسگرهای مختلف از دوربین های ارزان قیمت گرفته تا لیدار پیچیده و رادار انجام می شود. علاوه بر این، اتومبیل خودمختار برای اهداف مختلف و انواع محیط به سنسورهایی نیاز دارد. این موارد شامل نور مرئی (روز)، حسگرهای مادون قرمز (زمان شب یا نور کم) و مادون قرمز حرارتی برای شناسایی موجودات زنده است. مکانیزمی که حسگرهای مختلف را برای داشتن یک دید دقیق، یکپارچه و جامع از داده های سنجش شده ترکیب می کند، همجوشی حسگر نامیده می شود که ترکیبی از داده های آرایه ای از سنسورهای مختلف است. تحولات اخیر در همجوشی حسگر دلگرم کننده است و می‌تواند در فناوری اتومبیل‌های خودمختار تجاری گنجانده شود به عبارت دقیق‌تر، روش‌های تشخیص اشیای مبتنی بر همجوشی سنسور، در مقایسه با روش های سنتی تشخیص اشیا، دقت بهتری را ارائه می دهند[۱۵].

۶-۱-۶-۱- هوش مصنوعی

یادگیری ماشین[۳۳]، یادگیری عمیق[۳۴] و تکنیک های مبتنی بر هوش مصنوعی [۳۵]برای اتومبیل های خودران ضروری است. دلیل اصلی اهمیت این فناوری‌ها، محیط و رفتار غیر قابل پیش‌بینی اجسام اطراف است. بیشتر الگوریتم‌ها و مکانیزم‌های مربوط به بینایی ماشین مانند شناسایی شی، درک، شناسایی صحنه، بازسازی و تخمین،

[33] Machine Learning
[34] Deep Learning
[35] Artificial Intelligence

از مکانیزم های یادگیری ماشین و یادگیری عمیق استفاده می کنند. در این بخش، ما پیشرفت های اخیر در یادگیری ماشین و یادگیری عمیق بهینه شده برای فناوری اتومبیل خودمختار را ارائه می دهیم. علاوه بر مشارکت-های قبلی یادگیری ماشین، آزمایش نرم افزار همچنین با کمک تکنیک‌های یادگیری ماشین در اتومبیل‌های خودمختار انجام می شود. در نرم افزارهای سنتی، منطق عملیاتی به صورت دستی نوشته می شود و در یک سری موارد آزمایشی آزمایش می شود، در حالی که در نرم افزار مبتنی بر شبکه عصبی عمیق[۳۶]، این نرم افزار با کمک مجموعه داده های بزرگ فرا می گیرد و سازگار می شود[۱۵].

برخی از مدل های معروف یادگیری عمیق رایج که در اتومبیل های خودمختار استفاده می شوند را مرور می کنیم. یکی از جنبه های اساسی اتومبیل های خودمختار، ادراک است و کاندید مناسبی برای استفاده از مدل های یادگیری عمیق است. تحریک اتومبیل خودمختار به شدت به ادراک بستگی دارد و بنابراین، برای اتومبیل‌های خودمختار مهم است که از توانایی مانند درک انسان برخوردار باشند. مدل های یادگیری عمیق همچنین برای تصمیم گیری آگاهانه به پردازش داده های حسی عظیم کمک می کنند. علاوه بر ادراک، سایر الزامات عملکردی اتومبیل خودمختار که با یادگیری عمیق پشتیبانی می شوند عبارتند از تشخیص صحنه، شناسایی شی (مانع، ماشین، عابر پیاده و پوشش گیاهی)، تشخیص فعالیت، شناسایی فعالیت های انسانی، تشخیص محیط، تشخیص علائم جاده ای، تشخیص چراغ‌های راهنمایی و تشخیص نقاط کور.مدل های معروف یادگیری عمیق مورد استفاده در فناوری اتومبیل های خودمختار برای دستیابی به اهداف فوق در جدول ۳-۱ آورده شده است[۱۵].

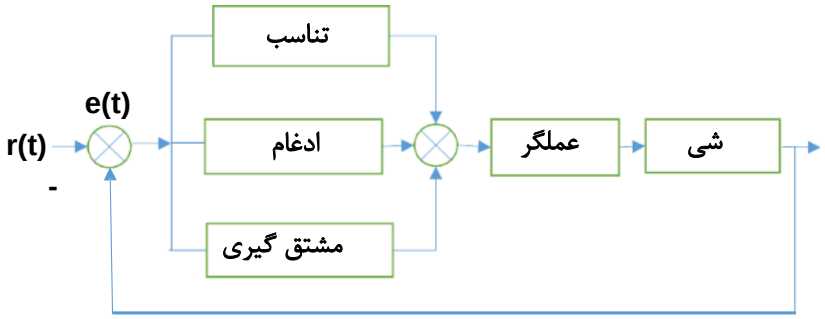
جدول ۳-۱، مدل های معروف یادگیری عمیق مورد استفاده در فناوری اتومبیل های خودمختار[۱۵].

End-to-end Learning	Convolutional Neural Network (CNN)
Deep CNN	Fully Convolutional Network (FCN)
Deep Neural Network (DNN)	Belief Networks
Deep Reinforcement Learning (DRL)	Deep Boltzmann Machines (DBM)
	Deep Autoencoders

[36] Deep Neural Network (DNN)

۷-۱-۶-۱- الگوریتم تناسبی-انتگرالی-مشتقی[۳۷]

این الگوریتم به اختصار PID نامیده می‌شود. الگوریتم PID یا الگوریتم PID بهبود یافته معمولاً توسط سیستم کنترل وسیله نقلیه اتخاذ می شوند. الگوریتم کنترل PID متداول ترین الگوریتم کنترل در فرآیند تولید صنعتی فعلی است. اصل الگوریتم PID در شکل ۸-۱ نشان داده شده است. در این شکل، r(t) سیگنال ورودی، e(t) سیگنال خطای بازخورد، u(t) سیگنال کنترل محاسبه شده توسط الگوریتم کنترل PID و c(t) سیگنال خروجی واقعی فعلی توسط جسم کنترل شده است. با توجه به شکل ۸-۱، الگوریتم PID از سه پارامتر و عملیات ریاضی شامل تناسب، یکپارچه سازی و تمایز برای کنترل هدف استفاده می کند. تفاوت بین هدف کنترل و مقدار واقعیت به عنوان ورودی حلقه بازخورد برای تنظیم موقعیت هدف استفاده می شود[۱۶].



شکل ۲۰-۱، شمای کلی الگوریتم PID[۱۶].

الگوریتم PID کلاسیک دارای مشکلاتی مانند تنظیم پارامتر پیچیده، سازگاری کم و غیره است. به خصوص هنگامی که سیستم انتقال بسیار غیر خطی است و تداخل طولی بسیار پیچیده است، دقت کنترل بسیار کم است. الگوریتم های بهبود یافته PID که در جدیدترین اتومبیل‌های خودران استفاده شده است، می توانند بر اشکالات الگوریتم PID کلاسیک غلبه کنند. کنترل طولی با استفاده از روش کنترل تخصصی که توسط قوانین تخصصی حاصل از تجربه رانندگی تنظیم شده است، به دست آمد. مشخص شد که حتی وقتی سیستم غیرخطی و تداخل طولی پیچیده است، دقت کنترل با استفاده از کنترل طولی بسیار بالا می رود. همراه با کنترل طولی، در صورت تغییر سریع و غیرقابل پیش بینی مدل و سیگنال ورودی هم، سیستم می تواند به طور خودکار جبران کند. بنابراین ماشین خودران می تواند در هر نوع محیطی به طور پایدار و دقیق رانندگی کند[۱۶].

[37] Proportional – Integral - Derivative

۲-۶-۱- فناوری‌های جانبی

فناوری‌های جانبی، فناوری‌هایی هستند که در کنار فناوری‌های اصلی به طراحی سامانه راننده خودکار کمک کرده و عملکرد آن را بهبود می‌بخشند. در ادامه بررسی بخشی از این فناوری‌های جانبی پرداخته شده است.

۱-۲-۶-۱- سیستم نظارت بر راننده[38]

سیستم نظارت بر راننده که با اختصار DMS نامیده می‌شود، سیستمی است که برای نظارت بر هوشیاری راننده طراحی شده است. هم‌چنین، این سیستم توانایی ارزیابی شرایط روحی و جسمی راننده، میزان پلک زدن را نیز داراست. در این سیستم وظیفه خود را به کمک پردازش تصاویر چهره راننده به وسیله دوربین داخل خودرو انجام می‌دهد. در اتومبیل‌های خودران نیز در سطوح مختلف نیز نیاز به توجه راننده به عملکرد اتومبیل در طول مسیر وجود دارد. به همین دلیل، این سیستم در کنار سامانه‌های راننده خودکار مورد استفاده قرار می‌گیرد[۱۷].

۲-۲-۶-۱- سیستم کنترل کروز تطبیقی[39]

سیستم کنترل کروز، کنترلری در جهت طولی است. وظیفه این سیستم، حفظ فاصله با وسیله نقلیه جلویی و ثابت نگه داشتن سرعت اتومبیل در ایمن با استفاده از شتاب و ترمزگیری است. دلیل این امر کاهش مصرف سوخت و هم‌چنین حفظ ایمنی سرنشینان خودرو است. این سیستم با استفاده از اسکنر لیزری نصب شده در جلوی خودرو، موقعیت خودروی جلویی و فاصله آن را تخمین می‌زند. سپس با استفاده از سیستم کنترلی، فرمان مناسب به خودرو اعمال می‌شود. این سیستم یکی از فناوری‌های جانبی مورد استفاده در سامانه‌های خودران خودرو است[۱۸].

۳-۲-۶-۱- فناوری ۵جی[40]

نسل پنجم اینترنت، قابلیت‌های زیادی را به خودروهای خودران اضافه کرده است. با استفاده از این فناوری، امکان ارتباط خودروها با یکدیگر، بروز رسانی داده‌های ناوبری، جزئیات جاده‌ها خصوصا در زمان شرایط جوی نامطلوب و... فراهم می‌گردد. هم‌چنین، قابلیت کنترل خودروهای خودران به وسیله اپراتور خارجی در مواقع ضروری امکان‌پذیر می‌گردد. سرعت انتقال داده‌ها در این فناوری بسیار بالا است.دلیل اهمیت این امر، نیاز به سرعت عملکرد مطلوب سامانه راننده خودمار متناسب با سرعت حرکت خودرو است. هم‌اکنون، مجهزسازی بزرگ‌راه‌ها و جاده‌های اصلی در اروپا به فناوری نسل پنجم اینترنت آغاز شده است[۱۹].

[38] Driver Monitoring System
[39] Adaptive Cruise Control System
[40] 5G

۴-۲-۶-۱- ارتباط خودروها با هم (V2V)

یکی از مهمترین مزایای یک شبکه سریع و بدون تأخیر، توانایی ارتباط اتومبیل های خودمختار با یکدیگر است. این نوع ارتباط یکپارچه به اتومبیل های خودمختار امکان می دهد تا اطلاعات مربوط به موقعیت فعلی، مسیر و خطرات موجود در جاده را مبادله کنند. به عنوان مثال، وقتی دو اتومبیل در یک بزرگراه حرکت می کنند و ماشین جلویی، از طریق سنسورهای موجود در آن، وضعیت خطرناک جاده را تشخیص می دهد، می تواند اطلاعات را به ماشین پشتی انتقال داده تا بتواند ترمز و تنظیم مسیر خود را شروع کند. علاوه بر این ، با داشتن یک شبکه کامل از وسایل نقلیه‌ی بهم پیوسته، ازدحام ترافیک می تواند کاهش یابد زیرا وسایل نقلیه قادر به تصمیم گیری هوشمندانه در مورد مسیر فعلی خود برای حفظ سرعت ثابت جریان خودرو هستند[۷].

۵-۲-۶-۱- ارتباط خودرو با زیرساخت‌ها(V2I)

علاوه بر برقراری ارتباط با سایر وسایل نقلیه، اتومبیل های خودران متصل به شبکه ۵جی همچنین می توانند با عناصر مختلف زیرساختی که جاده‌ها و سایر سیستم‌های حمل و نقل ما را تشکیل می دهند ارتباط برقرار کنند. به عنوان یک مثال ساده، اگرچه یک ماشین خودمختار قادر است شما را از نقطه A به نقطه B برساند، اما چگونه وسیله نقلیه می داند کجا پارک کند؟ با رفت و آمد مداوم اتومبیل، ضروری است که وسایل نقلیه خودران بتوانند مسیر خود را از قبل برای حفظ کارایی مطلوب برنامه ریزی کنند. بنابراین، اطلاعات مربوط به مکان‌های پارکینگ موجود به دست آمده از طریق سنسورهایی که نظارت بر اشغال یک مکان پارک دارند، می تواند از طریق هوا به وسیله نقلیه خودران فرستاده شوند. پس از دریافت این اطلاعات توسط وسیله نقلیه، می توان آن فضا را برای آن اتومبیل خاص در نظر گرفت و این رزرو را می توان از طریق ابر پخش کرد تا چندین اتومبیل بدون راننده برای همان پارکینگ درگیر نشوند[۷].

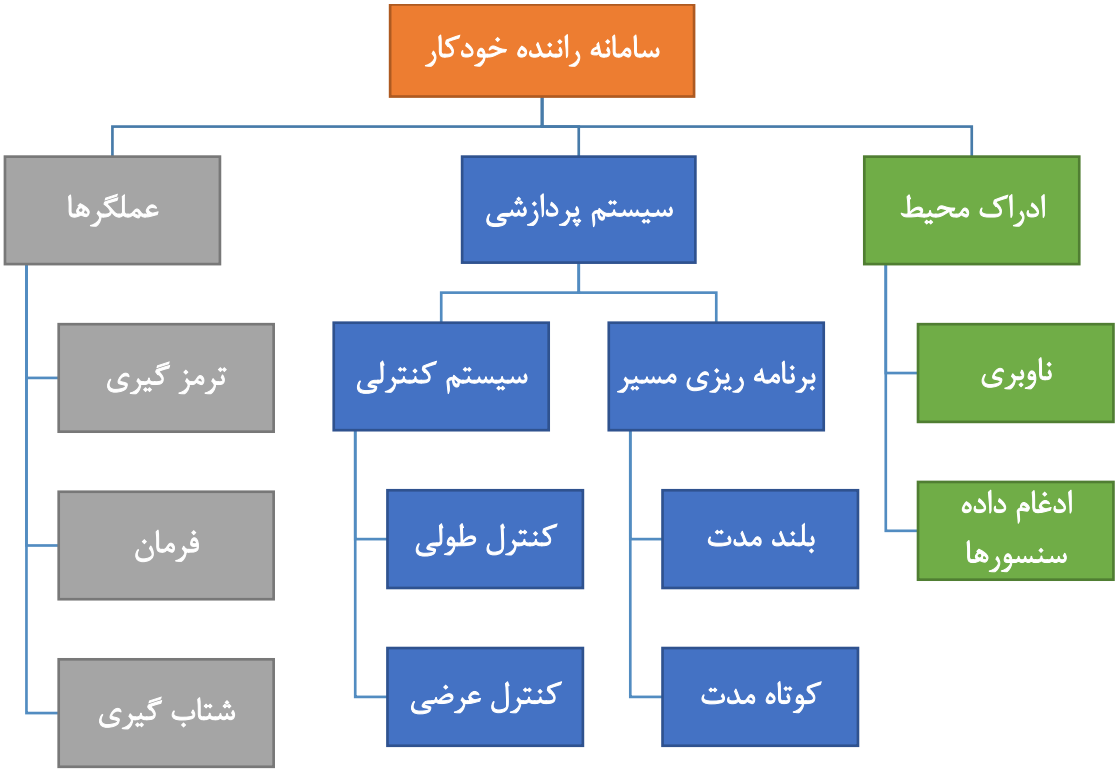
۱-۶-۲-۶- ارتباط خودرو با عابرین(V2P)

در حالی که ارتباط بین وسایل نقلیه و زیرساخت ها مهم است، مهمتر آن است که وسایل نقلیه از عابران پیاده و محل دقیق آنها آگاهی داشته باشند. اکثر ما بدون تلفن‌های هوشمند خود یا نوعی دستگاه آماده برای اینترنت (مثلاً ساعت هوشمند، تبلت، کتابخوان الکترونیکی و...) از خانه خارج نمی شویم و این بدان معناست که تقریباً همیشه به یک شکل به اینترنت متصل هستیم. جالب اینجاست که بسیاری از این دستگاه ها مانند تلفن‌های هوشمند توانایی استفاده از GPS برای تعیین محل دقیق استفاده از دستگاه را دارند. با داشتن یک شبکه 5G،

این اطلاعات را می توان فوراً به یک وسیله نقلیه خودمختار که در نزدیکی آن سفر می کند انتقال داد و آن را از محل عابر پیاده در هر زمان آگاه کرد. با این سطح از اتصال، وسایل نقلیه‌ی بدون راننده قادر خواهند بود به واکنش پویا نسبت به موقعیت عابر پیاده با اقداماتی نظیر ترمز و فرمان اتوماتیک که به نوبه خود باید خیابان های ما را برای پیاده روی بسیار ایمن تر کند[۷].

۷-۱- مسئله‌ها و زیر مسئله‌ها

در این بخش، مسئله‌ها و زیر مسئله‌ها توضیح داده شده است. برای درک بهتر، تقسیم‌بندی مسئله‌ها در شکل ۲۱-۱ آورده شده است.



شکل ۲۱-۱، تقسیم‌بندی مسئله‌ها و زیرمسئله‌ها.

در ادامه به بررسی هر یک از مسائل و بخش‌های مختلف آن پرداخته شده است.

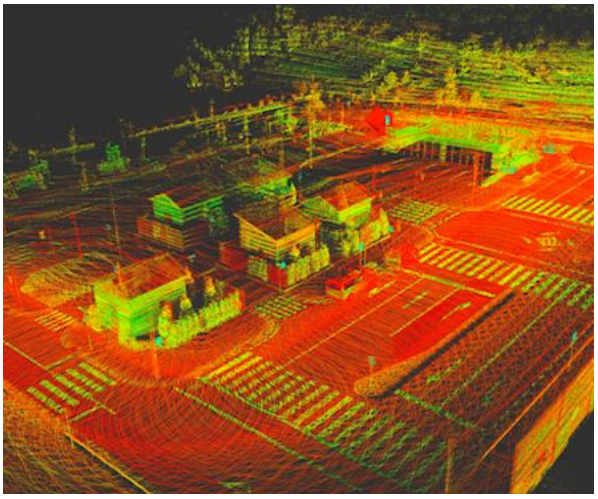
۱-۷-۱- ادراک محیط

درک محیط اطراف خودرو برای یک خودرو حائز اهمیت حیاتی بوده و از آن جهت که خودرو باید بداند که در محیط اطرافش چه می گذرد که بتواند برای اقدامات آینده‌اش به نحوی تصمیم‌گیری کند که دقت و امنیت به حداکثر مقدار و خطا به کمترین مقدار ممکن برسد.

این درک از محیط با استفاده از داده‌هایی که از سنسورهای مختلفی که روی خودروی‌خودران تعبیه شده‌اند و همچنین استفاده از داده‌های ناوبری به دست می‌آید، سنسورهای مختلفی که استفاده شوند شامل سنسور های لیدار[۴۱]، رادار[۴۲]، فاصله‌سنج‌های فراصوتی[۴۳]، دوربین‌های مختلف و ... می باشند که در بخش فناوری‌ها به ذکر توضیحات بیشتر برای هر کدام پرداخته شده است. همچنین علاوه بر خود خودرو و سایر موانع که شامل موانع ثابت[۴۴] مانند: درختان، دیوار، گاردریل، جدول، خودروهای پارک شده و موانع متحرک[۴۵] است مانند: سایر خودروهای در حال حرکت، عابرین پیاده، حیوانات در مسیر و غیره می باشند. یک خودروی خودران باید بتواند علاوه بر این که به خوبی این موانع را تشخیص دهد ، موقعیت خود و آن‌ها در فضای سه بعدی و ابعاد و فاصله های هرکدام را تشخیص دهد[۲۰].

تشخیص موانع تمام کاری نیست که باید برای درک محیط استفاده شود ، خودروی خودران باید با استفاده از اطلاعات جمع شده یک نمایش نقشه‌ای[۴۶] از محیط داشته باشد ، علاوه بر نقشه‌ی دو بعدی که شامل خطوط خیابان، قسمت‌های اشغال شده و نشده ی خیابان می باشد، نیازمند نقشه‌ی سه بعدی که به صورت ابر نقاط[۴۷] می باشد نیز هست. در تصویر شماره ۲۲-۱ ، نقشه ابر نقاط نمایش داده شده است که با استفاده از سنسور لیدار تهیه شده است. در ادامه بخش‌های مختلف ادراک محیط توضیح داده شده است[۲۱].

[41] Lidar
[42] Radar
[43] Ultrasonic Sensors
[44] Static
[45] Dynamic
[46] Map Representation
[47] Point Cloud



شکل ۲۲-۱ ، نقشه ابر نقاط تهیه شده توسط لیدار[۲۱].

۱-۱-۷-۱- ناوبری[۴۸]

ناوبری به دانش راهیابی بین دو نقطه توسط یک سیستم اطلاق می شود، در خودروهای خودران و به صورت کلی در هر سیستمی که نیازمند است از نقطه‌ای به نقطه‌ای دیگر حرکت کند ناوبری مورد نیاز است. در یک خودروی خودران نیز هدف نهایی رفتن از یک نقطه مبدا به یک نقطه مقصد است، در نتیجه علاوه بر موقعیت‌یابی خودرو در یک مقیاس کوچک(درک محیط اطراف) نیازمند تعیین دقیق مختصات خودرو در مقیاس بزرگ هستیم[۲۲].

سیستم ناوبری خودرو خودران بر پایه GPS، سیستمی است که به صورت زنده اطلاعات جغرافیایی خودرو را با اتصال به چندین ماهواره ی GPS گرفته و می تواند مشخصه‌ی طولی و عرضی، سرعت و جهت سیستم را نشان دهد که در مسیریابی و تصمیم‌گیری خودرو نقش اساسی دارد. برای اتصال به ماهواره‌های GPS نیز نیازمند سنسورهای خاصی هستیم که در داخل خودرو خودران باید تعبیه شده باشند. این سنسورها که از نوع سنسورهای خارجی[۴۹] محسوب می‌شوند، در داخل واحد سنجش اینرسی[۵۰] که به اختصار IMU نامیده می شود قرار دارند که اطلاعات گفته شده را به پردازشگر منتقل می‌کنند[۲۳].

[48] Navigation
[49] Exteroceptive
[50] Inertial Measurement Unit

۲-۱-۷-۱- ادغام داده‌های سنسورها

در این قسمت، داده‌های خام تولید شده توسط سنسورها جمع‌آوری می‌شود. پس از این مرحله نرم‌افزار باید بتواند که ویژگی‌های محیط را شناسایی کند و همچنین موقعیت خودرو در سطح جاده را نسبت به سایر اجزا را مشخص کند. روش‌های مختلفی برای ادغام داده‌های سنسورها وجود دارد که می‌توان به روش بیزین[۵۱]، فیلتر کالمن[۵۲][۲۴] و روش‌های احتمالاتی و آماری دیگر اشاره کرد[۲۵].

نکته‌ای که این سیستم مورد اهمیت قرار دارد، سرعت متفاوت ورودی‌های سنسورها محسوب می‌شود. یکی از راهکارها، بافر کردن داده‌ها است تا زمانی که داده همه سنسورها جمع‌اوری شود. اما این کار می تواند موجب شود که داده‌ها زمانی در دسترس قرار گیرند که دیگر زمان تصمیم‌گیری از دست رفته باشد و این موضوع عملکرد سیستم کنترلی را تحت تاثیر خواهد گذاشت. بنابراین می‌توان از روش‌هایی استفاده کرد که در هر لحظه بر اساس داده‌های موجود در آن لحظه تصمیم‌گیری انجام شود[۲۶].

۲-۱-۷- سیستم پردازشی

پس از دریافت داده‌ها از محیط و موقعیت یابی خودرو، این داده‌ها باید به یک سیستم کنترلی داده شوند تا فرامین مورد نیاز برای هدایت خودرو داده شود. به طور کلی دو بخش اصلی کنترلی در خودرو، کنترل سرعت و جهت خودرو محسوب می‌شوند. همچنین دو شتاب عرضی و طولی برای خودرو در نظر گرفته می‌شود که سیستم کنترلی باید فرامین لازم را برای کنترل این دو شتاب و تعیین مقدار بهینه آن‌ها تولید کند.

شاید یکی از مهم‌ترین چالش‌ها در طراحی یک خودروی خودران، طراحی سیستم کنترلی آن باشد. این سیستم باید بتواند داده‌های سنسورها را تفسیر و پردازش کند و سپس تصمیم‌گیری‌های لازم را داشته باشد. همچنین امکان ارتباط میان خودروهای مختلف و همچنین ارتباط با سرورهای ابری را نیز فراهم کند. در ادامه به سیستم‌های مختلف واحد پردازشی خواهیم پرداخت.

۱-۲-۷-۱- سیستم کنترلی

سیستم کنترلی شامل دو بخش سیستم کنترل طولی و سیستم کنترل عرضی است. در ادامه به بررسی هر یک از این بخش‌ها پرداخته شده است.

51 Baysian

52 Kalman filter

۱-۱-۲-۷-۱- سیستم کنترل طولی[۵۳]

چند نوع داده برای کنترل طولی خودرو مورد نیاز است: سرعت و شتاب خودرو، فاصله تا خودروی جلویی و سرعت و شتاب خودروی جلویی. سرعت و شتاب خودروی کنترل شونده را می‌توان با استفاده از سنسورهای OEM به دست آورد. فاصله تا خودروی جلویی را می‌توان با سنسورهایی مانند لیدار، رادار، اولتراسونیک و یا با تکنیک‌هایی از پردازش تصویر به دست آورد. رایج‌ترین سنسور برای این کار، سنسور رادار است. برای اندازه‌گیری سرعت خودروهای مجاور نیز دو راه وجود دارد. یکی از راه‌ها، اندازه گیری آن با استفاده از سنسورهای موجود بر روی خودروی اصلی و اندازه‌گیری نسبی سرعت و شتاب است. اما راه دیگر، برقراری ارتباط با خودروهای مجاور است. باید این نکته را مدنظر قرار داد که ارتباط میان دو خودرو می‌تواند در موارد قابلیت اطمینان پایینی داشته باشد[۲۶].

همانطور که گفته شد، یکی از عملکردهای سیستم کنترلی طولی، دنبال کردن خودروهای دیگر است که در آن باید سرعت و شتاب خودرو درنظر گرفته شود. چالش‌های اصلی این بخش در تنظیم صحیح سرعت در فواصل مختلف و مدلسازی صحیح دینامیک غیر خطی خودرو محسوب می‌شود. درضمن شتاب‌گیری و ترمزگیری باید به نحوی باشد که آرامش سرنشینان حفظ شود و در ضمن مصرف سوخت نیز در حالت بهینه قرار بگیرد[۲۷].

۱-۷-۲-۱-۲- سیستم کنترل عرضی[۵۴]

در این سیستم، کنترل خودرو در راستای عرضی صورت می‌گیرد. بنابراین وظایفی مانند حفظ پایداری خودرو در پیچ‌ها بر عهده این سیستم است. اما اکثر وظایف سیستم کنترلی به صورت ترکیبی از کنترل عرضی و طولی است. به عنوان مثال زمانی که سیستم وظیفه رانندگی میان خطوط را دارد، هر دو سیستم کنترل عرضی و طولی در این موضوع نقش دارند. برای رانندگی میان خطوط، مهم‌ترین مسئله، شناسایی صحیح خطوط است. مهم‌ترین سیستم برای شناسایی خطوط نیز پردازش تصویر محسوب می‌شود[۲۷].

موضوع دیگری که توسط سیستم کنترل عرضی صورت می‌گیرد، تغییر خط است. در محیط‌های شهری، برای تغییر خط باید به محدودیت‌های رانندگی و قوانین آن نیز توجه شود. متدهای مختلفی برای کنترل خودرو در چنین شرایطی تاکنون ارائه شده است که در همه آن‌ها، دقت در داده‌های سنسورها و شناسایی دقیق خودروهای عبوری از اطراف اهمیت دارد و باید به آن توجه شود. در اینجا نیز فیلتر کالمن برای پیش‌بینی وضعیت خودرو می‌تواند مورد استفاده قرار بگیرد[۲۷].

53 Longitudinal Control

54 Lateral control

۲-۲-۷-۱- برنامه ریزی مسیر

برنامه‌ریزی حرکت[۵۵] یا برنامه‌ریزی مسیر[۵۶] یک مسئله محاسباتی است که با توجه به شرایط داخلی و خارجی خودرو، اقدام به برنامه‌ریزی برای محاسبه بهترین مسیر مورد نظر برای رسیدن به مقصد می‌کند. این برنامه ریزی‌ها شامل دو نوع برنامه‌ریزی کوتاه‌مدت[۵۷] و برنامه‌ریزی بلندمدت[۵۸] است[۲۵].

۱-۲-۲-۷-۱- برنامه‌ریزی بلندمدت

برنامه‌ریزی بلندمدت شامل یافتن مسیر بهینه از مبدا به مقصد است ، به طور مثال یافتن بهترین مسیر از شهر الف به شهر ب با توجه به مسائلی همچون ترافیک موجود در مسیرهای مختلف، دقت به نوع خیابان‌ها (یک طرفه یا دو طرفه بودن) شرایط محیطی آن مسیرها (آیا این مسیر مورد نظر باز است و یا به دلایلی بسته است) و سایر شروطی که سیستم باید با توجه اطلاعات آن منطقه و نقشه‌های موجود آن را انجام دهد، حل این مسئله این برنامه‌ریزی می‌تواند بر اساس محدودیت‌های مختلف انجام شود. برای مثال آیا خودرو سعی دارد کوتاه ترین مسیر را طی کند یا این که می‌خواهد سریع‌ترین مسیر را طی کند؟ تمامی این مسائل در زمان رسیدن از مبدا به مقصد، میزان مصرف سوخت، میزان استهلاک خودرو و غیره اثر گذار هستند و در نتیجه سیستم باید قابلیت انتخاب مسیر بهینه را در برنامه ریزی بلند مدت داشته باشد[۲۸].

۲-۲-۲-۷-۱- برنامه‌ریزی کوتاه‌مدت

برنامه‌ریزی کوتاه‌مدت به تصمیماتی که خودرو باید در آن لحظه و یا یا لحظاتی بعد بگیرد اطلاق می شود، برای مثال کاهش یا افزایش سرعت خودرو در موقعیت‌های خاص، سبقت‌گرفتن، تغییر لاین، توقف غیرمنتظره سیستم برای پیشگیری از تصادف و غیره در دسته تصمیمات کوتاه مدت قرار می‌گیرند. این تصمیمات باید به گونه‌ای باشند که بتوانند علاوه بر تامین امنیت خودرو و سرنشینان، امینت سایر افراد و خودروها را تامین کند و در عین حال حرکت خودرو را به در جهت رسیدن به مقصد تنظیم کند. برنامه‌ریزی کوتاه‌مدت تا حد زیادی به درک محیط وابسته است و خودرو در تصمیم گیری از اطلاعات دریافت شده از سنسور های مختلف استفاده می کند[۲۸].

[55] Motion Planning
[56] Path Planning
[57] Short-term Planning
[58] Long-term Planning

۳-۷-۱- عملگرها

پس از واحد کنترلی، سیستم عملگرها وجود دارد که دستورات سیستم کنترلی را اجرا می‌کند. به طور کلی می‌توانیم بگوییم که برای هدایت یک خودرو سه کار اصلی باید انجام شود: فرمان‌دهی، ترمزگیری و شتاب‌گیری. زمانی که راننده‌ای خودرو را هدایت نمی‌کند، ابزارهای هدایت مکانیکی مانند فرمان و پدال‌ها جای خود را باید به سیستمی دهند که بتوانند دستورات دیجیتال را اجرا کنند. اصطلاحا به این سیستم رانندگی سیمی [۵۹] گفته می‌شود. در ادامه به معرفی سه زیر سیستم این سیستم خواهیم پرداخت[۲۹-۳۱].

۱-۷-۳-۱- سیستم ترمز

در خودروهای مدرن، سیستم ترمز عملکردی فراتر از کاهش سرعت خودرو را بر عهده دارد. به نحوی که سیستم ترمز باید با سیستم فرمان‌دهی نیز در ارتباط باشد و بسته به حالت‌های مختلف، ترمزگیری شرایط متفاوتی باید داشته باشد. همچنین سیستم ترمز در کنترل پایداری خودرو در پیچ‌ها به کمک دیگر سیستم‌ها می‌آید. پس در پیاده سازی سیستم ترمز باید به سیستم‌های جانبی که برای ایمنی و پایداری خودرو مورد استفاده قرار می‌گیرند نیز توجه داشت و آن‌ها را پیاده سازی کرد.

۱-۷-۳-۲- سیستم فرمان

شاید مهم‌ترین عضو در راه‌اندازی یک سیستم خودران، سیستم فرمان محسوب شود. برای فرمان‌دهی به خودرو در هر سرعتی، به یک گشتاور خاصی نیاز است. برای تامین این گشتاور می‌توان از یک موتور DC به همراه یک گیربکس استفاده کرد. همچنین نوع فرمان‌دهی در سرعت‌های بالا و پایین با یکدیگر تفاوت دارند. سیستم‌های فرمان‌دهی به یک سیستم فیدبک نیز مجهز هستند که زاویه دقیق فرمان را پس از وارد کردن گشتاور فرمان‌دهی به سیستم اطلاع می‌دهد.

در طراحی این سیستم موارد مهمی هستند که می‌توانند در نحوه عملکرد سیستم اثرگذار باشند. مهم‌ترین عامل، سرعت است که اگر سرعت افزایش پیدا کند، مقدار گشتاور مورد نیاز برای فرمان‌دهی و حداکثر زاویه فرمان‌دهی تغییر خواهد کرد. همچنین شرایط محیطی نیز بر روی سیستم فرمان اثر گذار هستند. به عنوان مثال زمانی که جاده برفی یا بارانی باشد، ممکن است در مواردی خودرو دچار بیش‌فرمانی شود که هدایت خودرو را سخت خواهد کرد. یا در مواردی باد شدید می‌تواند سیستم فرمان‌دهی را تحت تاثیر خود قرار دهد. بنابراین فیدبک سیستم در این موارد مهم است و خودرو باید نسبت به این موارد پایداری خود را حفظ کند.

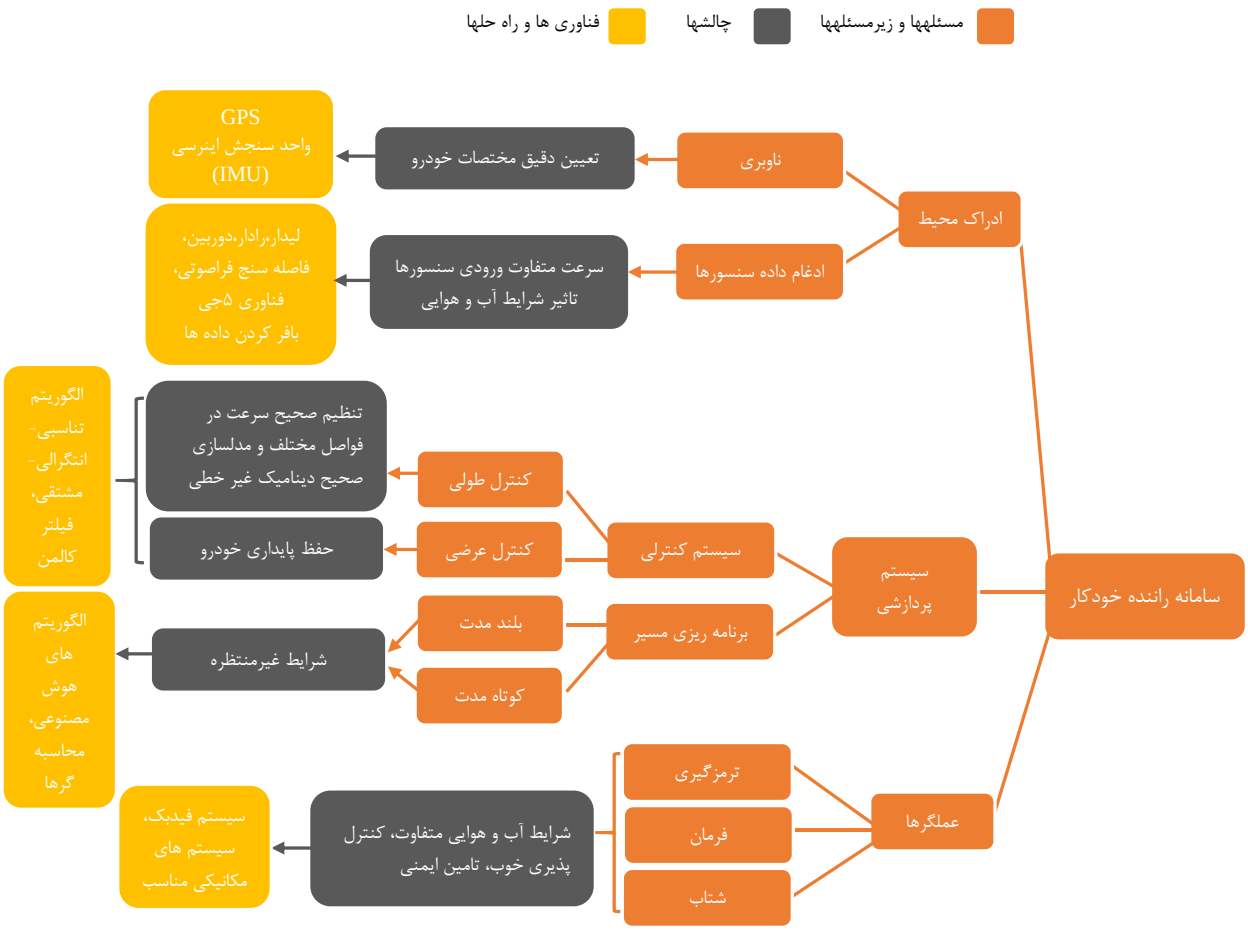
---

[۵۹] Drive-by-Wire

۱-۷-۳-۳- سیستم شتاب‌گیری

در حال حاضر نیز بسیاری از خودروهای غیر خودران نیز از اتصالات غیر مکانیکی برای شتاب‌دهی به خودرو استفاده می‌کنند. این سیستم به نسبت دو سیستم دیگر از پیچیدگی کمتری برخوردار است. با استفاده از یک موتور می‌توان دریچه سوخت را باز و بسته کرد و بسته به میزان فرمان داده شده، خودرو شتاب‌گیری خواهد کرد.

به طور کلی ارتباط و شبکه‌ی بین زیرمسائل مربوط به پروژه، چالش ها و فناوری‌ها را می توان به صورت نمودار شکل ۲۳-۱ نمایش داد.



شکل ۲۳-۱، مسائل، زیر مسائل و چالش‌های سامانه راننده خودکار.

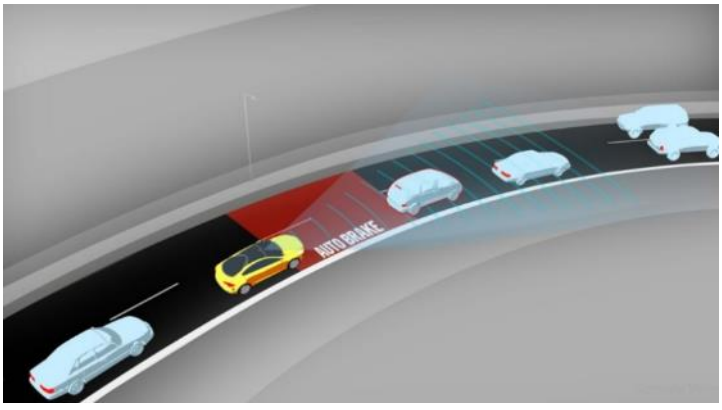
۸-۱- ویژگی‌ها و قابلیت‌های سامانه راننده خودکار

در این بخش به معرفی ویژگی هایی که یک خودرو خودران می‌تواند دارا باشد، پرداخته می شود و در توضیح آن ویژگی به فناوری مربوطه نیز اشاره می گردد.

۱-۸-۱- ترمز هوشمند

برای سیستم ترمزهوشمند رفتارهای متفاوتی را در خودروها می توان یافت. در اینجا به دو نمونه ترمز هوشمند که شرکت های سوبارو[60] و ولوو[61] از ان استفاده می کنند اشاره می شود. در خودروهای شرکت سوبارو، از دو نوع دوربین سیاه و سفید استفاده می شود که مانند چشم انسان عمل کرده و سرعت و فاصله خودرو جلویی را تشخیص می دهند. این دوربین ها در بالای شیشه جلویی خودرو تعبیه می شوند، و هر ۰.۱ ثانیه عمل اسکن کردن را برای یافتن تناقض بین رنگ های صفحه پس زمینه و و سطوح عمودی، انجام می دهند. نرم افزاری که در این سیستم به کار می رود، باید توانایی تشخیص نوع تصویری که توسط دوربین گرفته شده است را داشته باشد، و در صورت وجود مانع در مقابل خودرو، دستورهای لازم را صادر کند. [۳۲]

شرکت ولوو، از لیدار برای سیستم ترمز هوشمند خود استفده می کند. از انجایی که لیدار، بهترین عملکرد خود را در بازه فواصل کم دارد، این شرکت دوربین هایی را در بالای شیشه جلویی خودرو و رادارهایی روی سپر قرار داده است که در مواقعی که سرعت خودرو زیاد است، فواصل دورتر نیز قابل رؤیت باشند و در صورت وجود مانع، سیستم ترمز هوشمندبه موقع وارد عمل شود. رادارها می توانند تا چند صد متر را ببینند، اما نمی توانند تشخیص دهند که در حال رؤیت چه چیزی می باشند. در این حالت دوربین ها وظیفه تشخیص انچه رادارها می‌بینند را بر عهده می‌گیرند. [۳۲]. شکل ۲۴-۱ اشاره به همین ویژگی دارد.



شکل۲۴-۱، ویژگی ترمز هوشمند[۳۳].

[60] Subaru
[61] Volvo

۲-۸-۱- حفظ خودرو در بین خطوط

سیستم تشخیص خطوط[۶۲] با استفاده از تکنیک های پردازش تصویر،برای مثال تبدیل هاف[۶۳]، خط کشی های خیابان را تشخیص می دهد. تصویرها توسط دوربین هایی که در قسمت وسط و بالای شیشه جلویی قرار دارند، گرفته می شوند و در صورت انحراف از خط کشی ها، سیستم کنترل فرمان، خودرو را به بین خطوط هدایت می کند[۳۴].

ترکیب این سیستم با کروز کنترل تطبیقی می‌تواند شرایطی را ایجاد کند که برای مدت زمان طولانی، خودرو بدون دخالت راننده در یک مسیر مشخص به حرکت خود ادامه دهد[۳۵] . شکل ۲۵-۱ به این قابلیت اشاره می کند.

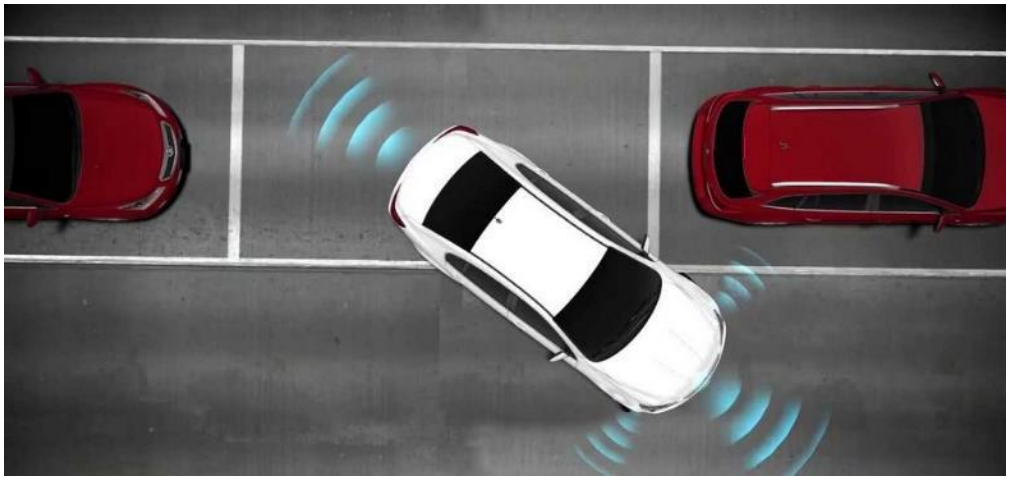
شکل ۲۵-۱ قابلیت حفظ خودرو در بین خطوط[۳۳].

۳-۸-۱- پارک خودکار۶۴

سیستم‌های پارک اتوماتیک، راه‌های گوناگونی برای درک اشیای اطراف خود دارند. برخی از آنها دارای سنسورهایی هستند که در اطراف سپرهای عقب و جلوی خودرو هستند و به‌عنوان گیرنده و فرستنده کار می‌کنند. این سنسورها، سیگنال‌هایی را به اشیای اطراف اتومبیل می‌فرستند و دوباره آنها را دریافت می‌کنند. سپس کامپیوتر خودرو با محاسبه‌ی مقدار زمان صرف‌شده برای بازگشت سیگنال‌ها، موقعیت اشیای پیرامون را محاسبه می‌کند. سیستم‌های دیگری موجود بوده که دارای دوربین‌هایی هستند که بر روی سپرهای خودرو قرار گرفته‌اند یا برای

[62] Lane centering
[63] Hough transform
[64] Auto park

کشف اشیا از رادار استفاده می‌کنند. البته نتیجه‌ی نهایی در همه‌ی آنها یکسان است. اتومبیل می‌تواند اتومبیل‌های پارک شده‌ی دیگر و فضای مناسب برای پارک و فاصله‌ی مناسب تا لبه‌ی جدول را تشخیص بدهد و سپس به‌درستی پارک شود. سنسورهای جانبی پارک خودکار، معمولا از سنسورهای الکترومغناطیسی یا اولتراسونیک (فراصوتی) برای تشخیص فاصله با اشیای اطراف خودرو استفاده می‌کنند و وظیفه اندازه گیری جای پارک را دارند تا به راننده اطلاع دهند که ایا خودرو در این فضا امکان پارک دارد یا خیر. پروسه پارک به وسیله سنسورها و سیستم کنترل فرمان انجام می گیرد[۳۳]. شکل ۲۶-۱ نمونه پارک زاویه ای را نشان می دهد.



شکل۲۶-۱، نمونه پارک زاویه ای خودکار[۳۳].

۴-۸-۱- تشخیص تابلوهای ایمنی

این سیستم با اسکن محیط اطراف، می‌تواند تابلوهای ایمنی کنار جاده را تشخیص داده و پس از تطبیق، از طریق یک صفحه نمایش کوچک، راننده را مطلع سازد. نحوه عملکرد این سیستم بدین شکل است که دوربین‌های نصب شده به محض برخورد با الگوهای از پیش‌تعیین‌شده اقدام به ترجمه تابلو می‌کنند و پس از تطبیق‌دادن تصاویر ثبت‌شده، این تابلو را در معرض نمایش راننده قرار می‌دهند. این ویژگی در خودروهای نیمه‌خودران به راننده این قابلیت را می‌بخشد که در صورت عدم مشاهده علائم ایمنی، از عوارض پیش‌رو و یا محدودیت‌های موجود اطلاع یابد. همچنین در کنار نمایش تابلوها قابلیت هشدار محدودیت سرعت به راننده را نیز داراست و به محض برخورد با تابلوهای هشدار سرعت، راننده را مطلع خواهد ساخت[۳۳].

تشخیص تابلوهای راهنمایی از طریق پردازش تصویر صورت می گیرد که این پروسه به سه قسمت اصلی تقسیم میشود: [۳۶]

- پیش پردازش[۶۵]: تبدیل RGB به HSV برای دقیق تر دیدن رنگ ها
- بازرسی[۶۶]: اسکن کردن تصاویر برای یافتن نواحی مختلف تصویر
- شناخت[۶۷]: شناخت علائم با استفاده از مدل های مختلف یادگیری ماشین برای مثال SVM[68]

در شکل ۲۷-۱ به نمونه ای از قابلیت هشدار به راننده هنگام برخورد با تابلوی محدودیت سرعت، اشاره شده است.

شکل۲۷-۱، قابلیت خواندن علائم رانندگی[۲]

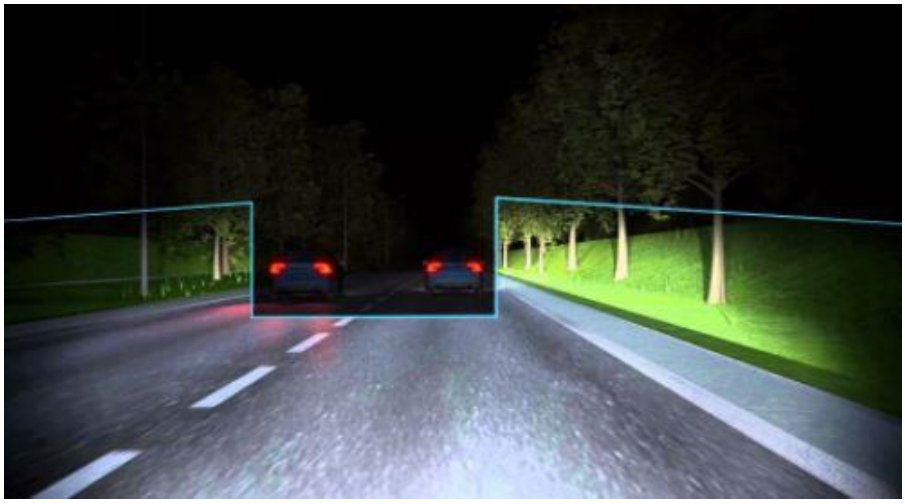
۵-۸-۱- چراغ هوشمند

تامین نور کافی برای افزایش دید راننده در شب یکی از مهم‌ترین وظایف چراغ‌های جلوی خودرو برشمرده می‌شود اما تابش مستقیم آن در چشمان راننده خودروهای روبه‌رویی ممکن است باعث بروز حوادثی شود. اغلب در جاده‌های کم‌نور رانندگان تمایل زیادی به قرار دادن چراغ‌ها در وضعیت بالا دارند اما هر چند این مسئله می‌تواند ایمنی و ضریب خطر حرکت خودرو را کاهش دهد اما خودروهای رانندگان روبه‌رویی با چالش دید مواجه خواهند شد. برای جلوگیری از حوادث احتمالی ناشی از این مسئله، چراغ‌های هوشمندی طراحی شده‌اند که در صورت مواجه‌شدن با یک خودرو، جهت خود را به شکلی منحرف می‌کنند که دید راننده روبه‌رویی را تحت تاثیر قرار نداده و نور کافی را در اختیار راننده بگذارند[۳۳].

برای مثال لامپ های دیجیتال جدید مرسدس بنز بسیار دقیق تر از هدلامپ های ال ای دی قبلی آن هستند و هر یک از آنها دارای بیش از یک میلیون پیکسل بوده و قادر به کنترل محل و زاویه نوردهی در جلوی خودرو

[65] Pre-processing
[66] Detection
[67] Recognition
[68] Support vector machine

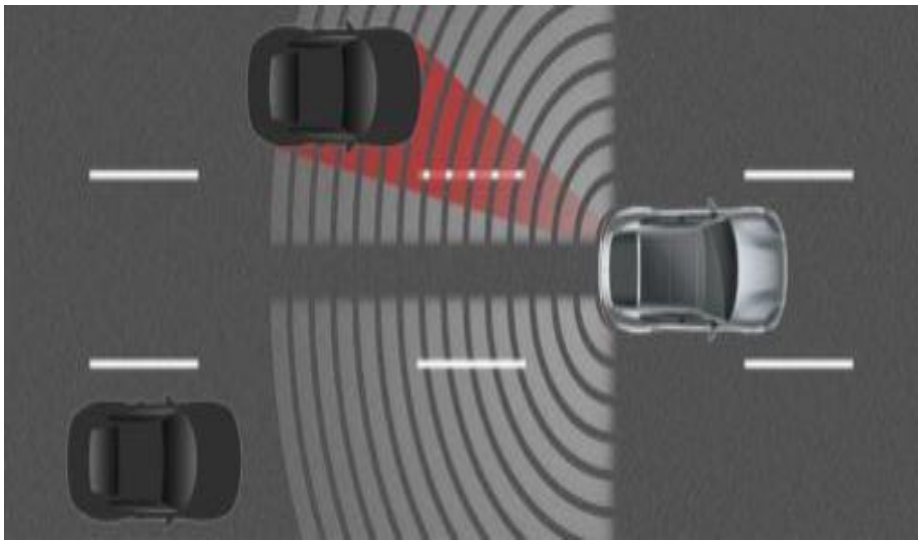
است.لامپ های یادشده مجهز به حسگرهایی هستند که میزان درخشش هر پیکسل از نور لامپ را مشخص می کنند و بنابراین افراد پیاده یا رانندگان خودروهای عبوری بر اثر شدت نور تابیده شده از لامپ های یادشده دچار مشکل نمی شوند [۳۶] . در شکل ۲۸-۱، به نمونه ای از تنظیم خودکار چراغ های خودرو اشاره شده که نور کمتری به خودروهای جلویی می رسد تا از اذیت شدن دیگر رانندگان جلوگیری شود.



شکل۲۸-۱ ، نمونه ای از چراغ هوشمند[۳۳].

۶-۸-۱- هشدار تغییر خط

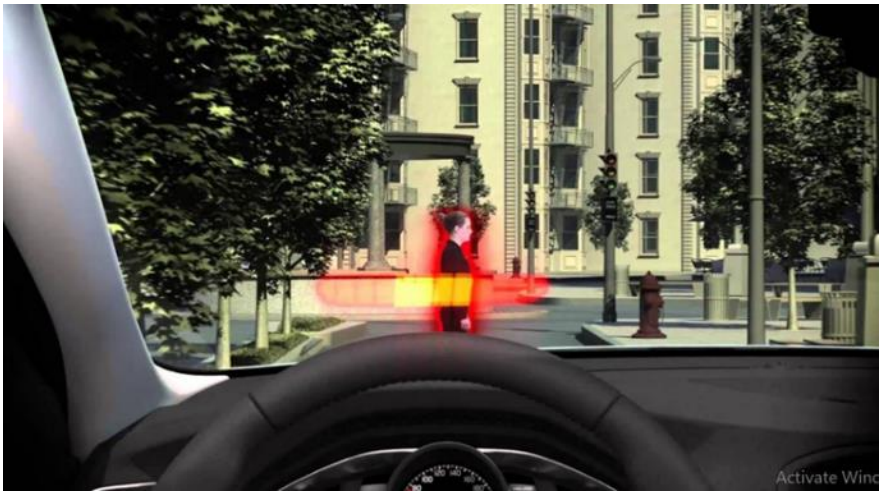
بسیاری از حوادث زمانی اتفاق می‌افتند که راننده بدون بررسی جوانب خودرو، اقدام به تعویض خط می‌کند وبه دلیل عدم مشاهده خودروهای اطراف منجر به حادثه می‌شود. این سیستم به کمک سنسورهای نصب شده در دو گوشه انتهایی خودرو، راننده را از وجود سایر خودروها و احتمال برخورد در صورت تغییر خط مطلع می‌سازد. چراغ‌های هشدار نصب شده روی آینه‌های جانبی در صورت وجود احتمال برخورد در حین تغییر خط، به رنگ نارنجی درمی‌آیند تا راننده با مشاهده آنها از وجود خودرویی در نقطه کور آینه، مطلع شده و از بروز حادثه جلوگیری کند. این سیستم یکی از بزرگ‌ترین معایب آینه‌های جانبی را حل کرده و نقاط کور را نیز به خوبی پوشش داده است. شکل ۲۹-۱ این قابلیت را نشان می دهد[۳۵].



شکل۲۹-۱، قابلیت هشدار تغییرخط[۳۵].

۷-۸-۱- تشخیص عابرین پیاده

تشخیص و شناخت عابران پیاده به وسیله لیدار سه بعدی و الگوریتم طبقه بندی که برای مثال با SVM اموزش داده شده باشد، استفاده می شود.خودرو با مقایسه موقعیت و جهت سرعت عابرپیاده با اطلاعاتی که از کناره و جدول خیابان، از طریق دوربین های خود دریافت کرده، میتواند تشخیص دهد که ایا عابرپیاده مشاهده شده، در وسط خیابان است یا در نزدیکی جدول خیابان قرار دارد و با استفاده از این اطلاعات، اقدامات لازم را انجام دهد.[۳۵] در شکل ۳۰-۱ میتوان این ویژگی را دید.



شکل ۳۰-۱، ویژگی تشخیص عابرپیاده[۳۵].

۹-۱- بازار هدف

در شاخه بازار هدف، مشتریان بالقوه و بالفعل ، بازار هدف داخلی و خارجی، حجم بازار و ارائه می‌شود. با توجه به اینکه موضوع سامانه راننده خودکار در سطح ۵ استاندارد SAE در ایران و در جهان هنوز در مراحل تحقیق و توسعه است و به طور فراگیر تولید نشده است، موضوعات مرتبط در رده‌های پائین‌تر سامانه راننده خودکار در این گزارش تشریح می‌شود.

۱-۹-۱- مشتریان بالقوه و بالفعل :

بخش‌هایی از سامانه راننده خودکار در خودروهای موجود درکشور پیاده‌سازی شده است. به عنوان مثال سیستم پارک خودکار، مکان یابی هوشمند و پیشنهاد مسیر، سامانه کنترل سرعت(کروز کنترل)، سامانه هشدار تغییر لاین، هشدار دنده عقب و برخی سامانه‌های دیگر امروزه در خودروهای شخصی به کار گرفته شده اند. بنابراین عموم مردم یکی از مهم ترین مشتریان سامانه راننده خودکار هستند. علاوه بر مصرف شخصی، سامانه راننده خودکار در سیستم حمل و نقل عمومی نیز نقش مهم و کاربردی می‌تواند ایفا کند و یکی از مشتریان بالقوه دیگر این سامانه سازمان‌های حمل ونقل عمومی و خصوصی هستند. به عنوان مثال، در گام اول پیاده سازی سامانه راننده خودکار در تاکسی های فرودگاهی و تاکسی‌های اینترنتی می‌تواند بخشی از نیازهای این سازمان‌ها را برطرف کند و امنیت و سودآوری بهتری برای آن ها به همراه داشته باشد. گام بعدی کاربرد و توسعه مشتریان این سامانه استفاده از سامانه راننده خودکار برای اتوبوس‌های شهری و بین شهری و کاربرد در ناوگان حمل ونقل سنگین و نیمه سنگین جاده ای‌است. این موارد در برخی از شرکت‌های خارجی در حال طی مراحل تست و ارزیابی است.

۲-۹-۱- بازار هدف داخلی و خارجی سامانه راننده خودکار:

بازار OEM: این بازار را خودروسازان کشور تشکیل می دهند که شرکتهای تأمین کننده آنها اقدام به تأمین قطعات مورد نیاز خود از قطعهسازان مینمایند. شرکتهای ساپکو (تأمین کننده شرکت ایران خودرو) شرکت سازهگستر سایپا (تأمین کننده شرکت سایپا) شرکت اپکو (تأمین شرکت ایران خودرو دیزل) از شرکت اصلی فعال در این بازار به شمار می‌آیند.

بازار OES :این بازار خدمات پس از فروش خودروها میباشد که وابسته به شرکتهای خودروساز است. سازمان خدمات پس از فروش ایران خودرو (ایساکو) ، شرکت سایپا یدک، شرکت گسترش خدمات پارس خودرو، مزدا یدک از شرکتهای این گروه محسوب میشوند.

بازار AM :این بازار قطعات خودرو شامل کلیه لوازم یدکی فروش های مختلف در سطح کشور میباشد که به صورت آزاد (بدون ارتباط خاص با خودرو سازان) اقدام به فروش قطعات خودرو مینمایند.

برای کسب اطلاعات از بازار داخلی با کمی تحلیل درمیابیم که این سامانه صرفا روی خودروهای لوکس داخلی نصب خواهد شد. با توجه به همین مسئله در ابتدا مشتریان این خودروها شرکت‌های سفارشی‌ساز زیر مجموعه دوخودرو ساز بزرگ کشور هستند. شرکت آپکوی ایران خودرو و شرکت آپشن سایپا یدک با سفارش گیری از مشتریان خود، این سامانه را برروی خودروها نصب می‌کنند.

با ادامه پیشروی این سامانه ها در کشور در مرحله بعد نوبت خودروسازان می‌رسد، که این سیستم را روی خودروهای خود پیاده‌سازی کنند. با توجه به این که سیستم راننده خودکار یا کمک راننده در کشور ما یک آپشن لوکس حساب می‌شود، با مراجعه به منابع تیراژ خودروهای لوکس را استخراج می‌کنیم. تولید مجموعه محصولات دنا در سال ۱۳۹۸ به تعداد ۳۲۴۱۹ عدد بوده است[۳۷].

در مورد بازارهای خارجی باید گفت این سامانه برای خودروسازهایی مناسب است که هنوز به دنبال تحقیق و توسعه خودروهای خودران نرفته‌اند. این خودروسازان، خودروساز درجه دوم به حساب می‌آیند. به عنوان نمونه داچیا رومانی جزء این دسته محسوب می‌شود که در محدوده خودروهای اقتصادی کار می‌کند. در مورد بازارها خارجی باید اضافه کرد که، رقابت بسیار شدید بوده و یا باید کیفیت بسیار بالایی ارائه داد و یا این که قیمت بسیار کمی برای سامانه در نظر گرفت تا بتوان جای صادرات این سامانه را باز کرد.

در صورتی که سامانه ها خودروهای خودران آزمایش خودرا پس داده و به سطوح بالاتری از خودرانی دست یابند، مشتری‌های سازمانی به علاقه‌مندان این محصولات اضافه می‌شوند. از جمله آن‌ها شرکت‌های بزرگ تاکسی اینترنتی از جمله اسنپ و تپسی هستند. با توجه به این شرکت اوبر در حال استفاده و تست خودروهای خودران است، دور از ذهن نیست که در کشور ما هم این مهم روی دهد.

۱۰-۱- بازیگران صنعت:

در شاخه بازیگران صنعت، اشخاص، شرکت‌ها و سازمان‌های موثر، اطلاعات تولید کنندگان و وارد کنندگان بیان می‌گردد. بازیگران صنعت این خودروهای خودران را می‌توان به سه دسته بزرگ تقسیم کرد. دسته اول بازیگرانی که وظیفه ایجاد بستر و زیر ساخت های این سامانه را دارند. دسته دوم را بازیگرانی تشکیل میدهند که وظیفه اجرایی داشته و باید تولید انبوه این خودروها را به عهده بگیرند. دسته سوم به بخش خدماتی و بیمه باز میگردد.

۱-۱۰-۱- بازیگران مرحله زیرساخت

در این دسته سازمان‌هایی نقش دارند که باید زمینه را برای تولید و بکار گیری این محصول فراهم کنند. این بازیگران خود به چند زیر دسته تقسیم می‌شوند:

۱-۱۰-۱-۱- زیردسته حقوقی

برای نخستین بار در خردادماه سال ۱۳۹۶ نخستین خودروی خودران در ایران با موفّقیّت آزمایش شد و در همان ماه رئیس پلیس راهور ناجا تردّد خودروهای خودران را به دلیل فقدان قوانین مربوط و مشخّص نبودن مسئول در تصادفات احتمالی این خودروها، ممنوع اعلام کرد[۳۸]. بنابراین از موانع بسیار بزرگ سر راه خودروهای خودران، خلاهای قانونی آن میباشد که باید توسط کارشناسان خبره ترافیک، حقوق، خودرو و پلیس راهنمایی و رانندگی حل شود. بنابراین سازمان‌های دخیل در این مسئله شامل سازمان ترافیک و شهرداری، پلیس راهنمایی و رانندگی به علاوه دانشگاه‌های حقوق، برق، مکانیک و غیره می‌باشد.

۱-۱۰-۱-۲- زیردسته علمی-تکنولوژی

اگرچه صنعت خودرو با پیشرفت های استثنایی تکنولوژیکی و نوآورانه روبه روا ست، دورانا نتظار برای تحقق عملی آن به طرز دردناکی آهسته است. روند پذیرش نوآوری های جدید به طور معمول زمان بر بوده (نیاز به زمان قابل توجه ۴ تا ۶ سال دارد) و نیازمند صرف هزینه برای آزمایش، تعیین کیفیت و نیز صدور گواهینامه برای محصول جدید است. علاوه بر این، تفکر سنتی رایج در صنایع خودرویی، به ویژه تفکر غالب تولیدکنندگان و تأمین کنندگان اصلی، در مواجهه با یک فرایند تحقیق و توسعه غیرانعطاف پذیر است و با ایجاد مانع برای تفکر فراقالبی به نوبه خود سبب کندتر شدن نوآوری و پذیرش محصولات جدید می شود. بنابراین استارت آپ ها نقشی کلیدی در ارائه سریع تر و ارزان تر فناوری های باز به بازار دارند. به منظور تحققا ینا مر، تولیدکنندگان اصلی پلتفرم خود را توسعه داده یا برنامه های شتاب دهنده ای همچون سرمایه گذاری ژنرال موتور، استارت آپ گاراژ شرکت بی ام دبلیو، شتاب دهنده تکنولوژی جگوار لند روور، و سرمایه گذاری ولوو اجرا می‌کنند، جایی که شراکت/سرمایه گذاری مشترک با سرمایه حداقل 100 هزار دلار تا میلیاردها دلار مشاهده می شود[۳۹].

متاسفانه در کشور تا الان فقط یک استارتاپ موفق در زمینه خودروهای خودران موجود است و در این زمینه خلاهای بسیاری حس می‌شود و باید دولت در این زمینه اقداماتی مشوقانه انجام دهد.

۱-۱۰-۱-۳- زیردسته آموزشی

به هر حال زمانی که یک تکنولوژی جدید و بروز ایجاد شده و میخواهد به مرحله تولید برسد، نیاز است آموزش‌هایی تا افراد و شرکت‌ها را برای مواجهه با تکنولوژی جدید آماده کند. در این بخش دانشگاه‌ها و نمایندگان شرکت‌های دانش‌بنیان نقش اساسی را ایفا می‌کنند.

۲-۱۰-۱- تولید کنندگان

دسته دوم متعلق به تولیدکنندگان خودرو و قطعه‌سازان می‌باشد. سازمان‌های این دسته با استفاده از تکنولوژی فراهم شده توسط استارتاپ‌ها اقدام به تولید انبوه خودرو خودران و قطعات می‌کنند. مهم ترین اعضای این دسته شامل شرکت ایرانخودرو، سایپا، پارس‌خودرو و کرمان‌موتور می‌باشند.

۳-۱۰-۱- خدمات پس از فروش

دسته سوم بازیگران با اهمیت، به خدمات پس از فروش و بیمه مربوط می‌شود. تمامی شرکت‌های بیمه کننده و تمامی تعمیرکارها و غیره جزو این دسته قرار می‌گیرند. در بخش خدمات پس از فروش نیاز است که آموزش هایی به افراد داده شود تا بتوانند خدمات مورد نیاز را ارائه دهند.

سامانه راننده خودکار به طور جزئی و در برخی از خودروهای خارجی و در برخی موارد در خودروهای جدید داخلی پیاده‌سازی شده است. برای بررسی آمار و اطلاعات موجود از این خودروها نگاهی به آمار تولید و واردات خودرو در کشور و برخی آمارهای جهانی می‌اندازیم؛ همچنین برآوردهای اقتصادی و ظرفیت‌های موجود در کشور مطابق با منبع [۴۰] بیان می‌شود. درضمن این آمار، اطلاعات واردات و تولیدکنندگان خودرو و زیرسیستم‌های هوشمندسازی نیز ارائه می‌گردد.

۱۱-۱- آمار واردات، صادرات و ظرفیت تولید

خودروهای خودران سطوح مختلفی دارند که سطح ۵ نشان دهنده خودران بودن کامل خودرو بدون نیاز به هرگونه دخالت انسان می باشد. از انجایی که این نوع خودروها هنوز در مرحله ساخت ازمایشی هستند و به تولید انبوه نرسیده اند، در این بخش به اطلاعاتی در مورد امار صادرات و واردات محصولات مشابه از جمله خودروهای خودران سطوح پایین‌تر بسنده می‌شود.

۱-۱۱-۱- خودروهای هیبریدی وارداتی

قابلیت هایی مانند سیستم هوشمند تنظیم شدت جریان ورودی هوا در جلو پنجره، کروز کنترلر و سیستم کنترلینگ تعبیه شده در فرمان ، آینه الکتروکرومیک ، چراغ های جلو به سیستم تطبیقی با خودرو های جلویی و سیستم گردش در سر پیچ ها، صفحه نمایش چند منظوره ۱۰.۲۵ اینچی لمسی با قابلیت Gesture Control برای کنترل شرایط داخلی و خارجی خودرو – مسیر یاب – سیستم مالتی مدیا و تماس های ورودی، قابلیت استارت خودرو با ریموت ، سیستم پارک هوشمند تنها با استفاده از ریموت کنترل خودرو ، دوربین دید عقب ، سنسور های مجاورتی سراسری به همراه سیستم کمک پارک ، شارژر وایرلس گوشی های هوشمند ، سیستم هوشمند رانندگی تطبیقی متناسب با شرایط و نوع جاده ، سیستم انتقال قدرت Steptronic ، قابلیت انتخاب

حالت رانندگی میان سه حال SPORT, COMFORT , ECO PRO ، دوربین ۳۶۰ درجه که یک تصویر سه بعدی از نمای بالا را به نمایش می گذارد ، دوربین دید در شب که می تواند هر موجود زنده ای را در تاریکی مطلق تشخیص داده و به راننده اطلاع دهد ، سیستم مانیتورینگ فشار باد تایر ها ، سیستم هشدار نقاط کور، در جدیدترین و لوکس ترین خودروهای وارداتی، باعث شد تا این خودروها جزو محصولات مشابه خودرو خودران، به شمار آیند[۴۱].

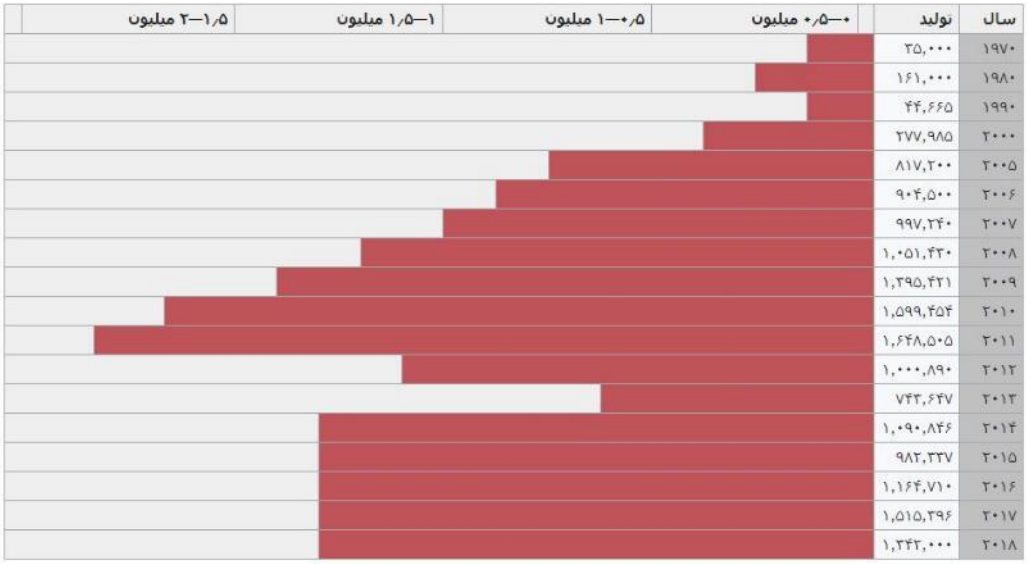
مبلغی که برای وارد کردن این خودروها در سال ۱۳۹۷ ثبت شده است، بالغ بر ۵۷ میلیون دلار می باشد[۴۲].

نمونه دیگری از محصولات مشابه را می‌توان خودروهای برقی و بدون نیاز به نیروی محرکه خارجی، در نظر گرفت. دلیل شباهت این خودروها به خودروهای خودران، برقی بودن انها است. زیرا صنعت خودرو در عین حالی که در مسیر خودران شدن می باشد، در مسیر برقی شدن و استفاده کمتر از سوخت های فسیلی نیز می باشد.

هزینه واردات این خودروهای برقی در سال ۱۳۹۷، حدود ۵ میلیون دلار ثبت شده است[۴۲].

۲-۱۱-۱- ظرفیت تولید

برای تعیین ظرفیت تولید، در بهترین حالت می‌توان ظرفیت تولید تمام خودروسازان کشور را باهم جمع نمود. واضح است که تمام خودروسازان، مشتریان بالقوه ایده خودرو خودران نیستند و شماری از آن‌ها به دلایل متعددی از جمله هزینه ساخت و یا ریسک شکست این نوع خودروها، از ساخت انها در ابتدای امر خودداری خواهند کرد.



شکل۳۱-۱، امار تولید سالیانه کشور ایران[۴۳]

با توجه به شکل ۳۱-۱ مشاهده می شود که در سال ۱۳۹۷، تعداد خودروهای تولیدی، ۱۳۴۲۰۰۰ بوده است.

جدول ۴-۱ ، تعداد اشتغال و ظرفیت واحدهای فعال تولید خودرو سواری به تفکیک استان تا پایان مهر ۱۳۹۶

استان	تعداد واحد	اشتغال (نفر)	ظرفیت اسمی (دستگاه)
آذربایجان شرقی	۲	۱۳۱۸	۲۶۵,۰۰۰
اصفهان	۳	۱۶۳۷	۱۵۳,۰۳۰
تهران	۳	۳۸۷۴۳	۱,۰۷۸,۵۰۰
خراسان رضوی	۲	۱۳۶۰	۱۱۵,۰۰۰
سمنان	۱	۱۵۰	۱۰۰,۰۰۰
فارس	۲	۴۱۲	۳۱,۰۰۰
کرمان	۲	۴۲۲۴	۶۶,۷۵۰
لرستان	۱	۱۲۲	۲۰,۰۰۰
مازندران	۱	۳۰۰	۳۰,۰۰۰
مرکزی	۲	۸۶۰	۷۰,۰۰۰
منطقه آزاد ارس	۱	۲۵	۵,۰۰۰
کل	۲۰	۴۹۱۵۱	۱,۹۳۴,۲۸۰

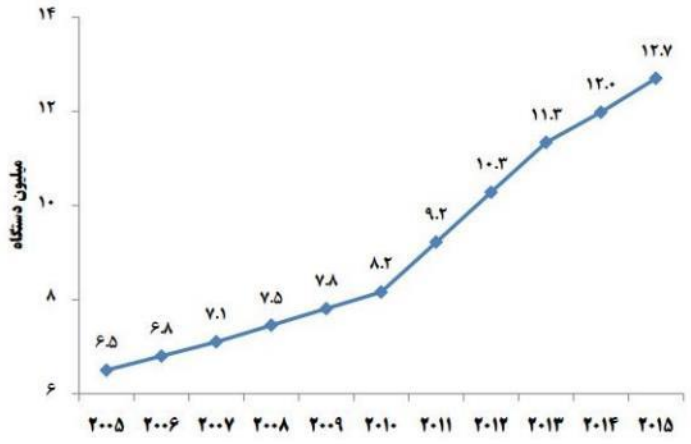
جدول ۵-۱ ، روند تولید و ظرفیت خودرو سواری در کشور(هزار دستگاه)

عنوان	۱۳۸۵	۱۳۸۶	۱۳۸۷	۱۳۸۸	۱۳۸۹	۱۳۹۰	۱۳۹۱	۱۳۹۲	۱۳۹۳	۱۳۹۴	۱۳۹۵	هفت‌ماهه ۱۳۹۶
آمار تولید	۹۲۳.۸	۷۹۵.۴	۸۸۱.۸	۱۱۹۳.۲	۱۳۵۹.۶	۱۴۲۱.۱	۷۸۸.۴	۶۲۴.۸	۹۵۸.۳	۸۹۲.۶	۱۲۵۵.۲	۷۷۱.۹
ظرفیت	۶۹۱	۸۸۹	۹۴۴	۱۳۴۴	۱۳۷۴	۱۸۵۴	۱۹۰۰	۱۹۰۰	۱۹۰۰	۱۹۰۰	۲۰۴۸	۲۰۴۸

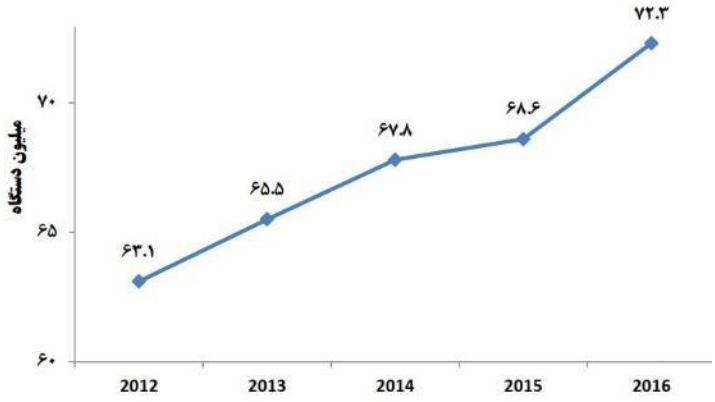
جدول ۶-۱ ، روند تولید جهانی خودرو طی سال های ۲۰۱۲ تا ۲۰۱۶ (میلیون دستگاه)

کشور	رتبه	۲۰۱۲	۲۰۱۳	۲۰۱۴	۲۰۱۵	۲۰۱۶	سهم از کل دنیا	درصد تغییر
کل (۵۳ کشور)	–	۶۳.۱	۶۵.۵	۶۷.۸	۶۸.۶	۷۲.۳	۱۰۰	۵
چین	۱	۱۵.۵	۱۸.۱	۱۹.۹	۲۱.۱	۲۴.۴	۳۳.۸	۱۶
ژاپن	۲	۸.۶	۸.۲	۸.۳	۷.۸	۷.۹	۱۰.۹	۱
آلمان	۳	۵.۴	۵.۴	۵.۶	۵.۷	۵.۷	۷.۹	۱
آمریکا	۴	۴.۱	۴.۴	۴.۳	۴.۲	۳.۹	۵.۴	۵–
کره جنوبی	۵	۴.۲	۴.۱	۴.۱	۴.۱	۳.۹	۵.۳	۷–
هند	۶	۳.۳	۳.۲	۳.۲	۳.۴	۳.۷	۵.۱	۸
اسپانیا	۷	۱.۵	۱.۸	۱.۹	۲.۲	۲.۴	۳.۳	۶
مکزیک	۸	۱.۸	۱.۸	۱.۹	۲.۰	۲.۰	۲.۸	۱
برزیل	۹	۲.۶	۲.۷	۲.۵	۲.۰	۱.۸	۲.۵	۱۲–
انگلستان	۱۰	۱.۵	۱.۵	۱.۵	۱.۶	۱.۷	۲.۴	۹
فرانسه	۱۱	۱.۷	۱.۵	۱.۵	۱.۶	۱.۶	۲.۳	۵
چک	۱۲	۱.۲	۱.۱	۱.۲	۱.۲	۱.۳	۱.۹	۸
ایران	۱۳	۰.۹	۰.۶	۰.۹	۰.۹	۱.۲	۱.۶	۳۴
روسیه	۱۴	۲.۰	۱.۹	۱.۷	۱.۲	۱.۱	۱.۶	۸–
اسلواکی	۱۵	۰.۹	۱.۰	۱.۰	۱.۰	۱.۰	۱.۴	۰.۱
اندونزی	۱۶	۰.۷	۰.۹	۱.۰	۰.۸	۱.۰	۱.۳	۱۷
ترکیه	۱۷	۰.۶	۰.۶	۰.۷	۰.۸	۱.۰	۱.۳	۲۰
تایلند	۱۸	۰.۹	۱.۱	۰.۷	۰.۸	۰.۸	۱.۱	۶
کانادا	۱۹	۱.۰	۱.۰	۰.۹	۰.۹	۰.۸	۱.۱	۱۰–
ایتالیا	۲۰	۰.۴	۰.۴	۰.۴	۰.۷	۰.۷	۱.۰	۸

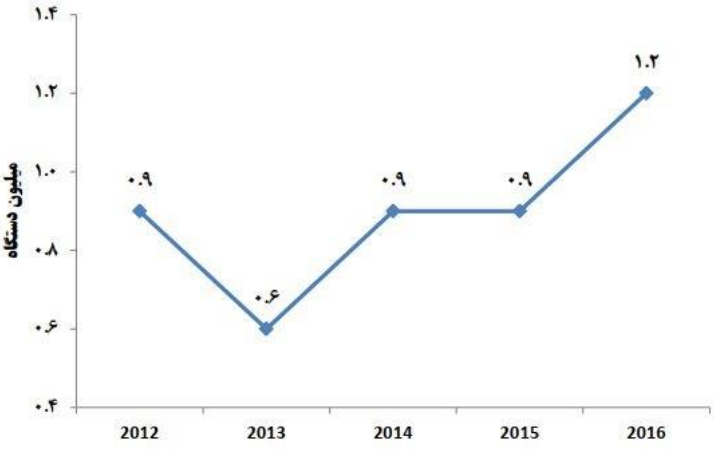
ماخذ: انجمن جهانی خودرو oica



شکل ۳۲-۱، روند تعداد خودروهای سواری تولید شده در جهان



شکل ۳۳-۱، روند تعداد خودروهای سواری تولید شده در ایران



شکل ۳۴-۱، روند تعداد خودروهای سواری مورد استفاده در ایران

جدول ۷-۱، روند ارزش صادرات خودرو سواری - (میلیارد دلار)

کشور	رتبه	۲۰۱۲	۲۰۱۳	۲۰۱۴	۲۰۱۵	۲۰۱۶
جهان	-	۶۵۱	۶۷۸	۷۰۹	۶۷۷	۷۰۰
آلمان	۱	۱۴۷	۱۴۹	۱۶۰	۱۵۳	۱۵۲
ژاپن	۲	۹۷	۹۲	۸۹	۸۶	۹۲
آمریکا	۳	۵۵	۵۷	۶۲	۵۵	۵۴
کانادا	۴	۴۷	۴۵	۴۵	۴۵	۴۹
انگلستان	۵	۳۴	۳۸	۴۲	۳۹	۴۱
کره جنوبی	۶	۴۲	۴۴	۴۵	۴۲	۳۷
اسپانیا	۷	۲۵	۲۹	۳۲	۳۳	۳۶
مکزیک	۸	۲۹	۳۲	۳۲	۳۳	۳۱
بلژیک	۹	۲۷	۳۲	۳۰	۲۸	۳۰
چک	۱۰	۱۵	۱۵	۱۸	۱۷	۱۹
ایران	۹۹	۰.۰۰۴	۰.۰۰۸	۰.۰۴۰	۰.۰۰۳	۰.۰۰۳

ماخذ: Trademap.org

جدول ۸-۱ ، روند ارزش واردات خودرو سواری - (میلیارد دلار)

کشور	رتبه	۲۰۱۲	۲۰۱۳	۲۰۱۴	۲۰۱۵	۲۰۱۶
جهان	-	۶۵۶	۶۸۳	۷۱۳	۶۸۶	۷۰۷
آمریکا	۱	۱۴۹	۱۵۵	۱۵۶	۱۶۹	۱۷۳
آلمان	۲	۴۲	۴۱	۴۷	۴۶	۵۱
انگلستان	۳	۳۵	۳۹	۴۶	۴۹	۴۶
چین	۴	۴۵	۴۷	۶۰	۴۴	۴۴
فرانسه	۵	۳۰	۳۰	۳۱	۲۹	۳۲
بلژیک	۶	۲۶	۳۰	۲۷	۲۸	۳۱
ایتالیا	۷	۲۰	۲۰	۲۳	۲۳	۲۷
کانادا	۸	۲۶	۲۶	۲۷	۲۶	۲۶
اسپانیا	۹	۱۰	۱۱	۱۵	۱۶	۱۸
استرالیا	۱۰	۱۸	۱۸	۱۶	۱۵	۱۶
ایران	۳۴	۲.۳	۳.۱	۳۶	۲۶	۳.۲

ماخذ: Trademap.org

جدول ۹-۱ ، ارزش واردات خودرو سواری طی سال های ۱۳۸۹ تا ۱۳۹۶ (میلیون دلار)

محصول	۱۳۸۹	۱۳۹۰	۱۳۹۱	۱۳۹۲	۱۳۹۳	۱۳۹۴	۱۳۹۵	هفت ماهه ۱۳۹۶
خودرو سواری	۸۹۴	۸۳۰	۱۰۹۹	۱۶۲۲	۲۱۴۶	۱۲۲۷	۲۰۰۸	۱۲۹۲

جدول ۱۰-۱، ارزش صادرات خودرو سواری طی سال های ۱۳۸۹ تا ۱۳۹۶(میلیون دلار)

محصول	۱۳۸۹	۱۳۹۰	۱۳۹۱	۱۳۹۲	۱۳۹۳	۱۳۹۴	۱۳۹۵	هفت ماهه ۱۳۹۶
خودرو سواری	۳۸۹	۱۲۹	۳۲۳	۷۱	۱۰۱	۱۴	۴۰	۸.۲

جدول ۱۱-۱ ، ارزش صادرات خودروهای سواری طی سال های ۱۳۸۹ تا ۱۳۹۶ به تفکیک کشور (میلیون اختیار)

کشور	۱۳۸۹	۱۳۹۰	۱۳۹۱	۱۳۹۲	۱۳۹۳	۱۳۹۴	۱۳۹۵
عراق	۳۸۲.۵	۱۲۶.۰	۳۱۷.۲	۳۹.۹	۸۴.۳	۱۱.۶	۳۲.۱
الجزایر						۰.۸	۲.۲
سوریه	۰.۲				۲.۱	۰.۰۳	۱.۶
لبنان	۰.۰۲		۰.۰۱				۱.۳
ترکمنستان	۰.۸	۰.۱	۰.۱	۰.۷	۰.۹	۰.۱	۱.۱
چین	۰.۰۳	۰.۱	۰.۰۱	۰.۳	۰.۱	۰.۰۳	۰.۳
یمن	۰.۰۲				۰.۴		۰.۲
ساحل عاج	۰.۰۱		۰.۰۲	۰.۰۲	۰.۲		۰.۱
آذربایجان	۰.۶	۰.۱	۲.۵	۱۵.۱	۶.۲		۰.۱
آلمان	۰.۴	۰.۶	۰.۲	۰.۱	۰.۱	۰.۲	۰.۱
جیبوتی						۰.۰	۰.۱
کره جنوبی	۰.۱	۰.۲	۰.۱	۰.۱	۰.۰۵	۰.۱	۰.۱
ترکیه	۱.۷		۰.۰۵	۰.۰۱	۰.۰۲	۰.۱	۰.۱
قزاقستان		۰.۱	۰.۳	۰.۳	۰.۰۴	۰.۰۲	۰.۰۵
ارمنستان	۰.۱	۰.۲		۰.۱	۰.۰۱	۰.۰۴	۰.۰۵
سایر	۲	۲	۳	۱۴	۶	۱	۰.۲
کل	۳۸۹	۱۲۹	۳۲۳	۷۱	۱۰۱	۱۴	۴۰

کشور	هفت ماهه ۱۳۹۶
عراق	۷.۰۹
آذربایجان	۰.۲۰
گینه	۰.۱۶
یمن	۰.۱۵
فرانسه	۰.۱۲
رومانی	۰.۰۹
کره جنوبی	۰.۰۷
ایتالیا	۰.۰۶
چین	۰.۰۵
امارات	۰.۰۴
المان	۰.۰۳
ارمنستان	۰.۰۳
چک	۰.۰۳
سوریه	۰.۰۲
افغانستان	۰.۰۱
سایر	۰.۰۲
کل	۸.۱۶

جدول ۱۲-۱، واردات خودرو سواری طی سال های ۱۳۸۹ الی ۱۳۹۶ به تفکیک کشور (میلیون دلار)

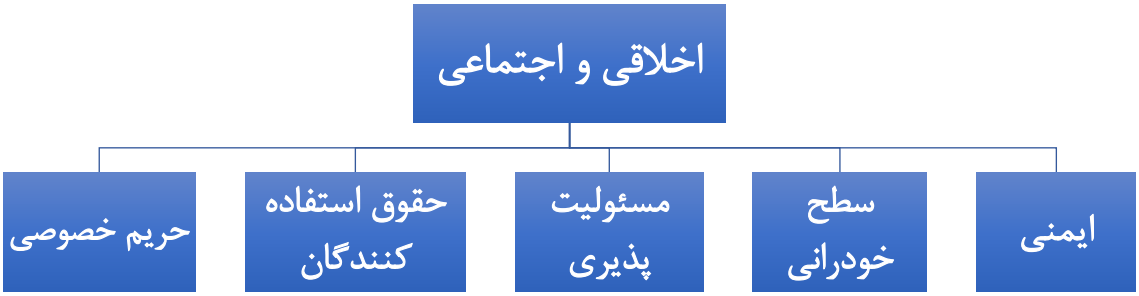
کشور	۱۳۸۹	۱۳۹۰	۱۳۹۱	۱۳۹۲	۱۳۹۳	۱۳۹۴	۱۳۹۵
امارات	۲۸۸	۳۴۳	۷۱۶	۱۳۵۸	۱۸۵۴	۸۸۷	۱۲۳۸
کره جنوبی	۳۶۴	۳۴۷	۲۳۰	۲۸	۲۴	۱۸۷	۳۱۴
آلمان	۲۷	۳۱	۲۲	۴	۴۲	۳۲	۱۶۴
ترکیه	۳	۱	۵	۹	۱۷	۱۶	۶۴
ژاپن	۵۴	۲۱	۲۱	۷	۱	۱۴	۴۹
کویت	۵۸	۱۷	۴۰	۵۷	۴۵	۱۷	۴۰
اسپانیا	۰.۱	۰	۰	۰.۰۳	۰	۳	۳۱
عمان	۶۰	۴۰	۲۳	۴	۳	۱۲	۲۹
گرجستان	۰	۰	۰	۱	۵	۵	۱۸
سوئد	۰	۰	۰	۰	۰	۰	۱۴
چین	۰.۲	۰.۴	۱۷	۱۳۶	۱۳۸	۳۸	۱۳
روسیه	۰.۰۵	۰	۰	۰	۰.۰۲	۶	۱۰
آمریکا	۰.۱	۰	۰	۰.۰۴	۰.۴	۱	۸
اتریش	۰.۱	۰.۱	۰	۰.۰۴	۰	۰.۲	۳
فرانسه	۲۱	۱۰	۸	۰.۰۳	۱	۰.۱	۱
سایر	۳۰	۲۱	۲۴	۱۷	۱۷	۷	۱۰
کل	۸۹۴	۸۳۰	۱۰۹۹	۱۶۳۲	۲۱۴۶	۱۲۲۷	۲۰۰۸

کشور	هفت ماهه ۱۳۹۶
امارات	۸۹۹.۷
المان	۸۹.۳
کره جنوبی	۵۶.۷
فرانسه	۵۴.۹
چین	۴۳.۱
ترکیه	۳۴.۵
کویت	۲۱.۸
گرجستان	۱۲.۵
ژاپن	۱۱.۰
اسپانیا	۹.۵
ایتالیا	۷.۴
روسیه	۶.۰
عمان	۵.۰
اکوادور	۴.۹
فیلیپین	۴.۰
سایر	۳۱.۷
کل	۱۲۹۱.۸

۱-۱۲- پیامد های ساخت خودروی خودران

در صورت به ثمر رسیدن محصول نهایی یعنی یک خودروی خودران، پیامدها و تاثیرات زیادی می تواند در زمینه های اقتصادی، اخلاقی و اجتماعی، سیاسی، فنی و زیست محیطی داشته باشد.

۱-۱۲-۱- اخلاقی و اجتماعی



شکل ۳۵-۱ ، پیامدهای اخلاقی و اجتماعی خودروهای خودران

۱-۱۲-۱-۱- ایمنی

هنگامی که از SDV به طور گسترده ای استفاده شود، این سوال پیش می آید که آیا خودروهای غیرخودکار به دلایل ایمنی باید ممنوع شوند یا خیر. از آنجا که عرضه SDV به صورت فراگیر نبوده است، این مسئله تاکنون سوال مهمی نبوده، اما مقامات ایمنی جاده ای ایالات متحده و انگلیس اعلام کرده اند که این امر در ۵۰ سال آینده اجتناب ناپذیر است. در همین حال، گروه هایی مانند Humans Against Autonomous Vehicles (HAAV) به شدت با SDV مخالفت کرده اند زیرا آن ها بر این عقیده اند که این خودرو ها از ایمنی کافی برای رانندگی برخوردار نیستند[۴۴].

۱-۱۲-۱-۲- سطح خودرانی خودرو

در خودرو های خودران ما انتخاب و توانایی تصمیم گیری خود را در ناوبری خودرو از دست می دهیم. به طور کلی SDV ها طوری برنامه ریزی شده اند که محدودیت سرعت و قوانین جاده را رعایت می کنند، بنابراین آزادی رانندگی از بین می رود. اخیراً در کالیفرنیا ، یک زن باردار زایمان کرد و مجبور شد سریعاً به بیمارستان منتقل شود، اما به دلیل تنظیم محدودیت سرعت SDV که تقریبا منجر به تاخیر در به دنیا آوردن یک نوزاد شد، مشکلاتی را ایجاد کرد[۴۴].

۱-۱۲-۱-۳- مسئولیت پذیری

این نگرانی وجود دارد که SDV ها اراده آزاد و مسئولیت اخلاقی ما را تهدید می کنند، زیرا به دلیل التزام زیاد بر الگوریتم ها و هوش مصنوعی، روند نگران کننده ای برای تغییر مسئولیت توسط دارندگان وسایل نقلیه مستقل وجود دارد و ترجیح می دهند با رانندگی در حالت خودران، خود را از قبول مسئولیت راحت کنند. در طی تحقیقاتی که در سال های اخیر انجام شده است، بسیاری از رانندگان گفته اند که آنها هیچ گونه مسئولیتی در قبال اقدامات SDV خود ندارند و متعاقباً نباید در قبال حوادث ناشی از وسیله نقلیه آنها پاسخگو یا مسئول باشند[۴۴].

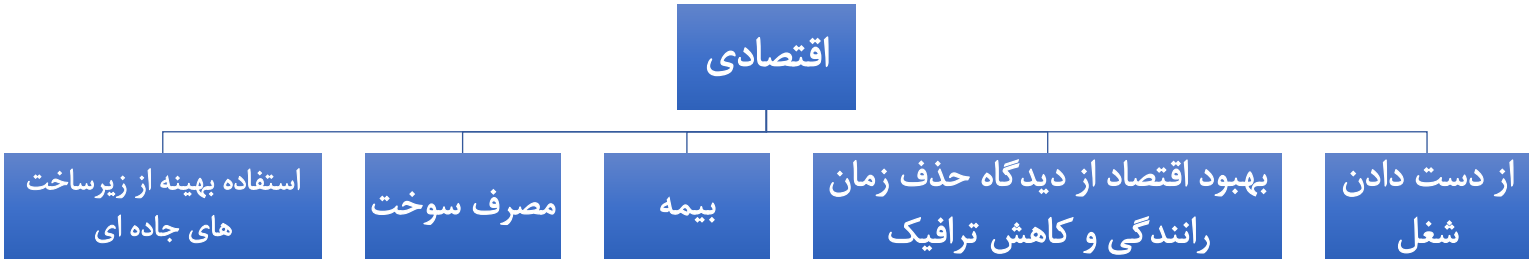
۱-۱۲-۱-۴- حقوق استفاده کنندگان

SDV ها این امکان را فراهم می سازند تا افراد بیشتری مانند افراد مسن ، معلول و نابینا از خودرو استفاده کنند. ولی هنوز خودرو های خودران در سطح ۳ می باشند و برای استفاده این افراد از خودرو های خودران، نیازمند سطح ۴ و ۵ در خودرو های خودران هستیم[۴۴].

۱-۱۲-۱-۵- حریم خصوصی

حریم خصوصی سیاست گذاران را مجبور کرده است روش هایی را برای اطمینان از حفظ حریم خصوصی و امنیت داده ها شناسایی کنند و اگر SDV قانون را نقض کند، آیا پلیس مجاز به دسترسی به آن است یا خیر. نهادهای نظارتی تشخیص داده اند که برای محافظت از اطلاعات شخصی فرد، باید سطح بالایی از رمزگذاری، ناشناس ماندن و تجمیع آن ها اعمال شود. بسیاری از تولیدکنندگان اتومبیل در حال ترویج رویکردی هستند که تلاش می کند داده ها را در داخل ماشین پردازش کند، نه اینکه به ارائه دهندگان خدمات مختلف یا اشخاص ثالث منتقل شود. اما تولیدکنندگان رعایت آن را از نظر فنی چالش برانگیز می‌دانند[۴۴].

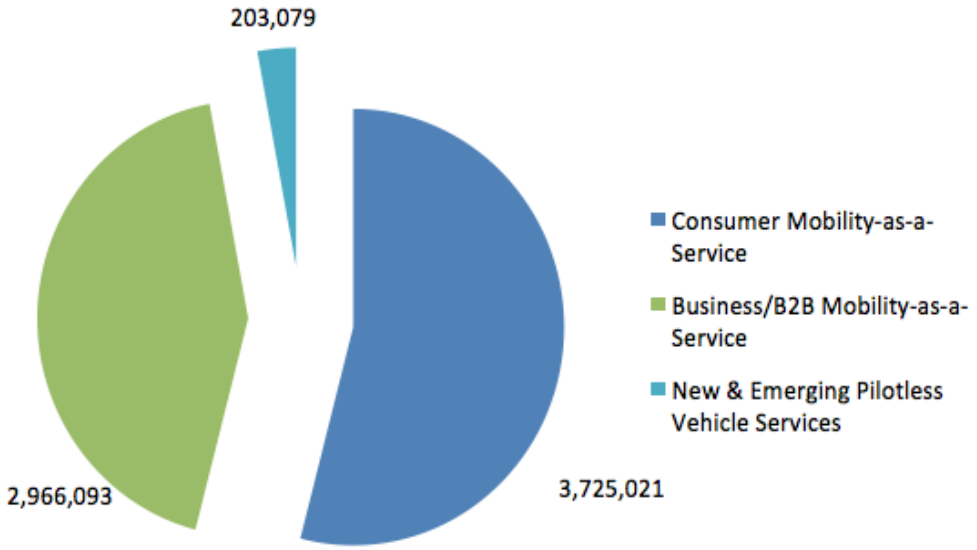
۲-۱۲-۱- اقتصادی



شکل ۳۶-۱ ، پیامدهای اقتصادی خودروهای خودران

وقتی رانندگی خودران به واقعیت تبدیل شود، چه تأثیراتی بر اقتصاد و جامعه ما خواهد گذاشت؟

طبق برآورد شرکت Intel و Strategy Analytics که در ژوئن ۲۰۱۷ اعلام شد ، تأثیرات اقتصادی خودروهای مستقل در سال ۲۰۵۰ بالغ بر ۷ تریلیون دلار خواهد بود. مبلغ دلار نشان دهنده یک ارزش تازه ایجاد شده یا اقتصاد مسافر[1] جدید است که براساس این فرض محاسبه می شود که وسایل نقلیه سطح ۵ کاملاً خودکار تا سال ۲۰۵۰ در جاده ها خواهند بود.



شکل ۳۷-۱ درآمد خدمات جهانی حاصل از رانندگی خودران در سال ۲۰۵۰ (بر حسب میلیون دلار آمریکا)[۴۵].

آنها همچنین فرض کردند که مصرف کنندگان و مشاغل به جای داشتن اتومبیل از پیشنهادات Mobility as a Service (MaaS) استفاده خواهند کرد و کسانی که برای کار با اتومبیل رفت و آمد می کردند مسافر می شوند و زمان رفت و آمد را به کار دیگری می گذرانند. بدین ترتیب، این برآورد دامنه بسیار گسترده ای از تأثیرات بالقوه را منعکس می کند.

در مقاله Economic Effects of Automated Vehicles [۴۶] اثرات اقتصادی وسایل نقلیه کاملاً خودکار در ایالات متحده به گونه ای تخمین زده شده است که از نظر سود اقتصادی خالص برای جامعه اندازه گیری می شود، تا با ادغام یافته های مطالعات متعدد و با استفاده از داده های اضافی، یک دیدگاه جامع ارائه دهد که می توان خلاصه ای از تأثیرات اقتصادی گفته شده در این مقاله را در جدول زیر مشاهده کرد.

جدول ۱۳-۱ خلاصه ای از تأثیرات اقتصادی (در کل صنعت و اقتصاد)[۴۶]

Industry	Size of Industry ($ billions)	Dollar Change in Industry ($ billions)	Percent Change in Industry (%)	$/Capita ($)
Insurance	180	-108	-60%	339
Freight Transportation	604	+100	+17%	313
Land Development	931	+45	+5%	142
Automotive	570	+42	+7%	132
Personal Transportation	86	-27	-31%	83
Electronics & Software	203	+26	+13%	83
Auto Repair	58	-15	-26%	47
Digital Media	42	+14	+33%	44
Oil & Gas	284	+14	+5%	44
Medical	1067	-12	-1%	36
Construction & Infrastruc	169	-8	-4%	24
Traffic Police	10	-5	-50%	16
Legal Profession	277	-3	-1%	10
Industry-specific Total	4480	418	9%	1312

Economy-wide Total	Dollar Change in Industry ($ billions)	$/Capita ($)
Productivity	448	1404
Collisions (*)	488	1530
Economy-wide Total	936	2934
Collision Value Overlap (*)	-138	-432
Overall Total	1217	3814

علاوه بر نکات گفته شده، شرکت ها به دنبال بهبود بهره وری حمل و نقل هستند و در نهایت با عملی شدن سیستم های خودرو خودران بسیاری از مشاغل مربوط به خودرو و رانندگی از بین می روند. این امر باعث می شود

در شرکت های مربوط به حمل و نقل بار، بار بیشتری را با هزینه کمتری حمل کنند و در نتیجه منافع اقتصادی افزایش می یابد.
تحقق رانندگی خودکار به طور چشمگیری تعداد تصادفات رانندگی ناشی از خطاهای انسانی را کاهش می دهد، که مسئول بیش از ۹۰٪ تصادفات رانندگی است و نیازی به بیمه نامه های اتومبیل را از بین می برد. تقاضا برای خدمات تعمیر خودرو، خدمات پزشکی و پلیس راهنمایی و رانندگی نیز کاهش می یابد.
در مورد اثرات کل اقتصادی، وقتی سیستم های خودرا به صورت گسترده یاده شود، ازدحام ترافیک در جاده ها به طور قابل توجهی کاهش می یابد. بهره وری بهبود یافته در استفاده از جاده ها به معنای افزایش ظرفیت جاده ها است و تخمین زده می شود که این امر باعث کاهش تقریباً ۴۸۸ میلیارد دلاری صرفه جویی در هزینه از طریق کاهش آسیب ها و مرگ های ناشی از برخورد اتومبیل شود. نفوذ وسایل نقلیه خودکار باعث کاهش ساعات غیرمولد آمریکایی ها می شود که ۲.۷ میلیارد ساعت رانندگی را برای کار خود سپری می کنند و به آنها امکان می دهد تا از زمان آزاد شده برای کار یا انجام فعالیت های دیگر استفاده کنند و در نتیجه ۴۴۸ میلیارد دلار بهره وری راننده را افزایش می دهند. تأثیرات کلی اقتصادی وسایل نقلیه خودران، که هم اثرات خاص صنعت و هم اثرات اقتصادی را با هم ترکیب می کنند، به بیش از ۱.۲ تریلیون دلار می رسد[۴۶].

۳-۱۲-۱- محیط زیست

فناوری خودرو های خودران می تواند از طریق الگوریتم هایی با به حداقل رساندن انرژی با رانندگی کارآمد تر، مسیریابی بهینه، شتاب مناسب، سرعت هوشمند (مانند تناسب سرعت نسبت به وسایل نقلیه اطراف) و کاهش بیکاری، میزان مصرف سوخت را بهینه کند. در سطوح بالاتر ، خودرو های خودران خطر حوادث رانندگی را به حداقل می رسانند، به تجهیزات ایمنی کمتری نیاز دارند و ساخت وسایل نقلیه سبک تر را نیز امکان پذیر می کنند. فناوری SDV می‌تواند ناوگان خدماتی را قادر سازد تا وسایل نقلیه را براساس نیازهای مسافر ارسال کنند، که این خود بهره‌وری سوخت را تا ۳۰ تا ۳۵ درصد بهبود بخشد.
برخی از فعل و انفعالات در سطح خودرو دارای اثرات منفی زیست محیطی هستند. سنسورها و تجهیزات ارتباطی که قابلیت های SDV را اضافه می کنند ممکن است منجر به تغییر شکل آیرودینامیکی SDV ها شوند و مزایای توضیح داده شده در بالا را جبران کنند. حال ممکن است بگوییم سرعت خودرو های خودران می توانیم افزایش دهیم و با سرعت متوسط در بزرگراه ها کار کنند. از آنجا که کشش آیرودینامیک به شدت در سرعت‌های بالاتر افزایش می‌یابد، محدودیت‌های افزایش سرعت ممکن است استفاده بیشتر از انرژی و انتشار گازهای گلخانه‌ای را ایجاد کند[۴۶].

۴-۱۲-۱- سیاست

در ابعاد کلان و در سطح بین المللی مصرف کنندگان تمایل دارند برای دریافت فناوریهای پیشرفته و امکانات بیشتر در خودرو هزینههای بالاتری پرداخت کنند. از این رو، برای دولت ها سرمایه گذاری در این زمینه صرفه اقتصادی دارد. هم اینک کشورهای آمریکا، آلمان و ژاپن در حال سرمایه گذاری در زمینه خودروهای خودران هستند. گسترش روز افزون سیستمهای ماهوارهای و جی پی اس که روی خودروهای هوشمند امروزی نصب است راه را برای ظهور خودروهای بدون راننده هموار کرده است و قطعا سرمایه گذاری در این حوزه، باعث قدرت گرفتن دولت ها و داشتن دست بالا در توسعه ی سیاست های خارجی خواهد بود.

۱۳-۱- نمونه‌های مشابه

در ادامه به بررسی نمونه‌های مشابه سامانه‌های راننده خودکار در خودورهای خودران پرداخته شده است. به دلیل محدودیت تعداد کلمات، تنها به بررسی چند مورد بسنده شده است.

۱-۱۳-۱- محصولات تسلا

سطح هوشمندی: سطح ۲ و ۳

سیستم خودران تسلا یکی از تکنولوژی‌های برتر این شرکت به شمار می‌رود که در سال ۲۰۱۷ و بعد از گذشت یک آپدیت وارد نسل ۲.۱ خود شده است. این فناوری هم‌اکنون روی خودروهای مختلف کمپانی تسلا از جمله مدل ۳، مدل X، مدل Y و مدل S قرار دارد[۴۷]. در ادامه قابلیت‌های سامانه راننده خودکار تسلا آورده شده است:

قابلیت های Autopilot یا سطح هوشمندی ۲ تسلا : [۴۷]

- سیستم تثبیت سرعت با نام تجاری
- سیستم هدایت خودکار خودرو با نام تجاری

قابلیت های Full Self-Driving یا سطح هوشمندی ۳ تسلا :[۴۷]

- سیستم مسیریابی و هدایت خودکار با نام تجاری
- سیستم تغییر لاین خودکار با نام تجاری
- سیستم پارک خودکار با نام تجاری
- سیستم کنترل از راه دور با نام تجاری
- سیستم کنترل از راه دور خودکار با نام تجاری

- سیستم شناسای
- ی علائم و چراغ های راهنمایی و رانندگی با نام تجاری
- در آینده : فناوری هدایت خودکار در خیابان های شهری که قرار است به زودی به خودرو ها اضافه شود.

در جدول ۱۴-۱، محصولات خودران تسلا به همراه مشخصات آورده شده است:

جدول ۱۴-۱، معرفی محصولات شرکت تسلا[۴۸-۵۱].

نام مدل	Autopilot (سطح۲)	Full Self-Driving (سطح ۳)	سخت افزار های مهم	سنسورهای مهم
Model S	دارد	ماژولار اضافه می شود	۱.دوربین (۸ عدد) ۲.پردازنده	۱.اولتراسونیک ۲.رادار
Model 3	دارد	ماژولار اضافه می شود	۱.دوربین (۸ عدد) ۲.پردازنده	۱.اولتراسونیک ۲.رادار
Model X	دارد	ماژولار اضافه می شود	۱.دوربین (۸ عدد) ۲.پردازنده	۱.اولتراسونیک ۲.رادار
Model Y	دارد	ماژولار اضافه می شود	۱.دوربین (۸ عدد) ۲.پردازنده	۱.اولتراسونیک ۲.رادار

۲-۱۳-۱- ویمو [69]

سطح هوشمندی: سطح ۴

ویمو تنها شرکتی است که آزمایش ماشین‌های خودران را بدون آن‌که راننده‌ای روی صندلی قرار داشته باشد یا سرنشین کناری کاری انجام دهد آزمایش می‌کند.دستاوردهای گوگل در ارتباط با یادگیری ماشینی و تعامل بهتر هوش مصنوعی با محیط اطراف به پیشرفت ویمو کمک زیادی کرده است[۵۲]. قابلیت‌های مهم این خودرو به شرح زیر است[۵۳]:

- سیستم مسیر یابی و مکان یابی خودکار و هوشمند و تعیین تراژکتوری مسیر
- سیستم تشخیص خطوط، تابلو ها و علائم راهنمایی و رانندگی
- سیستم درک محیط بسیار قوی با قابلیت تمایز بین انسان ها، خودرو ها و موانع
- سیستم پیش بینی مسیر حرکت انسان ها و خودرو های اطراف

[69] Waymo

در جدول ۱۵-۱ مشخصات این خودرو آورده شده است:

جدول ۱۵-۱، مشخصات خودروی خودران ویمو[۵۳].

نام مدل	Fully self-driving (سطح۴)	سخت افزارهای مهم	سنسورهای مهم
Waymo Taxi	دارد	۱.دوربین ۳۶۰ درجه ۲.پردازنده	۱.لیدار ۲.رادار

۳-۱۳-۱- لوسید۷۰

سطح هوشمندی : سطح ۳

لوسید موتورز در سال ۲۰۰۷ تأسیس شده اما با گذشت ۱۳ سال از افتتاح، هنوز محصولی تجاری در کلاس خودروهای جاده‌ای نداشته است. این برند، برای حضوری قدرتمند در سطح اول صنعت وسایل نقلیه برقی تلاش می‌کند و رقابت با تسلا را مهم‌ترین هدف خود می‌داند. براساس اخبار منتشر شده، شرکت لوسید بعد از سال‌ها انتظار، موفق به تولید خودرویی در حد و اندازه‌های رقابت با تسلا مدل S شده است. این خودرو، مطابق انتظار و اطلاعاتی که پیش‌تر شنیده شده بود، یک سدان لوکس و پهن‌پیکر با نام ایر است که برتری‌هایی نسبت به تولیدات تسلا دارد. قابلیت‌های این خودرو به شرح زیر است[۵۴]:

- دید ۳۶۰ درجه
- هشدار نقطه کور
- تشخیص رنگ چراغ راهنمایی
- ترمز خودکار
- هشدار تصادف، هنگام عبور از تقاطع
- کروزکنترل هوشمند با قابلیت توقف و حرکت خوکار
- هدایت خودکار بین خطوط
- تغییر هوشمند زاویه روشنایی چراغ‌های اصلی
- پارک خودکار
- کنترل خطر هنگام خروج از پارک
- مدیریت ترمز هنگام مانور

در جدول ۱۶-۱ مشخصات این خودرو آورده شده است:

[70] Lucid

از دیگر شرکت‌های حاضر در صنعت خودروهای خودران می‌توان به موارد نام برده شده در جدول ۱۷-۱ اشاره کرد:

جدول ۱۶-۱ مشخصات خودرو لوسید[۵۴].

نام مدل	Fully self-driving (سطح۳)	سخت افزارهای مهم	سنسورهای مهم
لوسید ایر	دارد	۱.دوربین (۱۴ عدد) ۲.پردازنده	۱. لیدار ۲. رادار ۳. اولتراسونیک

جدول ۱۷-۱، شرکت‌های دیگر حاضر در صنعت خودروهای خودران[۵۵-۵۹].

نام شرکت	سال تاسیس	کشور	سطح هوشمندی	سخت افزار های مهم	سنسورهای مهم
فولکس واگن	۱۹۳۷	آلمان	۴	۱.دوربین (۴ دوربین) ۲.پردازنده	۱.لیدار ۲.رادار ۳.اولتراسوند
مرسدس بنز	۱۹۲۶	آلمان	۳ و ۴	۱.دوربین ۳۶۰ درجه ۲.پردازنده	۱.لیدار ۲.رادار
بی ام دبلیو	۱۹۱۶	آلمان	۱ و ۲ (سطح ۳ رونمایی شده)	۱.دوربین (۸ دوربین) ۲.پردازنده	۱.لیدار
آرگو ای آی	۲۰۱۶	آمریکا	۴	۱.دوربین ۲.پردازنده	۱.لیدار ۲.رادار
بایدو	۲۰۰۰	چین	۴	۱.دوربین (۵ دوربین) ۲.پردازنده	۱.لیدار

در داخل کشور تنها یک نمونه از سامانه‌های راننده خودکار وجود دارد. این نمونه در ادامه آورده شده است:

۴-۱۳-۱- خلیج فارس

سطح هوشمندی: سطح ۳

متخصصان شرکت دانش‌بنیان صنعت و فناوری هوشمند خلیج فارس (PGITIC) مستقر در پارک علم و فناوری استان هرمزگان بعد از سه سال تحقیق و تست در قالب چهار گروه الکترونیک، کنترل، مکانیک و نرم‌افزار، موفق به طراحی و ساخت اولین سیستم هوشمند کنترل خودرو بدون راننده در ایران شدند.پروژه‌ی توسعه‌ی فناوری خودران شرکت صنعت و فناوری هوشمند خلیج فارس در سال ۲۰۱۰ و با انجام فاز تحقیق و توسعه کلید خورده است. عملیات توسعه‌ی این پروژه در سال ۲۰۱۲ رسما آغاز شده و طراحی و پیاده‌سازی آن در سال ۲۰۱۴ انجام شده است. در نهایت فاز اول این پروژه در سال ۲۰۱۶ بهره‌برداری شد و طی سال ۲۰۱۷ نیز به‌طور رسمی رونمایی شد. این سیستم بعد از نصب بر روی خودرو توانست نخستین تست خود را به طول ۲۵ کیلومتر در مسیر تعیین شده بدون خطا و دخالت انسان با موفقیت به پایان برساند[۶۰]. مشخصات این محصول در جدول ۱۸-۱ آورده شده است:

جدول ۱۸-۱، مشخصات سامانه راننده خودکار خلیج فارس[۶۰].

نام مدل	سطح ۳ هوشمندی	سخت افزارهای مهم	سنسورهای مهم
ماژول متصل شونده(خودرو نیست)	در ابعاد تست دارد نه ابعاد صنعتی	۱.دوربین (۳ عدد) ۲.پردازنده ۳. جی پی اس	۱.اولتراسونیک ۲.اسکنر لیزری

۲- روش‌های علمی

۲-۱- جمع آوری و تحلیل اطلاعات علمی

امروزه شرکت‌های خودروسازی مختلفی در سراسر جهان روی ساخت و طراحی خودروهای خودران فعالیت می‌کنند و پیش‌بینی می‌شود که تا چند سال آینده، خودروهای بدون‌راننده جای خودروهای عادی امروزی را بگیرند. دانشگاه‌ها و سازندگان مختلف در سراسر جهان برروی صنعت خودروی بدون‌راننده سرمایه‌گذاری می‌کنند. با توجه به استانداردهای مختلف بررسی‌شده و طراحی‌شده، معماری یک خودرو خودران به صورت کلی به دو قسمت اصلی درک محیط ۷۱ و تصمیم‌گیری ۷۲ تقسیم می‌شود.

۲-۱-۱- ساختار کلی ارتباط مقالات

به‌طور کلی، معماری سیستم خودروهای خودران را می‌توان در دو سیستم ادراک و سیستم تصمیم گیری و زیر سیستم آن‌ها خلاصه کرد. سیستم ادراک ۷۳ وظیفه تخمین وضعیت خودرو و ایجاد نمایش داخلی از محیط (به سیستم خودران) را با استفاده از داده های گرفته شده توسط سنسورها، مانند لیدار ۷۴، رادار ۷۵، دوربین، سیستم موقعیت یابی جهانی ۷۶، واحد اندازه گیری اینرسی ۷۷، کیلومترشمار و اطلاعاتی در مورد مدل سنسورها، شبکه راه، قوانین راهنمایی و رانندگی، دینامیک خودرو و غیره را بر عهده دارد. سیستم تصمیم گیری ۷۸ با توجه به وضعیت فعلی خودرو و محیط و همچنین قوانین راهنمایی و رانندگی و امنیت و آسایش مسافران وظیفه هدایت اتومبیل از موقعیت اولیه تا هدف نهایی تعریف شده توسط کاربر را دارد. در نمودار ۱-۱، ساختار ارتباطی مقالات مطالعه شده به صورت خلاصه آورده شده است.

در ادامه، به جزئیات هر یک از سیستم‌های ادراک و تصمیم گیری، تکنیک‌های مورد استفاده برای پیاده سازی و انواع مختلف آن‌ها پرداخته شده است.

71 Perception

72 Decision-Making

73 Perception

74 Light Detection and Ranging (LIDAR)

75 Radio Detection and Ranging (RADAR)

76 Global Positioning System (GPS)

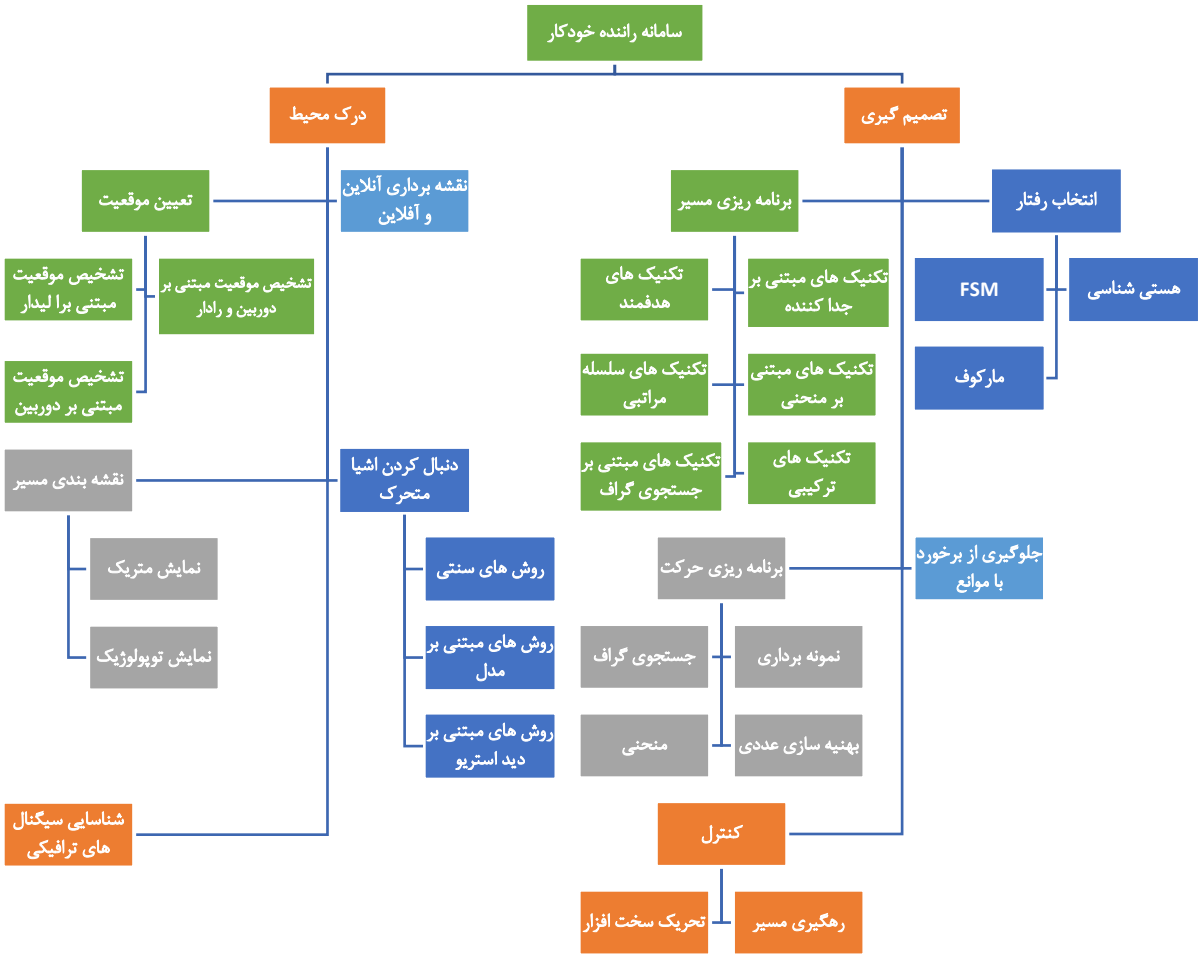
77 Inertial Measurement Unit (IMU)

78 Decision-Making

۲-۱-۲- درک محیط توسط خودرو خودران

در این بخش به بررسی تحقیقات و متدهای مهم ارائه‌شده در ادبیات برای سیستم درک محیط توسط خودروهای-خودران پرداخته می‌شود. این متدها شامل متدهای مختلف تعیین‌موقعیت، نقشه‌سازی آفلاین موانع اطراف، نقشه-سازی برای محیط، تشخیص موانع متحرک و تشخیص و شناسایی سیگنال‌های راهنمایی و رانندگی می باشند.

نمودار ۱-۲، ساختار کلی ارتباطات مقالات.



۱-۲-۱-۲- تعیین موقعیت

سیستم تشخیص موقعیت وظیفه‌ی تشخیص و تخمین موقعیت و نحوه‌ی قرارگیری خودرو خودران را بر عهده دارد. بسیاری از سیستم‌های تعیین موقعیت موجود بر پایه‌ی استفاده از اطلاعات GPS می‌باشند. لکن سیستم‌های

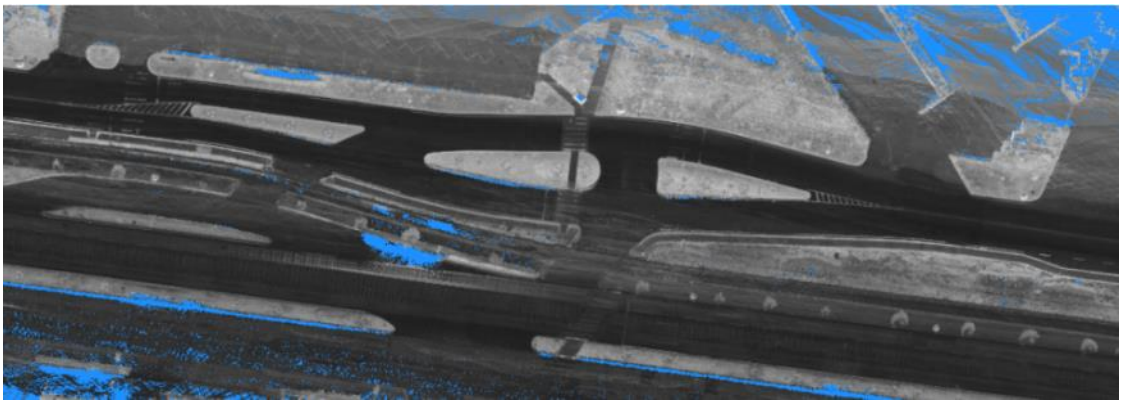
مبتنی بر GPS در بسیاری از موارد نتیجه‌ی معتبر ندارند، برای مثال در تونل‌ها، مسیرهای شهری محدودشده با ساختمان‌های بلند و غیره. روش های تشخیص

موقعیت زیادی وجود دارد که بر پایه‌ی سیستم GPS نمی‌باشد. این سیستم‌ها را می‌توان در سه دسته ی زیر طبقه‌بندی کرد: سیستم‌های مبتنی بر لیدار، سیستم‌های مبتنی بر لیدار و دوربین، سیستم‌های مبتنی فقط بر دوربین. در ادامه به معرفی هر کدام از این روش‌ها و مزایا و معایب آن‌ها پرداخته شده‌است.

۲-۱-۲-۱-۱- تشخیص موقعیت مبتنی بر لیدار

در این روش با استفاده از سنسور لیدار به تهیه‌ی یک نقشه‌ی سه‌بعدی طوری از محیط پرداخته می‌شود. سنسور لیدار یک سنسور چرخشی می‌باشد که با محاسبه‌ی زمان تابش و بازتابش پرتوهای نوری به تعیین موقعیت میپردازد. از مزایای این روش می‌توان گفت که دقت به نسبت خوبی دارد. از معایب آن نیز می‌توان به گران‌قیمت بودن سنسور لیدار اشاره کرد. در یکی از مقالات میانگین خطای مربعات حدود ۱۲ سانتی متر معرفی شده‌است.

لونیسون یک روش موقعیت‌یابی ارائه دادند که از نقشه‌های آفلاین که از بازتابش پرتوهای لیدار در آن استفاده شده است[۶۱]. آن‌ها از سنسور Velodyne HDL-64E استفاده کردند. همچنین ورونس در سال ۲۰۱۵ یک روش ارائه کرد که در آن تشخیص‌موقعیت از مقایسه‌ی نقشه‌های ماهواره‌ای با نقشه‌های به دست‌آمده از لیدار استفاده می‌شود.[۶۲]



شکل ۱-۲. نقشه ی مبتنی بر لیدار به دست آمده در [۶۲].

۲-۱-۲-۱-۲- تشخیص موقعیت مبتنی بر دوربین و لیدار

در این روش، از لیدار برای ساخت یک نقشه‌ی سه‌بعدی استفاده می‌شود و به صورت همزمان از یک دوربین برای تعیین موقعیت خودرو نسبت به این نقشه استفاده می‌شود. برای مثال در [۶۳] یک روش ارائه شد که در آن تصاویر استریو روی ابرنقاط سه‌بعدی مشخص می‌شوند. این نقشه‌ی سه‌بعدی از قبل توسط یک شرکت نقشه‌برداری

انجام شده بود که شامل اطلاعات طولی و عرضی و اطلاعات سنسور لیدار می‌باشد. در [۶۴] یک روش تشخیص موقعیت برای خودروخودران ارائه شده‌است که تصاویر پانوراما‌ی زمین که با تصاویر ماهواره‌ای تطبیق داده‌می‌شوند. تصاویر به دو بخش زمین و غیرزمین تقسیم‌بندی می‌شوند و این تقسیم‌بندی با استفاده از داده‌های سنسور لیدار انجام می‌شود.

۳-۱-۲-۱-۲- تشخیص موقعیت مبتنی بر دوربین

برخی روش‌ها برای موقعیت‌یابی تنها از دوربین استفاده می‌کنند که این روش از کم‌هزینه ترین روش‌ها می‌باشد. برای مثال در [۶۵] یک روش بر اساس سنجش‌بصری۷۹ و نقشه‌های مسیر استفاده شده‌است. آن‌ها با استفاده از نقشه‌های OpenStreet، اطلاعات مربوط به تقاطع‌ها و خیابان‌های مختلف را به‌دست می‌آورند سپس یک نمایش گراف‌مانند از مسیر به‌وجود می‌آید. همچنین یک مدل احتمالاتی از این که خودرو چگونه این مسیر را طی می‌کند به دست می‌آورند و با استفاده از اطلاعات دوربین، موقعیت خودرو نسبت به نقشه را محاسبه می کنند. در برخی دیگر روش‌ها مانند [۶۶] از تصویر یک نقشه‌ی ویژگی۸۰ استفاده می‌شود و با دو روش انجام می‌شود: روش مبتنی بر بر ویژگی نقطه‌ای که در آن تصویر هر لحظه‌ی دوربین با یک سری از تصاویر که قبلا عملیات نقشه‌برداری روی آن‌ها انجام شده‌است مقایسه می‌شود و اطلاعات مورد نیاز استخراج می‌شود. دیگر روش، روش مبتنی بر ویژگی خطی می‌باشد که در آن، نقشه به‌صورت نیمه‌اتوماتیک با استفاده از ویژگی‌هایی مااند خطوط عمودی و افقی جاده به دست می‌آید. در سایر روش‌ها مانند [۶۷]، همانند روش قبلی سعی می‌شود که با استفاده از اطلاعات دوربین به یک نقشه دست‌یافت ولی این بار از ویژگی‌های بافتی محیط استفاده می‌شود.

۲-۲-۱-۲- نقشه‌برداری آنلاین و آفلاین محیط غیرساختاری

سیستم نقشه‌بردار۸۱ در خودروی‌خودران وظیفه‌ی محاسبه و تولید نقشه‌ی محیط اطراف را برعهده دارد. این سیستم‌ها زیربنای اساسی برای ناوبری در محیط‌های غیرساختاری(پیاده روها، تابلوهاو ...) می‌باشند. نمایش محیط معمولا به دو روش توپولوژیکال و متریک انجام می‌شود. در نقشه‌بندی توپولوژیکال محیط به صورت یک گراف مدل می‌شود که در آن گره‌ها مکان‌ها یا ویژگی‌های خاص می‌باشند و یال‌ها بیانگر ارتباطات توپولوژیکال (مانند فاصله، نزدیکی، موقعیت و...) می‌باشد. روش متریک غالبا محیط را به سلول‌هایی با فاصله‌ی منظم تقسیم می‌کند

79 Visual Odometry

80 Feature map

81 Mapper

و این تقسیم‌بندی به اندازه، شکل و یا موقعیت ویژگی‌ها بستگی ندارد. در نتیجه این روش، معمول‌ترین روش برای نقشه‌بندی محیط می‌باشد.

از معمول‌ترین روش‌های نشان‌دادن محیط در خودروهای خودران، استفاده از توری اشتغال۸۲ می باشد که در [۶۸] معرفی شد. در این نقشه‌ها، فضای محیط به صورت سلول‌های گسسته‌ی هم‌اندازه تقسیم‌بندی می‌شود که معمولا در اندازه‌های سانتی‌متر هستند. هر سلول شامل احتمال اشغال بودن آن سلول استو اطلاعات داخل هر سلول به صورت مستقل برای آن سلول بروز می‌شود. اطلاعات ۳ بعدی به‌دست آمده می‌تواند بر روی تصویر دوبعدی نگاشت شود. از معایب این روش آن است که تنها سلول‌هایی که در دیدرس سنسور هستند آپدیت می‌شوند[۶۹].

در [۷۰] روش موقعیت‌یابی و نگاشت همزمان (Simultaneous Localization and Mapping83) ارائه شده‌است که در آن یک سری محدودیت‌های جزئی از داده‌های سنسور استخراج می‌شود و با حل کردن این محدودیت‌ها، قادر به تولید یک گراف گسسته‌ی دقیق از محیط می‌باشد.

۳-۲-۱-۲- نقشه‌بندی مسیر۸۴

روش‌های نگاشت یادشده در ۲-۳-۱ این امکان را برای خودروهای خودران فراهم می‌کنند که در محیط‌های صاف حرکت کنند. اما برای محیط‌هایی مانند بزرگراه‌ها که یک سری‌قوانین راهنمایی و رانندگی موجود است، خودروهای خودران به یک نقشه‌بند مسیر۸۵ نیاز دارد که قالب مسیرها و خطوط را مشخص کند. این عملیات به طور معمول با دو روش متریک و توپولوژیکال انجام می‌شود.

۱-۳-۲-۱-۲- نمایش متریک

در این روش نقشه به صورت یک نقشه‌ی توری۸۶ است که هر قسمت از محیط در یک سلول آن ذخیره شده‌است. این روش به دلیل هزینه‌ی نگهداری بسیار زیاد و توان پردازشی مورد نیاز بالا زیاد مورد استفاده نمی‌باشد. در [۷۱] یک روش مبتنی بر نمایش متریک ارائه شده است که در آن، نقشه شامل سلول‌های ۲۰ سانتی‌متر در ۲۰ سانتی‌متر می باشد که هرکدام دارای یک کد می‌باشد. کدهای غیرصفر بیانگر این است که آم سلول متعلق به یک خط مسیر می‌باشد. این کدها شامل اعداد ۱ تا ۱۶ می‌باشند که بیانگر فاصله‌ی نسبی از آن سلول به مرکز خط

82 Occupancy Grid Map

83 SLAM

84 Road Mapping

85 Road Mapper

86 Grid Map

می‌باشد. در این روش از شبکه‌های عصبی عمیق برای تشخیص کد سلول‌هایی که در آن‌ها داده وجودندارد و یا اطلاعات به‌دست آمده قابل اعتماد نیست استفاده شده است. برای آموزش این شبکه از یک دیتاست با ده ها کیلومتر مسیر استفاده شد که توانست به دقت ۸۳.۷ درصد دست یابد.

۲-۳-۲-۱-۲- نمایش توپولوژیک

دنباله‌ای از نقاط که مرکز خطوط جاده را نشان می‌دهند تشکیل شده است. این دنباله می‌تواند به صورت دستی و یا نیمه‌خودکار و یا به صورت خودکار از نقشه‌های OGM یا تصاویر ماهواره‌ای به دست آید. برای مسابقات دارپا ۲۰۰۵، از یک نوع نقشه‌ی توپولوژیکی به اسم فایل‌دیتای مشخص‌کننده‌ی مسیر (Route Data Definition File87) استفاده شد که یک قالب از داده است که شامل مختصات نقاط مسیر و سایر اطلاعات متناظر مانند ارتفاع و طول می‌باشد.

برنده‌ی مقام اول مسابقات دارپا ۲۰۰۷ از یک مدل گرافی برای خودروخودران خود استفاده کردند[۷۲]. هر گره بیانگر یک نقطه‌ی مسیر و یال‌های جهت‌دار بیانگر تمام مسیرهایی است که آن نقطه را به سایر نقاط متصل می‌کند. طول (هزینه) هر یال با در نظر گرفتن موارد مختلفی تعیین شده‌است مانند زمان مورد انتظار برای گذر از مسیر، طول مسیر و پیچیدگی مسیر. نویسندگان از برچسب‌گذاری دستی تصاویر هوایی استفاده کردند.

در سال ۲۰۱۱، رام و همکاران نقشه‌های خیابان‌باز(OpenStreetMap۸۸) را ارائه کردند که یک نقشه‌ی قابل ویرایش رایگان را در اختیار جهانیان قرار می‌داد. تولید و رشد OSM از محدودیت‌های استفاده یا دردسترس بودن اطلاعات نقشه‌ها در سراسر جهان انگیزه‌گرفته است. همچنین در دسترس بودن تصاویر و نقشه‌های ارزان-قیمت ماهواره‌ای به پیشرفت هرچه بیشتر این روش کمک شایانی کرده است. OSM مدلی از محیط به‌دست می‌دهد که شامل سه عنصر اصلی می‌باشد: گره‌ها، مسیرها و ارتباطات.[۷۳] گره‌ها بیانگر نقاط جغرافیایی، مسیرها بیانگر مجموعه‌ای از گره‌ها و ارتباطات شامل هر تعدادی از گره‌ها و مسیرها می‌باشد که ارتباطی با هم دارند.

۴-۲-۱-۲- دنبال‌کردن اشیا متحرک

دنبال‌کننده‌ی اشیا متحرک(Moving Object Tracker89) مسئولیت تشخیص و دنبال کردن موانع متحرک در طاراف خودروخودران را برعهده دارد. وجود این زیرسیستم برای عدم برخورد خودروخودران با موانع ضروری

87 RDDF

88 OSM

89 MOT

می‌باشد. موقعیت اشیا و موانع معمولا از داده‌های به دست‌آمده از سنسورهایی مانند لیدار تهیه می‌شوند. روش‌های مختلفی برای تشخیص موانع وجود دارد که به بررسی برخی از آن‌ها می‌پردازیم.

۱-۴-۲-۱-۲- روش‌های سنتی

روش‌های سنتی شامل سه قسمت اصلی بخش‌بندی داده، فیلترکردن داده و نسبت‌دهی داده می‌باشد. در قدم نسبت‌دهی داده، قسمت‌های مختلف داده به بخش‌های مختلف اشیا متحرک نسبت داده‌می‌شوند. در قسمت فیلترکردن داده، برای هر هدف، یک موقعیت تخمین زده می‌شود. تخمین موقعیت با میانگین‌گیری هندسی از داده‌های نسبت داده‌شده به هر هدف انجام می‌شود و سپس با استفاده از فیلترکالمن بروزرسانی می‌شوند.

آمارال و همکاران در [۷۴] یک روش سنتی ارائه دادند که در آن دنبال کردن اشیا محترک توسط ابر نقاط سه‌بعدی به دست‌آمده از سنسور لیدار انجام می‌شود. در این روش اطلاعات ابرنقاط به بخش‌های مختلف تقسیم‌بندی می‌شوند که این تقسیم‌بندی بر اساس فاصله‌ی اقلیدسی انجام می‌شود. ژنگ و همکاران نیز در [۷۵] یک باندینگ باکس شکل برای هر خوشه از اطلاعات ایجاد کردند و از ابعاد باندینگ‌باکس نوع آن را تشخیص می‌دادند در این روش از یک دوربین برای فیلتر کردن داده‌های موجود در ابر نقاط که متعلق به اشیا متجرک نیست استفاده می‌شود.

۲-۴-۲-۱-۲- روش‌های مبتنی بر مدل[۹۰]

در این‌روش، اطلاعات مورد نیاز به صورت مستقیم از مدل‌های فیزیکی سنسورها و مدل‌هندسی اشیا استفاده می‌شود. سپس فیلترهای غیرپارامتری (مانند فیلتر ذره‌ای) استفاده می‌شود.[۷۶]. تقسیم‌کردن داده‌های نسبت‌داده شده به اهداف مورد نیاز نیست. در [۷۷] یک روش مبتنی برمدل برای تشخیص و دنبال کردن اشیا ارائه کردند که در خودروی خودران شرکت کننده‌ی دانشگاه استنفورد در مسابقات دارپا ۲۰۰۷ از آن استفاده شد.

۳-۴-۲-۱-۲- روش‌های مبتنی بر دید استریو[۹۱]

این روش مبتنی بر تصاویر رنگی[۹۲] و اطلاعات عمقی محیط می‌باشد که این اطلاعات عمقی از دو تصویر چپ و راست به دست می‌آید. با بخش‌بندی این اطلاعات می‌توان به موقعیت اشیا مختلف دست پیدا کرد. اس و همکاران یک روش در [۷۸] ارائه دادند که فقط از اطلاعات یک دوربین که جلوی خودرو نصب شده است استفاده می‌کنند. در این روش از ماشین بردار پشتیبان (Support Vector Machine[93]) برای طبقه‌بندی استفاده می‌شود.

90 Model Based

91 Stereo

92 RGB

93 SVM

که با استفاده از این اطلاعات تشخیص می‌دهند که هر قسمتی از تصویر متعلق به اشیا متحرک می‌باشد یا خیر. همچنین زیگلر و همکاران در [۷۹] یک ساختار ارائه دادند که با شناسایی پیکسل به پیکسل و استفاده از الگوریتم SGBM به تعیین یک نقشه‌ی عمقی برای هر تصویر می‌پردازند. و این اطلاعات برای تشخیص اشیا متحرک در محیط استفاده می‌شوند.

۵-۲-۱-۲- شناسایی سیگنال‌های ترافیکی

زیرسیستم تشخیص دهنده‌ی سیگنال‌های ترافیکی(Traffic Signalization Detector94) وظیفه‌ی تشخیص و شناسایی سیگنال‌های قوانین راهنمایی و رانندگی را برعهده دارد که خودرو بتواند تصمیمات مناسب با توجه به قوانین و مقررات رانندگی را در شرایط مختلف بگیرد. بخش‌های مختلفی در این حوزه وجود دارد که مهمترین بخش‌ها عبارت اند از: شناسایی چراغ‌های راهنمایی رانندگی، شناسایی سیگنال‌های ترافیکی، و شناسایی پیاده‌روها در محیط خودروی خودران. روش‌های مورد استفاده برای شناسایی سیگنال‌های ترافیکی مختلف بسیار مشابه می‌باشد و معمولا می‌توان از یک روش در سایر بخش‌ها استفاده کرد.

در برخی روش‌های مبتنی بر یادگیری، می توان از ویژگی‌هایی استفاده کرد که به صورت دستی به دست آمده اند استفاده کرد. می‌توان گفت از سال ۲۰۰۴ به بعد از روش‌های مبتنی بر یادگیری بیشتر استفاده شود. از اولین متدهای مبتنی بر روش‌های یادگیری می‌توان به الگوریتم‌های پشت‌سرهم۹۵ استفاده می‌شود که در [۸۰] معرفی شدند. که سپس الگوریتم‌های مبتنی بر هیستوگرام گرادیان‌ها(Historgram of Gradients96) در [۸۱] معرفی شدند که پیشرفت‌های زیادی به نسبت روش‌های قبلی داشتند.

در ادامه روش‌های مبتنی بر شبکه‌های عصبی ارائه شدند که دقتی بسیار بالاتر از روش‌های قبلی مبتنی بر یادگیری و مدل داشتند. از معتبرترین این روش‌ها می‌توان به الگوریتم YOLO97 اشاره کرد. در آن به کمک ایده‌ی مستیطل‌های لنگری (Anchor Boxes) عمل شناسایی قسمت‌های مختلف سیگنال‌های راهنمایی و رانندگی و حتی موانع با دقت بسیار بالا انجام می‌شود[۸۲].

94 TSD

95 Cascade

96 HoG

97 You Only Look Once

۳-۱-۲- تصمیم گیری در خودروهای خودران

در این بخش، به مرور تکنیک های مربوطه در ادبیات سیستم تصمیم‌گیری اتومبیل‌های خودران که شامل برنامه-ریزی مسیر، انتخاب رفتار، برنامه‌ریزی حرکت و زیر سیستم‌های کنترل پرداخته شده است.

۱-۳-۱-۲- برنامه‌ریزی مسیر[98]

زیر سیستم مربوط به برنامه‌ریزی مسیر وظیفه محاسبه مسیر W از طریق یک شبکه جاده‌ای، از موقعیت اولیه خودرو خودران تا موقعیت نهایی تعریف شده توسط راننده را دارد. یک مسیر توالی ایستگاه های بین راهی است، به عنوان مثال $W = \{w_1, w_2, \dots, w_{|W|}\}$ به‌طوری که هر $w_i = (x_i, y_i)$، یک جفت مختصات است. اگر شبکه راه با یک گراف جهت‌دار وزنی نشان داده شود که رئوس آن ایستگاه های بین راهی باشد، یال‌ها جفت ایستگاه های بین راهی را به هم متصل می کنند، و وزن یال‌ها نشان دهنده هزینه پیمایش آن مسیر است. بنابراین مشکل محاسبه یک مسیر می تواند به مسئله یافتن کوتاه‌ترین مسیر در یک گراف جهت دار وزنی کاهش می یابد. با این حال، برای شبکه های بزرگ جاده ای، پیچیدگی زمانی الگوریتم های کوتاه ترین مسیر کلاسیک، مانند Dijkstra [۸۳] و A* [۸۴]، آنها را غیر عملی می کند.

در دهه گذشته، پیشرفت‌های چشمگیری در عملکرد الگوریتم‌های برنامه ریزی مسیر در شبکه‌های جاده ای وجود داشته است. الگوریتم های تازه توسعه‌یافته می توانند مسیر رانندگی را در میلی ثانیه یا کمتر، حتی در مقیاس های بزرگ محاسبه کنند. برای بررسی الگوریتم های برنامه ریزی مسیر متناسب با خودروهای خودران، به مقاله [۸۵] رجوع شده است. تکنیک های برنامه ریزی مسیر در شبکه های جاده ای از نظر زمان جستجو، زمان پیش‌پردازش، استفاده از فضا و مقاوم بودن نسبت به اغتشاش، به تکنیک‌های زیر تقسیم می شوند.

۱-۱-۳-۱-۲- تکنیک های هدفمند[99]

تکنیک های هدفمند، با جلوگیری از درنظر گرفتن راس‌هایی که در راستای راس هدف نیستند، جستجو از راس مبدا به راس هدف را هدایت می کنند. A* یک الگوریتم کوتاه ترین مسیر جهت دار هدف است. با استفاده از یک تابعی باعث می شود رئوس نزدیک به هدف زودتر بررسی شوند. الگوریتم هدفمند دیگر Arc Flags [۸۶] است. در طول مرحله پیش‌پردازش، گراف به بخش‌هایی با تعداد کمی از رئوس مرزی و تعداد یکسان راس تقسیم می شود و سپس الگوریتم اجرا می شود. روش Arc Flags دارای زمان پیش‌پردازش بالایی است، اما سریع‌ترین زمان پرس‌وجو در بین تکنیک‌های هدفمند را دارد.

98 Route & path planning

99 Goal-directed techniques

۲-۱-۳-۱-۲- تکنیک های مبتنی بر جداکننده[100]

تکنیک های مبتنی بر جداکننده بر اساس جدا کردن راس ویا یال کار می‌کنند. جداکننده راس (یا یال) زیرمجموعه کوچکی از رئوس (یا قوس) است که با حذف آن گراف به چندین بخش یکسان تجزیه می‌شود که در ادامه برای محاسبه گراف همپوشانی استفاده می‌شود. الگوریتم مسیریابی چند سطحی با عملکرد بالا[101] [۸۷] یکی از انواع این روش است که با اضافه کردن یال میانبر به گراف، زمان جستجو مسیر را به میزان قابل‌توجهی کاهش می دهد، اما در ازای آن، میزان استفاده از حافظه و زمان پیش پردازش افزایش می یابد.

۳-۱-۳-۱-۲- تکنیک های سلسله مراتبی[102]

تکنیک های سلسله مراتبی از سلسله مراتب ذاتی شبکه‌های جاده‌ای بهره می برند، که در آن جاده‌های اصلی مانند بزرگراه‌ها یک شبکه فرعی شریانی کوچک تشکیل می‌دهند. مرحله پیش پردازش با توجه به ساختار کوتاه-ترین مسیر واقعی، اهمیت رأس ها یا یال‌ها را محاسبه می‌کند. الگوریتم انقباض سلسله مراتبی[103] [۸۸] یک تکنیک سلسله مراتبی است که ایده ایجاد میانبرهایی برای عبور از رئوس با اهمیت کم را اجرا می کند. الگوریتم REACH [۸۹] یک تکنیک سلسله مراتبی دیگر است که در مرحله پیش‌پردازش، اندازه گیری مرکزیت (مقادیر reach) راس‌ها را محاسبه می‌کند و از آن‌ها برای جستجو استفاده می‌کند.

۴-۱-۳-۱-۲- تکنیک های مبتنی بر منحنی[104]

تکنیک های مبتنی بر منحنی، یک مجموعه از نقاط شناخته شده قبلی را می گیرند (به عنوان مثال، ایستگاه‌های بین راهی یک مسیر) و مجموعه‌ای جدید از نقاط را تولید می کنند که یک مسیر صاف را نشان می دهد. معمول ترین تکنیک های مبتنی بر منحنی قابل استنباط برای برنامه ریزی مسیر اتومبیل های خودران، منحنی های spline هستند.

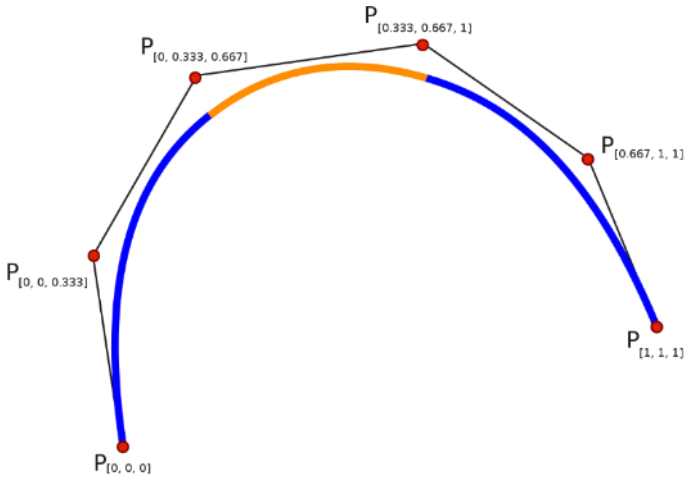
100 Separator-based techniques

101 The High-Performance Multilevel Routing (HPML)

102 Hierarchical techniques

103 The Contraction Hierarchies (CH)

104 Interpolating curve-based techniques



شکل ۲-۲. منحنی spline

منحنی spline یک منحنی پارامتری چند جمله ای به صورت تکه‌ای است که به زیر فواصل تقسیم می‌شود و می‌تواند به عنوان منحنی چند جمله ای تعریف شود. محل اتصال بین هر بخش، گره (یا نقاط کنترل) نامیده می‌شود که معمولاً دارای محدودیت صافی هستند. این نوع منحنی هزینه محاسباتی کمی دارد زیرا رفتار آن با گره‌ها تعریف می‌شود. با این حال، نتیجه آن ممکن است بهینه نباشد، زیرا بیشتر به جای محدودیت‌های جاده بر پیوستگی قطعات تمرکز دارد[۹۰]. در مقاله [۹۱] چو و همکارنش برای برنامه ریزی مسیر، از منحنی spline مرتبه سوم استفاده کردند. آنها با استفاده از طول منحنی و فاصله نسبت به خط مرکزی، مجموعه ای از spline های پارامتریک مرتبه سوم را تولید می‌کنند که نشان دهنده نامزدهای مسیر احتمالی است. مسیر بهینه براساس ایمنی راحتی مسیر انتخاب می شود.

۵-۱-۳-۱-۲- تکنیک های مبتنی بر جستجوی گراف[۱۰۵]

تکنیک‌های مبتنی بر جستجوی گراف بهترین مسیر را بین وضعیت فعلی خودرو و حالت هدف در یک فضای حالت که در قالب گراف نشان داده شده را جستجو می‌کند. متداول‌ترین روش های مبتنی بر جستجوی گراف برای برنامه ریزی مسیر اتومبیل‌های خودران، Dijkstra و انواع A* است. الگوریتم Dijkstra [۸۳] کوتاه‌ترین مسیر بین گره اولیه و گره هدف یک گراف را پیدا می کند. آرنای و همکاران [۹۲] از الگوریتم Dijkstra برای محاسبه یک مسیر برای خودرو خودران Verdino استفاده کرد. باچا و دیگران[۹۳] الگوریتم Dijkstra را برای ساخت مسیری برای ورود و خروج از پارکینگ در خودرو خودران Odin به کار گرفت.

۶-۱-۳-۱-۲- تکنیک های ترکیبی

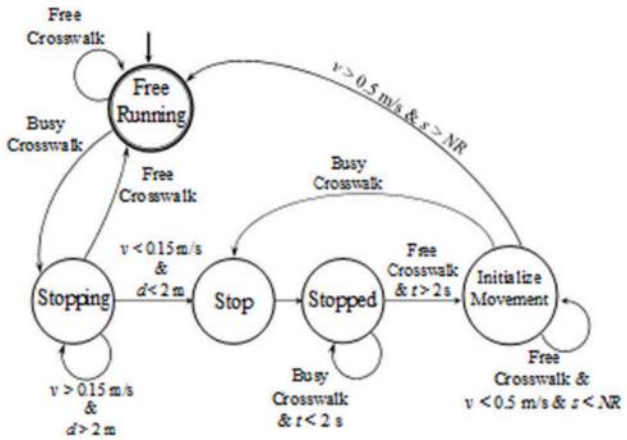
تکنیک‌های نام‌برده را می توان در الگوریتم های ترکیبی باهم استفاده کرد به‌طوری که از ویژگی‌های مختلف گراف بهره برداری شود. الگوریتم REAL [۹۴] ترکیبی از REACH و ALT است. الگوریتم ReachFlags [۹۵] ترکیبی از REACH و Arc Flag است. الگوریتم CHASE [۹۵]، CH را با Arc Flag ترکیب می کند. بدین

105 Graph search based techniques

ترتیب می توان با ترکیب چندین تکنیک، و بهره گیری از موازی کاری CPU ها و GPU های چند هسته‌ای، سرعت فرآیند را تسریع کرد.

۲-۳-۱-۲- انتخاب رفتار[106]

زیر سیستم انتخاب رفتار، وظیفه انتخاب رفتار رانندگی فعلی مانند ماندن بین خطوط، تصمیم‌گیری در تقاطع، کنترل چراغ راهنمایی و غیره را با انتخاب رفتار مناسب را دارد انتخابگر رفتار با توجه به رفتار رانندگی فعلی و جلوگیری از برخورد با موانع ایستا و متحرک در محیط در چارچوب زمانی افق تصمیم گیری، هدفی را انتخاب می کند. یک خودرو خودران باید با شرایط مختلف ترافیکی جاده ای و شهری و برخی از آنها به طور همزمان (به عنوان مثال تقاطع‌ها) کنار بیاید. مباحث پیچیده ای درباره تصمیمات اخلاقی در انتخاب رفتار مطرح می‌شود. سیستم در یک حادثه اجتناب ناپذیر چه کاری را انتخاب خواهد کرد؟ اولویت ایمنی مسافر است یا ایمنی عابر پیاده؟ تحقیقاتی در این زمینه انجام شده است از جمله مقالات [۹۶] [۹۷] [۹۸].



شکل ۳-۲ . روش انتخاب رفتار در مواجهه با عابران پیاده در معابر[۹۹]. در تصویر ، v سرعت ماشین است ، d فاصله بین ماشین و خط ایستگاه عابر پیاده است ، t زمانی است که پیاده رو آزاد است ، s موقعیت فعلی ماشین است ، و NR نقطه غیربرگشتی بعد از خط عابرپیاده است.

۱-۲-۳-۱-۲- تکنیک های مبتنی بر FSM[107]

در روشهای FSM ، یک فرایند تصمیم گیری مبتنی بر قانون[108]های از قبل تنظیم شده، برای انتخاب اقدامات تحت سناریوهای مختلف ترافیک اعمال می شود. اشکال عمده این روش دشواری مدل‌سازی، عدم قطعیت‌ها و سناریوهای پیچیده ترافیک شهری است. یکی از کاربردهای موفقیت آمیز FSM برای ترافیک جاده‌ای شهری ،

106 Behavior selection

107 Finite State Machines (FSM)

108 Rule-based decision process

استفاده آن توسط تیم جونیور در مسابقات شهری دارپا بود[۱۰۰]. آنها برای چندین سناریوی ترافیک شهری از FSMاستفاده کردند. با این حال، این رقابت مجموعه‌ای محدود از سناریوهای شهری را ارائه می دهد.

ابرهارد و همکاران[۱۰۱] روند انتخاب رفتار را به عنوان شبکه ای از اتوماتای ترکیبی و درختان تصمیم گیری مدل سازی کرد. به منظور کاهش پیچیدگی مدل، وظیفه رانندگی به یک مجموعه محدود از حالت‌های هدایت جانبی(فرمان) و طولی(گاز وترمز) تقسیم گردید. این سیستم در یک سناریوی واقعی توسط گروه تحقیق و فناوری BMW آزمایش شد. Okumura و همکاران [۱۰۲] FSM را با یک ماشین بردار پشتیبانی۱۰۹ ترکیب کرده و یک طبقه بندی برای فرآیند انتخاب رفتار سطح بالا در شرایط دوربرگردان ایجاد می کند. روند تصمیم گیری از دو بخش تشکیل شده است. ابتدا SVM با توجه به وضعیت خودرو و داده های درک موجود، یک عمل انتخاب می‌کند، سپس یک FSM عمل را پردازش کرده و دستورات سطح بالا را تولید می‌کند.

۲-۲-۳-۱-۲- تکنیک های مبتنی بر هستی شناسی۱۱۰

هستی شناسی چارچوبی برای نمایش دانش است که می تواند برای مدل سازی مفاهیم و روابط آنها استفاده شود. ژائو[۱۰۳] از یک پایگاه دانش مبتنی بر هستی شناسی برای مدل سازی مقررات راهنمایی و رانندگی و داده های حسگر برای کمک به اتومبیل های خودران برای درک جهان استفاده کرد. آنها برای ساختن سیستم تصمیم گیری خود ، پایگاه دانش مبتنی بر هستی شناسی را به صورت دستی ساختند. آنها روی موقعیت های ترافیکی متمرکز شدند که در تقاطع‌ها و جاده های باریک اتفاق می افتد. این سیستم با در نظر گرفتن مقرراتی مانند قوانین حق تقدم تصمیماتی را اتخاذ می کند و دستوراتی مانند " "Stop، "ToLeft" یا "Give Way" را به سیستم برنامه ریزی مسیر برای تغییر مسیر یا توقف می فرستد. از معایب این روش، نیاز به طراحی یک مدل دقیق جهانی متشکل از مواردی مانند خطوط نقشه برداری و قوانین راهنمایی و رانندگی در همه وضعیت‌ها است که معمولاً به صورت دستی توسط انسان انجام می شود.

۳-۲-۳-۱-۲- تکنیک های مبتنی بر فرایندهای تصمیم گیری مارکوف۱۱۱

چارچوب فرایند تصمیم گیری تا حدودی قاب مشاهده مارکوف۱۱۲ نه تنها به عدم اطمینان در انتقال اقدامات بین وضعیت‌ها، بلکه به عدم اطمینان در ادراک نیز می پردازد. این الگوریتم در یک فرآیند تکرارشونده مقدار ارزش

109 Support Vector Machine (SVM)

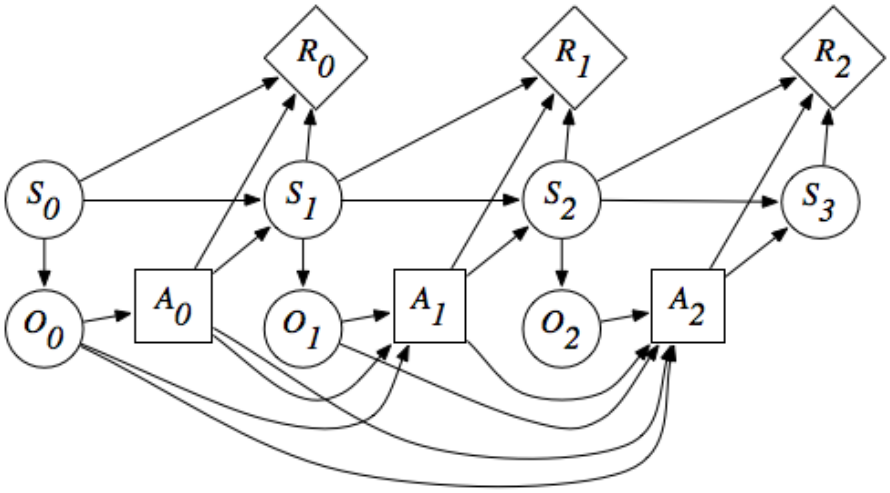
110 Ontology-based techniques

111 Markov decision processes-based techniques

112 Partially Observable Markov Decision Process (POMDP)

هر حرکت را برای تخمین سیاست کنترل بهینه تعمیم می دهد[۱۰۴]. مقاله [۱۰۵] این الگوریتم را برای انتخاب رفتار به منظور انجام تغییر مسیر هنگام رانندگی در محیط های شهری اعمال می کنند.

برشتل[۱۰۶] با توجه به تعاملات شرکت کنندگان در جاده، از یک روش مداوم POMDP برای استدلال در مورد اشیا پنهان و عدم اطمینان در مشاهده استفاده کرد. آنها از ایده فرض سیاستهای محدود برای سرعت بخشیدن به برنامه ریزی استفاده کردند. در اولین گام، راحتی و کارایی با در نظر گرفتن هزینه برای شتاب و ترمز برای رسیدن به یک منطقه هدف، توسط یک تابع پاداش بهینه می‌شود. این مرحله فقط به وضعیت خودرو و هدفی که قبلاً تعریف شده بستگی دارد. در مرحله دوم، هزینه بیشتری برای برخورد با سایر خودروها در جاده در نظر گرفته می شود.



شکل ۴-۲. POMDP به عنوان یک شبکه تصمیم گیری پویا.

۳-۳-۱-۲- برنامه‌ریزی حرکت

زیر سیستم برنامه‌ریزی حرکت مسئول محاسبه یک مسیر، از وضعیت فعلی خودرو خودران تا هدف است. این مسیر باید به مسیر تعریف شده توسط زیرسیستم انتخاب رفتار تا آنجا که ممکن است نزدیک باشد. ضمن اینکه محدودیت‌های حرکتی و دینامیکی خودرو را برآورده کند و ایمنی و راحتی مسافران را فراهم کند. تعریف یک مسیر به دو شکل اساسی صورت می‌گیرد. ممکن است به صورت دنباله ای از دستورات تعریف شود، به عنوان مثال $T_c = \{c_1, c_2, \dots, c_{|T|}\}$ ، به‌طوری که هر دستور $c_k = (v_k, \varphi_k, \Delta t_k)$ که در هر c_k ، v_k سرعت مطلوب در زمان k ، φ_k زاویه فرمان مورد نظر در زمان و Δt_k مدت زمان c_k است. یا ممکن است به صورت دنباله ای از حالتها تعریف شود $T_s = \{s_1, s_2, \dots, s_{|T|}\}$ به‌طوری که هر حالت $s = (p_k, t_k)$ که p_k یک ژست(حالت) و t_k زمان مطلقی است که انتظار می رود این ژست در آن حاصل شود. روش‌های برنامه ریزی حرکتی

را می‌توان به طور عمده در چهار کلاس تقسیم بندی کرد: روش‌های مبتنی بر جستجوی گراف، نمونه‌گیری، درون یابی منحنی و بهینه سازی عددی[۱۰۷] [۱۰۸] .

۱-۳-۳-۱-۲- تکنیک های مبتنی بر جستجوی گراف[113]

تکنیک های مبتنی بر جستجوی گراف برای برنامه ریزی حرکت ، روش های برنامه ریزی مسیر را به منظور تعیین تکامل وضعیت ماشین در طول زمان گسترش می‌دهند. متداول ترین روش‌های برنامه‌ریزی مسیر حرکت مبتنی بر جستجوی گراف برای خودرو خودران، شبکه‌های حالت[114]، باند الاستیک[115] (EB) و *A است.

شبکه حالت [۱۰۹] یک روش جستجو گراف است که در آن رأس‌ها حالت‌ها و لبه‌ها را نشانگر مسیرهایی می داند که حالاتی را که از محدودیت های حرکتی را راضی می‌کند، به هم متصل می‌کند. به این ترتیب، مسیر حرکت به سمت هدف به عنوان دنباله ای از یال‌ها در نمودار نشان داده می‌شود. شبکه‌های حالت قادر به کنترل متغییرهای مختلفی مانند موقعیت، سرعت و شتاب هستند و برای محیط های پویا مناسب هستند. با این حال، آنها هزینه محاسباتی بالایی دارند، زیرا آن‌ها هر راه حل ممکن را در گراف ارزیابی می کنند[۱۰۷].

روش دیگر مبتنی بر جستجوی گراف، نوار الاستیک است که ابتدا در برنامه ریزی مسیر و بعداً در حرکت مورد استفاده قرار گرفت. باند الاستیک یک نمودار جستجو با رئوس و لبه های الاستیک است. یک راس الاستیک با افزودن راس فضایی با یک لبه درون و خارج که رئوس فضایی همسایه را به هم متصل می کند، تعریف می‌شود. مسیر توسط یک الگوریتم بهینه سازی یافت می‌شود که دو نیرو را متعادل می کند، نیروهای دافعه تولید شده توسط موانع خارجی و نیروهای انقباضی تولید شده توسط نقاط همسایه برای از بین بردن شل شدن باند. این تکنیک تداوم و ثبات را نشان می دهد، دارای زمان اجرای آن نامشخص است و به یک مسیر اولیه نیاز دارد.

گو و همکاران [۱۱۰] یک روش برنامه ریزی مسیر فضا-زمان جدا شده را پیشنهاد می‌کند. برنامه ریزی مسیر به سه مرحله تقسیم می شود. در مرحله اول، با توجه به محدودیت های جاده و موانع، یک مسیر عملیاتی بدون برخورد از یک باند الاستیک استخراج می‌شود. در مرحله دوم، یک پروفایل سرعت تحت چندین محدودیت، یعنی محدودیت سرعت و وجود مانع، شتاب جانبی و شتاب طولی پیشنهاد می‌شود. سرانجام، با توجه به مسیر و مشخصات سرعت، مدارها با استفاده از مارپیچ های مسیر پارامتریک محاسبه می شوند. مسیرها با شبیه سازی حرکات آینده آنها در برابر تمام موانع ایستا و متحرک ارزیابی می‌شوند.

113 Graph search-based techniques

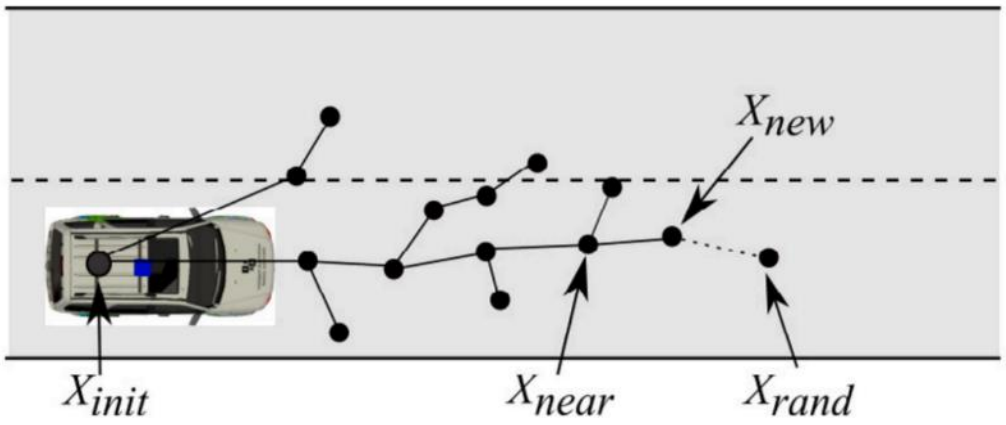
114 state lattice

115 Elastic Band (EB)

۲-۳-۳-۱-۲- تکنیک های مبتنی بر نمونه‌برداری

تکنیک‌های مبتنی بر نمونه برداری به طور تصادفی فضای حالت را جستجو می‌کنند و به دنبال ارتباط بین وضعیت فعلی ماشین و حالت هدف بعدی هستند. بیشترین استفاده از روش مبتنی بر نمونه گیری برای برنامه ریزی مسیر حرکت خودرو خودران، جستجوی سریع درخت تصادفی۱۱۶ (RRT) است.

روش های RRT برای تولید مسیر[۱۱۱]، به تدریج با استفاده از نمونه های تصادفی از فضای حالت، یک درخت جستجو از حالت فعلی خودرو ایجاد می‌کند. برای هر حالت تصادفی یک فرمان کنترل به نزدیکترین راس درخت اعمال می‌شود تا بتواند حالت جدیدی را در نزدیکترین حالت ممکن به حالت تصادفی ایجاد کند. شکل ۵-۲ مثالی از مسیر ایجاد شده توسط روش RRT از حالت فعلی اتومبیل به حالت تصادفی را نشان می دهد. روشهای RRT برای فضاهای با ابعاد بالا هزینه محاسباتی کمی دارند و همیشه اگر راه حل وجود داشته باشد و به آن زمان کافی داده شود، یک راه حل پیدا می‌کنند. [۱۰۷]. رادلی [۱۱۲] یک روش RRT برای برنامه ریزی مسیر حرکت خودرو پیشنهاد می‌کند. آن‌ها متغییرهای جدیدی را برای RRT استاندارد ارائه می‌دهد تا موقعیت حالت های تصادفی را به سمت مسیر سوق دهد، امیدوار کننده ترین دستورات کنترل را برای گسترش حالت ها انتخاب کند، حالت های غیر امیدوار کننده را کنار بگذارد و مجموعه‌ای از حالت ها را که بهترین مسیر را تشکیل می‌دهند انتخاب کند. همچنین از بخشی از مسیر ساخته‌شده در چرخه برنامه‌ریزی قبلی استفاده مجدد می‌کند. دو و همکاران[۱۱۳] یک روش RRT را پیشنهاد می‌کنند که از رفتار بصری رانندگان در جاده ها برای نمونه‌گیری از RRT استفاده می‌کند.



شکل ۵-۲. مثالی از یک مسیر تولید شده با روش RRT، X_{init} حالت اولیه خودرو است، state حالت تصادفی است، X_{near} نزدیکترین رأس درخت به X_{rand} است و X_{new} حالت جدید است.

116 Rapidly-exploring Random Tree (RRT)

۳-۳-۱-۲- تکنیک های مبتنی بر منحنی[117]

تکنیک‌های مبتنی بر منحنی، مجموعه ای از وضعیت های شناخته شده قبلی (به عنوان مثال مسیرها) را درونیابی و مسیری صاف‌تر ایجاد می‌کند که محدودیت‌های حرکتی و دینامیکی خودرو، راحتی، موانع و سایر پارامترها را در نظر می‌گیرد. متداول ترین تکنیک مبتنی بر منحنی برای طراحی مسیر حرکت خودروهای خودران، منحنی های پارچه‌ای[118] هستند[۱۰۷]. منحنی پارچه ای به دلیل انتگرال‌هایی که تعریف می‌کند، هزینه محاسباتی بالایی دارد[۱۰۷]. آلیا[۱۱۴] برای برنامه ریزی مسیر از شاخکهای پارچه ای استفاده می‌کند. شاخک ها با استفاده از یک کنترل کننده رهگیری مسیر که دستورات زاویه فرمان را از پارامترهای هندسی شاخک ها گرفته و یک بررسی برخورد روی نقشه شبکه اشغال انجام می دهد و آن‌ها را به عنوان قابل ناوبری یا قابل ناوبری طبقه بندی می‌کند. موهاگیر و همکاران[۱۱۵] همچنین برای برنامه ریزی مسیر از شاخکهای پارچه ای استفاده می‌کنند. با این حال، آن‌ها از روش الهام گرفته شده از فرآیند تصمیم‌گیری مارکوف (MDP) برای انتخاب بهترین شاخک استفاده می‌کنند.

۴-۳-۱-۲- تکنیک های مبتنی بر بهینه سازی عددی[119]

تکنیک‌های مبتنی بر بهینه سازی عددی، یک تابع را با متغیرهای محدود به حداقل یا حداکثر می رسانند. متداول ترین تکنیک های مبتنی بر بهینه سازی عددی برای برنامه ریزی مسیر حرکت خودرو خودران، روش‌های بهینه سازی تابع و روش‌های پیش بینی مدل هستند.

روش های بهینه سازی تابع با به حداقل رساندن یک تابع هزینه که محدودیت های مسیر مانند موقعیت، سرعت و شتاب را در نظر می‌گیرد، یک مسیر پیدا می کند. این روش ها می توانند به راحتی محدودیت‌های حرکتی و دینامیکی خودرو و محدودیت های محیط را در نظر بگیرند. با این حال، آنها هزینه محاسباتی بالایی دارند، زیرا فرایند بهینه سازی در هر وضعیت حرکت انجام می‌شود و به ایستگاه‌های مسیر بستگی دارد[۱۰۷]. زیگلر، بندر، دانگ و دیگران[۱۱۶] از یک روش بهینه سازی تابع برای برنامه ریزی مسیر خودرو خودران برتا استفاده می‌کند. آنها مسیر بهینه را پیدا می‌کنند که یک تابع هزینه را به حداقل می‌رساند و از محدودیت‌های مسیر پیروی می‌کند.

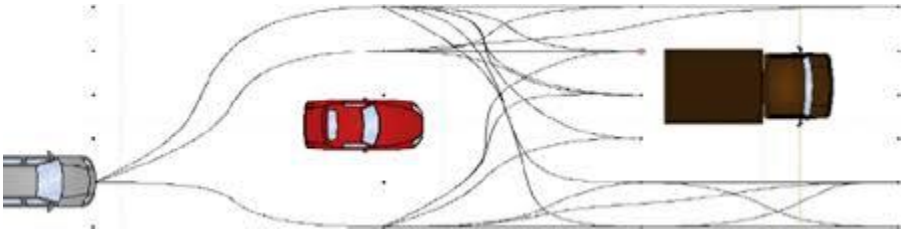
روش های پیش بینی مدل برای برنامه‌ریزی مسیر[۱۱۷] با استفاده از مدل هایی که وضعیت‌های آینده را پیش بینی می‌کنند، دستورات کنترلی عملی را به صورت پویا صادر می‌کنند. از آن‌ها می توان برای حل مسئله تولید دستورات کنترل پارامتری استفاده کرد که محدودیت های وضعیت را برآورده می‌کند و پویایی آنها را می توان با

117 Interpolating curve-based techniques

118 Clothoid curves

119 Numerical optimization-based techniques

معادلات دیفرانسیل بیان کرد. فرگوسن و همکاران [۱۱۸] از یک روش پیش بینی مدل برای برنامه ریزی مسیر حرکت اتومبیل Boss (اتومبیل دانشگاه کارنگی ملون که مقام اول را در چالش شهری DARPA 2007 بدست آورد) استفاده کرد[۱۱۹]. برای محاسبه مسیر، آنها از یک الگوریتم بهینه‌سازی استفاده می‌کنند که به تدریج تقریب اولیه پارامترهای کنترل مسیر را اصلاح می‌کند تا زمانی که خطای نقطه پایان مسیر در یک حد قابل قبول باشد.



شکل ۶-۲. برنامه ریزی حرکت در خودرو خودران

۴-۳-۱-۲- جلوگیری از برخورد با موانع[۱۲۰]

زیر سیستم جلوگیری از برخورد با موانع، مسیر محاسبه شده توسط زیر سیستم برنامه‌ریزی حرکت را دریافت می‌کند و در صورت لزوم آن را تغییر می دهد (به طور معمول سرعت را کاهش می دهد) تا از برخورد جلوگیری کند. درمورد روشهای انجام عملکردهای زیر سیستم جلوگیری از موانع، ادبیات زیادی وجود ندارد. گیدولینی [۱۲۰] یک زیر سیستم جلوگیری از مانع پیشنهاد می‌کند که، در هر چرخه برنامه حرکت، یک نقشه آنلاین دریافت می کند که نمایانگر محیط اطراف ماشین، وضعیت فعلی اتومبیل خودران در نقشه آنلاین و مسیر برنامه ریزی شده توسط برنامه ریز حرکت است. این زیرسیستم مسیر را شبیه سازی می‌کند و اگر پیش‌بینی شود تصادفی در مسیر رخ می‌دهد ، جلوگیری کننده از مانع سرعت خطی خودرو خودران را کاهش می دهد تا از تصادف جلوگیری کند. شکل ۷-۲ نمودار بلوکی جلوگیری از مانع را نشان می دهد.

120 Obstacle avoidance

شکل ۷-۲. نمودار بلوکی از سیستم جلوگیری از مانع که توسط Guidolini ارائه شده است [۱۲۰]. این سیستم به عنوان ورودی نقشه آنلاین، وضعیت فعلی خودرو و مسیر را دریافت می‌کند و مسیر را شبیه‌سازی می‌کند. در صورت بروز هرگونه تصادف، سیستم جلوگیری از مانع باعث کاهش سرعت خطی خودرو خودران می‌گردد.

کائو [۱۲۱] از داده های رادار برای محاسبه فاصله بین اتومبیل خودران و موانع موجود در محیط استفاده می کند. هی و دیگران[۱۲۲] سیستمی را برای جلوگیری از برخورد براساس معماری کنترل سلسله مراتبی متشکل از یک لایه تصمیم گیری پیشنهاد کرد، که شامل یک مدل ارزیابی خطر پویا است که به طور مداوم ریسک برخورد را محاسبه می‌کند.

۵-۳-۱-۲- کنترل[121]

زیر سیستم کنترل مسیر تولید شده توسط زیر سیستم برنامه‌ریزی حرکت، که در نهایت توسط زیر سیستم جلوگیری از مانع اصلاح می شود را دریافت می‌کند و دستورات را برای محرک فرمان، دریچه گاز و ترمزهای ماشین خودران محاسبه و ارسال می‌کند.

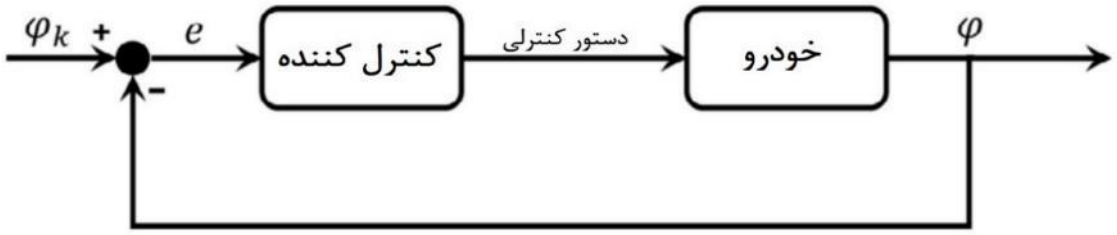
۱-۵-۳-۱-۲- روش های کنترلی تحریک مستقیم سخت افزار[122]

روشهای کنترل تحریک مستقیم سخت افزاری مستقیماً ورودیهای محرک فرمان، دریچه گاز و ترمزهای ماشین را از مسیر محاسبه شده توسط زیر سیستم برنامه‌ریز حرکت محاسبه می‌کند و خطاهای متغییرها را با توجه به مقدار خواسته شده کاهش می‌دهد. یکی از متداول ترین روش های تحریک مستقیم سخت افزار برای اتومبیل

121 Control

122 Direct hardware actuation control methods

های خودران، کنترل بازخورد۱۲۳ است. این روش شامل اعمال دستورات کنترلی، بررسیم مقادیر v و φ ، و محاسبه دستورات کنترلی آینده برای اصلاح خطاها (تفاوت بین v و φ فعلی و v و φ تنظیم‌شده). شکل ۸-۲ نمودار کنترل بازخورد را نشان می‌دهد. بلوک کنترل کننده ممکن است از انواع کنترل کننده ها تشکیل شده باشد و بلوک خودرو سیستم مورد کنترل است.



شکل ۸-۲. نمودار بلوکی زیرسیستم کنترل

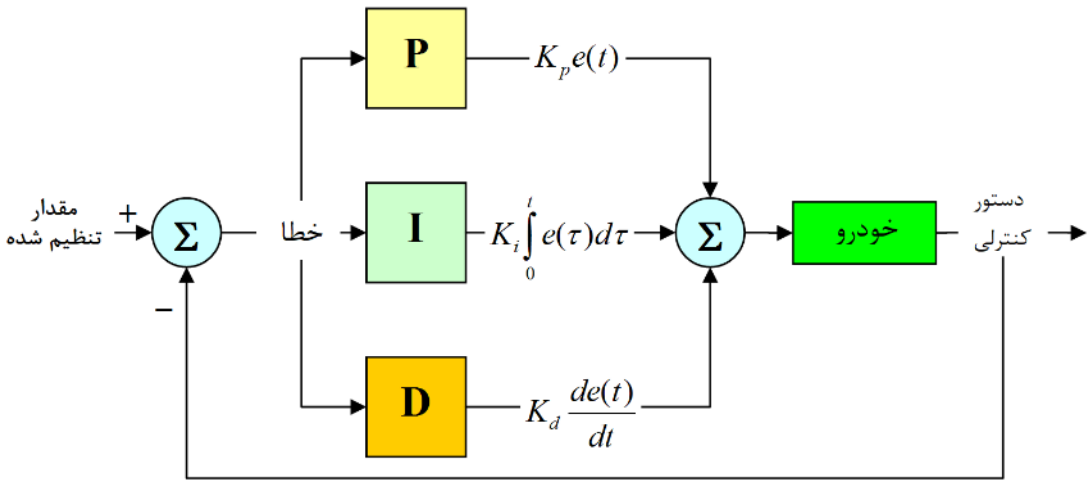
فانکه [۱۲۳] از یک روش کنترل بازخورد برای پیاده سازی زیرسیستم کنترل کننده آئودی TTS استفاده کرد. ساختار سخت افزاری آئودی TTS برای دستیابی به کنترل زمان واقعی در فرکانس ۲۰۰ هرتز طراحی شده است. زیگلر، بندر، شریبر و همکاران[۱۱۶] از یک روش کنترل بازخورد برای زیر سیستم کنترل کننده برتا۱۲۴ استفاده کردند. سیستم کنترل کننده برتا قادر به رانندگی با سرعت ۱۰۰ کیلومتر در ساعت از طریق مسیر ۱۰۳ کیلومتری Bertha-Benz-Memorial-Route بود.

یکی دیگر از روش های محبوب تحریک سخت افزار برای خودروهای خودران، روش کنترل تناسبی-انتگرالی-مشتقی۱۲۵ است. این روش شامل تعیین ورودی سخت افزار مورد نظر و اندازه گیری خطا است که میزان فاصله وضعیت فعلی سخت افزار از وضعیت ورودی سخت افزار مورد نظر را تعیین می کند [۱۲۴]. در کنترل کننده PID Kp برابر ضریب مقدار خطا، Ki برابر ضریب انتگرال خطا، و Kd برابر ضریب مشتق خطا می باشد. ژائو [۱۲۵] از یک روش PID تطبیقی مبتنی بر تکنیک حداقل واریانس تعمیم یافته برای پیاده سازی زیرسیستم کنترل کننده خودرو خودران Pionner هوشمند استفاده کرد. زیر سیستم کنترل کننده هوشمند Pionner قادر به رانندگی با سرعت ۶۰ کیلومتر در ساعت بود.

123 Feedback control

124 Bertha

125 Proportional Integral Derivative (PID)



شکل ۹-۲. دیاگرام بلوکی کنترل کننده PID

لوینسون و همکاران [۱۲۶] نیز ترکیبی از روش کنترل مدل پیش بین۱۲۶ و یک روش کنترل PID را برای زیر سیستم کنترل کننده اتومبیل رباتیک جونیور[۱۲۷] به کار گرفت (ماشین دانشگاه استنفورد که در سال ۲۰۰۷ در جایگاه دوم مسابقات شهری دارپا قرار گرفت).

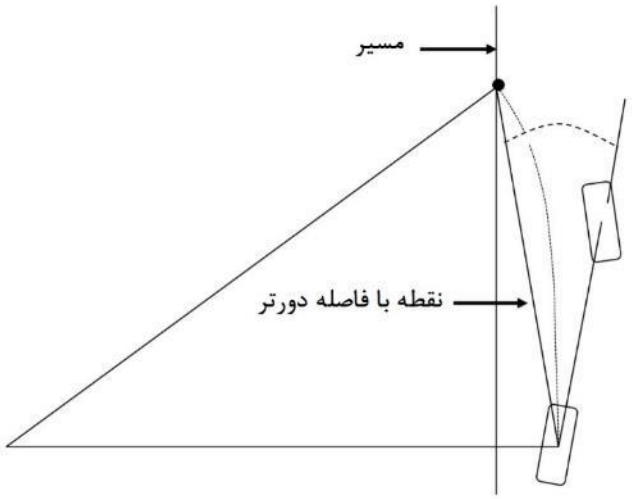
۲-۵-۳-۱-۲- روش های رهگیری مسیر۱۲۷

روش های رهگیری مسیر اجرای برنامه محاسبه شده توسط زیرسیستم برنامه‌ریز حرکت را پایدار می‌کند تا خطای ناشی از مدل خودرو را کاهش دهد. این روش‌ها را می‌توان تکنیک‌های ساده برنامه‌ریزی مسیر درنظر گرفت. روش پیگیری خالص۱۲۸ [۱۲۸] به دلیل اجرای ساده آن، به طور گسترده در خودروهای خودران برای رهگیری مسیر استفاده می‌شود. این روش شامل یافتن یک نقطه در مسیر در فاصله دورتر از مسیر فعلی و چرخاندن چرخ جلو است به طوری که یک قوس دایره ای مرکز محور عقب را با نقطه مسیر متصل می‌کند. همانطور که در شکل ۱۰-۲ نشان داده شده است. پیشنهادات زیادی برای بهبود روش پیگیری خالص وجود دارد. سامسون[۱۲۹] پیشنهاد می کند از موقعیت چرخ عقب به عنوان خروجی سخت افزار برای تثبیت اجرای برنامه حرکت استفاده شود. ترون[۱۳۰] رویکرد کنترل اتومبیل راننده استنلی را ارائه می دهد که شامل موقعیت چرخ جلو به عنوان متغیر تنظیم شده است. زیر سیستم کنترل کننده استنلی قادر به رانندگی با سرعت ۶۰ کیلومتر در ساعت بود.

126 Model predictive control (MPC)

127 Path tracking methods

128 Pure pursuit method



شکل ۱۰-۲. هندسه روش پیگیری خالص

روش کنترل مدل پیش‌بین(MPC) به طور گسترده‌ای در خودروهای خودران استفاده می‌شود. این روش با استفاده از مدل حرکت خودرو برای شبیه سازی و بهینه سازی خروجی ها با درنظرگرفتن پیش بینی آینده، ورودی های فرمان کنترل انتخاب می‌کند که منجر به خروجی مورد نظر سخت افزار می‌شود[۱۳۱]. کوگا و همکاران [۱۳۲] از روش MPC برای ردیابی مسیر یک ماشین الکتریکی کوچک استفاده می‌کنند. آنها از مدل استاندارد دوچرخه برای پیش بینی حرکت ماشین استفاده می‌کنند. زیرسیستم کنترل‌کننده ماشین الکتریکی قادر بود ماشین را تا سرعت ۲۰ کیلومتر در ساعت هدایت کند. مقاله [۱۳۳] رویکرد کنترل یک آئودی TTS را ارائه می‌دهد، که در آن از یک مدل حرکت پویا از ماشین[۱۳۴] برای فعال کردن ردیابی مسیر در محدوده های هندلینگ استفاده می‌شود.

۲-۲- یافتن راه حل زیر مسائل

در این بخش، زیر مسائل موجود در پروژه مورد بررسی قرار گرفته است. با توجه به خواسته‌های پروژه در گام حل تئوری پروژه، این بخش به دو قسمت کلی تقسیم می‌شود.

- پارک خودکار
- ادراک محیط

در ادامه به بررسی هر یک از زیر مسائله پرداخته شده و راه‌حل‌های مسائل بیان شده است.

۱-۲-۲- پارک خودکار

سیستم پارک خودکار از دوربین یا سیستم رادار و حسگرهای مختلف برای تشخیص فضای مناسب پارک و موقعیت نسبی ماشین استفاده می کند. سپس، هنگام شروع عمل پارک توسط راننده، به طور خودکار مسیر پارک را برنامه ریزی می کند. هم‌چنین، راننده در هنگام پارک نیز به ترمز و سرعت خودرو دسترسی داشته و توانایی کنترل آن را داراست. این ایده اولین بار توسط فولکس واگن در سال ۱۹۹۲ ارائه شد. شرکت‌های ولو۱۲۹ و بی ام و۱۳۰ نیز سیستم پارک خودکار خود را توسعه داده‌اند[۱۳۵]. این سیستم شامل سیستم فرمان قدرت الکتریکی، سیستم تشخیص محیط با استفاده از سنسور فراصوت و سیستم کنترل مسیر برای برنامه ریزی و ردیابی مسیر برای پارک کردن خودکار است. سیستم پارک می‌تواند بر اساس چهار حسگر فراصوت به طور خودکار فضای پارک مناسب را جستجو کرده و سپس سیگنالی را برای روشن کردن سیستم پارک خودکار به راننده ارسال کند. سیستم دستیار پارک لکسوس۱۳۱ دارای صفحه نمایشی برا نمایش تصویری از محیط اطراف خودرو است تا امکان پارک برای راننده نیز فراهم شود[۱۳۷]. این سیستم توانایی تعیین مسیر و نمایش آن بر روی صفحه نمایش را داراست.

سیستم پارک خودکار از بخش های مختلفی تشکیل شده است. در ابتدا لازم است سیستم فضای مناسب پارک را تشخیص داده شود. هم‌چنین، نوع پارک نیز مشخص می‌شود. پس از آن، خودرو متوقف شده تا بلوک برنامه‌ریزی مسیر با استفاده از الگوریتم خاصی مسیر پارک را ایجاد کند. پس از آن، مسیر تولید شده به کنترلر ارسال شده و کنترلر با ارسال سیگنال به صورت زاویه فرمان و حرکت اتومبیل عملیات پارک خودکار را بر حسب مسیر تعیین شده، انجام می‌دهد. با استفاده از داده های سنسورها، زاویه فرمان و سنسور سرعت چرخ تخمین موقعیت بر

129 Volvo

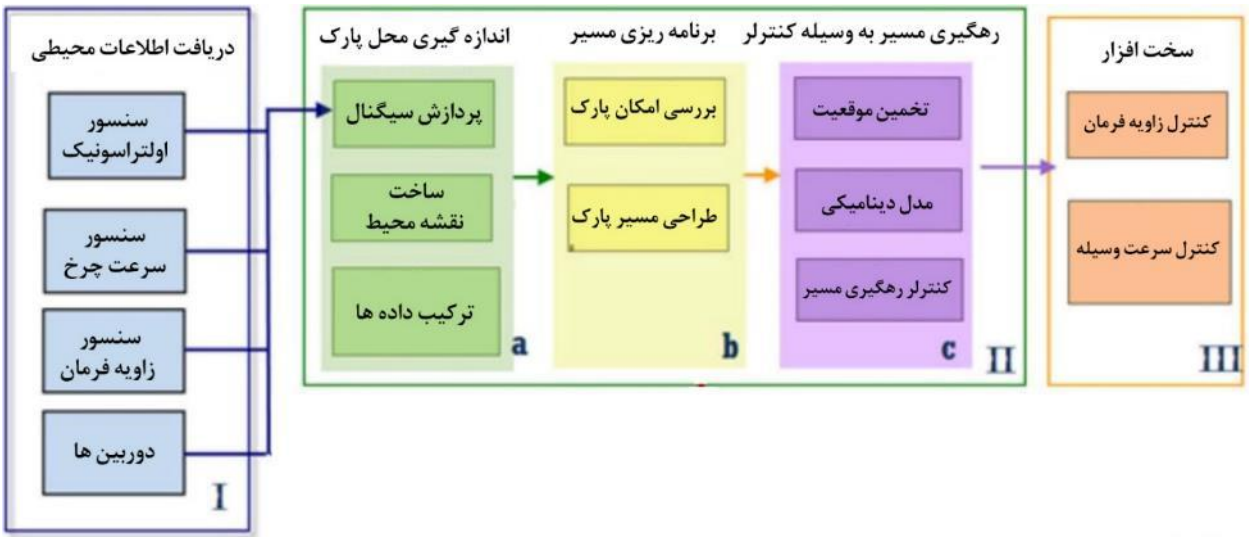
130 BMW

131 Lexus

اساس مدل دینامیکی خودرو انجام می‌شود. در شکل ۱۱-۲، نمای کلی از اجزای سیستم پارک خودکار نشان داده شده است. با توجه به شکل ۱۱-۲، پارک خودکار شامل سه چالش اصلی است:

- ادارک محیط اطراف
- برنامه‌ریزی مسیر پارک
- رهگیری مسیر برنامه‌ریزی شده

در ادامه به بررسی پژوهش‌های انجام‌شده در قسمت‌های مختلف سیستم پارک خودکار پرداخته شده است.



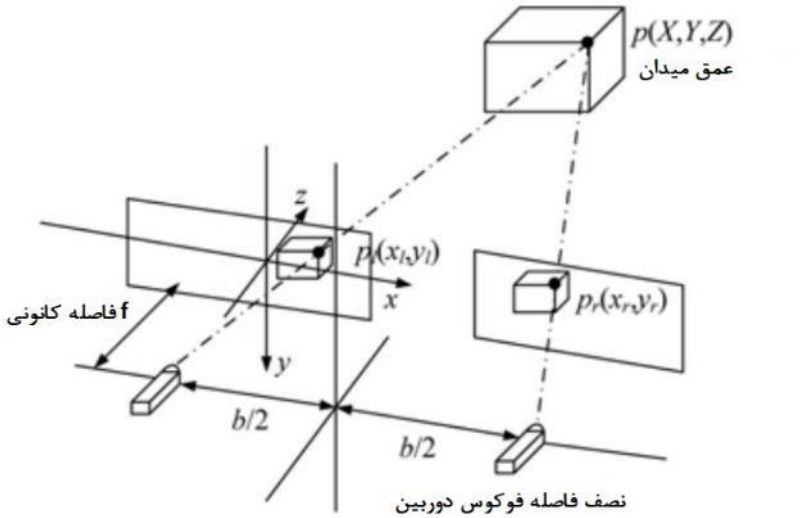
شکل ۱۱-۲، نمای کلی از ساختار سیستم پارک خودکار[۱۳۸].

۱-۱-۲-۲- ادراک محیط اطراف

بیشتر تصادفات به دلیل خطاهای انسانی مانند عدم آگاهی، حواس پرتی، خواب آلودگی، مهارت کم و خستگی رخ می‌دهد. سیستم‌های پیشرفته کمک راننده با نظارت محیط رانندگی و هشدار خطرات احتمالی به رانندگان، می‌تواند خطاهای انسانی را کاهش دهد. سیستم کمک راننده می‌تواند تصاویر نویزی را از طریق عملکردهای الکترونیکی تصحیح اعوجاج، ترمیم کند و تصویر دقیق پشت خودرو را به رانندگان ارائه دهد. هم‌چنین می‌تواند از پردازنده های تصویر برای تخمین دقیق فاصله بین اتومبیل و سایر موانع بهره ببرد. این سیستم قادر است که با ایجاد خطوط ثابت و پویای تصاویر ویدئویی، نمای دقیق پارک را در اختیار راننده قرار دهد. در زمینه استفاده از سنسورها و دوربین‌ها پژوهش‌های مختلفی انجام شده است. در ادامه به بررسی اجمالی این پژوهش‌ها پرداخته شده است.

۱-۱-۱-۲-۲- دوربین‌ها

یکی از سیستم های ادراک بصری در پارک خودکار بر اساس چهار دوربین است[۱۳۹]. اطلاعات اطراف خودرو به صورت زمان واقعی توسط دوربین هایی با زاویه دید عریض گرفته می شود. تصاویر سه بعدی از محیط، توسط یک سیستم درک بصری به یک پلان تبدیل می‌شوند، که توانایی کمک به راننده در پارک خودرو را داراست. از طریق تشخیص اندازه و مرز فضای پارکینگ توسط سیستم درک بصری، سیستم کمکی پارکینگ می‌تواند به طور خودکار در مورد پارک موازی یا عمودی تصمیم بگیرد. ژیانگ و ژنگ از سیستم دید دو چشمی به همراه ژیروسکوپ و سنسورهای فراصوت و مادون قرمز برای ادراک محیط کرده‌اند[۱۴۰]. یکی از روش های ادراک بصری، استفاده از سیستم بینایی دو چشمی یا استریو۱۳۲ است که براساس هندسه استریو، اطلاعات سه بعدی محیط را در اختیار سیستم قرار می‌دهد. نمایی از این سیستم بینایی در شکل ۱۲-۲ نشان داده شده است.

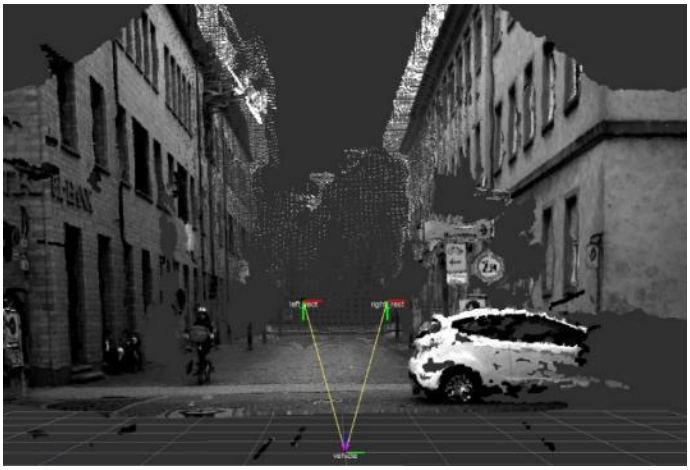


شکل ۱۲-۲، نمای کلی از سیستم استریو[۱۳۹].

سیستم بینایی دو چشمی برای بدست آوردن اطلاعات سه بعدی از چشم انسان تقلید می کند. این یک روش مناسب و مهم در زمینه دید رایانه است. همانطور که در شکل ۱۲-۲ نشان داده شده است، دو دوربین به صورت افقی قرار گرفته اند. $p_R(x_R, y_R), p_L(x_L, y_L)$ مختصات پیکسل های تصویر در دوربین های چپ و راست هستند. با استفاده از تصاویر چپ و راست، نقشه عمق تصویر و در نهایت ابر نقاط به دست می آید. $p(X, Y, Z)$ مختصات سه بعدی نقاط محیط در دستگاه مختصات دوربین است. با این روش می توان ابر نقاط محیط را بدست آورد. با استفاده از ماتریس تبدیل مختصات، دستگاه مختصات نقاط را تغییر داد. مزیت استفاده از سیستم دید دوچشمی (استریو) ارزان بودن آن نسبت به حسگرهای گران قیمتی مانند لیدار است. در شکل ۱۳-۲، ابر نقاط

132 Stereo

تولید شده به وسیله دو دوربین نشان داده شده است. ادراک بصری با هدف افزایش درک راننده از محیط پارک و هشدار خطر برخورد به راننده توسعه یافته است. هم‌چنین، استفاده از دوربین ها، آینه های دید عقب یا سایر تکنیک ها، به طور موثر مشکلات بینایی را برطرف کرده است.



شکل ۱۳-۲، ابر نقاط تولید شده با استفاده از دو دوربین[۱۴۰].

۲-۱-۱-۲-۲- سنسورها

سیستم پارک خودکار توانایی درک محیط خودرو، پردازش داده‌ها و کنترل با دقت بالا را دارد. درک محیط خودرو، معمولا توسط حسگرهایی با دقت بالا صورت می‌گیرد. حسگر اصلی و روش های پردازش سیگنال سیستم تشخیص فضای پارک را می توان به دو بخش تقسیم کرد.

۱-۲-۱-۱-۲-۲- روش مبتنی بر حسگرهای فراصوت

در مورد پارکینگ موازی از حسگرهای فراصوت بسیار استفاده می شود. اصل کار سیستم این است که وقتی وسیله نقلیه از موقعیت پارک خود عبور می کند، کنترل کننده سیگنال های حسگر فراصوت را دریافت کرده و برمبنای آن اطلاعات مربوط به فضای پارک را شناسایی و ذخیره می کند سپس نقشه ای از منطقه پارک ایجاد می کند.

پارومچیک و لاگیر از سنسورهای فراصوتی برای شناسایی محیط اطراف و جمع آوری اطلاعات استفاده کرده-اند[۱۴۱]. لاگر و همکاران به طراحی روشی جدید از خودروی هوشمند بر پایه سنسورها پرداخته‌اند[۱۴۲]. وجه تمایز این پژوهش با دیگر مقالات، استخراج پایگاه داده برای انجام مانورهای خودروی هوشمند است. ژیانگ به بررسی سیستمی شامل سنسورهای فراصوت برای انجام پاک خودکار موازی پرداخته است[۱۴۳]. جونگ و همکاران به روشی برای استفاده از سنسورهای اولتراسونیک با هزینه پایین ابداع کرده‌اند[۱۴۴]. در این پژوهش از دو سنسور اولتراسونیک در قسمت مجانبی خودرو برای انجام پارک موازی خودکار استفاده شده است.

۲-۲-۱-۱-۲-۲- روش مبتنی بر حسگرهای رادار

در این روش نیز با استفاده از داده های حسگر رادار، بر نقشه دو بعدی محیط برای شناسایی فضاهای مناسب پارک، به دست می‌آید. تحقیقات بسیاری در این زمینه انجام شده است. حسگرهای فراصوت برای تشخیص موانع اطراف خودرو و اطلاعات پارک استفاده می شوند. با در نظر گرفتن خطاهای اندازه‌گیری، زاویه فرمان و سرعت خودرو، جلوگیری از برخورد و شرایط محیطی، ابتدا خودرو در موقعیت مناسب برای شروع پارک قرار می گیرد. سپس با استفاده از عملکرد کنترل شده طراحی شده از زاویه فرمان و سرعت، خودرو درجهت پارک کنترل می شود. به دلیل محدودیت های اندازه پارک و خطای اندازه گیری، پارک خودرو فقط با یک با اندازه گیری دشوار است. درنتیجه اندازه‌گیری اطلاعات حسگرها ومتغیرها، به صورت زمان واقعی ضروری است.

به دلیل ماهیت حسگرهای فراصوت، قدرت تشخیص آن‌ها از رادارها کمتر است. اما شایان ذکر است که استفاده از حسگرهای فراصوت باعث کاهش هزینه در سیستم پارک خودکار می‌گردد. شی و ونگ سیستم پارک خودکاری بر پایه رادار با دقت بالا طراحی کرده‌اند[۱۴۵]. در این پژوهش، رادار محل مناسب برای پارک را تشخیص می‌دهد.

۳-۲-۱-۱-۲-۲- روش مبتنی بر ترکیب سنسورها

به مرور زمان روش‌های پیچیده تر نیز توسط محققان دیگر مورد بحث قرار گرفته است. تشخیص فضاهای پارک با روش ترکیب داده های چند حسگر (حسگر مادون قرمز، حسگر فراصوت، رادار) و استفاده از روش های ادراک بصری یکی از مباحث جدید این حوزه است. استفاده از حسگر فراصوت در کنار انکودر، می تواند به بدست آوردن اطلاعات محیط کمک کند. موقعیت یابی خودرو نیز با ترکیب داده‌های رادار و حسگر فراصوت بدست آمده و استفاده از روش فیلتر کالمان حاصل می شود و می تواند روند پارک را ایمن و دقیق کند. استفاده از یک روش کنترل، مثلا کنترل فازی، فیدبک حالت، شبکه عصبی و... نیز باعث بهبود پارک می‌شود.

بی و تانگ به طراحی سیستم پارکی بر پایه ترکیب حسگرهای مختلف پرداخته‌اند[۱۴۶]. روش پیشنهادی در این پژوهش در جاده‌های واقعی مورد ارزیابی قرار گرفته و نتایج به دست آمده نشان دهنده دقت مطلوب سیستم بر پایه ترکیب سنسورهای مورد استفاده است. کانگ و همکاران به روشی برای تشخیص محل پارک به پایه داده‌های سنسور فراصوت و مادون قرمز پرداخته‌اند[۱۴۷]. چو و همکاران به طراحی سیستم پارک خودکار بر پایه شبکه وایرلس و سنسورهای مادون قرمز برای شناسایی محیط پرداخته‌اند[۱۴۸].

۲-۱-۲-۲- برنامه‌ریزی مسیر

برنامه‌ریزی مسیر مسئله مهمی است که باید در سیستم پارک خودکار مورد توجه قرار گیرد. در حال حاضر، با توجه به کسب اطلاعات محیطی در اطراف خودرو، برنامه ریزی مسیر را می توان به دو دسته تقسیم کرد. اول، یک روش برنامه ریزی جهانی برای مسیر مبتنی بر اطلاعات شناخته شده محیط. دوم، یک روش برنامه ریزی

مسیر محلی مبتنی بر اطلاعات محیطی نامشخص مانند روش میدان مصنوعی و پتانسیل [۱۴۹]، روش هیستوگرام میدانی برداری [۱۵۰] و روش میدانی بردار قطب [۱۵۱] . در روش برنامه ریزی مسیر محلی، اطلاعات محیطی وسیله نقلیه ناشناخته یا تا حدودی ناشناخته است، یعنی اندازه، شکل یا موقعیت مانع را نمی توان مستقیماً از طریق سنسورها بدست آورد.

۱-۲-۱-۲-۲- برنامه‌ریزی مسیر جهانی[۱۳۳]

با محاسبه حداقل شعاع دایره‌های چرخش بر اساس محدودیت های هندسی محیط، یک مسیر مطلوب به دست می‌آید. بنابراین، طراحی آن نیاز به تضمین برخورد نداشتن خودرو دارد. تحقیقات معمول در ادامه آورده شده است. دوبینز با استفاده از حداقل شعاع های چرخش، کوتاه ترین روش‌های برنامه‌ریزی مسیر خودرو را پیشنهاد می‌کند[۱۵۲]. جهت حرکت وسیله نقلیه از یک موقعیت شروع دلخواه به هر موقعیت هدف است. طبق تحقیقات دوبینز ، ریدز و شپ در مورد مشکل کوتاه‌ترین مسیر وسیله نقلیه با تغییر جهت حرکت[۱۵۳] و هارتمن و کانایاما یک روش برنامه‌ریزی مسیر هموار را بر اساس منحنی های مکعبی توسعه می‌دهند[۱۵۴]. براسل و شوتر پیشنهاد می‌کنند که یک مسیر عملی شامل خطوط مستقیم، قوس ها و مارپیچ ها باشد[۱۵۵]. آلبادا و بوئر از یک سری مسیرهای مارپیچی تولید شده توسط کنترل کننده برای حرکت خودرو در امتداد مارپیچ برنامه‌ریزی شده استفاده می‌کند[۱۵۶].

پس از برنامه ریزی مسیر، برای کنترل زاویه هدایت و سرعت وسیله نقلیه در ردیابی دقیق مسیر هدف ، طراحی کنترل کننده ردیابی ضروری است. در فرآیند کنترل ردیابی می توان از شبکه عصبی، کنترل فیدبک و کنترل پیش‌بین[۱۳۴] استفاده کرد [۱۵۷-۱۵۸]. برنامه ریزی مسیر جهانی یک روش حلقه باز است که در صورتی که اطلاعات محیط حسگر به خوبی شناخته شود، قابل اجرا است. زیرا دستورالعمل حرکت خودرو قبل از شروع پارک تصمیم گرفته می‌شود. این روش برای محیط بدون تغییر مناسب است، چرا که سازگاری ضعیفی با محیط پویای دارای عدم قطعیت‌ها نشان می‌دهد. از مزایای این روش این است که به سنسورهای پیچیده ای احتیاج ندارد و ابتدا مسیرها را به صورت آفلاین[۱۳۵] برنامه‌ریزی می‌کند. بنابراین، به هیچ عملیاتی در زمان واقعی نیاز ندارد و نیازهای پردازنده‌ها را کاهش می‌دهد. با این حال، اشکال این است که مسیرهای به دست آمده با روش برنامه ریزی جهانی ممکن است پرپیچ و خم باشند، و هنگام ردیابی مسیرهای پر پیچ و خم خطاهای زیادی وجود دارد.

133 Global Path Planning

134 Predictive Control

135 Offline

این روش از قابلیت انطباق کمتری برخوردار است، در زمان واقعی نیست و نمی‌تواند خطاهای ناشی از حرکت خودرو را کنترل کند.

۲-۲-۱-۲-۲- برنامه ریزی مسیر محلی[136]

توانایی در زمان واقعی شاخص بسیار قابل توجهی در مسئله تولید مسیر است. همانطور که فناوری کنترل ربات به پیشرفت خود ادامه می‌دهد، تولید و کنترل خط سیر در زمان واقعی نیازهای بالاتری دارند که باعث به وجود آمدن بسیاری از روش‌های جدید برای برنامه ریزی سیر در زمان واقعی می‌شود [۱۶۰-۱۵۹]. اکنون، روش کنترل هندسی غیرخطی به طور مکرر نقل می‌شود [۱۶۲-۱۶۰]. ایده اصلی آن استفاده از مشخصه‌های مسطح دیفرانسیل[137] برای حل مشکلات تولید مسیر در زمان واقعی است. برای دستیابی به مسیرهای بهینه، وو و همکاران یک برنامه ریزی حرکتی مطلوب را تحت شرایط محدودیت‌های سینماتیک و پویایی تحلیل می کنند[۱۶۳]. از پژوهش‌های انجام شده، می‌توان فهمید که حل مشکلات برنامه ریزی مسیر با استفاده از مفهوم مسطح بودن دیفرانسیل، مزایای زیادی به همراه دارد.

۳-۲-۱-۲-۲- سایر تحقیقات

مطالعات بیشتر در این زمینه بر طراحی مسیرهای مختلف پارک متمرکز است که شامل قوس، خطوط مستقیم و مسیرهای سینوسی است. لاوموند و همکاران با به کارگیری روش عملکرد مداوم به پارکینگ موازی دست یافتند[۱۶۴]. این روش به شرح زیر است:

۱. برنامه ریزی مسیر برای جلوگیری از برخورد، و در نظر نگرفتن محدودیت‌های ناقص و محدودیت‌های هندسی.

۲. محاسبه دوباره مسیر در مواجهه با شرایط محدود کننده.

۳. با مقایسه و تجزیه و تحلیل این دو مسیر می توان مسیر متفاوتی را بدست آورد.

علی رغم اینکه روش لاوموند می تواند پارکینگ موازی را تحقق بخشد، اما زمان مصرفی این راه حل، متاثر از موقعیت اولیه خودرو و محدودیت محیطی سخت است. به منظور جستجوی موقعیت اولیه برای پارک، الگوریتم رگرسیون توسط لاگیر و پرومچک اتخاذ شده است[۱۶۶-۱۶۵]. روش طراحی مسیر سینوسی خاص، یک نقطه اولیه (نقطه اولیه یک متغیر است) را به یک نقطه هدف (موقعیت پارک) متصل می کند. هر بار ، ماشین برای تست برخورد از نقطه اولیه به موقعیت هدف حرکت می کند. هنگامی که آزمایش ممکن است منجر به برخورد شود، ماشین می تواند موقعیت اولیه را برای انجام مجدد عملیات جایگزین کند، تا زمانی که مسیر پارک بدون

136 Local Path Planinng

137 Differential Flatness Characters

برخورد را پیدا کند. این روش بسیار پیچیده و وقت گیر است. لاگیر و همکاران برای بدست آوردن یک عمل ساده و ذخیره اطلاعات نقطه اولیه، محاسبات نقطه اولیه عملی را به صورت آفلاین انجام می دهند[۱۶۸-۱۶۷]. این فرآیند پارک را ساده می کند، اما این روش همچنان به محاسبه مکرر نیاز دارد. در مقاله او ، حرکت ماشین با معادله زیر توصیف شده است:

$$\begin{cases} \dot{x} = v\cos\varphi\cos\theta \\ \dot{y} = v\cos\varphi\sin\theta \\ \dot{\theta} = v\sin\varphi/L \end{cases}$$

که φ زاویه فرمان، θ زاویه تحول مختصات، v (t) سرعت حرکت نقطه میانی محور جلو، فاصله دو محور جلو،L شعاع لاستیک ماشین و φ و v دو دستور کنترل ماشین هستند. از آنجا که زاویه فرمان اتومبیل از نظر مکانیکی محدود است، محدودیت زیر (حداکثر محدودیت انحنا) باید در نظر گرفته شود: $|\varphi| \le \varphi_{\max}$

جیانگ و سناویرن مسیری را طراحی می‌کنند که قوسها را با تعدادی خط مستقیم ترکیب می کند، اما فقط می تواند با یک فضای پارک مشخص سازگار شود و عملکرد آن محدود است[۱۶۹]. خو و همکاران از یک ماشین ناوبری برای تکمیل مراحل پارک اتوماتیک با استفاده از روش برنامه ریزی مسیر چند جمله ای استفاده می کنند[۱۷۰].

برخی از محققان فرایند پارکینگ را به چندین مرحله تقسیم می‌کنند و الگوریتم های کنترل موثر را پیشنهاد می‌دهند. لو و همکاران از حسگرهای مادون قرمز برای تشخیص اطلاعات موانع در اطراف خودرو استفاده می-کنند[۱۷۱]. یک روش پارک تخت سه مرحله ای پیشنهاد شده است.

- مرحله اسکن : حسگرهای مادون قرمز برای اسکن مکان های پارک و یافتن فضای پارک مناسب استفاده می‌شود.
- آغاز پارک کردن: مسیری با توجه به فضای پارک برای عمل پارک کردن محاسبه می‌شود و وسیله نقلیه به شروع مکان پارک هدایت می‌شود.
- مرحله ردیابی مسیر: وسیله نقلیه به طور خودکار کنترل می‌شود تا مسیر برنامه‌ریزی شده برای تکمیل پارک موازی را ردیابی کند.

لی و همکاران در مورد کنترل اتوماتیک پارکینگ تحقیقاتی انجام داد[۱۷۲]. آنها از منظر قید غیریکپارچگی، یک الگوریتم کنترل ساده و موثر پیشنهاد می‌کنند و ثبات الگوریتم را تجزیه و تحلیل می‌کنند. همچنین مشکل پارکینگ را به دو مرحله اصلی تقسیم می‌کنند. ابتدا زاویه و جهت اتومبیل را نسبت به فاصله افقی فضای پارک

تنظیم کنید و سپس فاصله طولی را تنظیم می‌کنند. یک آزمایش بر روی یک ماشین ربات انجام شده و اثربخشی الگوریتم کنترل طراحی شده تأیید می‌شود. لنو و همکاران ، برخی از تأثیرات فضای پارک محدود روی سیستم پارک اتوماتیک را مطالعه می‌کنند[۱۷۳]. رویکرد آن‌ها نه تنها شامل کنترل زاویه فرمان، بلکه کنترل سرعت نیز می باشد. یک مدل ریاضی از سیستم پارک نیز مورد مطالعه قرار گرفته است. پایداری مدل با استفاده از شبیه سازی آن و روش لیاپونوف اثبات شده است، سپس از طریق آزمایش اتومبیل ربات سیار امکان روش پارک موضعی دایره ای را بررسی می‌کند.

ادبیاتی که در بالا ذکر شد نشان می دهد بسیاری از مطالعات در مورد برنامه ریزی مسیر بسیار قابل توجه است. دانشمندان مسیرهای مختلفی طراحی می‌کنند از جمله، مسیر سینوسی اتصال نقطه اولیه و نقطه هدف (موقعیت پارک)، مسیری که قوس ها را با تعدادی خط مستقیم ترکیب می‌کند و مسیری که چند جمله ای ها را با موقعیت و جهت متقارن ترکیب می کند و ... . برخی از آنها قید عدم یکپارچگی را در مطالعات خود در نظر گرفته‌اند. به علاوه، برخی از محققان طرح کاملی از پارک را طراحی کرده و روشهای مناسبی را برای کنترل زاویه فرمان و سرعت خودرو پیشنهاد می‌کنند.

۳-۱-۲-۲- روش‌های کنترلی برای رهگیری مسیر

برای کنترل فرآیند پارک خودکار روش‌های مختلفی وجود دارد. از جمله این روش‌ها می‌توان به منطق فازی، شبکه عصبی و روش‌های کنترلی مبتنی بر نامساوی‌های خطی اشاره کرد.

سوگنو و موراکامی از یک مدل خودرو به همراه سنسورها و ریزپردازنده‌ها برای طراحی قوانین فازی کنترل پارک خودرو استفاده کرده‌اند[۱۷۴]. هم‌چنین، سوگنو و همکاران چهار سال بعد از سخت افزار مشابه و قوانین فازی استخراج شده برای کنترل پارک عمودی خودکار استفاده کرده‌اند[۱۷۵]. گوین و همکاران از دانشگاه استنفورد در سال ۱۹۹۰ در پژوهشی به بررسی پارک خودکار یک تریلی بر پایه شبکه عصبی پرداخته‌اند[۱۷۶]. در پژوهش دیگری که توسط کوسکو و همکاران انجام شده، به مقایسه منطق فازی و شبکه‌های عصبی در سیستم‌های کنترل معکوس یک تریلر پرداخته شده است[۱۷۷]. از نتایج به دست آمده این پژوهش می‌توان به مزایای بیشتر منطق فازی در مقایسه با شبکه‌های عصبی در هنگام کنترل حرکت معکوس خودروها اشاره کرد.

یاسونوبو و همکاران از منطق فازی برای استخراج تجربه رانندگی در پارک خودکار استفاده کرده‌اند[۱۷۸]. جنکینز و یوها به طراحی کنترل نوینی بر پایه شبکه عصبی پرداخته‌اند. در این پژوهش داده‌های تجربی بر اساس آموزش شبکه با استفاده از مدل دینامیکی خودرو به دست آمده اند. در پژوهش دیگری، داکسونجر و اشمیت به مطالعه سیستم کنترلی بر اساس تجربه رانندگان پرداخته‌اند[۱۷۹]. وجه تمایز این پژوهش، ترکیب شبکه عصبی و منطق فازی برای انجام پارک خودکار است. کنترلر حرکتی با ترکیب شبکه عصبی مصنوعی، فیدبک خطی و پیشخوراند

غیرخطی برای انجام پارک خودکار توسط گورینوسکی و همکاران طراحی شده است[۱۸۰]. لیتچ و رابرت به بهبود الگوریتم‌های ژنتیک برای استفاده در پارک خودکار پرداخته‌اند[۱۸۱]. در پژوهشی که توسط لئو و کیم انجام شده، از روش نقشه‌برداری سلولی برای ساخت قوانین فازی بهینه استفاده شده است[۱۸۲]. کنترلر فازی طراحی شده با این قوانین، توانایی برنامه‌ریزی یک مسیر به سمت محل پارک از هر شرایط اولیه دلخواهی را داراست. گومز، براوو و همکاران، به طراحی الگوریتمی برای جلوگیری از برخورد با موانع پرداخته‌اند[۱۸۳]. در پژوهشی که توسط این افراد انجام شده، از منطق فازی برای کنترل پارک خودکار با اقدام بهینه بر اساس اطلاعت محیطی استفاده شده است. در پژوهشی که توسط سو، جین و همکاران انجام شده، به روشی بر پایه برنامه‌ریزی بایسین برای پارک خودکار موازی پرداخته شده است[۱۸۴]. هم‌چنین، از ربات مدلی برای شبیه‌سازی روش پیشنهادی در پارک موازی به صورت خودکار استفاده شده است. نتایج به‌دست آمده، نشان دهنده صحت و درستی روش پیشنهادی است. لی و همکاران به طراحی کنترل فازی خودکار با سه نوع منطق فازی پرداخته‌اند[۱۸۵]. این کنترل فازی خودکار، توانایی پارک موازی، پارک عمودی و دور زدن خودرو را داراست. تمرکز این پژوهش بر روی طراحی کنترلر فازی و انتخاب خودکار قوانین فازی مطلوب جهت کنترل هر یک از اعمال مذکور است. ژائو و همکاران به پژوهش در رابطه با طراحی الگوریتمی برای پارک خودکار با استفاده از منطق فازی در فضای کوچک پرداخته‌اند[۱۸۶]. این محققان میزان اعتمادپذیری پارک خودکار با کنترلر فازی مذکور را با استفاده از شبیه-سازی‌های عددی تعیین کرده‌اند. هم‌چنین، ژائو و همکاران به مطالعه سیستم‌های فازی ژنتیک و کنترلرهای فازی تصمیم‌گیرنده برای استفاده از توانایی یادگیری الگوریتم ژنتیک در تنظیم قوانین فازی پرداخته‌اند. مولر و همکاران با استفاده از روش کنترل پیشخوراند و پلتفرم شرکت فولکس‌واگن برای طراحی کنترلر پارک خودکار پرداخته-اند[۱۸۷]. در پژوهش دیگری لی و لی به طراحی کنترلر فازی تکاملی برای پارک خودکار پرداخته‌اند[۱۸۸]. بالیانس و همکاران به طراحی کنترلر ترکیبی فازی و تناسبی-مشتقی-انتگرالی برای پارک موازی یه خودرو مدل پرداخته‌اند[۱۸۹]. وجه تمایز این پژوهش با دیگر مقالات در نظر گرفتن محدودیت فضا برای انجام مانور پارک است. هم‌چنین، عد برخورد با موانع دیگر هدف این پژوهش است. علاوه‌بر این، مقاوم بودن کنترلر در برابر اغتشاشات نیز در نظر گرفته شده است. نتایج پژوهش نشان دهنده بهبود علمکرد کنترلر ترکیبی رهگیری مسیر در مقایسه با کنترلر تناسبی-مشتقی-انتگرالی است.

یکی از دیگرروش‌های کنترلی برای رهگیری مسیر ، کنترل پیش‌بین مدل است. کنترل پیش‌بین مدل، یک سیگنال کنترلی بهینه با توجه به رعایت قیود مختلف ارائه می‌دهد. عیب اصلی این روش کنترلی حجم محاسبات بالا است. در حالی که، سعی پژوهشگران کاهش حجم محاسبات بدون کاهش دقت بوده است. علاوه بر این، امکان این وجود دارد که بار محاسباتی این روش برای سیستم‌های دینامیکی که نیازمند پاسخ سریع هستند، بسیار سنگین باشد. اویاما و نوناکا به طراحی کنترلری بر اساس کنترل پیش‌بین مدل با توجه به قیود حرکتی و سرعت پاسخ برای یک

خودرو مدل پرداخته‌اند[۱۹۰]. در پژوهش دیگری که توسط این پژوهشگران انجام شده، قید طول جاده برای کنترلر پیش‌بین مدل خودرو لحاظ شده است[۱۹۱]. ژیا و همکاران به طراحی کنترلر پیش‌بین مدل با هدف دنبال کردن مسیر از پیش تعیین شده با توجه به انرژی مصرف شده و برای مدل خودرو پرداخته‌اند[۱۹۲]. در پژوهش دیگری که توسط شین و همکاران انجام‌شده است، از شبکه‌عصبی به جای مدل دینامیکی برای کنترلر پیش‌بین استفاده شده است[۱۹۳]. دلیل این امر آن است که مدل دینامیکی خودرو دارای عدم قطعیت‌هایی است که بر عملکرد سیستم کنترلی تاثیر گذار است. استفاده از این روش به کاهش عدم قطعیت‌ها و بهبود عملکرد کنترلر کمک کرده است. ژیانگ و همکاران به طراحی کنترلر رهگیر مسیر با استفاده از روش کنترل پیش‌بین مدل پرداخته‌اند[۱۹۴]. در این پژوهش، از مدل خطی‌سازی شده خودرو برای کاهش حجم محاسبات استفاده شده است. هم‌چنین، قیود حرکتی سیستم همانند بیشنیه زاویه فرمان برای کنترل حرکت در طی مسیر در نظر گرفته شده است. در پژوهش دیگری که توسط کیم و همکاران انجام شده ، به طراحی کنترلر رهگیری مسیر با استفاده از کنترل پیش‌بین مدل پرداخته شده است[۱۹۵]. تفاوت این پژوهش با دیگر مقالات در استفاده از سنسورهای خودرو برای جلوگیری از برخورد با موانع در حین طی مسیر است. در پژوهشی دیگر، طراحی کنترلر رهگیر مسیر برای پارک موازی و بر پایه کنترل پیش‌بین مدل توسط هائو وهمکاران انجام شده است[۱۹۶]. لحاظ کردن قیود حرکتی و زاویه فرمان به همراه سرعت خودرو به عنوان پارامترهای خروجی کنترلر از مزایای این پژوهش است. شبیه‌سازی‌های رایانه‌ای به همراه تست‌های تجربی نشان دهنده عملکرد مطلوب کنترلر پیش‌بین مدل در مقایسه با کنترلر تناسبی-مشتقی-انتگرالی است.

یکی از پژوهش‌های نوآورانه در این زمینه توسط هانگ و همکاران انجام شده است[۱۹۷]. در این پژوهش، از کنترل تطبیقی برای رهگیری مسیر پارک استفاده شده است. این روش کنترلی شامل الگوریتم کنترلی، الگوریتم تخمین پارامترها و الگوریتم تنظیم مجدد پارامترها است. این روش کنترلی به دلیل استفاده از داده‌های ورودی و خروجی سیستم نیاز به اطلاعات مدل دینامیکی نداشته و برای خودروهای مختلف قابل استفاده است. هم‌چنین، از مولفه سرعت به عنوان یک متغیر برا انجام عملیات پارک استفاده شده است. علاوه بر این، از مزایای این روش می‌توان به حجم محاسبات کمتر و مقاوم‌بودن بیشتر در برابر عدم قطعیت‌ها اشاره کرد.

از دیگر روش‌های کنترلی برای رهگیری مسیر استفاده از کنترل مود لغزشی۱۳۸ است. این روش دارای حجم محاسبات کم و سرعت عملکرد بالا است. در حالی که ، با پدیده‌ای به نام چترینگ۱۳۹ همراه خواهد بود. در این پدیده، سیگنال کنترلی برای رسیدن به مقدار مطلوب دچار نوسان خواهد بود[۱۹۸]. ژو و همکاران، به طراحی

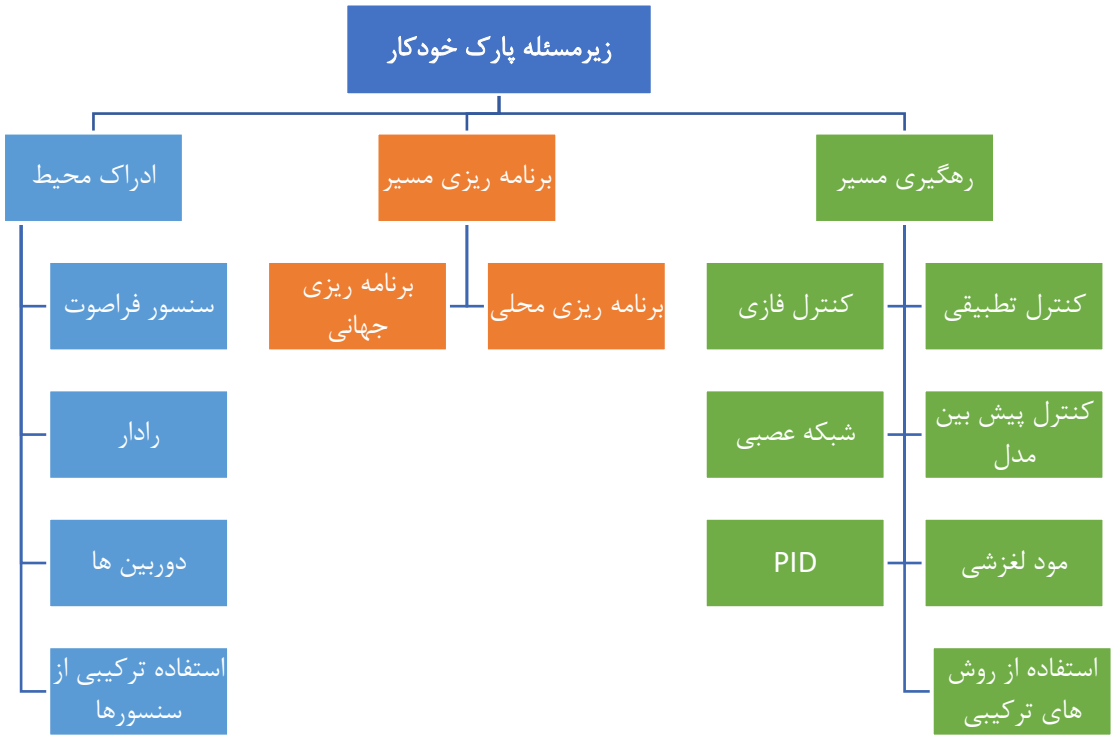
138 Sliding Mode Control

139 Chattering

یک کنترلر رهگیری مسیر در پارک موازی بر پایه مود لغزشی تطبیقی پرداخته‌اند[۱۹۹]. در این پژوهش، از شناسایی در لحظه۱۴۰ مدل به جای معادلات دینامیکی استفاده شده است. مزیت این روش، بهبود دقت کنترلر و هم‌چنین مقاوم شدن سیستم در برابر عدم قطعیت‌ها است. هم‌چنین، قید محدودیت زاویه فرمان با استفاده از تابع اشباع برای کنترلر تعریف شده است. علاوه‌براین، تحلیل پایداری سیستم نیز در نظر گرفته شده است.

۴-۱-۲-۲- جمع بندی

در این قسمت چالش‌های مسئله پارک خودکار بررسی گردید. پژوهش‌های انجام گرفته برای چالش به صورت خلاصه آورده شده است. با توجه به تحقیقات انجام شده، راه‌حل‌های ارائه شده برای مسئله پارک خودکار به صورت خلاصه در شکل ۱۴-۲ آورده شده است.



شکل ۱۴-۲، راه حل‌های ارائه شده برای زیر مسئله پارک خودکار به تفکیک چالش.

با توجه به پژوهش‌های مرور شده، چالش‌های موجود در طراحی سیستم پارک خودکار به شرح زیر است:

- **راحتی کاربرد:** پیاده سازی سیستم پارک خودکار تا حد ممکن ساده باشد.
- **مطابقت با مدل های مختلف خودرو:** به دلیل وجود تفاوت در معادلات دینامیکی خودروهای مختلف، سیستم باید توانایی کنترل خودروهای مختلف با معادلات متفاوت را دارا باشد.

140 Online Identification

- **حجم محاسبات کنترلر:** با توجه به سخت افزار مورد استفاده، کنترلر طراحی شده باید توانایی پاسخ در زمان مناسب با توجه به سرعت عملکرد خودرو را دارا باشد.
- **دقت عملکرد:** انجام مانور پارک خودکار با رهگیری دقیق مسیر برنامه‌ریزی شده، از برخورد خودرو با موانع جلوگیری کرده و پارک خودرو به درستی در محل تعیین شده، انجام می‌پذیرد.
- **مقاوم بودن در برابر اغتشاشات:** سنسورهای مورد استفاده در سیستم خودروی خودران، دارای اغتشاشات هستند. سیستم پارک خودکار باید در برابر اغتشاشات اطلاعات دریافتی از سنسورها مقاوم باشد.
- **عدم قطعیت های مدل خودرو:** دینامیک خودروها، دارای پارامترهایی است که بر حسب زمان تغییر می‌کنند. هم‌چنین، رفتار دینامیکی خودرو دارای عدم قطعیت‌هایی در معادلات می‌باشد. سیستم پارک خودکار باید توانایی کنترل خودرو با وجود عدم قطعیت‌ها را داشته باشد.
- **پایداری سیستم:** فرامین کنترلی سیستم پارک خودکار باید به صورتی تعیین شود که سیستم دچار ناپایداری نشود.
- **لحاظ کردن قیود حرکتی:** سیستم دینامیکی خودرو شامل قیود و محدودیت‌هایی است. از جمله این محدودیت‌ها می‌توان به محدودیت زاویه چرخش فرمان و سرعت پارک اشاره نمود. لحاظ کردن این محدودیت‌ها در سیستم پارک خودکار بسیار مهم است.

با توجه به توضیحات و پژوهش‌های ذکر شده، در فصل چهار به انتخاب روش‌های مناسب برای پارک خودکار و تطبیق راه‌حل‌ها با چارچوب تعیین شده برای پروژه پرداخته شده است.

۲-۲-۲- ادراک محیط

خودرو خودران با استفاده از تجهیزات مختلفی از جمله رادار، لیدار، جیپیاس و بینایی ماشین و با استفاده از تکنیکهای همچون پردازش تصویر، شبکههای عصبی مصنوعی، الگوریتمهای فرا ابتکاری و داده کاوی این دادههای مختلف، با جهان بیرون ارتباط برقرار میکند و به اصطلاح آن را درک می کند. با توجه به این که پروژه ی مسابقه در حیطه ی شبیه سازی تعریف شده است، عمده ی توجه ما روی بخش پردازش تصویر، شبکه های عصبی و بینایی ماشین است و بخش های کار با سنسور در این پروژه که در حوزه ی SIL(Software in the Loop) تعریف شده جایگاهی ندارد.

بینایی ماشین شامل روشهای مربوط به دستیابی تصاویر، پردازش، آنالیز و درک محتوای آن است. این سیستم تصاویر دنیای بیرون را به عنوان ورودی دریافت و دادههای عددی یا سمبلیک را به عنوان خروجی تولید مینماید. این سیستم با الگو برداری از سیستم بینایی انسان در رایانه شبیهسازی شده است.

به طور کلی حوزه ی ادراک محیط با بینایی ماشین در خودرو های خودران را می توان به زیر مسائل مختلفی از جمله شناسایی علائم و چراغ های راهنمایی رانندگی، شناسایی اشیای ثابت و دنبال کردن اشیای متحرک، شناسایی لاین های جاده، تفکیک اشیا از هم و فاصله سنجی از اشیا و لبه ها تقسیم کرد و ضمنا اکثر این زیر مسائل را می توان با راه حل های مختلفی از جمله روش های پردازش تصویر کلاسیک و روش های مبتنی بر شبکه های عصبی عمیق حل نمود. لازم به ذکر است که لزوما تمام زیر مسائل معرفی شده در حیطه ی پروژه تعریف شده، در مسابقه نمی گنجند و تنها تعدادی از آن ها در بخش پیاده سازی مورد استفاده قرار می گیرند، اما در این بخش، جمع آوری اطلاعات درباره ی آن ها و بررسی و تحلیل آن ها خالی از لطف نیست. در ادامه ضمن ارائه توضیحات و محتوای مرتبط با موضوع پروژه، با استناد به مقالات و رساله های داخلی و خارجی، چالش ها و زیر مسائل مختلف قسمت ادراک محیط را مشخص می کنیم و راه حل های مختلف و متنوعی را برای هر کدام از زیر مسائل و چالش های مختلف این قسمت بررسی می کنیم و آن ها را جمع آوری می کنیم.

۱-۲-۲-۲- شناسایی علائم و چراغ های راهنمایی رانندگی[141]

در این بخش، قصد داریم تا راه حل های مختلف و متنوعی را برای زیر مسئله ی شناسایی علائم و چراغ های راهنمایی رانندگی بررسی کنیم و با چالش ها و روش های موجود در هر کدام از راه حل ها آشنا شویم.

۱-۱-۲-۲-۲- روش کلاسیک و مبتنی بر پردازش تصویر

Haar-Cascade یک الگوریتم تشخیص شی مبتنی بر یادگیری ماشین است که برای شناسایی اشیا موجود در یک تصویر یا فیلم و براساس ویژگی های استخراج شده استفاده می شود. برای مثال در تصویر در بخش تشخیص تابلو، از الگوریتم Haar-Cascade استفاده شده است. بعد از تشخیص تابلو نیاز به تعیین نوع تابلو است که یکی از معروف‌ترین الگوریتم های طبقه بندی در یادگیری ماشین،SVM می باشد که برای تصویر زیر بر اساس مجموعه داده‌های موجود آموزش داده شده است[۲۰۰].

141 Sign Detection


شکل ۱۵-۲ تشخیص تابلو STOP با استفاده ازHaar-Cascade شکل ۱۶-۲ تعیین کلاس تابلو با استفاده از SVM

۲-۱-۲-۲-۲- روش مبتنی بر شبکه های عصبی عمیق

سیستم‌های تشخیص علائم ترافیکی[۱۴۲] یک مؤلفه کلیدی در کاربردهای جهان واقعی، مانند رانندگی خودکار، نظارت ترافیکی، ایمنی راننده، حفظ شبکه جاده‌ای و تحلیل صحنه‌های ترافیکی است. در ادامه برترین سیستم های تشخیص شی[۱۴۳] مانند (Faster R-CNN, R-FCN, SSD YOLOV2) و ترکیب آنها با چندین استخراج کننده ویژگی[۱۴۴] مانند (Inception V2, Inception Resnet V2, Resnet V1 50, Resnet V1 101 Mobilenet V1, Darknet-19) مورد تجزیه و تحلیل قرار گرفته است. سیستم‌های تشخیص علائم ترافیکی به صورت دو مورد زیر تعریف می شوند:

- تشخیص علائم راهنمایی و رانندگی[۱۴۵]
- طبقه بندی علائم راهنمایی و رانندگی[۱۴۶]

برای تصمیم گیری در مورد اینکه کدام یک از تشخیص دهنده ها برای یک کاربرد خاص که در اینجا تشخیص علائم راهنمایی و رانندگی مناسب است، نه تنها معیارهای دقت استاندارد مثل میانگین متوسط دقت (mAP) مهم هستند، بلکه عوامل دیگری مانند مصرف حافظه و مدت زمان اجرا نیز نقشی اساسی دارند. چندین مجموعه داده علائم راهنمایی و رانندگی در دسترس از کشورهایی مانند ایالات متحده، بلژیک، آلمان، کرواسی، ایتالیا، سوئد و چین جمع آوری شده است. در ادامه به بررسی این روش ها که آزمایش خود را بر روی مجموعه داده های تشخیص علائم راهنمایی و رانندگی آلمان[۱۴۷] انجام شده، پرداخته می شود.

142 TSRS

143 Object detection

144 Feature extractor

145 Traffic sign detection

146 Traffic sign recognition

147 GTSDB

در این بخش، عملکرد آزمایش های آشکارساز علائم راهنمایی و رانندگی، بررسی می شود. تجزیه و تحلیل هر یک از این آزمایشات شامل اقدامات مختلفی مانند دقت ، تعداد پارامترها ، عملیات نقطه شناور (FLOP) ، مصرف حافظه و زمان پردازش است . این مدل ها بر روی رایانه ای آموزش داده شده‌اند که با پردازنده مرکزی Intel Core i7-4770، ۱۶ گیگابایت رم وکارت گرافیک NVIDIA Titan Xp ارزیابی می شوند که ۳۸۴۰ هسته CUDA و ۱۲ گیگابایت رم دارد.

نتایج دقیق برای هر کلاس علامت راهنمایی و رانندگی در جدول ۱ همراه با پرسیژن۱۴۸، ریکال۱۴۹، میانگین پرسیژن ۱۵۰و میانگین IoU حاصل شده توسط هر تشخیص دهنده در جدول ۱-۲ ارائه شده است.

در کل در این مقاله ، یک مقایسه تجربی از هشت آشکارساز علائم راهنمایی و رانندگی بر اساس شبکه های عصبی عمیق ارائه شده است. همچنین جنبه های اصلی این آشکارساز ها مانند دقت ، سرعت ، مصرف حافظه ، تعداد عملیات و تعداد پارامترهای قابل یادگیری در CNN تجزیه و تحلیل شده است. همه مدل های مورد مطالعه در این مقاله از قبل با مجموعه داده های Microsoft COCO آموزش داده شده و پس از آن با مجموعه داده GTSDB تنظیم شده اند تا کلاس های مختلف علائم راهنمایی و رانندگی را بر اساس شکل و رنگ آنها شناسایی و طبقه بندی کنند[۲۰۱].

148 Precision

149 Recall

150 mAP

شکل ۱۷-۲ نمونه ای از نتایج مدلهای آشکارساز علائم راهنمایی و رانندگی[۲۰۱].

جدول ۱-۲ نتایج دقت بر روی مجموعه داده GTSDB (بر حسب درصد) که توسط هر مدل آشکارساز علائم راهنمایی و رانندگی بدست آمده[۲۰۱].

Model	Class	Avg. IoU	Precision	Recall	AP
Faster R-CNN Resnet 50	Prohibitory	82.52	91.38	98.75	98.62
	Mandatory	81.21	70.00	85.71	85.15
	Danger	85.07	79.45	92.06	90.78
Faster R-CNN Resnet 101	Prohibitory	87.29	90.29	98.14	98.13
	Mandatory	85.58	67.65	93.88	93.46
	Danger	87.05	85.51	93.65	93.64
Faster R-CNN Inception V2	Prohibitory	82.73	81.22	99.38	99.36
	Mandatory	79.66	62.50	81.63	80.47
	Danger	85.62	81.69	92.06	92.03
Faster R-CNN Inception Resnet V2	Prohibitory	91.37	96.99	100	100
	Mandatory	89.16	79.31	93.88	93.66
	Danger	90.11	92.19	93.65	93.65
R-FCN Resnet 101	Prohibitory	87.93	84.66	99.38	99.37
	Mandatory	85.37	76.67	93.88	92.58
	Danger	86.95	86.76	93.65	93.52
SSD Inception V2	Prohibitory	81.76	96.95	78.88	78.77
	Mandatory	80.85	90.00	55.10	54.46
	Danger	85.76	93.18	65.08	65.05
SSD Mobilenet	Prohibitory	80.49	92.50	68.94	67.03
	Mandatory	78.51	89.65	53.06	52.01
	Danger	81.11	79.63	68.25	65.85
YOLO V2	Prohibitory	73.96	92.31	89.44	88.73
	Mandatory	74.66	79.07	69.39	65.70
	Danger	75.82	94.55	82.54	82.06

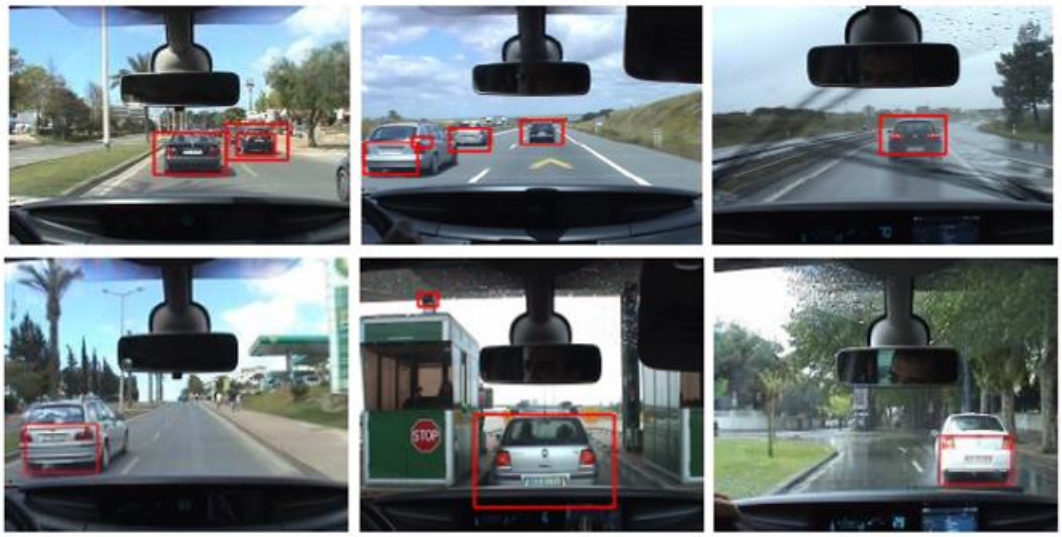
۲-۲-۲-۲- شناسایی اشیا[151] (خودرو ها و عابرین پیاده)

در این بخش، قصد داریم تا راه حل های مختلف و متنوعی را برای زیر مسئله ی شناسایی اشیا که شامل خودرو ها و عابرین پیاده می باشند، بررسی کنیم و با چالش ها و روش های موجود در هر کدام از راه حل ها آشنا شویم.

۱-۲-۲-۲-۲- روش کلاسیک و مبتنی بر پردازش تصویر

در این حوزه روشی بر اساس تشخیص ویژگی های اشیا درHaar-Cascade معرفی شده که برای تشخیص ماشین ها و عابرین پیاده در جاده های واقعی طراحی شده است. در این مقاله از سه دیتاست (TDS1,TDS2,TDS3) برای آموزش کسکید ها و همچنین سه دیتاست (PDS1,PDS2,PDS3) برای آزمایش عملکرد آن ها شده استفاده شده است که به شکل دستی توسط نگارندگان مقاله جمع آوری شده اند.

بر خلاف روشی که برای شناسایی تابلو بررسی شد که در آن نیاز بود تا در نهایت، نوع تابلو نیز مشخص گردد، در این روش تنها به شناسایی خودرو بسنده می شود و در این مرحله شناسایی نوع خودرو اهمیت کمتری دارد[۲۰۲].

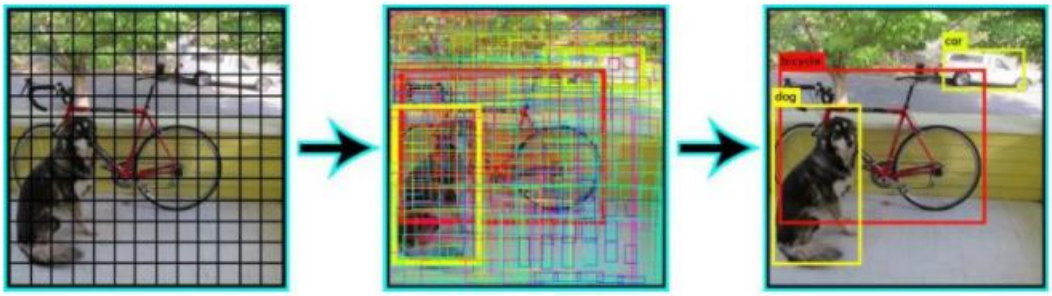


شکل ۱۸-۲ پیدا کردن اشیا (خودروها) در تصویر یا استفاده از Haar-Cascade [۲۰۲].

151 Object Detection

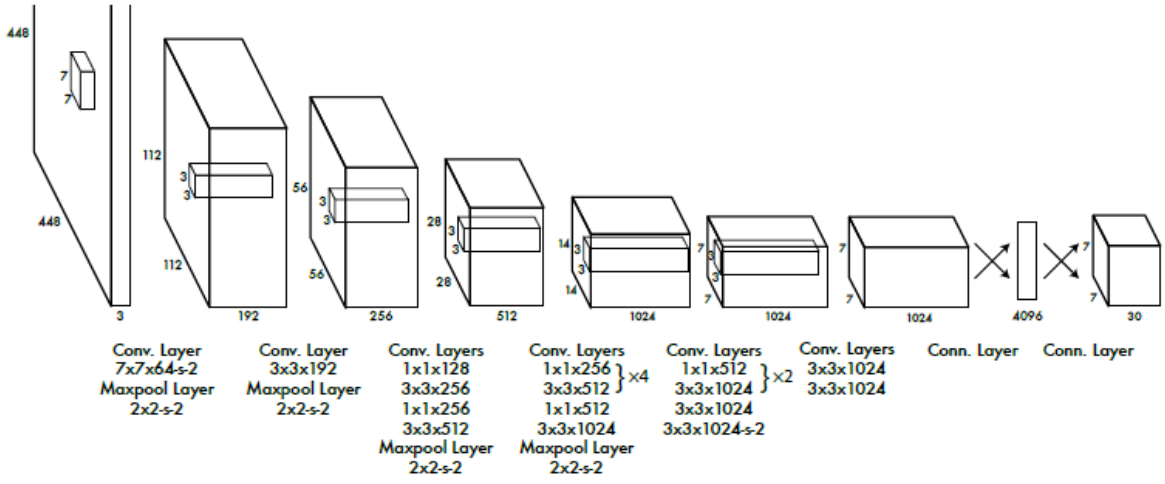
۲-۲-۲-۲-۲- روش مبتنی بر شبکه های عصبی عمیق

یولو، کامل‌ترین سیستم بلادرنگ (Real time) در یادگیری عمیق و حل مسائل تشخیص تصویر است که ورژن های مختلفی از آن در دسترس می باشد. همان‌طور که در تصویر زیر مشاهده می شود، این الگوریتم ابتدا تصویر را به بخش‌های مختلف تقسیم می‌کند و هر بخش را علامت‌گذاری می‌کند، سپس الگوریتم شناسایی را به صورت موازی برای تمامی این بخش‌ها اجرا می‌کند تا بررسی شود هر بخش به کدام دسته‌بندی تعلق می‌گیرد. بعد از شناسایی کامل اشیا، آن‌ها به هم متصل می‌شوند تا هر شیء اصلی یک جعبه باشد[۲۰۳].



شکل ۲-۱۹ نحوه عملکرد یولو در تشخیص اشیا موجود در تصویر[۲۰۳]

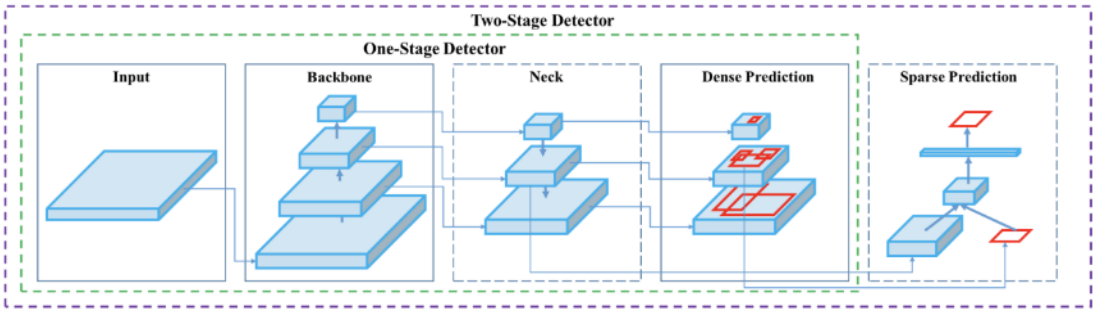
این عملکرد شناسایی به کمک شبکه ی زیر انجام می گردد :



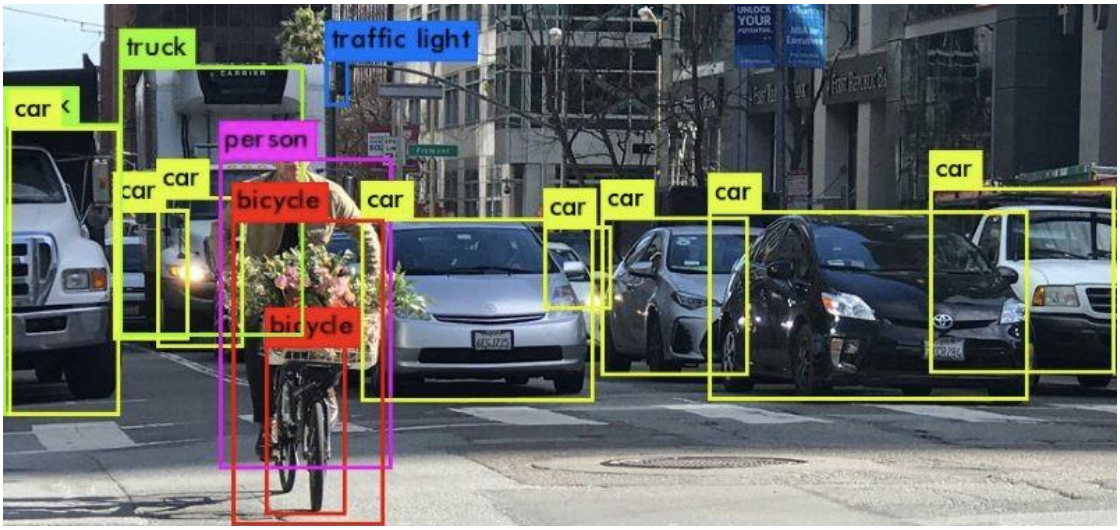
شکل ۲-۲۰ ساختار Yolov3 [۲۰۳]

اما یکی از ورژن های معروف و محبوب یولو، ورژن پنجم آن می باشد که سرعت و دقت آن به شکل قابل توجهی از بخش های دیگر بیشتر می باشد و در بخش چهارم به تفصیل به بررسی و مقایسه ی آن با دیگر روش ها خواهیم

پرداخت. قسمت پشتیبان[۱۵۲] این مدل که از آن عمدتا برای استخراج ویژگی های مهم از تصویر ورودی داده شده استفاده می شود، مدل بسیار قوی و شناخته شده ی CSP[۱۵۳] می باشد.[۲۰۴]



شکل ۲۱-۲ نحوه ی استخراج یک کلاس شی از یک تصویر با یولو [۲۰۴]



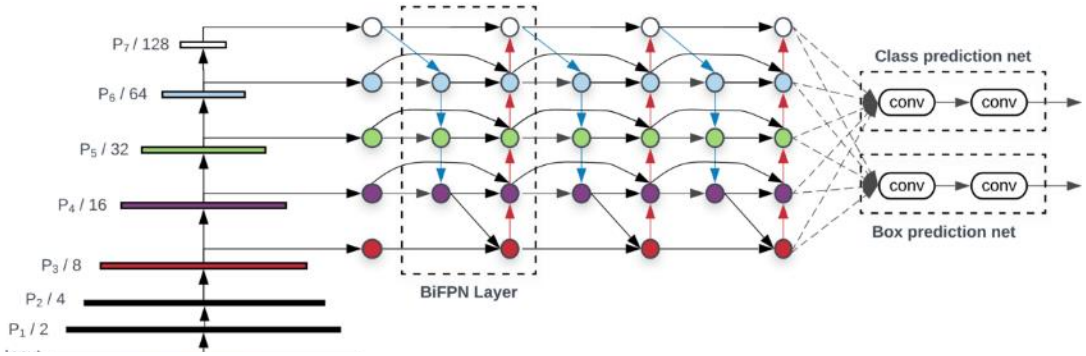
شکل ۲۲-۲ مشاهده کلاس های مختلف اشیا از جمله عابر دوچرخه سوار با شبکه ی یولو [۲۰۴]

EfficientDet -۲-۲-۲-۲-۳

با اختراع BiFPN ، خانواده جدیدی از آشکارسازها به نام EfficientDetایجاد شده است. معماری EfficientDetدر شکل زیر نشان داده شده است و از EfficientDetبه عنوان یک شبکه پشتیبان استفاده می کند. همانند یولو۵، مزایا و معایب این مورد نیز به طور مفصل در بخش چهار بررسی خواهد شد.[۲۰۵]

152 Backbone

153 Cross Stage Partial



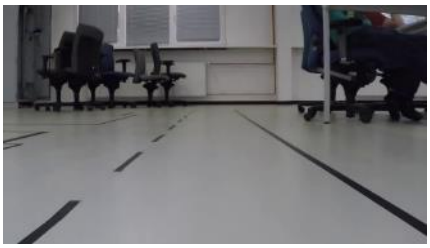
شکل ۲۳-۲ معماری EfficientDet [۲۰۵]

۳-۲-۲-۲- شناسایی خطوط جاده[۱۵۴]

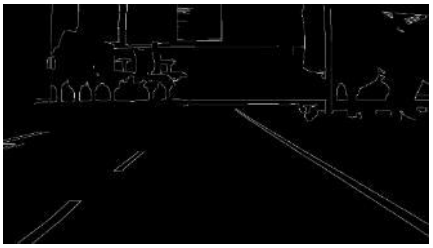
در این بخش، قصد داریم تا راه حل های مختلف و متنوعی را برای زیر مسئله ی شناسایی خطوط جاده بررسی کنیم و با چالش ها و روش های موجود در هر کدام از راه حل ها آشنا شویم.

۱-۳-۲-۲-۲- روش کلاسیک و مبتنی بر پردازش تصویر(روش مرسوم)

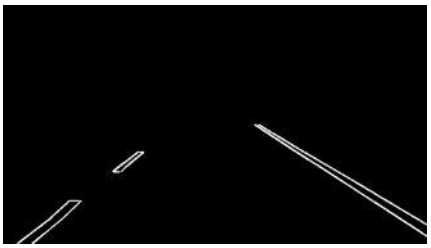
شناسایی خطوط جاده یکی از بخش های مهم برای یک خودرو خودران است که با استفاده از آن، خودرو مسیرحرکت خود را مشخص می کند. در این روش ابتدا از فیلتر های مختلفی از جمله فیلتر گوسی به منظور رفع نویز و هموارکردن تصویر استفاده می شود. سپس لبه های موجود در تصویر به کمک الگوریتم Canny [۲۰۶] پیدا می شوند و با بردن نقاط به فضای Hough می توان خطوط موجود در تصویر را پیدا کرد[۲۰۷]. در مرحله بعد با حذف قسمت های اضافی تصویر که محدوده آن با توجه به ابعاد خودرو و ارتفاع دوربین تعیین می شود، تصویری با خطوط مشخص و بدون نویز به دست می آید.



شکل ۲۴-۲ تصویر اصلی



شکل ۲۵-۲ لبه یابی با استفاده از الگوریتم Canny



شکل ۲۶-۲ بهبود خطوط پیدا شده با استفاده از Hough Transform

154 Lane and Road Detection

۲-۳-۲-۲-۲- روش مبتنی بر شبکه های عصبی عمیق

بر اساس دیتاست ها و نتایج بدست آمده، بنچمارکی برای شبکه های تشخیص خطوط جاده توسط سایت Paperswithcode گردآوری شده که در تصویر ۱۲ آورده شده است. در ادامه بهترین روش های موجود برای هر دیتاست مورد تحلیل قرار گرفته است[۲۰۸].

Benchmarks

TREND	DATASET	BEST METHOD	PAPER TITLE
	TuSimple	PINet	Key Points Estimation and Point Instance Segmentation Approach for Lane Detection
	CULane	LaneATT (ResNet-122)	Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection
	Caltech Lanes Washington	VPGNet	VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition
	Caltech Lanes Cordova	VPGNet	VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition
	BDD100k	ENet-SAD	Learning Lightweight Lane Detection CNNs by Self Attention Distillation

شکل ۲۷-۲ بنچمارک شبکه های موجود در تشخیص خطوط جاده[۲۰۸].

PINET ۲-۲-۲-۳-۳

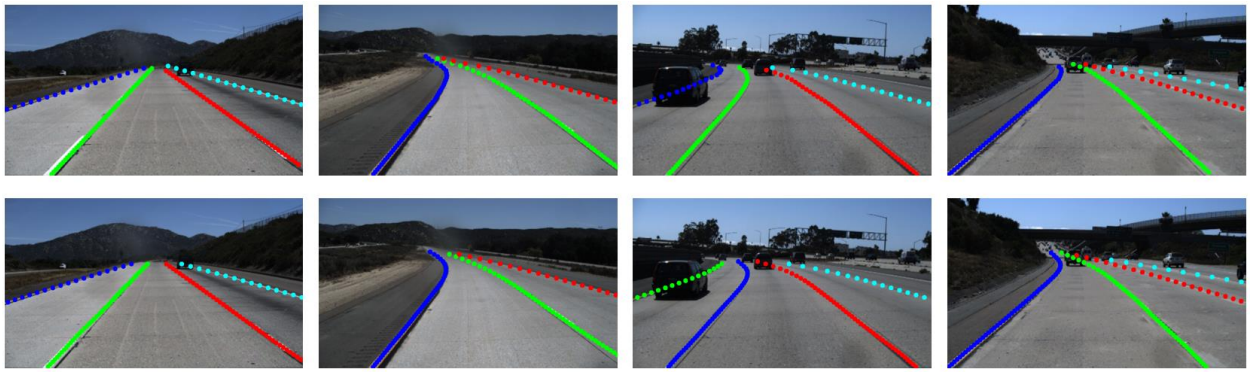
تکنیک‌های درک محیط برای رانندگی خودکار باید با شرایط مختلف سازگار باشد. در مورد تشخیص خط ترافیک بسیاری از شرایط باید در نظر گرفته شود، مانند تعداد خطوط ترافیک و قدرت محاسباتی سیستم هدف. برای رفع این مشکلات، در این مقاله روشی جدید پیشنهاد شده است که بر اساس تخمین نقاط کلیدی۱۵۵ و رویکرد تقسیم نمونه۱۵۶ عمل می کند. این شبکه شامل چندین شبکه ساعت شنی پشته ای۱۵۷ است که به طور همزمان آموزش می بینند. بنابراین می توان اندازه مدل های آموزش دیده را با توجه به توان محاسباتی محیط

155 Keypoint estimation

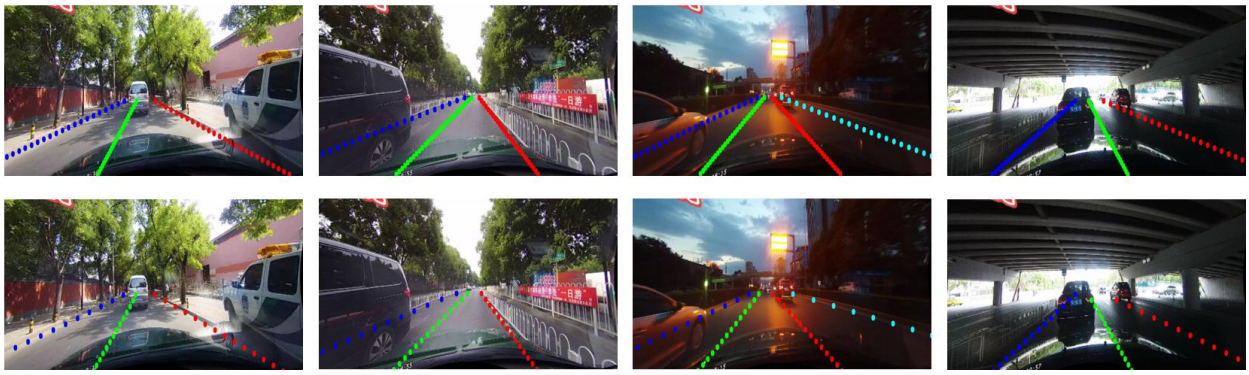
156 Instance Segmentation

157 Stacked hourglass networks

هدف انتخاب کرد. این شبکه در مجموعه داده های TuSimple و Culane ، که مجموعه داده های عمومی محبوبی برای تشخیص خطوط هستند، به دقت بالایی دست یافته که در دیتاست TuSimple دارای رتبه یک می باشد. در شکل های ۱۸-۳ و ۱۹-۳ می توان خروجی این شبکه را بر روی این دو مجموعه داده مشاهده کرد[۲۰۹].



شکل ۲۸-۲ نتایج بر روی دیتاست TuSimple، ردیف اول داده های مرجع و ردیف دوم نتایج پیش بینی شده توسط PINet [۲۰۹]



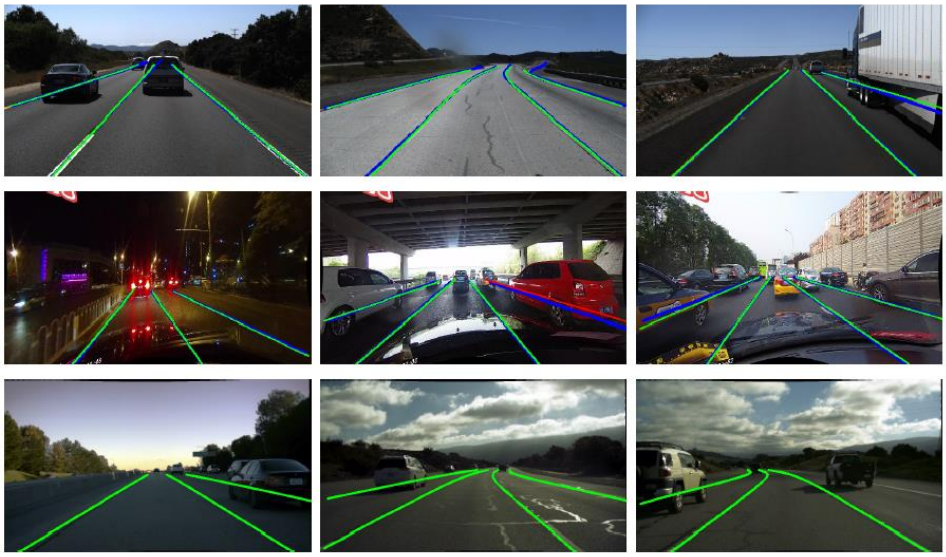
شکل ۲۹-۲ نتایج بر روی دیتاست CULane،ردیف اول داده های مرجع و ردیف دوم نتایج پیش بینی شده توسط PINet [۲۰۹]

LaneATT -۲-۲-۲-۳-۴

روش های مدرن تشخیص خطوط در شرایط پیچیده دنیای واقعی عملکرد چشمگیری داشته‌اند، اما بسیاری از آنها دارای مشکلات مربوط به حفظ کارایی در زمان واقعی هستند که برای وسایل نقلیه خودران مهم است. در این شبکه، یک مدل تشخیص خطوط عمیق مبتنی بر انکرها۱۵۸ ، که مانند سایر شبکه های های عمیق تشخیص اشیا، از انکرها برای مرحله ترکیب کردن ویژگی استفاده می کند. از آنجا که خطوط از یک الگوی منظم پیروی

158 Anchor

می کنند و ارتباط زیادی با هم دارند، فرض می شود که در برخی موارد ممکن است اطلاعات مکانی برای استنباط موقعیت آنها بسیار مهم باشد، به ویژه در شرایطی مانند انسداد[۱۵۹] ، نشانگرهای خط از دست رفته و موارد دیگر. بنابراین، در این کار مکانیسم توجه مبتنی بر انکر [۱۶۰]را ارائه می دهد که اطلاعات بدست آمده را ترکیب می کند. این مدل به طور گسترده بر روی سه دیتاست پرکاربرد یعنی TuSimple، CULane و LLAMAS ارزیابی شده است. نتایج موجود در تصویر نشان می دهد این شبکه نسبت به روشهای فعلی از کارآیی بالایی برخوردار است. همانطور که در شکل مشاهده می شود، این شبکه رتبه اول را در مجموعه داده Culane داراست[۲۱۰].



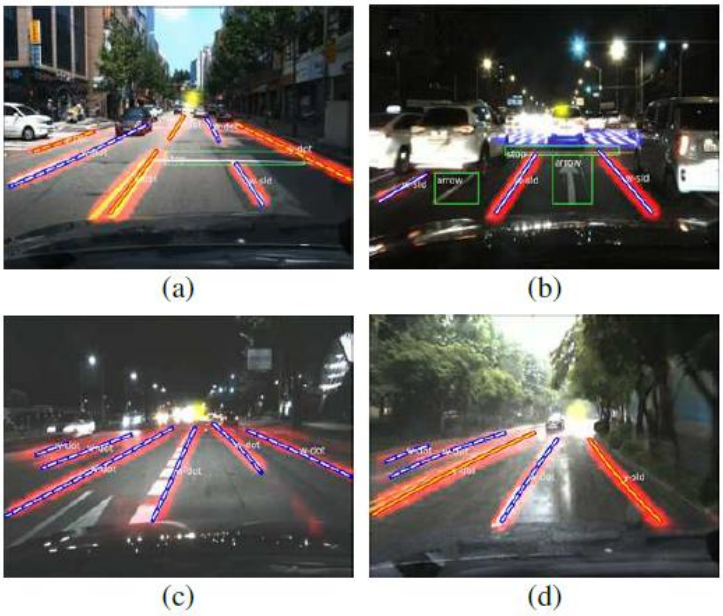
شکل ۳۰-۲ نتایج کیفی LaneATT در TuSimple (ردیف بالا) ، CULane (ردیف میانی) ، و LLAMAS (ردیف پایین). خطوط آبی داده های مرجع هستند، در حالی که خطوط سبز و قرمز به ترتیب خط درست تشخیص داده شده۱ و به اشتباه درست تشخیص داده شده۱ است[۷۷].

159 Occulusion

160 anchor-based attention

VPGNet -۲-۲-۲-۳-۵

در این مقاله، یک شبکه چند منظوره قابل آموزش از انتها به انتها[۱۶۱] ارائه شده است که دو کار تشخیص خطوط جاده و علامت گذاری های جاده را انجام می دهد که در شرایط نامساعد جوی نیز به خوبی عمل می کند. به عنوان مثال، تصاویر گرفته شده در روزهای بارانی تحت نور کم هستند، در حالی که جاده های مرطوب باعث انعکاس نور می شوند و ظاهر خطوط و خطوط جاده را مخدوش می کنند. در شب نیز تحریف رنگ زیر روشنایی رخ می دهد. برای رفع این نواقص، از یک معیار علامت گذاری خط و جاده استفاده شده است که شامل حدود ۲۰،۰۰۰ تصویر با ۱۷ کلاس خطوط علامت گذاری جاده و تحت چهار سناریو مختلف است: بدون باران، باران، باران شدید و شب. رویکرد حاصل VPGNet، می تواند خطوط جاده و خطوط علامت گذاری شده را تشخیص داده و طبقه بندی کند و با یک رویکرد جدید، نقاطی که مشخص نیستند را پیش بینی کند. نتایج تجربی نشان می دهد که این رویکرد تحت شرایط مختلف در زمان واقعی (۲۰ فریم در ثانیه) به دقت و کارایی بالایی دست می یابد که با توجه به تصویر ۱۲، رتبه یک را در دو مجموعه داده Caltech Lanes Washington و Caltech Lanes Cordova دارد[۲۱۱].



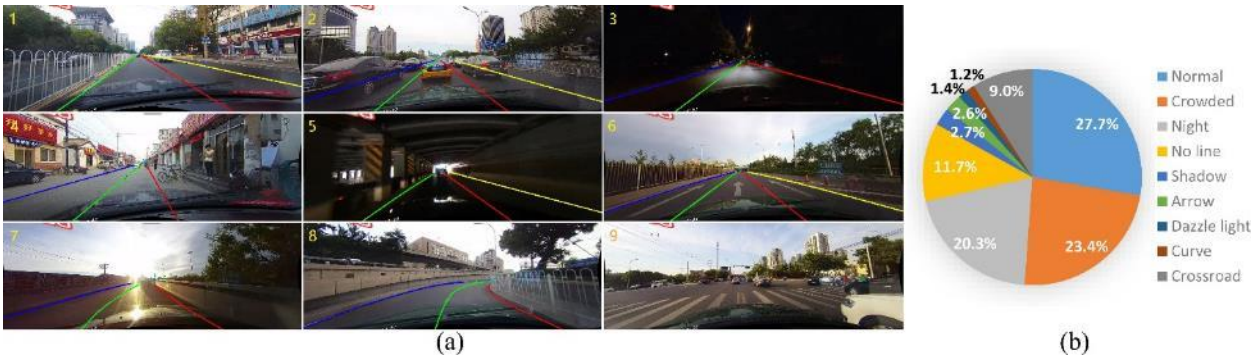
شکل ۳۱-۲ نمونه هایی از نتایج مشخص شده از خط و جاده توسط این شبکه: (الف) صحنه ای پیچیده شهری؛ (ب) تشخیص چندین علامت های موجود در جاده؛ ج) صحنه شب؛ (د) شرایط بارانی[۲۱۱].

161 End-To-End

ENet_SAD -۲-۲-۲-۳-۶

شبکه های کانوولوشنی معمولا به صورت انباشته شدن عملیات کانوولوشنی لایه به لایه ساخته می شوند. اگرچه CNN قابلیت بسیار قدرتمندی در استخراج ویژگی های با معنا از پیکسل های خام را دارد، اما ظرفیت آن برای بدست آوردن روابط فضایی پیکسل ها در میان ردیف ها و ستون های تصویر به طور کامل بررسی نشده است. این روابط برای یادگیری اشیا با معنا که دارای شکل های قوی اما انسجام ظاهری ضعیف، مانند خطوط ترافیکی، که معمولاً مسدود هستند یا حتی روی سطح جاده رنگ آمیزی نمی شوند، مهم هستند.

در این مقاله، برای رفع مشکل گفته شده SCNN ۱۶۲را پیشنهاد شده است، که شبکه های کانوولوشنی لایه به لایه را به کانوولوشن های قطعه قطعه شده در فضای ویژگی‌شان تعمیم می دهد، بنابراین امکان انتقال پیام بین پیکسل ها در ردیف ها و ستون های یک لایه را فراهم می کند. این ویژگی این قابلیت را به ما میدهد، اشکال و اجسامی که دارای روابط فضایی قوی اما سرنخ های ظاهری کمتری هستند را تشخیص دهیم. در شکل ۸ نمونه ای از خروجی این شبکه آورده شده است[۲۱۲].



شکل ۳۲-۲ (الف) مثال‌هایی از مجموعه داده برای سناریوهای مختلف. (ب) نسب سناریو های مختلف در آموزش این شبکه[۲۱۲].

۴-۲-۲-۲- ردیابی اجسام متحرک۱۶۳

ردیابی اشیا به معنای تخمین وضعیت شی هدف موجود در صحنه با توجه به اطلاعات قبلی است. به طور کلی ردیابی اشیا به دو دسته تقسیم می شود،

- Single Object Tracking

162 Spatial CNN

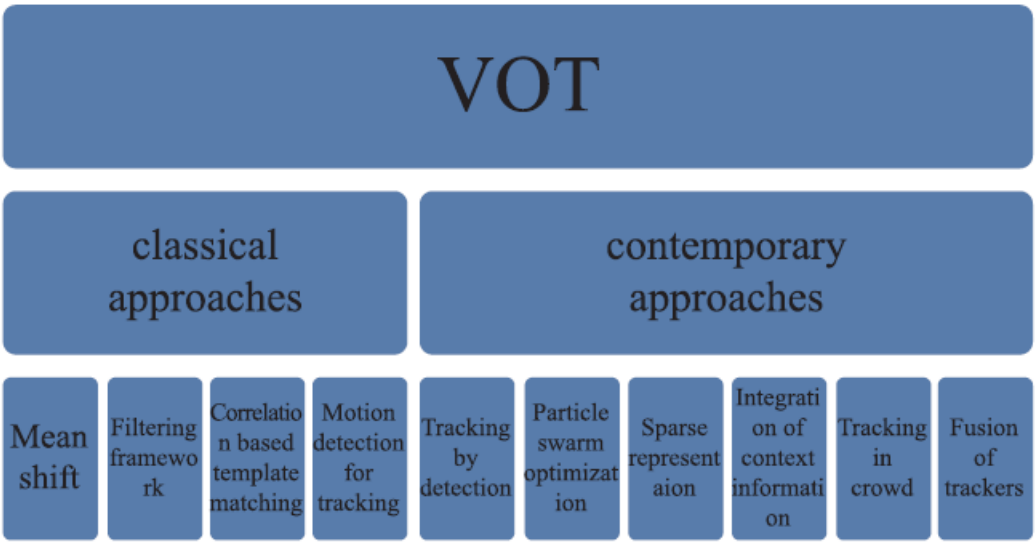
163 Object Tracking

- Multiple Object Tracking

در این بخش، قصد داریم تا راه حل های مختلف و متنوعی را برای زیر مسئله ی ردیابی اجسام متحرک ، بررسی کنیم و با چالش ها و روش های موجود در هر کدام از راه حل ها آشنا شویم.

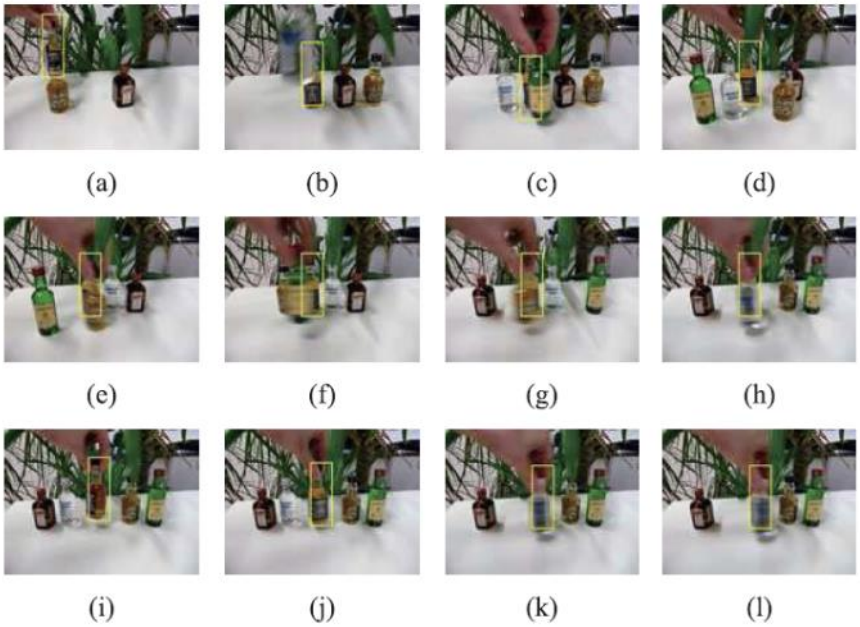
۱-۴-۲-۲-۲- روش کلاسیک و مبتنی بر پردازش تصویر

با توجه به دیاگرام ۲۳-۳، مجموعه ای از روش های کلاسیک پردازش تصویر با هم ترکیب شده اند تا بتوان حرکت یک جسم متحرک را دنبال نمود. این روش ها را در شکل زیر تحت عنوان VOT یا visual object tracking مشاهده می شود:



شکل ۳۳-۲ مجموعه ای از روش های کلاسیک برای ردیابی اجسام متحرک[۲۱۳]

در بحث ردیابی اجسام متحرک، رویکردهای کلاسیک و همچنین الگوریتم های ردیابی معاصر مورد بررسی قرار می‌گیرد. میانگین تغییر، فیلتر کالمن، تشخیص حرکت و الگوریتم های مبتنی بر تطبیق الگو به عنوان رویکردهای کلاسیک برای ردیابی اشیا ارائه می شوند، در حالی که ردیابی توسط تشخیص اشیا، هوش ازدحام ، نمایش پراکنده، ادغام اطلاعات و ردیابی در محیط شلوغ به عنوان الگوریتم های ردیابی معاصر مورد بحث قرار می گیرد[۲۱۳].

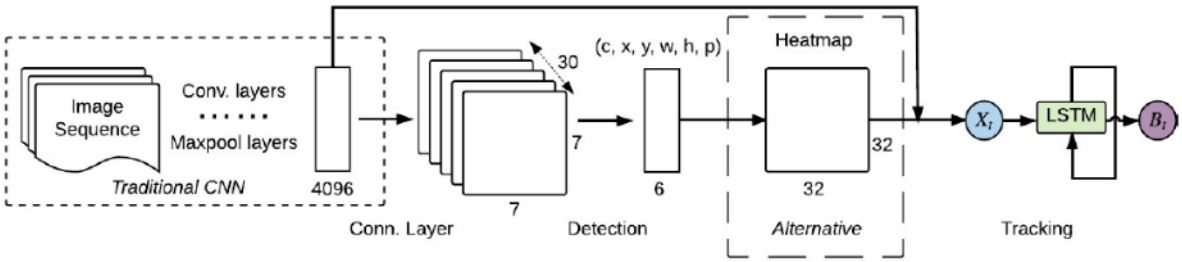


شکل ۳۴-۲ ردیابی شی مورد نظر در فریم های مختلف[۲۱۳].

۲-۴-۲-۲-۲- روش های مبتنی بر شبکه های عصبی عمیق

۱-۲-۴-۲-۲-۲- ROLO

رولو یک روش ردیابی برای یک شی واحد است که از ترکیب شناسایی اشیا۱۶۴ و شبکه‌های عصبی بازگشتی (LSTM) تشکیل شده است. در این شبکه یولو وظیفه استخراج ویژگی ها را بر عهده دارد و در هر فریم، LSTMیک بردار ویژگی ورودی به طول 4096 دریافت می‌کند و محل شی ردیابی شده را بر می گرداند[۲۱۴].



شکل ۳۵-۲ ساختار شبکه ROLO [۲۱۴]

164 Object detection

۲-۲-۲-۴-۲-۲- SiamMask

این شبکه مبتنی بر شبکه عصبی Siamese است. علاوه بر تولید جعبه های چرخشی دور شی با سرعت ۵۵ فریم در ثانیه، ماسک بخش بندی شی مورد نظر را نیز ارائه می دهد. برای دستیابی به این هدف، باید ابتدا یک جعبه را دور شی مورد نظر تنظیم کرد تا این شبکه بتواند شی مورد نظر را ردیابی کند. این بدان معنا است که ردیابی اشیا چندگانه (MOT) با SiamMask قابل اجرا نیست[۲۱۵].

شکل ۲-۳۶ مشخص کردن باکس اطراف شی

شکل ۲-۳۷ بدست آوردن ماسک و ردیابی آن توسط شبکه

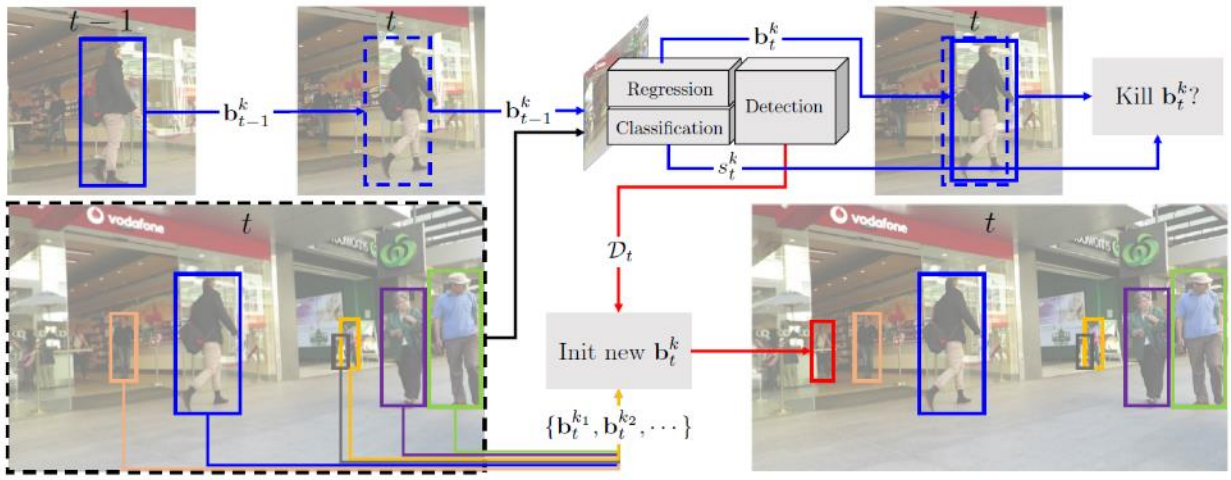
۲-۲-۲-۴-۲-۳- Deep SORT

Deep SORT الگوریتم پیشرفته شده SORT است که از نوآوری یادگیری معیار کسینوسی استفاده می کند. روش های کنترل مسیر۱۶۵ و کالمن فیلتر تقریبا مشابه روش SORT است، با این تفاوت که در این روش جعبه های محدود کننده با استفاده از یک شبکه عصبی کانوولوشن پیش آموزش دیده محاسبه می شوند. این روش یک نقطه شروع عالی برای تشخیص چندین شی موجود در تصویر است، زیرا اجرای آن ساده و دقت کاملی را ارائه می دهد و در عین حال به صورت زمان واقعی اجرا می شود[۲۱۵].

۲-۲-۲-۴-۲-۴- Tracktor++

این روش یکی از روش های محبوب در ردیابی اشیاست که رویکردی بسیار ساده دارد. این مدل با محاسبه رگرسیون جعبه محدود کننده، بدون نیاز به آموزش یا بهینه سازی در ردیابی داده ها، موقعیت یک شی را در فریم بعدی پیش بینی می کند. شناسایی اشیا در این روش با استفاده از Faster R-CNN صورت میگیرد[۲۱۶].

165 track handling



شکل ۳۸-۲ ساختار ردیابی اجسام توسط Tractor++ [۲۱۶]

۵-۲-۲-۲- تشخیص فاصله خودرو نسبت به خودرو های دیگر۱۶۶

در این بخش، قصد داریم تا راه حل های مختلف و متنوعی را برای زیر مسئله ی ۳-۲-۵تشخیص فاصله خودرو نسبت به خودرو های دیگر (به طور خاص خودروی جلویی) ، بررسی کنیم و با چالش ها و روش های موجود در هر کدام از راه حل ها آشنا شویم.

۱-۵-۲-۲-۲- با استفاده از پردازش تصویر کلاسیک:

در این بخش سعی خواهیم کرد فاصله تا ماشین را ارزیابی کنیم. هنگامی که توانستیم خودرو جلویی را شناسایی کنیم و نقطه میانی لبه پایین مستطیل انداخته شده دور آن را بدست آوریم، با استفاده از تبدیل پرسپکتیو، می توانیم موقعیت آن را بر روی تصویر تاب خورده محاسبه کنیم. روی تصویر بدست آمده، رابطه مستقیمی بین موقعیت پیکسل و فاصله در متر وجود دارد، بنابراین فاصله بین موقعیت محاسبه شده از نقطه میانی و پایین تصویر ضرب شده در تعداد متر در هر پیکسل ، نشان دهنده فاصله بین دو ماشین است. تصویر زیر نمونه ای از خروجی این فناوری محاسبه فاصله بین دو خودرو می باشد که از روش های کلاسیک پردازش تصویر بدست آمده است[۲۱۷].

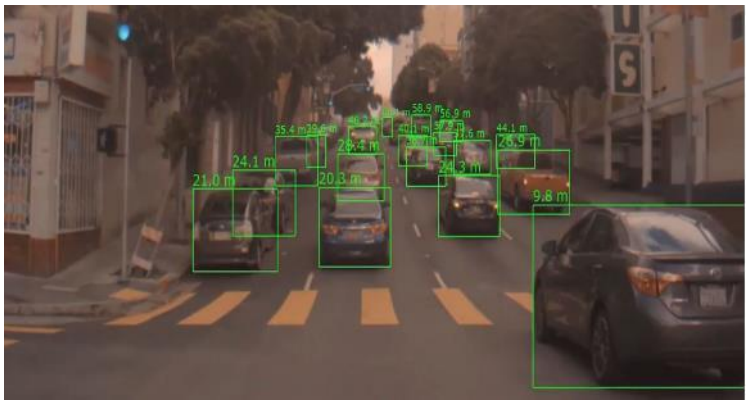
166Distance Measerment

۲-۵-۲-۲-۲- روش مبتنی بر شبکه های عصبی عمیق

با استفاده از شبکه های عصبی کانولوشنی و بازگشتی و داده های یک دوربین جلو میتوان فاصله خودرو جلویی را بدست آورد. این فناوری با استفاده از داده های حسگر رادار و لیدار به عنوان اطلاعات دقیق زمین، مسافت تا اجسام را پیش بینی می کند. ما می دانیم که این اطلاعات دقیق است زیرا بازتاب مستقیم سیگنال های رادار و لیدار منتقل شده، بدون در نظر گرفتن توپولوژی جاده، اطلاعات دقیق از فاصله را با جسم فراهم می کند.

با آموزش شبکه های عصبی بر روی داده های رادار و لیدار، این فناوری این امکان را می دهد تا فاصله از اشیا را از یک دوربین واحد تخمین بزنیم، حتی در زمانی که خودرو در سربالایی و سرپایینی قرار دارد. تصویر زیر نمونه ای از خروجی یک شبکه عصبی است که بر روی داده های رادار و لایدار آموزش داده شده است و فاصله سه بعدی از تصاویر دو بعدی را پیش بینی می کنند[۲۱۸].

شکل ۳۹-۲ بدست آورن Perspective Transform جاده و محاسبه فاصله پیکسلی نسبت به خودرو[۲۱۸]

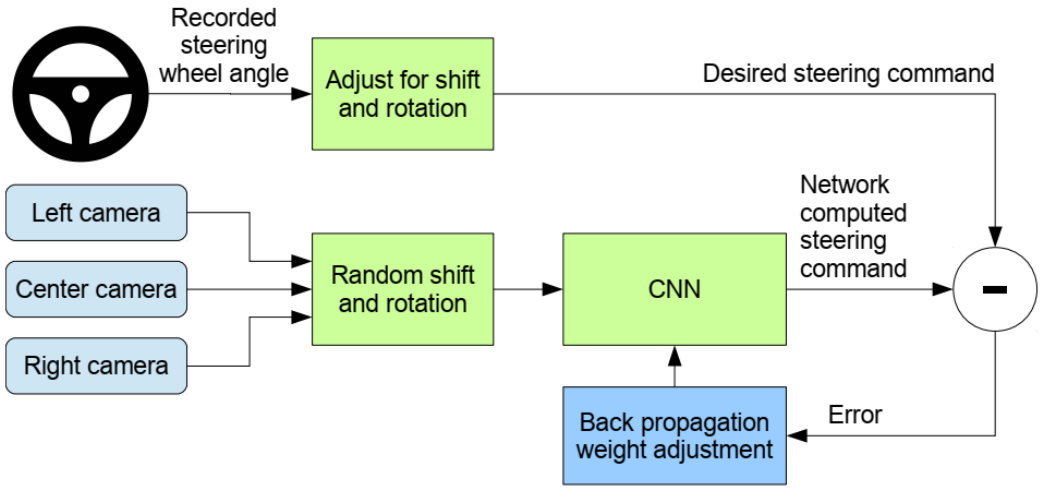


شکل ۴۰-۲ تعیین فاصله نسبت به خودرو های دیگر با آموزش شبکه های عصبی بر روی داده های رادار و لیدار[۲۱۸]

۶-۲-۲-۲- روش تقلید رفتاری[167]

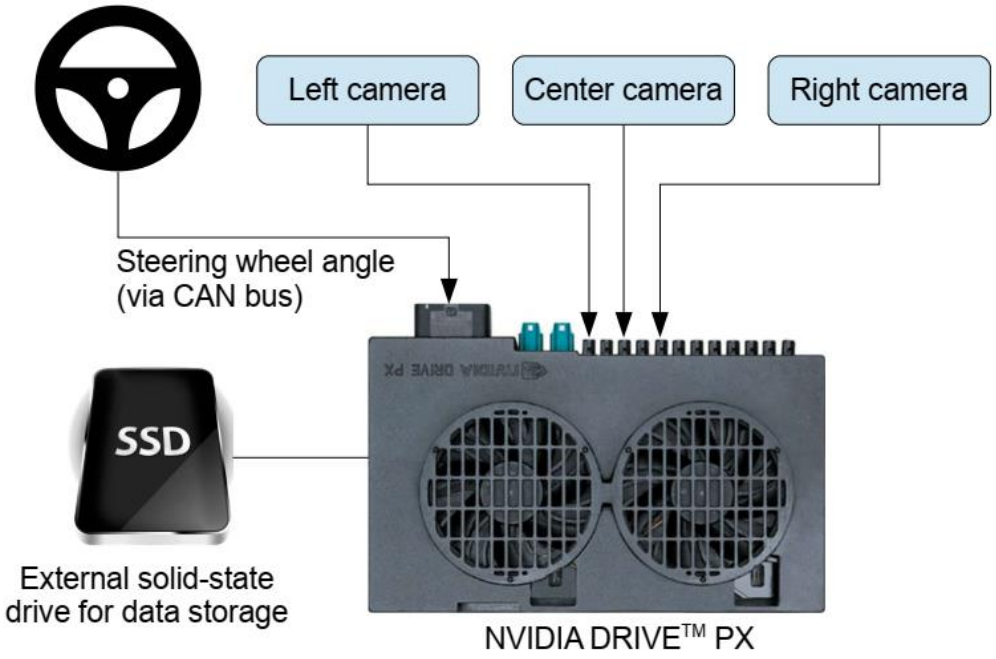
یکی از روش‌های کنترل خودروهای خودران استفاده از تقلید رفتاری انسان می‌باشد. بوجارسکی و همکاران در NVIDIA در سال ۲۰۱۶ با آموزش دادن یک شبکه ی کانولوشنی، موفق شدند که داده های دریافتی از یک دوربین را مستقیما به داده ی مورد نیاز برای چرخش فرمان خودرو تبدیل کنند. داده های مورد نیاز برای آموزش، شامل داده های جمع آوری شده از رانندگان انسان می باشد که در شرایط مختلف چه تصمیماتی را می گیرند. از نقاط قوت این روش، پایان به پایان بودن آن می باشد. به گونه ای که در این روش به صورت مجزا به استخراج ویژگی پرداخته نمی شود و مستقیما ورودی تصویر به خروجی زاویه ی فرمان تبدیل می شود[۲۱۹]. خلاصه ی روش به صورت زیر می باشد:

جمع آوری داده یرای این روش به این صورت است که یک راننده‌ی انسان برای مدتی رانندگی کرده و تصمیمات وی حین رانندگی برای چرخش فرمان ضبط می شود، در همه حال نیز تصاویر دریافتی از ۳ دوربین جلو، چپ و راست جمع‌آوری می شود. به این‌صورت که برای هر تصویر ضبط‌شده که ورودی خام شبکه می‌باشد، یک برچسب وجود دارد و آن زاویه‌ی فرمان خودرو در آن لحظه می باشد. سپس این تصاویر با برحسب‌های ذکر شده به شبکه داده شده و شبکه برای مدتی آموزش می بیند.



شکل۴۱-۲ نحوه ی انجام تقلید رفتاری [۸۷]

167 Behavioral Cloning



شکل ۴۲-۲ دیاگرام آموزش شبکه[۲۱۹]

برای تصمیم‌گیری نیز به این صورت عمل می شود که داده‌های دریافتی که همان تصاویر از ۳ دوربین جلو چپ و راست یا فقط تک‌دوربین جلو می‌باشد به شبکه داده شده و زاویه‌ی مناسب فرمان برای آن‌تصویر به دست می‌آید[۲۱۹]. علاوه بر داده‌های دوربین می‌توان از داده‌های سنسورهای مختلف نیز استفاده کرد.

۷-۲-۲-۲- تقسیم‌بندی[168]

این روش خود شامل چندین بخش مختلف است که ادامه آورده شده است.

۱-۷-۲-۲-۲- تقسیم بندی موردی [169]

علاوه بر شناسایی اشیا[170] که وجود اشیا مختلف در تصویر را تشخیص می‌دهد، گاها نیاز است که که دقیقا مشخص شود هرکدام از این اشیا در کجای تصویر هستند، برای این کار معمولا از روش‌های مبتنی بر یادگیری عمیق استفاده می‌شود که به توضیح آن پرداخته شده است. در سال ۲۰۱۸، هه و همکاران[171] یک روش عمومی و قابل انعطاف برای تقسیم‌بندی موردی ارائه کردند. این روش علاوه بر قابلیت تشخیص اشیا مختلف در تصویر، قابلیت خروجی دادن یک ماسک یا نقشه برای هر کدام از اشیا را دارا می‌باشد. این روش با اضافه کردن قابلیت پیش‌بینی همزمان ماسک هر شی و باندینگ‌باکس آن، عملکرد روش پیشین، Faster RCNN را بهبود داده است. همچنین این روش قابل تعمیم به سایر بخش های موجود در ادبیات مانند تشخیص موقعیت بدنی انسان را دارد. این روش که در سال ۲۰۱۷ ارائه شد، عملکرد بهتری نسبت به تمام الگوریتم های پیشین که برای تقسیم-بندی موردی ارائه شده‌بودند داشت. تقسیم‌بندی موردی در کل کار سختی محسوب می‌شود، زیرا علاوه بر آن‌که تمام اشیا موجود در تصویر باید به درستی تشخیص داده‌شوند، برای هرکدام از این‌اشیا باید مشخص شود که دقیقا در کدام قسمت تصویر قرار گرفته اند[۲۲۰]. در این روش از شبکه‌ی از پیش آموزش داده شده‌ی ResNet-101 به عنوان شبکه‌ی زیرساخت[172] استفاده شده است. ابتدا با استفاده از PRN173، باندینگ‌باکس‌های موجود در هر تصویر تخمین زده می‌شوند و سپس با استفاده از RoIPool، ویژگی های هر کدام از باندینگ‌باکس‌های تخمین‌زده‌شده استخراج می شود و سپس عملیات طبقه‌بندی و رگرسیون باندینگ باکس انجام می‌شود. مراحل ذکرشده تا اینجا در روش Faster-RCNN استفاده می‌شود. نوآوری جدید Mask-RCNN این است که برای هر کدام از باندینگ‌باکس‌های به‌دست‌آمده یک ماسک بخش بندی تولید می‌کند که این توالی برخلاف روش های قدیمی‌تر می‌باشد که ابتدا ماسک تصویر تولید می‌شد و سپس باندینگ‌باکس ها به دست می‌آمدند[۲۲۰].

168 Segmentation

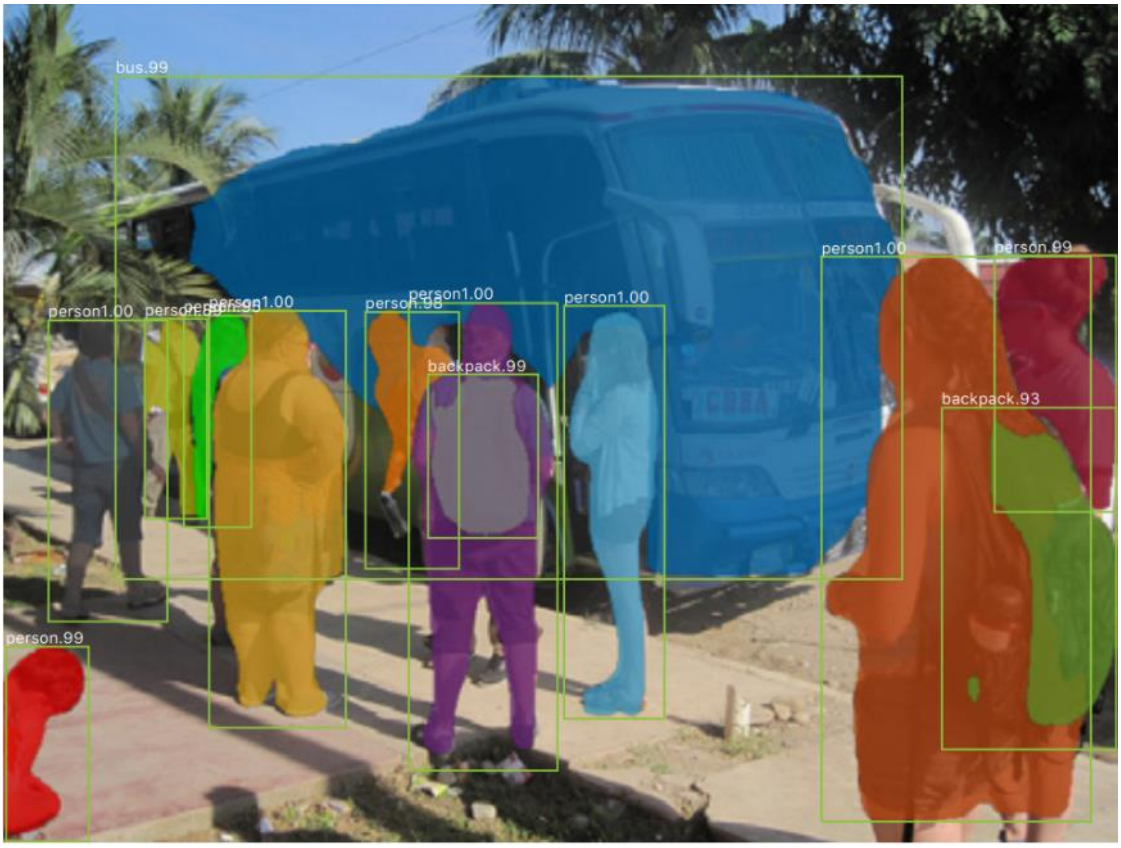
169 Instance Segmentation

170 Object Detection

171 He et al 2018

172 Backbone

173 Region Proposal Network



شکل ۴۳-۲ تشخیص اشیا و تقسیم بندی موردی در یک تصویر [۲۲۰].

۲-۷-۲-۲-۲- تقسیم‌بندی معنایی[174]

یکی از مواردی که در درک محیط باید به آن توجه شود، تقسیم‌بندی یا تفکیک معنایی آن محیط است. روش های مختلفی برای انجام این کار وجود دارد. در گذشته بیشتر از روش‌های مبتنی بر پردازش تصویر کلاسیک استفاده می‌شد. ولی با پیشرفت‌های انجام شده در قدرت پردازشی و الگوریتمی، روش‌های جدید غالبا مبتنی بر شبکه‌های عصبی عمیق می‌باشد. که در این‌گزارش به بررسی روش مبتنی بر یادگیری عمیق می‌پردازیم.

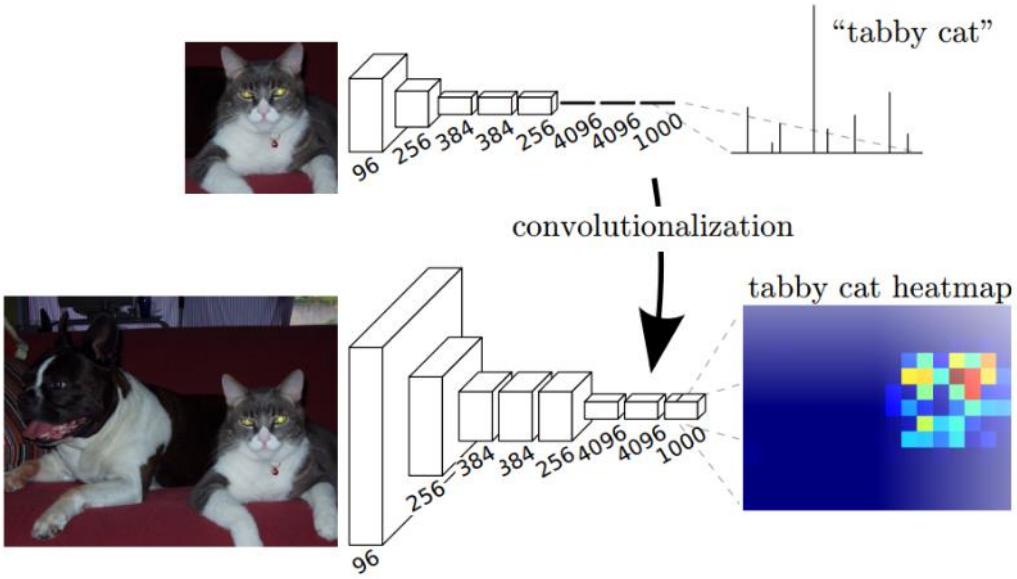
شبکه‌های عصبی کانولوشنی ابزار قدرتمندی هستند که ویژگی‌های سلسه‌مراتبی را به‌دست می‌دهند. لانگ**[175]** و همکاران نشان دادند که بااستفاده شبکه‌های عصبی کانولوشنی می‌توان تقسیم‌بندی معنایی انجام داد که می‌تواند عملکرد بهتری نسبت به سایر ابزارها داشته باشد. تفاوت این روش با سایر روش‌های موجود در ادبیات، استفاده کامل از شبکه های عصبی‌کانولوشنی و عدم استفاده از سایر روش‌های کلاسیک بوده است. در این مقاله جزئیات

174 Semantic Segmentation

175 Jonathan Long 2015

کاملی از شبکه‌های عصبی کاملاکانولوشنی داده‌شده است و با روش‌های قبلی مبتنی بر شبکه‌های عصبی کانولوشنی ارتباط داده‌شده است. در این تحقیق شبکه‌های معروف مانند VGGnet، GoogLeNet و AlexNet که در اصل برای انجام کلاس‌بندی استفاده می‌شوند، برای استفاده برای انجام تقسیم‌بندی معنایی تغییریافته‌اند و از اطلاعات یادگرفته شده توسط این‌شبکه‌ها برای انجام تقسیم‌بندی معنایی استفاده شده است. همچنین از یک ساختار اتصال‌پرشی بین لایه‌های عمیق و کم‌عمق شبکه ی اصلی استفاده شده است. این روش موفق شد که نسبت به بهترین روش آن‌موقع حدود ۲۰ درصد بهبود داشته‌باشد.

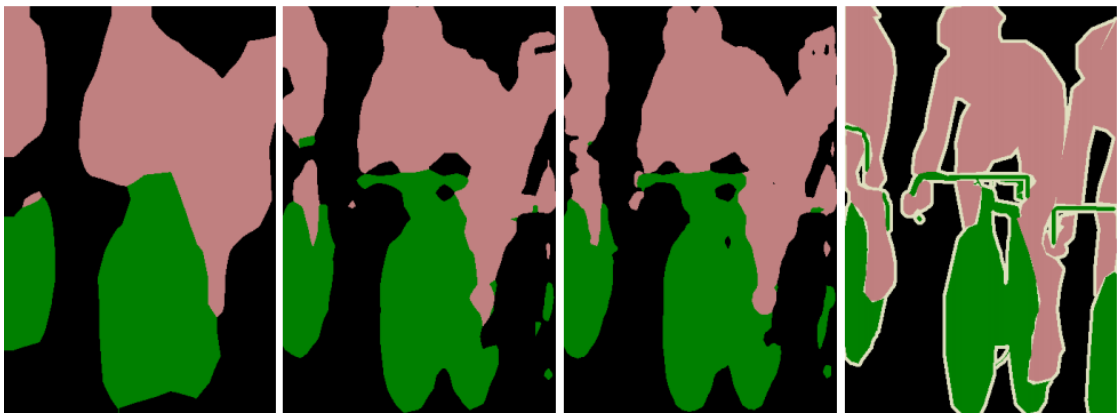
روش‌های کلاس‌بندی مانند LeNet، AlexNet و مدل‌های قبلی، معمولا یک تصویر با اندازه‌های معین را به‌عنوان ورودی دریافت کرده و خروجی‌های غیر فضایی به دست می‌دهند، به این صورت که خروجی آن‌ها یک عدد یا یک بردارکه در آن احتمال تعلق تصویر ورودی به یک یا چند مورد از کلاس‌های پیش تعیین‌شده مشخص شده است. اما شبکه های کاملا-کانولوشنی، ورودی با هر ابعادی را می پذیرند و خروجی آن‌ها به صورت یک نقشه ی تقسیم‌بندی می باشد که هر کدام از قسمت‌های تصویر با چه احتمالی شامل چه کلاسی می باشد. نمونه ای از این مقایسه در تصویر زیر نمایش داده شده است.



شکل ۴۴-۲ مقایسه‌ی شناسایی اشیا و تقسیم بندی معنایی [۲۲۱]

هرچه ابعاد قدم‌های طی‌شده[176] در این شبکه بیشتر می شود، خروجی دقت پایین‌تری پیدا می‌کند، که این مهم در تصویر زیر نمایش داده شده است. [۲۲۱]

176 Stride



شکل ۴۵-۲ مقایسه‌ی تقسیم بندی اشیا با قدم۱۷۷ های به ترتیب از چپ به راست ۸،۱۶،۳۲ پیکسلی و مقایسه با برچسب دستی (تصویر سمت راست) [۲۲۱].

۸-۲-۲-۲- مکان‌یابی۱۷۸

مسیریابی از مهمترین وظایفی است که خودروهای خودران که بر عهده‌دارند. مسیریابی نیازمند درک محیط فرای چیزی است که سنسورهایGPS به‌دست می‌دهند. لوینسون و همکاران۱۷۹ نشان دادند که داده‌های دریافتی از GPS، IMU و لیدار می تواند برای تولید نقشه‌هایی با دقت بالا از محیط اطراف خودرو مورد استفاده قرار گیرد. در این مقاله، آن‌ها به ارائه ی روش جدیدی می‌پردازند که نسبت به روش قبلی ارائه‌شده توسط خودشان بهبود چشمگیری داشته است. در این روش به جای مدل کردن محیط به عنوان یک توری‌فضایی۱۸۰ با سلول‌های ثابت، از یک توری احتمالاتی۱۸۱ استفاده می‌شود که هر سلول آن شامل اطلاعات توزیع گوسی مادون قرمز می باشد. آن‌ها این روش جدید تشخیص موقعیت خودرو را در مقایسه با روش قدیمی تست کردند که خطای میانگین مربعات روش جدید در محدوده ی ۱۰ سانتی متر بود. و این مهم به تیم این امکان را داد که خودرو را به صورت از راه دور تا ۱۰۰ ها کیلومتر در ترافیک زیاد شهری به حرکت درآورند.

177 Stride

178 Localization

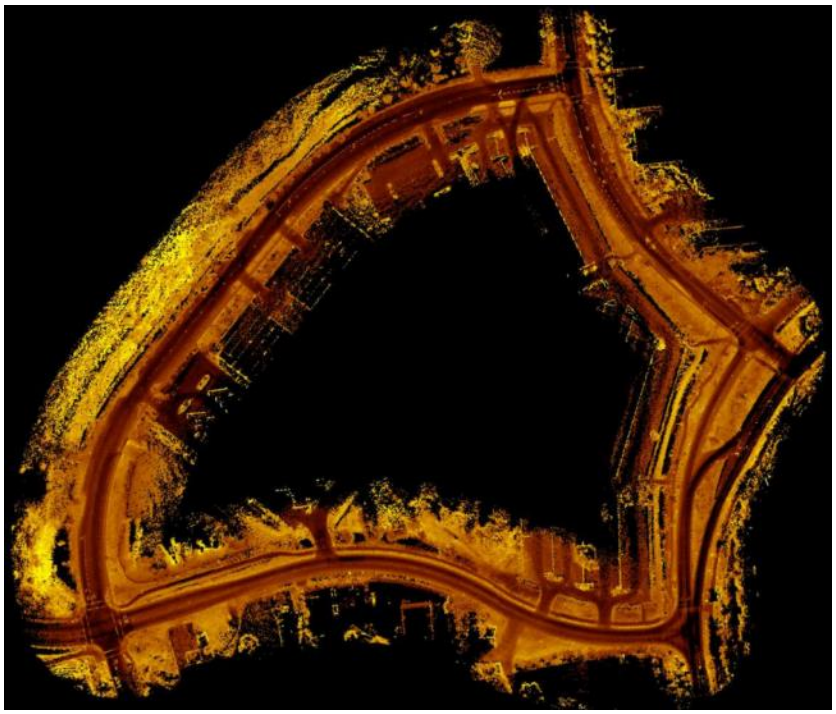
179 Levinson et al

180 Spatial Grid

181 Probabilistic Grid

صدف نهایی این مقاله ارائه‌ی یک نقشه‌ی سلولی-توری۱۸۲ می‌باشد که در هر سلول میانگین و واریانس بازتابش مادون‌قرمز در آن سلول موجود می باشد. روش کلی به صورت زیر می‌باشد:

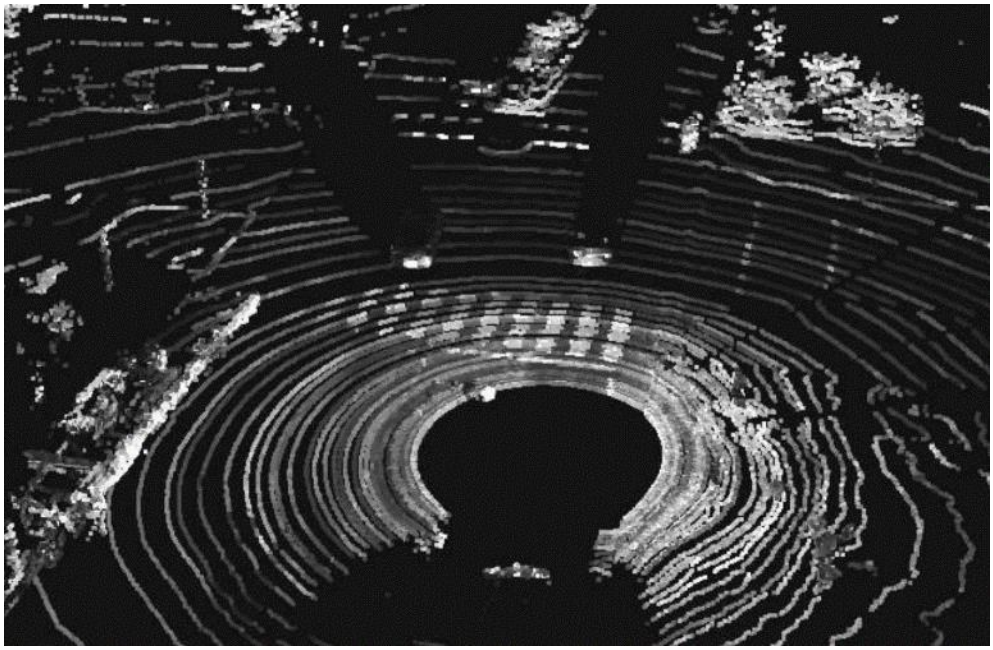
- در مرحله‌ی اول، مسیرها پس پردازش می‌شوند تا محل‌هایی که با هم همپوشانی دارند در یک راستا قرار گیرند.
- سپس شدت بازتابش هرکدام از پرتوهای تابیده‌شده کالیبره می‌شوند که منحنی‌برگشت مشابهی داشته‌باشند.
- در نهایت، داده‌های کالیبره‌شده از لیزر برای مسیرهایی که در یک راستا قرار گرفته‌اند، برروی یک نقشه‌ی پروضوح نمایش داده می‌شوند.[۲۲۲]



شکل ۴۶-۲ نقشه‌ی بازگشتی مادون قرمز [۲۲۲]

همان‌طور که ذکر شد، خروجی نقشه برای آنکه قابل استفاده باشد، باید کالیبره شده باشد. در تصاویر زیر خروجی کالیبره‌نشده و کالیبره‌شده نمایش داده‌شده است.

182 Grid Cell



شکل ۴۷-۲ نقشه با خروجی کالیبره نشده [۲۲۲]

شکل ۴۸-۲ نقشه با خروجی کالیبره شده [۲۲۲]

۳-۲- تحلیل راه حل های زیر مسئله ها

در این بخش با توجه به پژوهش‌های مرور شده در بخش قبل، به بررسی راه حل‌های نهایی برای زیر مسائل پرداخته شده است.

۱-۳-۲- پارک خودکار

ابتدا چارچوب حل مسئله را مشخص می‌کنیم . همان طور که در بخش سوم گفته شد، مسئله‌ی پارک خودکار به سه زیرمسئله‌ی ادراک، برنامه‌ریزی مسیر و دنبال کردن مسیر با روش کنترلی تقسیم می‌شود. مسئله‌ی تعریف شده در این مسابقه در قالب یک ماشین واقعی در دنیای حقیقی نیست و به صورت شبیه‌سازی می‌باشد. در نتیجه راه حل‌ها را با چارچوب حل مسئله انطباق می دهیم.

با توجه به مسئله‌ی تعریف شده، از زیرمسئله‌ی ادراک استفاده نمی شود و موقعیت اولیه خودرو و جای پارک از اول مشخص است. با این حال ما روش‌های مختلف ادراک را تحلیل و مقایسه کرده ایم. ولی در این پروژه از این بخش استفاده نمی شود.

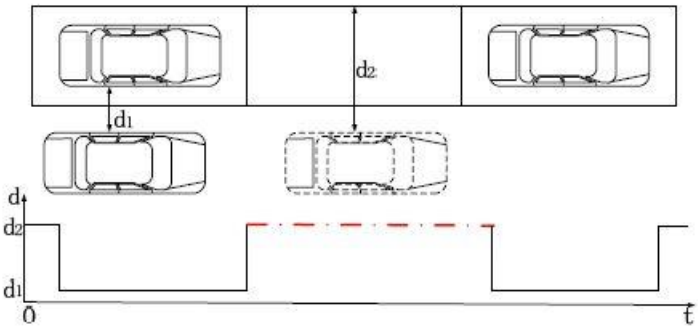
۱-۱-۳-۲- ادراک

مرحله‌ی اول پارک خودکار، ادراک محیط شامل موانع و سایر ماشین‌های پارک شده است. سیستم پارک اتوماتیک اطلاعات مربوط به فضا را در کنار ماشین اسکن می کند. اگر فضای خالی ای به منظور پارک یافت شود و تشخیص داده شود که فضای پارک برای اندازه ی اتومبیل مناسب است، سیستم وارد مرحله‌ی بعد یعنی برنامه ریزی مسیر می شود. با توجه به چارچوب مسئله و محیط شبیه سازی مسئله، استفاده از سنسورهای مختلف برای درک محیط به صورت داده‌های شبیه سازی می‌باشد. در این بخش، برخی مزایا و معایب سنسورها را برای درک محیط ذکر می کنیم.

۱-۱-۱-۳-۲- اولتراسونیک

کاوشگر اولتراسونیک می تواند اطلاعات فاصله را تشخیص دهد. اگر وسایل نقلیه کنار جاده پارک شده باشند ، سیستم d1 (متر) و در غیر این صورت d2 (متر) دریافت می کند. که در شکل ۵۰-۲ نشان داده شده است. اگر فضای پارکینگ در دسترس باشد، می توان نبض در خط نقطه ای را بدست آورد، سپس طول فضا با توجه به سرعت و مدت زمان پالس بدست می‌آید. این فضا در صورت طولانی تر بودن از کمترین طول مکان پارک ممکن برای اتومبیل، مناسب خواهد بود. اگر فضا در دسترس نباشد ، اتومبیل تا پیدا کردن مکان پیش می‌رود. پس از دریافت فضای موجود، سیستم وارد مرحله برنامه ریزی مسیر می شود[۲۲۳].

شکل ۲-۴۹ استفاده از سنسور اولتراسونیک در خودرو



شکل ۲-۵۰ موقعیت ماشین نسبت به ماشین‌های پارک شده

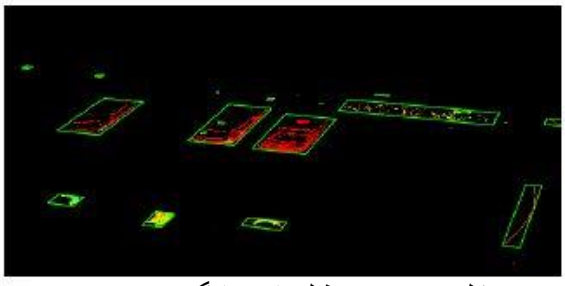
مزایا و معایب :

مزیت: استفاده از اولتراسونیک هزینه‌ی کمی به همراه دارد.

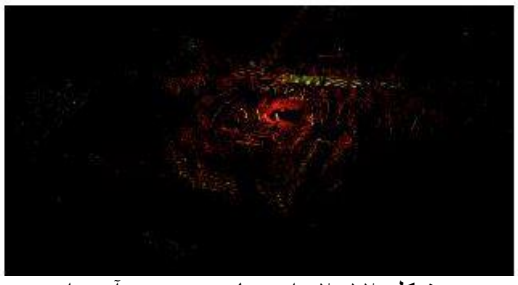
عیب: سنسور اولتراسونیک نمی تواند با شرایطی که دو طرف جای پارک خالی است استفاده شود.

۲-۱-۱-۳-۲- لیدار

در استفاده از لیدار، منطقه مورد نظر (ROI) را تعریف می‌کنیم. فاصله سمت چپ و راست LiDAR، ۸ و ۱۰ متر و فاصله تا جلو و عقب ۱۶ و ۸ متر است. ما به اطلاعات اتومبیل های پارک شده و موانع اهمیت می‌دهیم. داده های زمینی و داده های نقطه ای درخت برای ما بی فایده است. فاصله از LiDAR تا زمین ۲ متر است. و درخت بلندتر از LiDAR است. بنابراین از فیلتر برای حفظ داده ها $-2m<z<0m$ استفاده می شود. سرانجام داده های به دست آمده به صفحه XOY طرح‌ریزی می‌شوند و حداقل مستطیل محدود کننده (MBR) موانع و ماشین‌ها جایگزین می‌شوند. شکل ۲-۵۱ داده‌های به دست آمده و شکل ۲-۵۲ مستطیل‌های جایگزین شده را نشان می‌دهند[۲۲۴].



شکل ۲-۵۱ مستطیل‌های جایگزین شده[۹۲].



شکل ۲-۵۲ داده-های به دست آمده از لیدار [۹۲].

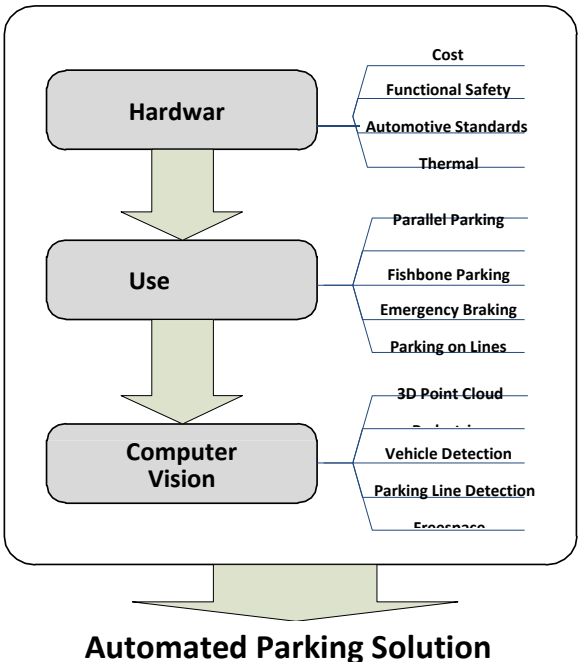
مزایا و معایب:

مزیت : با استفاده از لیدار می توان وضعیت پارکینگ موازی و یا پارکینگ عمود را بر اثر وضعیت ماشین های پارک شده استنباط کرد. همچنین دقت بسیار بالایی در این روش وجود دارد.

عیب: سنسورهای لیدار نسبت به سایر سنسورها بسیار گران هستند و صرفه‌ی اقتصادی ندارند.

۳-۱-۱-۳-۲- دوربین و پردازش تصویر

همان طور که در بخش سوم گفته شد برای ادراک محیط می توان از دوربین استفاده کرد. برای تشخیص مکان‌های پارک در صورت عدم وجود موانعی در تعریف جای پارک، تشخیص علائم جاده ای که فضای پارک را تعریف می کنند بسیار مهم است. در استفاده از دوربین و بینایی ماشین، می توان با استفاده از اصلاح تصویری، استخراج لبه و تجزیه و تحلیل فضا مکان مناسب پارک را حتی در نبود سایر ماشن‌های پارک شده تشخیص داد[۲۲۵]. در نمودار ۲-۵۳ الگوریتم‌های مختلف بینایی ماشین را مشاهده می کنیم.



نمودار۵۳-۲ الگوریتم‌های مختلف بینایی ماشین

مزایا و معایب:

مزیت: به طور کلی درک محیط به وسیله‌ی دوربین و بینایی ماشین از رزولوشن بیشتری برخوردار است و به وسیله‌ی پردازش تصویر می توان موانع را دسته بندی کرد و بین انسان، ماشین و سایر موانع تمایز قائل شد که این ویزگی در راستای پارک خودکار مفید خواهد بود و عملیات پارک را دقیق تر یکند.

عیب: دوربین‌ها دارای نقاط کور هستند و لذا ممکن است با تعداد پایین دوربین، بعضی موانع را نبینیم و برای حل این مشکل باید از تعداد بیشتری دوربین استفاده شود. و تعداد بیشتر دوربین هزینه‌ی بیشتری در پی خواهد داشت.

با توجه به ویژگی‌های گفته شده برای هر روش، استفاده از سنسورهای اولتراسونیک مقرون به صرفه تر است و پاسخگوی خواسته‌ی این زیر مسئله و پیدا کردن جای پارک و تشخیص موانع می باشد. اولویت دوم به دلیل تمام مزایای گفته شده، استفاده از دوربین و پردازش تصویر خواهد بود.

نکته : با همه‌ی این تفاسیر، همان طور که در ابتدا گفته شد، در چارچوب این مسئله از بخش ادراک استفاده نمی‌شود.

۲-۱-۳-۲- برنامه ریزی مسیر

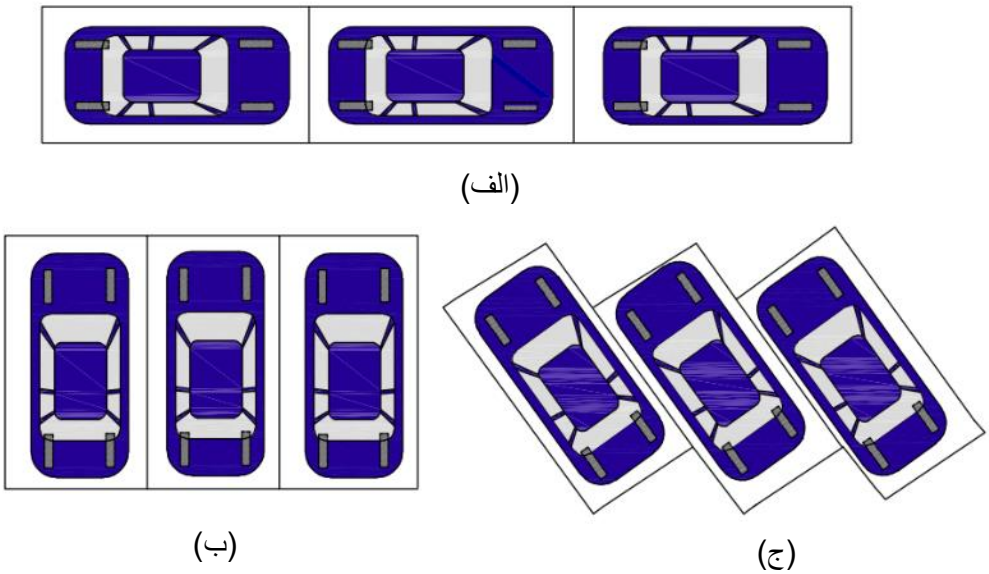
همان طور که در بخش سه گفته شد، برنامه ریزی مسیر را می توان به دو دسته‌ی روش برنامه ریزی جهانی، برای مسیر مبتنی بر اطلاعات شناخته شده محیط و روش برنامه ریزی مسیر محلی، مبتنی بر اطلاعات محیطی نامشخص تقسیم کرد. جدول ۲-۲ مزایا و معایب این دو روش را به نمایش می گذارد.

جدول ۲-۲، مزایا و معایب روش‌های برنامه‌ریزی مسیر

روش برنامه‌ریزی مسیر	مزایا	معایب
برنامه‌ریزی مسیر جهانی	معمولاً می تواند مسیر بهینه را پیدا کند	از قبل اطلاعات دقیق محیط باید دانسته شود. مقدار زیاد محاسبه و سازگاری ضعیف با محیط پویا
برنامه ریزی مسیر محلی	در زمان حال بودن، عملی بودن	دستیابی به مسیر بهینه جهانی دشوار است.

با مقایسه این دو روش و با توجه به چارچوب حل مسئله در محیط شبیه‌ساز، برای رسیدن به مسیر بهینه روش برنامه ریزی مسیر جهانی را انتخاب می کنیم.

انواع اساسی پارک عبارتند از: پارک موازی ، پارک عمودی و مورب. این پارک‌ها در شکل ۵۴-۲ نشان داده شده اند.



شکل ۵۴-۲ ، انواع روش‌های پارک. (الف) پارک موازی (ب) پارک عمودی (ج) پارک زاویه دار[۲۲۶].

- **پارک عمودی(گاراژ) :**

برای ما مهم است که مسیرهای مرجع معقولی را ارائه دهیم که اتومبیل بتواند با موفقیت ماموریت پارکینگ گاراژ را انجام دهد. اگر مسیر مرجع به دلیل پویایی بی مدل یا عدم دقت در مدل سازی، از درست عملی بودن دور باشد ، در این صورت خودرو قادر به پیگیری دقیق مسیر نیست. برای جلوگیری از ناهنجاری، ما باید یک مسیر مرجع دقیق برای ردیابی انتخاب کنیم. برای ارزیابی امکان سنجی و اثربخشی طرح، دو آزمایش برای پارک گاراژ - پارک گاراژ رو به عقب و پارک گاراژ رو به جلو ایجاد شده است. در پارک عقب گاراژ، طبق تجربه رانندگی، همیشه فرمان را به سمت راست و عقب ماشین می‌چرخانیم. سپس ماشین در هنگام ورود به گاراژ منجر به یک مسیر قوس می شود. بنابراین، یک چهارم دایره برای تشکیل این خط سیر استفاده می شود. از آنجا که امیدواریم ماشین بتواند مستقیماً در گاراژ عقب بماند، در انتهای چهارم دایره به عنوان یک مسیر دلخواه یک خط می کشیم. به عبارت دیگر، مسیر مرجع برای پارکینگ گاراژ به عقب شامل یک حرکت دایره ای و یک حرکت مستقیم است. مسیر مرجع عقب هنگام پارک عقب گاراژ به عنوان یک عملکرد نشان داده می شود[۲۲۷].

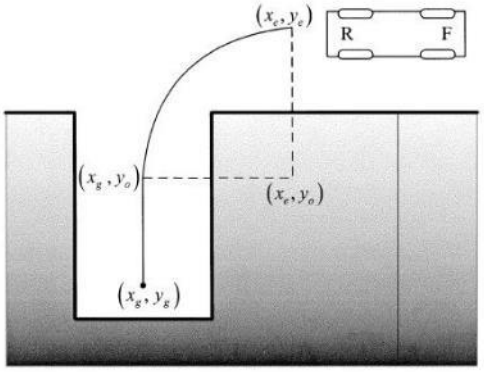
فرم کلی حرکت دایره ای به صورت زیر می‌باشد.

$$(x_r - x_e)^2 + (y_r - y_o)^2 = (x_e - x_g)^2.$$

و حرکت خطی نیز به صورت زیر است.

$$x_r = x_g \quad \text{and} \quad y_g \le y_r \le y_0.$$

شکل ۵۵-۲ پارک عمودی از عقب را نشان می‌دهد.



شکل ۵۵-۲ پارک عمودی از عقب[۲۲۷].

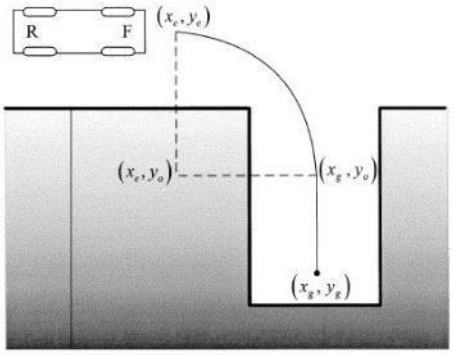
اگر وسیله نقلیه بتواند این مسیر مرجع را به طور کامل دنبال کند ، قطعاً می توان گفت که وسیله نقلیه به درستی در گاراژ پارک می کند. مسیر مرجع پارک گاراژ از جلو نیز به صورت زیر تعریف می شود.

$$(x_f - x_e)^2 + (y_f - y_o)^2 = (x_g - x_e)^2.$$

فرم کلی حرکت دایره ای به صورت روبه‌رو می‌باشد.

$$x_r = x_g \quad \text{and} \quad y_g \le y_r \le y_0.$$

حرکت خطی به به این صورت می باشد. شکل ۵۶-۲ پارک عمودی از جلو را نشان می‌دهد.



شکل ۵۶-۲ پارک عمودی از جلو[۲۲۷].

پارک مورب عملکردی شبیه به پارک عمودی دارد. با این تفاوت که زاویه‌ی پارک نیز باید در معادلات لحاظ شود.

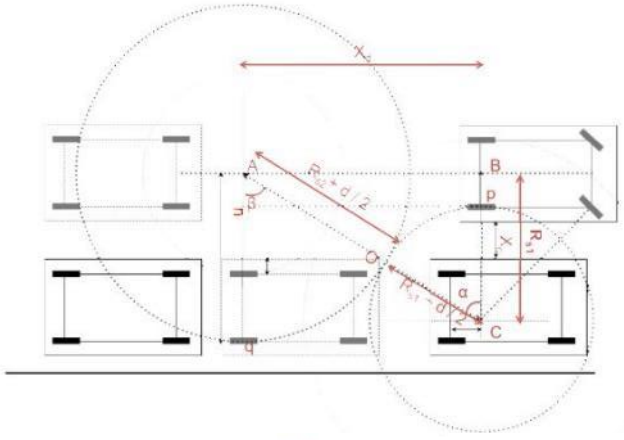
- **پارک موازی:**

به منظور اجرای پارک موازی، یک مسیر عملی و صاف را اتخاذ می‌شود به گونه ای که اتومبیل می‌تواند آن را ردیابی کند. در حقیقت، از منحنی های مختلفی برای ارائه مسیر مرجع استفاده شده است مانند دو قوس دایره ای، منحنی سینوسی، منحنی کسینوس، و یک منحنی چند جمله ای مرتبه پنجم و غیره. این منحنی ها با موقعیت اولیه ماشین و موقعیت نهایی پارکینگ تعیین می‌شوند. با مقایسه روش‌های مختلف یک مسیر بهینه به صورت زیر تعریف می‌شود.

برنامه ریزی مسیر شامل معادلات هندسی ساده در این سیستم است. مسیری که وسیله نقلیه قبل از مانور در محل پارک موازی طی می کند، کاملاً تراز شده، سه بخش در نظر گرفته شده است. همانطور که در شکل۷-۴ نشان داده شده است، یک خط مستقیم و دو قوس دایره. در طول کل کار پارک، چرخ فقط باید دو بار در نقطه "p" و "o" زاویه خود را تراز کرده و تغییر دهد. این امر نه تنها خطاهای احتمالی ناشی از فرمان مکرر را کاهش می دهد بلکه مصرف انرژی کمتری را نیز به همراه دارد و از این رو از نظر انرژی صرفه جویی و ساده است. کل مسیر پارک فقط از دانش فاصله بین اتومبیل و وسیله نقلیه از قبل پارک شده محاسبه می شود که از بخش ادراک فاصله بدست می آید[۲۲۸].

حالت ایده آل ، وقتی ماشین موازی با ماشین پارک شده باشد. خط سیر اتومبیل در شکل ۵۷-۲ نشان داده شده است.

برای در نظر گرفتن تمام شرایط ممکن مسئله، حالت غیر ایده آل را نیز در نظر می گیریم.

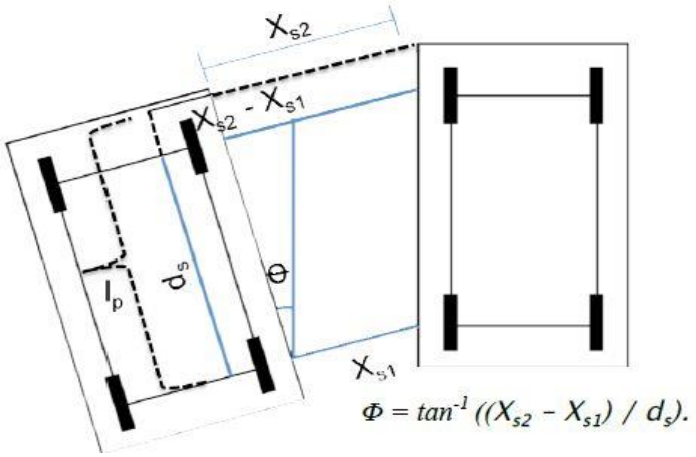


$$(X_d)^2 = (AB)^2 = (AC)^2 - (CB)^2 \qquad \alpha = \tan^{-1}(X_d / R_{s1}) \qquad \text{length(po)} = (\alpha / 360) \times 2\pi \times (R_{s1} - d/2)$$

$$\beta = \alpha. \qquad \text{length(oq)} = (\beta / 360) \times 2\pi \times (R_{s2} + d/2).$$

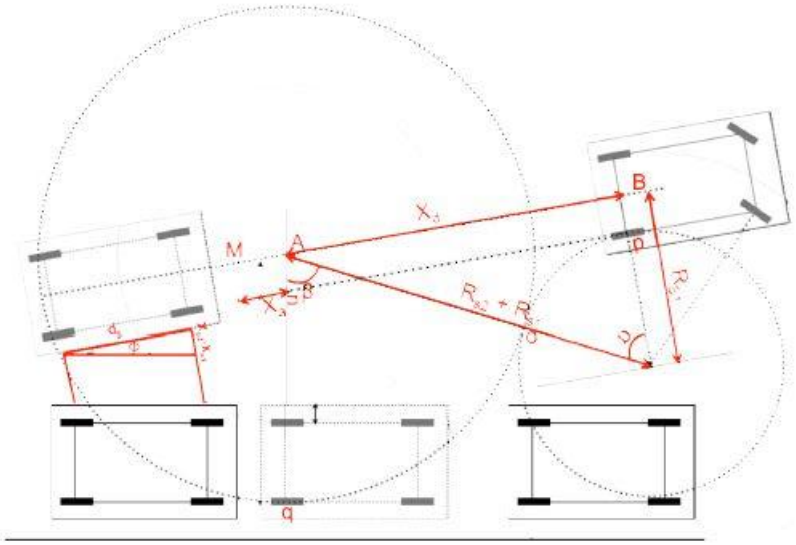
شکل ۵۷-۲ خط سیر حالت ایده آل[۲۲۸].

برخلاف حالت ایده آل، در محیط عملی بسیار محتمل است که هنگام پارک وسیله نقلیه ابتدا به موازات وسیله پارک شده قرار نگیرد.در این حالت زاویه شیب با مرجع تعیین می شود. شکل ۵۸-۲ زاویه‌ی ماشین را نشان می‌دهد.



شکل ۵۸-۲ زاویه‌ی ماشین نسبت به ماشین پارک شده[۲۲۸].

و مسیر طی شده در این حالت در شکل ۵۹-۲ مشخص شده است.

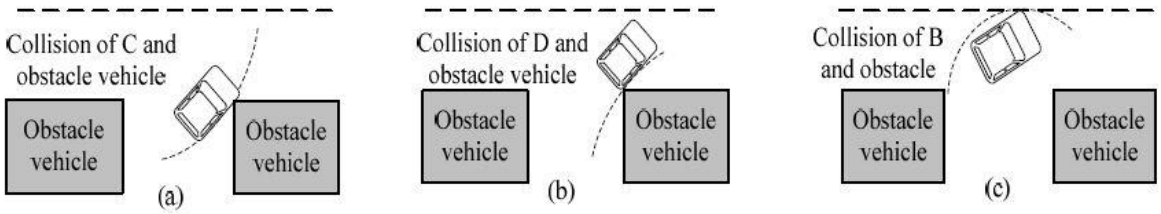


شکل ۵۹-۲ مسیر طی شده در حالت غیر ایده آل [۲۲۸].

در برنامه‌ریزی مسیر محدودیت‌ها نیز باید در نظر گرفته شوند. ابتدا ، هنگامی که چرخ جلو خودرو در حداکثر مقدار کار می کند، حداقل شعاع چرخش محدود می شود، ثانیا ، دامنه تغییر انحنای مسیر نباید بیش از بزرگترین محدودیت سرعت زاویه ای باشد. در غیر این صورت، از مسیر برنامه‌ریزی شده نمی توان برای ردیابی مسیر استفاده

کرد. علاوه بر این، در روند معکوس کردن ماشین از نقطه شروع پارکینگ به فضای پارکینگ نهایی، برخورد بین وسیله نقلیه و موانع به سه حالت نشان داده شده در شکل ۶۰-۲ می تواند باشد. که در برنامه‌ریزی مسیر باید جلوی این برخوردها گرفته شود[۲۲۹].

بدین صورت با استفاده از روش‌های انتخاب شده که توضیح داده شد، این راه حل‌ها با چارچوب زیرمسئله‌ی برنامه



شکل ۶۰-۲ حالت‌های برخورد وسیله نقلیه با مانع[۲۲۹].

ریزی مسیر انطباق پیدا می کنند که در این مسئله پارک موازی مد نظر است و در شرایط مختلف با داشتن نقطه‌ی اولیه خودرو و محل پارک، ابتدا با یک برنامه‌ریزی مسیر به محل شروع پارک رفته و سپس با ارائه‌ی مسیر مطلوب، خواسته‌ی این زیر مسئله یعنی پارک در محل داده شده را به طور کامل پیاده‌سازی می کنند.

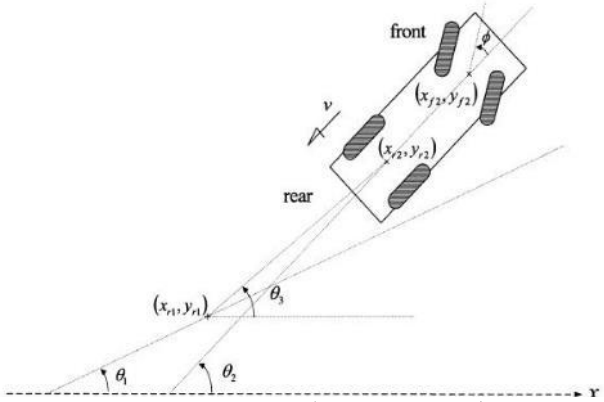
۳-۱-۳-۲- ردیابی مسیر با استفاده از روش‌های کنترلی

برنامه ریزی مسیر و کنترل ردیابی مسیر مکمل یکدیگر هستند. الگوریتم کنترل ردیابی مسیر یکی از فن آوری های کلیدی پارک خودکار است. از مهمترین مشکلات کنترل ردیابی مسیر پارک می‌توان به اطمینان از ردیابی مسیر، موقعیت و جهت گیری وسیله نقلیه هنگام اتمام مانور پارک اشاره کرد. همان طور که در بخش سه گفته شد، روشهای کنترلی زیادی برای ردیابی مسیر وجود دارد که در چارچوب مسئله می توان از آن‌ها استفاده کرد.

۱-۳-۱-۳-۲- کنترلر فازی

قوانین کنترلر فازی معمولاً توسط رفتار اپراتور انسانی تعیین می شود. در واقع ، کنترلر فازی الگوریتمی است که می تواند استراتژی کنترل زبانی را بر اساس دانش متخصص یا اپراتور به استراتژی کنترل خودکار تبدیل کند.

هسته‌ی آن مجموعه قوانین کنترل زبانی است. در چارچوب مسئله می توان یک کنترلر فازی دو ورودی - تک خروجی برای پارک خودکار اتخاذ کرد . شکل ۶۱-۲ حالت ماشین را نشان می‌دهد.

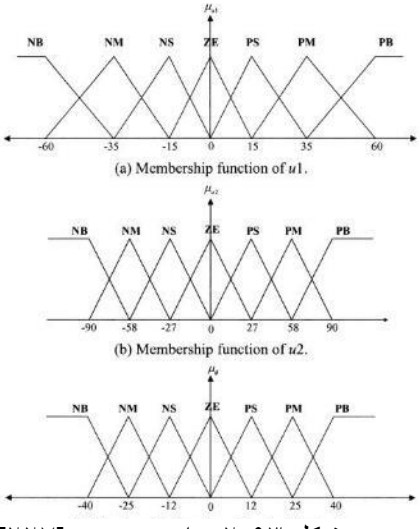


شکل ۶۱-۲ نمای کلی از وضعیت ماشین[۲۲۷].

متغیرهای ورودی در چارچوب مسئله می توانند به صورت روبه‌رو تعریف شوند. هم‌چنین، خروجی زاویه‌ی فرمان است.

$$u_1 = \theta_3 - \theta_1$$
$$u_2 = \theta_2 - \theta_1.$$

توابع عضویت، در شکل ۶۲-۲ نشان داده شده است، جایی که همه آنها به هفت پارتیشن فازی تجزیه می‌شوند، مانند بزرگ منفی (NB)، محیط منفی (NM)، کوچک منفی (NS)، صفر (ZE)، مثبت کوچک (PS)، محیط مثبت (PM) و مثبت بزرگ (PB). از آنجا که هر ورودی به هفت مجموعه فازی تقسیم شده است، باید چهل و نه قانون فازی تعیین شود که در شکل ۶۳-۲ نشان داده شده است[۲۲۷].



شکل ۶۳-۲ توابع عضویت[۲۲۷].

u1 \ u2	NB	NM	NS	ZE	PS	PM	PB
NB	ZE	NS	NM	NB	NB	NB	NB
NM	PS	ZE	NS	NM	NB	NB	NB
NS	PM	PS	ZE	NS	NM	NB	NB
ZE	PB	PM	PS	ZE	NS	NM	NB
PS	PB	PB	PM	PS	ZE	NS	NM
PM	PB	PB	PB	PM	PS	ZE	NS
PB	PB	PB	PB	PB	PM	PS	ZE

شکل ۶۲-۲ قوانین فازی[۲۲۷]

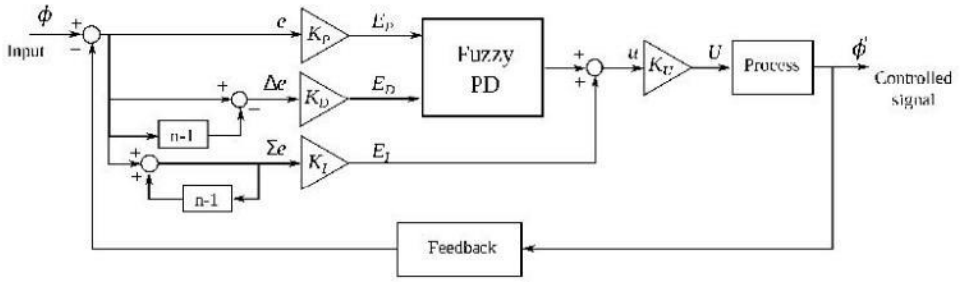
مزایا و معایب:

مزیت: مزیت اصلی این سیستم ها این است که آنها می توانند از دانش بشر در مسائلی استفاده کنند که ساختن مدلهای دقیق ریاضی دشوار یا بعضاً غیرممکن است. همچنین برای پیاده سازی روی سیستم‌های غیرخطی مناسب است.

عیب: به دست آوردن مدل‌های رفتاری انسان به منظور ایجاد قوانین کنترل زمانی پیچیده و زمان بر است. همچنین حجم محاسبات بالا است.

۲-۳-۱-۳-۲- کنترلر فازی بیشرفته ترکیبی با PID

هدف اصلی کنترل کننده به حداقل رساندن خطای کنترل با تنظیم خروجی کنترل کننده فرآیند است. سپس سیستم باید به یک نقطه ثبات برسد. پایداری به معنای این است که نقطه تنظیم شده بر روی خروجی بدون نوسانات اطراف آن نگه داشته می‌شود. این کنترل کننده ترکیب کنترل کننده فازی و PID است و متناسب با خطا پاسخ سریع ارائه می‌دهد، در حالی که از قسمت انتگرال یک تنظیم مجدد خودکار دارد که خطای حالت پایدار را برطرف می کند. عملکرد مشتق به کنترل کننده اجازه می‌دهد تا سریع به تغییرات خطا پاسخ دهد[۲۲۶]. بلوک دیاگرام این روش در حوزه زمان گسسته به صورت شکل ۶۴-۲ می‌باشد.



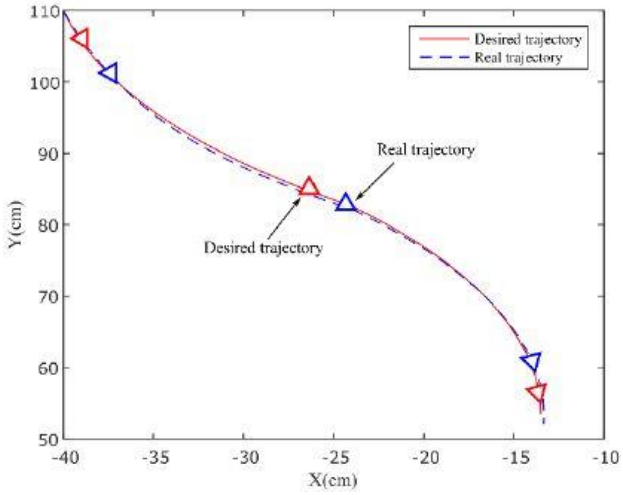
شکل ۶۴-۲ بلوک دیاگرام روش کنترلی فازی ترکیبی با PID[۲۲۶].

مزایا و معایب:

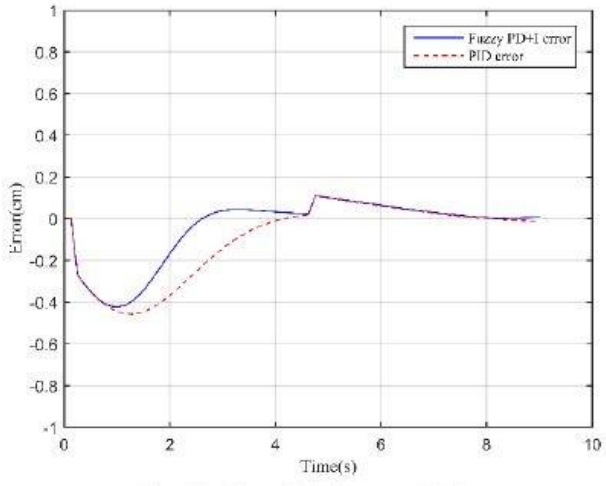
مزیت: دقت بیشتر و عدم نوسان حول مسیر

عیب: به دلیل وجود کنترلر فازی در بخشی از این کنترلر، مشکل پیچیدگی و زمان بر بودن به دست آوردن مدل‌های رفتاری انسان همچنان وجود دارد. و حجم محاسبات بالا است.

در شکل ۶۵-۲ دنبال کردن مسیر مورد نظر یک شبیه‌سازی را با استفاده از این روش مشاهده می کنیم. همچنین در شکل ۶۶-۲ مقایسه‌ی خطای دو کنترلر فازی پیشرفته و PID نشان داده شده است.



شکل ۶۵-۲ مسیر مرجع و مسیر دنبال شده[۲۲۶]



شکل ۶۶-۲ مقایسه‌ی دو کنترلر فازی پیشرفته و PID [۲۲۶]

۳-۳-۱-۳-۲- کنترلر مود لغزشی

کنترل حالت لغزشی (SMC) می تواند به طور مداوم با توجه به وضعیت فعلی سیستم (مانند خطا و مشتق آن) به طور هدفمند در فرآیند دینامیکی تغییر کند، که سیستم را مجبور می‌کند مطابق با مسیر حالت از پیش تعیین شده حرکت کند. کنترل کننده زاویه فرمان هدف را بر اساس موقعیت وسایل نقلیه و خطاهای جهت گیری با توجه به نقطه هدف در مسیر محاسبه خواهد کرد. سرعت خودرو تحت کنترل SMC نیست[۲۳۰].

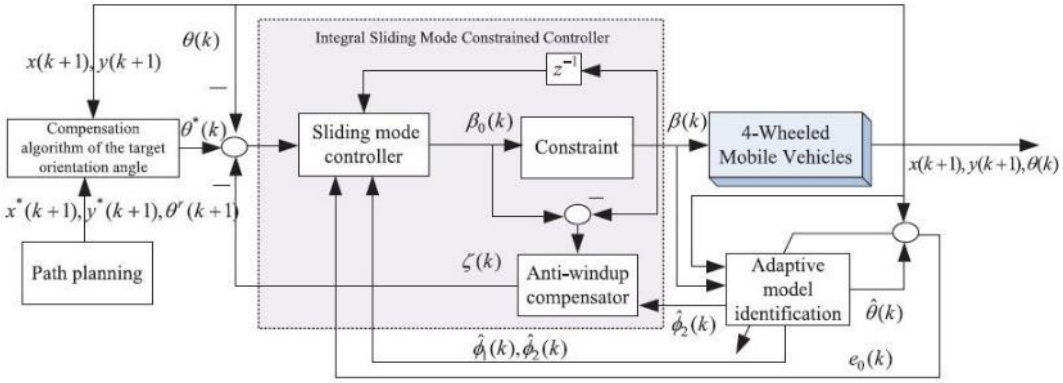
مزایا و معایب:

مزیت: از آنجا که هیچ ارتباطی با پارامترها و اختلالات شی ندارد، SMC دارای مزایای پاسخ سریع، عدم حساسیت به تغییرات پارامترها و اختلالات است.

عیب: تعداد کمی از طرح های اصلاح شده SMC مشکل اشباع ورودی کنترل را در نظر می‌گیرند. اگر محدودیت ورودی کنترل در طراحی کنترل کننده در نظر گرفته نشود، ممکن است اشباع انتگرال SMC ظاهر شود و سیستم کنترل حلقه بسته ممکن است ناپایدار شود.

Integral Sliding-Mode-Constrained-Control -۲-۳-۱-۳-۴

این الگوریتم کنترل پیشنهادی شامل یک الگوریتم شناسایی آنلاین داده محور، الگوریتم SMC انتگرال و الگوریتم جبران خسارت ضد باد است که فقط زاویه بدن و زاویه فرمان خودرو را در خود جای داده است و از هیچ اطلاعات مدل مکانیزم اتومبیل استفاده نمی کند. بلوک دیاگرام این کنترلر در شکل ۶۷-۲ آورده شده است[۲۳۰].



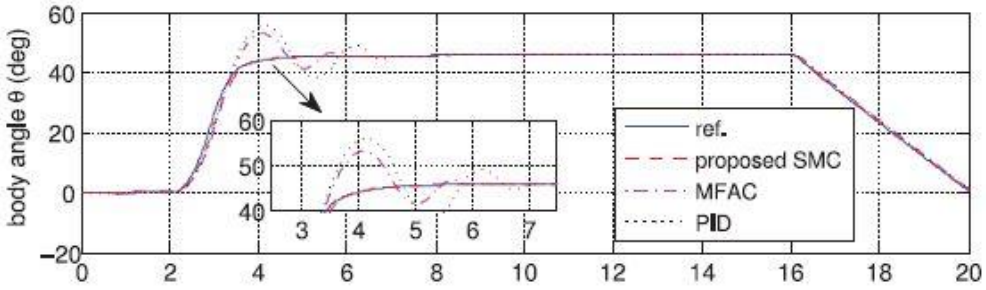
شکل ۶۷-۲ بلوک دیاگرام کنترلر SMC پیشرفته[۲۳۰].

مزایا و معایب:

مزیت: به راحتی قابل اجرا است و میزان محاسبات را کاهش می‌دهد، پیشنهادی نسبت به مقدار اولیه مشتقات جزئی حساس نیست، که استحکام قوی را فراهم می‌کند. همچنین می‌تواند به طور موثر مشکلات اشباع انتگرال و اشباع محرک را حل کند.

عیب: عیبی برای این روش یافت نشد.

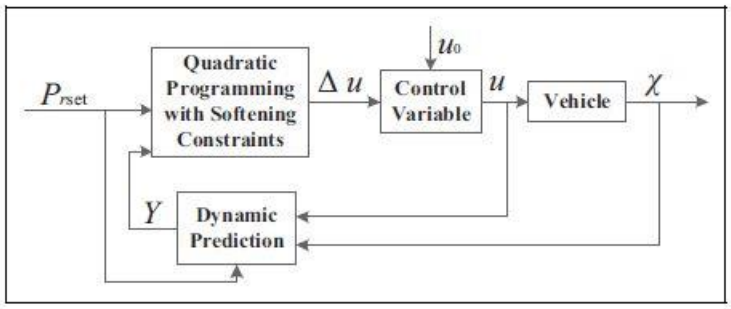
در شکل ۶۸-۲ مقایسه‌ی کنترلرهای پیاده‌سازی شده روی یک سیستم را مشاهده می کنیم. که این روش کنترلی بهترین عملکرد را داشته است و پس از آن روش کنترل تطبیقی که در ادامه آمده است . کنترلر PID نسبت به دو روش قبلی عملکرد پایین تری داشته است.



شکل ۶۸-۲ مقایسه‌ی کنترلرهای پیاده سازی شده[۲۳۰].

۵-۳-۱-۳-۲- کنترل پیش‌بین مدل

این روش کنترلی یک روش کنترل پیشرفته است که معمولاً برای کنترل یک فرآیند با محدودیت استفاده می‌شود. به عبارت دیگر، با پیش بینی وضعیت آینده سیستم و افزودن محدودیت هایی به متغیرهای ورودی، خروجی یا وضعیت آینده، محدودیت ها در یک مسئله برنامه نویسی درجه دوم یا برنامه نویسی غیرخطی بیان می‌شوند. در این روش فرض بر این است که متغیرهای کنترل پیش بینی شده قادر به نقض موقت محدودیت های سخت اصلی هستند، که مجموعه عملی موجود را اعمال می‌کند. در چارچوب مسئله کنترل کننده پارک MPC به طور هماهنگ سرعت و فرمان چرخ جلو را کنترل می کند[۲۳۱]. شکل ۶۹-۲ مدل استفاده از این روش را نشان می دهد.



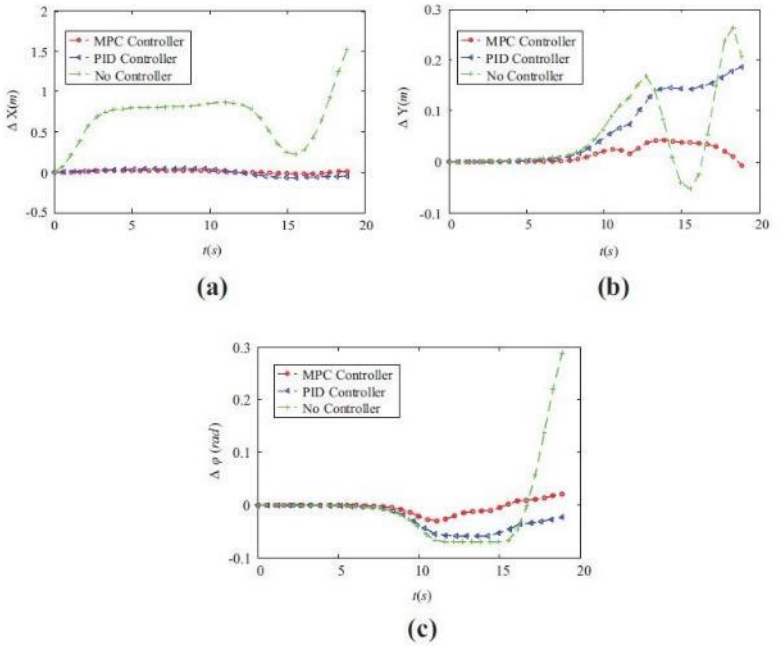
شکل ۶۹-۲ مدل کنترلی MPC[۲۳۱].

مزایا و معایب:

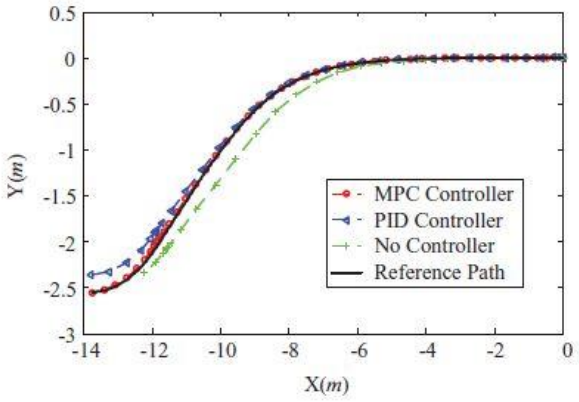
مزیت: برجسته ترین جذابیت MPC این است که می تواند محدودیت ها را به صراحت کنترل کند.

عیب: حجم محاسبات بالا از مهم‌ترین معایب کنترل پیش‌بین مدل است.

شکل‌های ۷۰-۲ و ۷۱-۲ نشان دهنده‌ی عملکرد بهتر یک سیستم با استفاده از این روش نسبت به روش کنترلی PID هستند.



شکل ۷۰-۲ مقایسه‌ی عملکردی روش‌های کنترلی[۲۳۱].



شکل۷۱-۲ مقایسه‌ی عملکردی روش‌های کنترلی[۲۳۱].

Iterative control -۲-۳-۱-۳-۶

این الگوریتم برای شبیه سازی حرکت کامل پارک کردن استفاده می شود. این روش یک روش تکراری تطبیقی است. در هر تکرار از y، y’ و y’’ به عنوان شرط اولیه جدید استفاده می شود و یک چند جمله ای درجه پنج پنجم جدید برای تعیین دستور حرکت بعدی محاسبه می‌شود. ماهیت انطباقی این الگوریتم اجازه نمی‌دهد "عدم

شعاع"، "عدم اطمینان زاویه ای" و اشکال دیگر خطا جمع شود، و هر مرحله از راه شروع جدید است. یک حلقه واحد شامل دو حرکت است، یک چرخش و یک حرکت مستقیم[۲۳۲].

مزایا و معایب:

مزیت: هر مرحله از راه یک شروع جدید است و خطاها با هم جمع نمی‌شوند و دقت کار بالا می‌رود.

عیب: به خاطر محاسبات در هر مرحله،میزان محاسبات بالا و سرعت این روش پایین است.

۷-۳-۱-۳-۲- کنترل تطبیقی

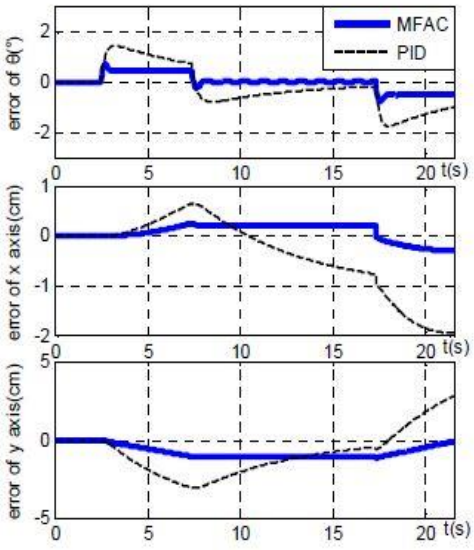
در این روش طرح کنترل تطبیقی مدل آزاد فقط به داده‌های ورودی و خروجی سیستم کنترل شده متکی است و شامل هیچ یک از اطلاعات مدل سیستم کنترل شده نیست. بنابراین می‌تواند کنترل تطبیقی و ساختاری پارامتری سیستم کنترل ناشناخته را تحقق بخشد. هدف کنترل سیستم پارک اتوماتیک حفظ سرعت ثابت و سپس کنترل زاویه جهت خودرو با کنترل زاویه چرخ جلو است که هدف آن برابر با زاویه جهت گیری اتومبیل است تا اطمینان حاصل شود که مختصات ماشین به طور مداوم در مسیر هدف قرار می‌گیرد[۲۳۲].

مزایا و معایب:

مزیت: این الگوریتم با پارامتر تنظیم و محاسبه کمی انجام می‌شود، بنابراین برای ماشی‌های مختلف مناسب است که بتوانند کنترل انطباقی سیستم پارک اتوماتیک را انجام دهند.

عیب: عیبی برای این روش یافت نشد.

در شکل ۷۲-۲ مقایسه‌ی این روش کنترلی را با کنترلر PID پیاده‌سازی شده روی یک سیستم مشاهده می کنیم که این روش عملکرد بهتری داشته است .



شکل ۷۲-۲ مقایسه‌ی روش‌های کنترلی[۲۳۲].

۴-۱-۳-۲- جمع‌بندی

برای مقایسه روش‌ها و جمع بندی، با توجه به مزیت‌ها و معایب گفته شده و همچنین مقایسه‌های انجام شده از نتایج شبیه‌سازی‌ها اولویت‌های زیر به ترتیب برای کنترل پارک خودکار انتخاب شده است:

۱) روش کنترلی Integral Sliding-Mode-Constrained-Control را به عنوان اولویت اول انتخاب می‌کنیم که در چارچوب مسئله، مسیر مورد نظر به خوبی دنبال شود

۲) کنترل پیش‌بین مدل

۳) استفاده از منطق فازی

هم‌چنین برای برنامه‌ریزی مسیر از روش برنامه‌ریزی جهانی استفاده می‌شود. در نهایت برای جمع‌بندی هماهنگ کردن راه حل‌های زیرمسئله‌ها، ابتدا سنسورهای خودرو فضای موجود برای پارک را بررسی می‌کنند. پس از آن، یک مسیر بدون برخورد محاسبه می شود. در آخرین مرحله، از یک استراتژی کنترل برای دنبال کردن مسیر محاسبه شده و فرار از هر مانع غیر منتظره استفاده می‌شود. وجود موانع غیرمنتظره می‌تواند منجر به محاسبه مجدد مسیر شود. که البته در مسئله‌ی تعریف شده مرحله‌ی پیدا کردن فضای پارک وجود ندارد.

۲-۳-۲- ادراک محیط

۱-۲-۳-۲- تشخیص اشیا[۱۸۳]

شناسایی اشیا به شناسایی مکان و اندازه اشیا مورد نظر اشاره دارد. شناسایی اشیا چه در حالت ایستا که شامل چراغ های راهنمایی و رانندگی و نشانه های عبور از جاده و چه در حالت پویا که شامل وسایل نقلیه، عابرین پیاده و دوچرخه سواران می شود، همواره مورد بحث در خودرو های خودران بوده است. در این بخش درباره پیشرفته‌ترین و به‌روزترین روش‌های پیشنهادی شناسایی اشیا بحث شده که این خود نقطه شروع چندین کار دیگر از جمله ردیابی اشیا و درک محیط می باشد[۲۳۳].

۱-۱-۲-۳-۲- شبکه YOLO

به طور کلی یک آشکارساز شی به منظور استخراج ویژگی های ورودی و ارسال این ویژگی ها به یک سیستم پیش‌بین طراحی شده است تا بتواند جعبه ای به دور شی تشخیص داده شده بیاندازند و کلاس آن را تشخیص دهد. یولو اولین آشکارسازی است که این روند را به صورت انتها به انتها[۱۸۴] طی می کند.

یولو شامل سه بخش است:

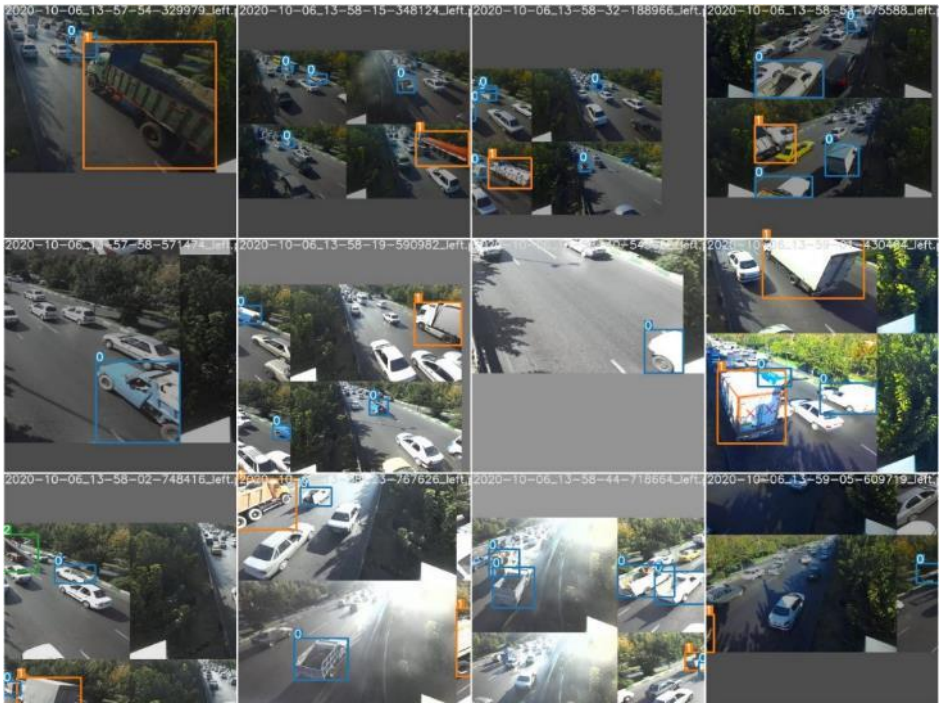
- **Backbone**: یک شبکه عصبی کانوولوشنی که وظیفه استخراج ویژگی را بر عهده دارد.
- **Neck**: به مجموعه ای از لایه‌هایی که به منظور ترکیب ویژگی‌های تصویر و انتقال آن‌ها به سمت پیش‌بینی کننده عمل می‌کنند، گفته می‌شود.
- **Head**: ویژگی ها را از Neck دریافت می کند و مراحل پیش بینی مکان جعبه و کلاس را انجام می دهد.

همانطور که در بخش ۳ توضیح داده شد، YOLO دارای ورژن های متفاوتی از ۱ تا ۵ می باشد که شروع شکوه آن از ورژن سه بود که توانست در سال ۲۰۱۸ دقت بی نظیری را ارئه دهد. در ادامه، در سال ۲۰۲۰ دو ورژن بعدی آن یعنی چهار و پنج در فاصله کوتاهی از یکدیگر منتشر شدند. YOLOv4 با موارد جدید حیرت انگیز معرفی شد، با اختلاف زیاد از YOLOv3 بهتر عمل کرد و همچنین به مقدار قابل توجهی دقت متوسط آن افزایش یافت. با فاصله کمی از منتشر شدن YOLOv4، YOLOV5 منتشر شد و این نسخه توانست و عملکرد بهتری نسبت به تمام نسخه های قبلی داشته باشد.

183 Object detection

184 End-to-end

روند آموزش شبکه برای بررسی عملکرد نهایی سیستم بسیار مهم هست، از این رو نسخه‌های چهار و پنج با نوآوری جدی خود در بخش تقویت داده‌ها[۱۸۵] که به نام موزاییک آگمنتیشن[۱۸۶] معروف است و بهبود تابع ضرر خود، توانستند دقت بالایی کسب کنند. نمونه ای از موزاییک آگمنتیشن در تصویر ۷۳-۲ آورده شده است.



شکل ۷۳-۲ نمونه‌ای از موزاییک آگمنتیشن.

همانطور که در تصویر ۷۳-۲ مشخص است، موزاییک آگمنتیشن چهار تصویر را به چهار موزاییک با نسبت تصادفی ترکیب می کند که این خود باعث بهبود عملکرد آموزش شبکه در دیتاست‌هایی مثل COCO که تعداد زیادی کلاس دارد، می‌شود. علاوه بر آن این دو شبکه با استفاده از ساختار CSPNet توانتسند مقدار محاسبات را تا حد زیادی کاهش دهند و سرعت استنتاج و آموزش شبکه و همچنین دقت را بهبود بخشند[۲۳۴].

بزرگترین سهم YOLOv5 ترجمه چارچوب Darknet به چارچوب PyTorch است. چارچوب Darknet به زبان C نوشته شده و امکان کنترل دقیق روی عملیات رمزگذاری شده در شبکه را فراهم می کند. از بسیاری جهات، کنترل زبان سطح پایین مزیت تحقیق است، اما می تواند انتقال آن را در بینش های تحقیق جدید کندتر کند و نحوه گسترش و پیشرفت آن را سخت می سازد. برتری که ورژن پنج نسبت به ورژن های پیشین خود دارد،

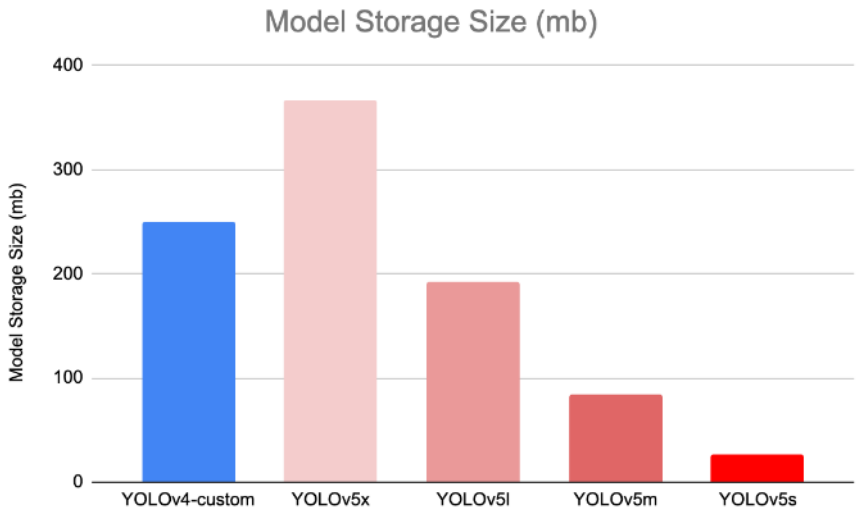
185 Augmentation

186 Mosaic Augmentation

استفاده از BiFPN ها در قسمت Neck و یادگیری خودکار جعبه انکرهای محدود کننده با استفاده از K-means و الگوریتم های ژنتیک است که این تغییرات گفته شده، سرعت ،کارآمدی و بهینه بودن این آشکارساز را نسبت به ورژن‌های پیشین خود برتری می دهد. کاربری و استفاده از YOLOv5 بسیار آسان است و با توجه به سخت افزار مورد نیاز کار، پنج مدل بر اساس زمان پردازش، دقت و تعداد پارامتر ها در اختیار می گذارد که می توان در جدول ۳-۲ و شکل ۷۴-۲ مشاهده کرد.

جدول۳-۲، مقایسه ای از مدل های مختلف YOLOV5 و YOLOV3 بر اساس سرعت، FLOPS، دقت و تعداد پارمترها[۲۳۵]

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}		params	FLOPS
YOLOv5s	37.0	37.0	56.2	**2.4ms**	**416**		7.5M	13.2B
YOLOv5m	44.3	44.3	63.2	3.4ms	294		21.8M	39.4B
YOLOv5l	47.7	47.7	66.5	4.4ms	227		47.8M	88.1B
YOLOv5x	**49.2**	**49.2**	**67.7**	6.9ms	145		89.0M	166.4B
YOLOv5x + TTA	**50.8**	**50.8**	**68.9**	25.5ms	39		89.0M	354.3B
YOLOv3-SPP	45.6	45.5	65.2	4.5ms	222		63.0M	118.0B

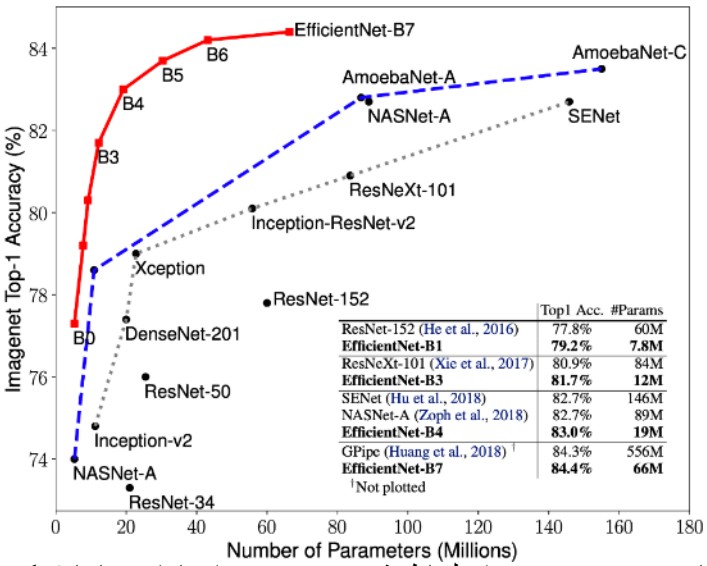


شکل ۷۴-۲ نمونه‌ای از موزاییک آگمنتیشن[۲۳۶].

بدلیل کاربری ساده و سبک بودن شبکه، برنامه آشکارساز YOLOv5 برای تلفن های موبایل نیز موجود است[۲۳۸-۲۳۷] .

EfficientDet -۲-۳-۲-۱-۲

تیم Goolge Brain در سال ۲۰۲۰ مدل EfficientDet را با هدف شکل دادن ساختار تصمیم گیری در یک چارچوب مقیاس پذیر برای شناسایی اشیا معرفی کرد. ریشه معرفی این شبکه به کمی پیشتر با معرفی ساختار EfficientNet بر می گیرد. در یک شبکه می توان هر لایه را گسترده تر کرد، تعداد لایه ها را عمیق تر کرد، می توان تصاویر را با وضوح بالاتر به شبکه داد و یا می توان ترکیبی از این پیشرفت ها را ایجاد کرد که تحقیق در همه این امکانات برای محققان یادگیری ماشین بسیار خسته کننده است. EfficientNet تصمیم گرفت تا روشی خودکار برای مقیاس گذاری معماری های مدل ConvNet تعریف کند. این مقاله به دنبال بهینه سازی عملکرد با توجه به مقدار عمق، عرض و وضوح آزاد داده شده است در حالی که در محدودیت حافظه و FLOP مشخص قرار داشته باشد. ایجاد تکنیک مقیاس گذاری مدل جدید یک گام بزرگ به جلو است و نویسندگان این مقاله با ایجاد یک معماری جدید ConvNet که نتایج را بهبود می بخشد، نتوانستند معماری معرفی کنند که مدل جدید از طریق جستجوی معماری عصبی کشف می شود. جستجوی معماری عصبی با در نظر گرفتن تعداد معینی از FLOPS و در نتیجه ایجاد یک ConvNet پایه به نام EfficientNet-B0 ، دقت را بهینه می کند. با توجه به نکات گفته شده EfficientNet یک جهش بزرگ نسبت به دیگر ساختار های ConvNet است که عملکرد آن از لحاظ دقت و تعداد پارامترها که به وضوح در تصویر ۲-۷۵ مشخص است، بهتر می باشد[۲۳۹].



شکل ۲-۷۵ مقایسه EfficientNet از لحاظ دقت بر حسب تعداد پارامترها با شبکه های دیگر [۲۴۰]

در EfficientDet علاوه بر ساختار EfficientNet که عملکرد آن را بهبود می دهد، از ایده BiFPN ها که یک شبکه وزنی دو طرفه به منظور ترکیب ویژگی آسان و چند مقیاسی است استفاده شده است که نه تنها با افزایش mAP بلکه با کاهش تعداد پارامترها و FLOP ها نیز عملکرد را بهبود می بخشد. در جدول ۴-۲ می توان مقایسه مدل های EfficientDet با دیگر شبکه های تشخیص دهنده اشیا را مشاهده کرد که در تمام شاخص‌ها (mAP، تعداد پارامترها، تعداد FLOPS، زمان پردازش در CPU وGPU) عملکرد بسیار قابل توجه و بهتری داشته است. همانطور که از جدول مشخص است، EfficientDet دارای سرعت قابل توجهی در استفاده از پردازنده های گرافیکی و هسته ای است که حتی برای برنامه‌های حساس که به تأخیر کم نیاز دارند، استفاده از آن مناسب است[۲۴۱،۲۴۲،۲۴۳].

جدول ۴-۲ مقایسه EfficientDet با دیگر آشکارسازها از لحاظ mAP، تعداد پارمترها، تاخیر و سر عت[۲۴۲]

Model	mAP	#Params	Ratio	#FLOPS	Ratio	GPU LAT(ms)	Speedup	CPU LAT(s)	Speedup
EfficientDet-D0	**32.4**	**3.9M**	**1x**	**2.5B**	**1x**	**16** ±1.6	**1x**	**0.32** ±0.002	**1x**
YOLOv3 [26]	33.0	-	-	71B	28x	51†	-	-	-
EfficientDet-D1	**38.3**	**6.6M**	**1x**	**6B**	**1x**	**20** ±1.1	**1x**	**0.74** ±0.003	**1x**
MaskRCNN [8]	37.9	44.4M	6.7x	149B	25x	92†	-	-	-
RetinaNet-R50 (640) [17]	37.0	34.0M	6.7x	97B	16x	27 ±1.1	1.4x	2.8 ±0.017	3.8x
RetinaNet-R101 (640) [17]	37.9	53.0M	8x	127B	21x	34 ±0.5	1.7x	3.6 ±0.012	4.9x
EfficientDet-D2	**41.1**	**8.1M**	**1x**	**11B**	**1x**	**24** ±0.5	**1x**	**1.2** ±0.003	**1x**
RetinaNet-R50 (1024) [17]	40.1	34.0M	4.3x	248B	23x	51 ±0.9	2.0x	7.5 ±0.006	6.3x
RetinaNet-R101 (1024) [17]	41.1	53.0M	6.6x	326B	30x	65 ±0.4	2.7x	9.7 ±0.038	8.1x
NAS-FPN R-50 (640) [5]	39.9	60.3M	7.5x	141B	13x	41 ±0.6	1.7x	4.1 ±0.027	3.4x
EfficientDet-D3	**44.3**	**12.0M**	**1x**	**25B**	**1x**	**42** ±0.8	**1x**	**2.5** ±0.002	**1x**
NAS-FPN R-50 (1024) [5]	44.2	60.3M	5.1x	360B	15x	79 ±0.3	1.9x	11 ±0.063	4.4x
NAS-FPN R-50 (1280) [5]	44.8	60.3M	5.1x	563B	23x	119 ±0.9	2.8x	17 ±0.150	6.8x
EfficientDet-D4	**46.6**	**20.7M**	**1x**	**55B**	**1x**	**74** ±0.5	**1x**	**4.8** ±0.003	**1x**
NAS-FPN R50 (1280@384)	45.4	104 M	5.1x	1043B	19x	173 ±0.7	2.3x	27 ±0.056	5.6x
EfficientDet-D5 + AA	**49.8**	**33.7M**	**1x**	**136B**	**1x**	**141** ±2.1	**1x**	**11** ±0.002	**1x**
AmoebaNet+ NAS-FPN + AA(1280) [37]	48.6	185M	5.5x	1317B	9.7x	259 ±1.2	1.8x	38 ±0.084	3.5x
EfficientDet-D6 + AA	**50.6**	**51.9M**	**1x**	**227B**	**1x**	**190** ±1.1	**1x**	**16**±0.003	**1x**
AmoebaNet+ NAS-FPN + AA(1536) [37]	50.7	209M	4.0x	3045B	13x	608 ±1.4	3.2x	83 ±0.092	5.2x
EfficientDet-D7 + AA	**51.0**	**51.9M**	**1x**	**326B**	**1x**	**262** ±2.2	**1x**	**24** ±0.003	**1x**

Haar Cascade -۲-۳-۲-۱-۳

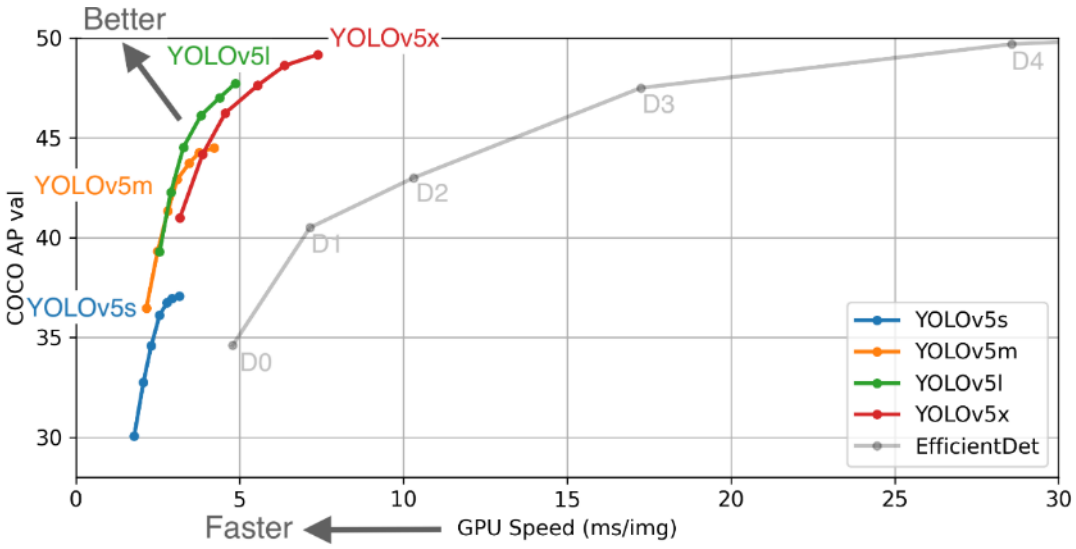
این روش یک رویکرد مبتنی بر یادگیری ماشین است که تابع کسکید۱۸۷ آن بر روی بسیار زیادی داده آموزش میبینند. پیاده سازی و استفاده از این الگوریتم بسیار ساده است و می توان آن را از کتابخانه OpenCV مشاهده کرد. این روش تا حد زیادی به محیط وابسته است و نیازمند تعیین پارامتر های آن به صورت دستی می باشد.

187 Cascade

آموزش Cascade از ابتدا کار بسیار سخت و طاقت فرسایی است و به نسبت شبکه های عصبی کانوولوشنی دقت به مراتب کمتری دارد و در کاربردهایی نظیر خودرو خودران که نیازمند دقت بالایی در تشخیص اشیا است، روش مناسبی نیست.

۴-۱-۲-۳-۲- جمع بندی

با توجه به نکات گفته شده و مسئله موجود که می بایست اشیا موجود در یک ویدئو شناسایی شوند، این بخش یک امر ضروری در بخش درک محیط می باشد. با توجه به اینکه هدف درک محیط اطراف خودرو خودران می باشد و اگر این امر درست صورت نپذیرد باعث ایجاد خسارت های جبران ناپذیری می شود، استفاده از روش Haar Cascade بدلیل دقت پایین در شناسایی اشیا و وابسته بودن پارامتر های آن به محیط، نسبت به دو روش دیگر از ضریب اهمیت پایین‌تری برخوردار است. با توجه به دو روش پیشنهادی دیگر، YOLOv5 فناوری جدید تری نسبت به EfficientDet می باشد و مقایسه این دو با یکدیگر در تصویر ۷۶-۲ آورده شده است. هنگامی که درباره کاربردهایی که نیازمند تاخیر کم در شناسایی هستند صحبت می شود هر دو شبکه عملکرد خوبی دارند ولی با این حال همانطور که در شکل ۷۶-۲ مشاهده میشود، YOLOV5 از نظر زمان پردازش و دقت عملکرد بهتری نسبت به EfficientDet دارد که این خود باعث می شود حتی بتوان از این شبکه در سخت افزارهای ضعیف تری مانند Raspberry pie استفاده و دقت قابل قبولی کسب کرد.



شکل ۷۶-۲ مقایسه عملکرد EfficientDet و YOLOv5 از لحاظ دقت بر حسب سرعت(هر دو شبکه تحت شرایط یکسان بر روی دیتاست COCO آموزش دیده‌اند)[۲۳۷]

۲-۲-۳-۲- بخش بندی

با توجه به روش‌های بررسی‌شده پیرامون تقسیم بندی موردی و معنایی و همچنین تشخیص اشیا، استفاده از الگوریتم‌های تشخیص اشیا بهینه ترین روش برای استفاده در پروژه می‌باشد. همان گونه که در شکل ۳۳-۳ تفاوت‌های دیداری تقسیم‌بندی موردی، تقسیم‌بندی معنایی و تشخیص اشیا مشخص است، نیاز اصلی در این پروژه شناسایی و شمارش اشیا می‌باشد. به‌علاوه، با این که تشخیص اشیا را می‌توان با تقسیم‌بندی موردی نیز انجام داد، الگوریتم‌های شناسایی اشیا با توجه ویژگی ساختاری که دارند، در کاری که انجام می‌دهند دقت بالاتری به نسبت الگوریتم‌های تقسیم‌بندی معنایی و ساختاری دارند. همچنین به سخت افزار پایین تری نیاز است.

مقایسه‌ی دو روش تقسیم‌بندی موردی[۱۸۸] و تشخیص اشیا[۱۸۹] در جدول ۵-۲ آمده است.

جدول ۵-۲ مقایسه دو روش تشخیص اشیا و تقسیم بندی موردی.

تشخیص اشیا	تقسیم‌بندی موردی
دور اشیا باندینگ باکس رسم می‌شود، که شی داخل آن باندینگ باکس قرار می‌گیرد.	برای هر شی یک ماسک تولید می‌شود که نشان-دهنده‌ی موقعیت دقیق آن شی هستند.
آموزش سریع‌تر به نسبت تقسیم‌بندی موردی	آموزش کند
ساده‌تر بودن کل فرایند	پیچیدگی بیشتر فرایند به نسبت تشخیص اشیا
کاربرد کمتر	کاربردهای بیشتر
دقت بیشتر در تشخیص اشیا مجزا	احتمال تشخیص دو شی به عنوان یک شی

با توجه به آن که در پروژه‌ی اصلی، شمارش خودرو ها مورد ارزیابی قرار می‌گیرد، روش پیشنهادی تشخیص اشیا می‌باشد.

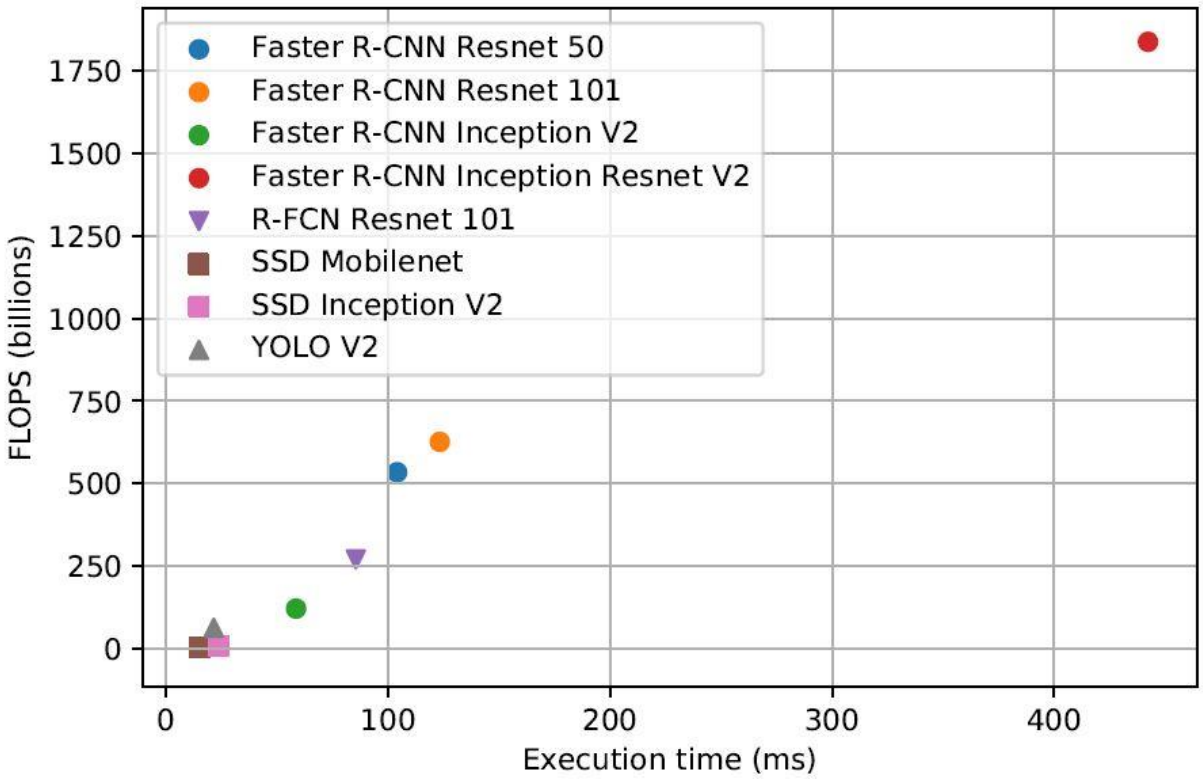
۳-۲-۳-۲- شناسایی علائم و چراغ های راهنمایی رانندگی

روش های معرفی شده برای شناسایی علائم و چراغ های راهنمایی رانندگی در این بخش بررسی می شود.

188 Instance Segmentation

189 Object Detection

شکل ۷۷-۲ تعداد FLOP را در برابر زمان اجرا نشان می دهد. استفاده از بلوک های متراکم در شبکه های رزیژوال ۱۹۰منجر به افزایش FLOP ها و زمان محاسبه برای هر دو آشکارساز Faster RCNN و R-FCN شده است. از طرف دیگر، SSD Mobilenet مدلی است که دارای کمترین میزان FLOP و کمترین زمان اجرا است. باید در نظر داشت که شمارنده FLOP ممکن است به دلیل فاکتورهای مختلف مانند بهینه سازی سخت افزار و ورودی و خروجی حافظه ، از نظر زمان اجرای واقعی خطی نباشد. این واقعیت را می توان در مقایسه مدل های YOLO V2 و SSD Inception V2 مشاهده کرد. مورد اول ۶۲.۷۸ میلیارد FLOP را در زمان کمتری نسبت به ۷.۵۹ میلیارد FLOP دیگری انجام می دهد.

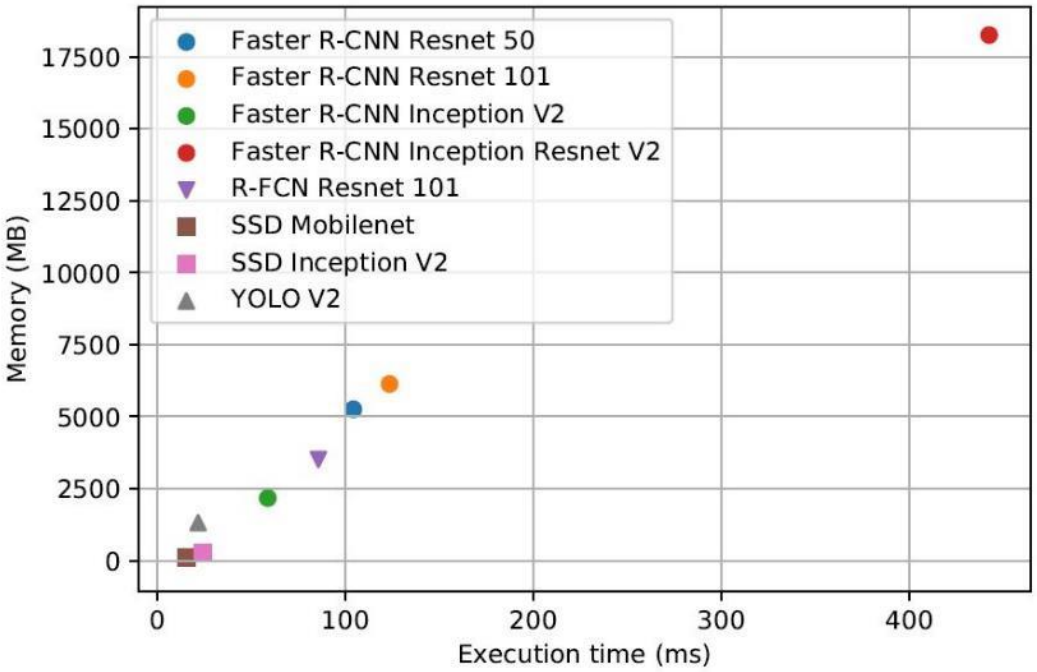


شکل ۷۷-۲ مقایسه روش های مختلف از لحاظ FLOPS بر زمان اجرا[۲۴۴]

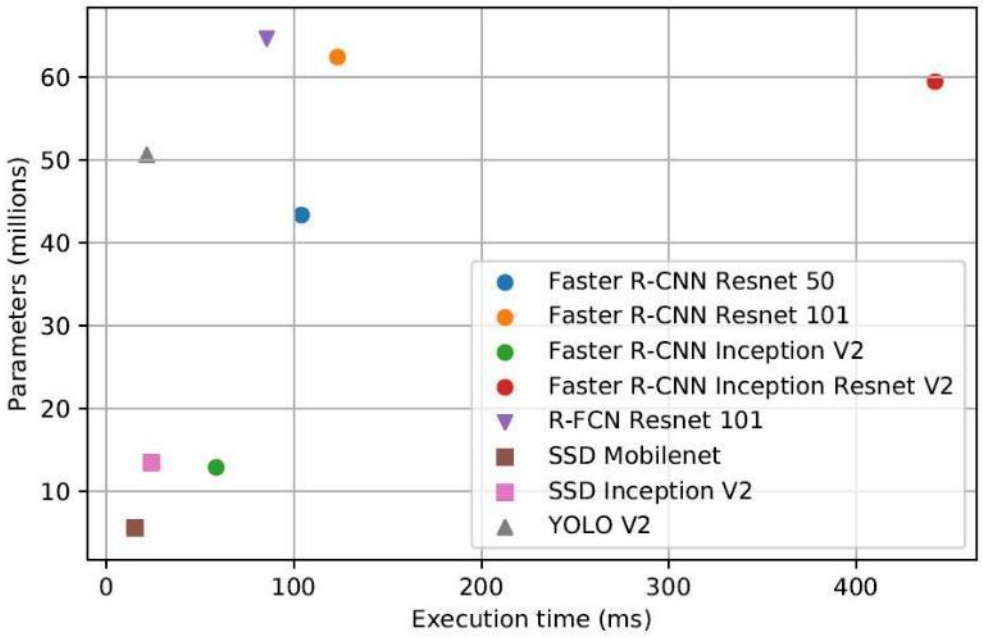
علاوه بر این ، تعداد پارامترهایی که هر شبکه عصبی باید یاد بگیرد (وزن و بایاس) ارتباط مستقیمی با زمان اجرای آنها ندارد. همانطور که در شکل ۷۸-۲ نشان داده شده است، می توان دریافت که مدل هایی که استخراج کننده

190 Residual

ویژگی آنها Resnet 101 است، حاوی میلیون ها پارامتر بیشتر از آشکارسازهایی است که زمان اجرای آن ها بیشتر یا مشابه این استخراج کننده است.



شکل ۲-۷۸ مقایسه روش های مختلف از لحاظ تعداد پارامترها بر زمان اجرا[۲۴۴].



شکل ۲-۷۹ مقایسه روش های مختلف از لحاظ مصرف حافظه بر زمان اجرا[۲۴۴]

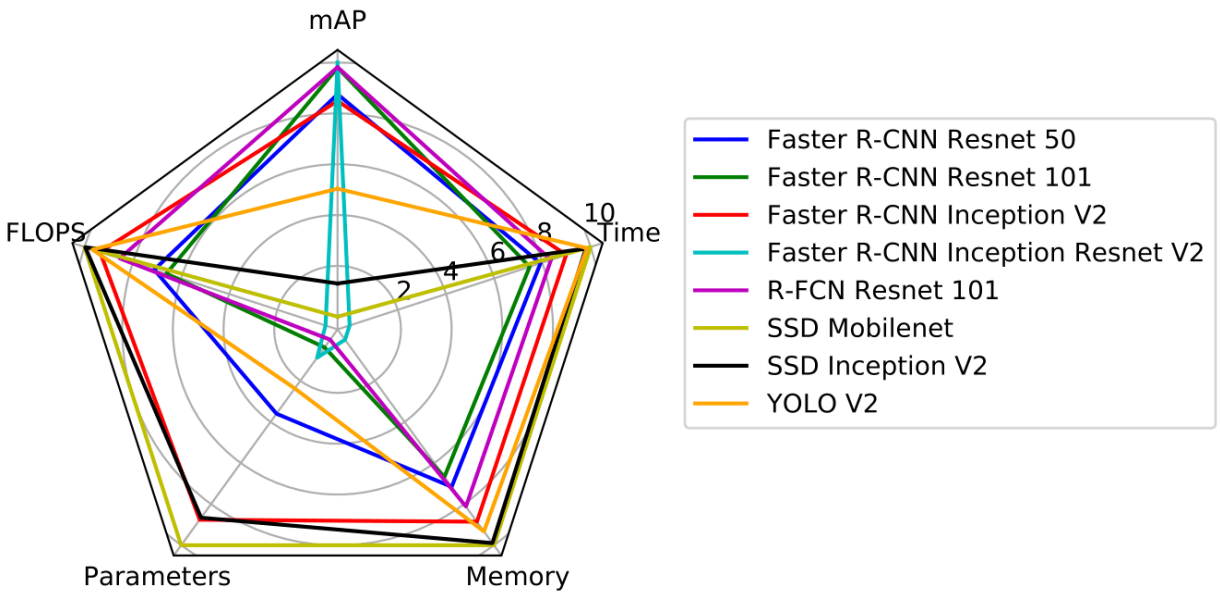
مصرف حافظه نیز یک عامل حیاتی است. این امر به این تصمیم گیری کمک می کند که سخت افزاری که در حال استفاده است آیا مناسب است یا خیر. شکل ۷۹-۲ مصرف حافظه را در برابر زمان اجرای مدلهای مورد بررسی شده ترسیم می کند که همبستگی خطی بالایی بین زمان اجرا و استخراج کننده های ویژگی قدرتمندتر که به حافظه بیشتری نیاز دارند، وجود دارد.

علاوه بر موارد شده، روش Haar-Cascade که یک الگوریتم تشخیص شی مبتنی بر یادگیری ماشین است نیز در بخش‌های پیشین معرفی شد. همانطور که در بخش شناسایی اشیا در مورد این الگوریتم توضیح داده شد این روش تا حد زیادی به محیط وابسته است و نیازمند تعیین پارامتر های آن متناسب با محیط تست می باشد. همانطور که گفته شد آموزش Cascade از ابتدا کار بسیار سختی است و به نسبت شبکه های عصبی از دقت به مراتب کمتری برخوردار است و در کاربردهایی نظیر خودرو خودران که نیازمند دقت بالایی در تشخیص تابلو ها و علائم راهنمایی و رانندگی است، روش مناسبی نیست[۲۴۴].

۱-۳-۲-۳-۲- جمع بندی

نمودار راداری در شکل ۸۰-۲ رسم شده است که پره های آن پنج عامل اندازه گیری شده یعنی mAP ، زمان اجرا ، FLOPها ، تعداد پارامترها و میزان استفاده از حافظه را نشان می دهد. به جز شاخص mAP، کمترین مقدار هر اندازه گیری به عنوان بهترین در نظر گرفته می‌شود. برای هر شاخص نیز مقادیر در محدوده [۰-۱۰] تبدیل شده‌اند. باید در نظر داشت که mAP ، زمان اجرا و مصرف حافظه مهمترین عوامل را تشکیل می دهند. با توجه به نتایج حاصل شده، مشاهده می شود بهترین مدل های کلی R-FCN Resnet 101 و Faster R-CNN Inception V2 می باشد[۲۴۴].

علاوه بر موارد مطرح شده، به منظور انطباق مسئله با محیط پروژه، به منظور مصرف بهینه در زمان می توان از روشی که در بخش شناسایی اشیا استفاده می شود مانند YOLOV5، برای شناسایی علائم راهنمایی و رانندگی نیز استفاده شود ولی با این تفاوت همه تابلو ها در یک کلاس در نظر گرفته شود و در انتها توسط یک طبقه بند ساده که نمونه های آن در بالا آورده شد، کلاس تابلو تشخیص داده شود که این کار باعث می شود هم در زمان پردازش صرفه جویی شود و هم دقت تحت تاثیر افزایش کلاس قرار نگیرد.



شکل ۸۰-۲ بررسی کلی مدلهای آشکارساز علائم راهنمایی و رانندگی[۲۴۴]

۴-۲-۳-۲- شناسایی خطوط جاده

شناسایی خطوط جاده به منظور تخمین زدن حالت خودرو با توجه به خطوط شناسایی شده امری ضروری در خودروهای خودران است. این اطلاعات می تواند به عنوان بازخورد موقعیت خودرو برای سیستم های کنترل وسیله خودکار ارائه شود. از چند دهه پیش تاکنون حجم زیادی از کارهای تحقیقاتی در این حوزه انجام شده است. با این حال، هنوز به طور کامل حل نشده و به دلیل طیف گسترده ای از عدم اطمینان در شرایط واقعی ترافیک جاده، به عنوان یک مشکل چالش برانگیز باقی مانده است[۲۴۵].

انتخاب روشی مناسب به منظور شناسایی خطوط جاده تا حد بسیار زیادی به شرایط محیط بستگی دارد.

به طور کلی یک روش کلاسیک پردازش تصویر و چهار روش با رویکرد شبکه‌های عصبی پیشنهاد شد که هر کدام از این شبکه ها متناسب با مجموع داده، دارای رتبه های یک در بنچمارک معرفی شده توسط سایت Paperswithcode بودند.

با توجه به چارچوب حل مسئله، اگر شرایط محیط اعم از نور و رنگ ثابت باشد، روش کلاسیک به خوبی پاسخگو این نیاز خواهد بود و می تواند دقت خوبی در زمان مناسبی ارائه دهد. ولی اگر شرایط محیطی متغیر باشد، به طور مثال در حالت واقعی باشد، ممکن است روش کلاسیک تا حد مناسبی شرایط مورد نیاز را برآورده نکند. در

حالت واقعی چالش های بسیار زیادی وجود دارد، مانند باران، برف، انسداد جاده، شب و ... و همانطور که گفته شد بدلیل اینکه الگوریتم های کلاسیک مبتنی بر محیط هستند باید از شبکه های عصبی معرفی شده، استفاده شود.

روش های جدید (مبتنی بر شبکه های عصبی) تشخیص خطوط در شرایط پیچیده دنیای واقعی عملکرد چشمگیری داشته‌اند، اما بسیاری از آنها دارای مشکلات مربوط به حفظ کارایی در زمان واقعی هستند که برای وسایل نقلیه خودران مهم است. از این رو دو روش PINet و LaneATT که در سال ۲۰۲۰ معرفی شدند، توانستند تا حد بسیار خوبی نسبت به این اتفاق منعطف باشند.

در شبکه PINet می توان اندازه مدل های آموزش دیده را با توجه به توان محاسباتی محیط هدف انتخاب کرد و دقت و کارایی شبکه را در عین بهینه کردن آن حفظ کرد و در شبکه LaneATT حتی در زمان‌هایی که نشانگرهای خط از دست رفته باشد باز هم دقت بسیار خوبی در زمان واقعی ارائه می‌دهد.

به طور کلی هر دو این روش‌ها توانایی پیاده سازی در همه محیط را دارند و با توجه به اینکه در شرایط محیطی متفاوتی که در مجموع داده پدید آمده، آموزش دیده‌اند، میتوان از این دو روش استفاده کرد و دقت قابل قبولی کسب کرد.

۵-۲-۳-۲- ردیابی اجسام

ردیابی برای وسایل نقلیه خودران شامل شناسایی دقیق و مکان‌یابی اشیا پویا در محیط اطراف خودرو است. ردیابی وسایل نقلیه در رانندگی خودران یک امر ضروری به منظور جلوگیری از برخورد به مانع، برنامه ریزی مسیر و تشخیص مسیر هدف است[۲۴۶]. درجدول ۶-۲ به تحلیل و بررسی روش‌های گفته شده پرداخته شده است.

در مسئله خودرو های خودران، به منظور بهبود عملکرد ردیابی اشیا از داده های سنسوری نیز استفاده می شود. با توجه به انطباق راه حل ها با چارچوب حل مسئله، بحث ردیابی اشیا در این پروژه مطرح نیست و با توجه به اینکه به داده های سنسوری دسترسی وجود ندارد و ادراک محیط در قالب یک ویدئو صورت می گیرد، نیازی به انجام این بخش در چارچوب این پروژه نمی باشد[۲۴۷].

جدول ۶-۲ مقایسه روش های ردیابی اجسام متحرک[۲۴۶،۲۴۷]

روش ها	مزایا	معایب
فیلتر کالمن	* می تواند شی یا نقاط را در تصاویر نویزی ردیابی کند.	* دارای محدودیت در تعریف متغیرهای حالت به صورت توزیع نرمال(گوسی) است.
KLT tracker	* از لحاظ زمانی کارآمد است. * در مقابل انسداد[۱۹۱] به خوبی عمل می کند.	* در این روش ردیابی چند بسیار پیچیده است.
میانگین تغییر	* قابل استفاده برای شرایط با رنگ غالب.	*وقتی پس زمینه مشابه شی هدف باشد، ردیابی با مشکل مواجه می‌شود.
ROLO	* آشکارساز آن مبتنی بر YOLO است.	* فقط توانایی ردیابی یک شی را دارد. * دقت مناسبی ندارد.
SiamMask	* ماسک شی را نیز بر می گرداند. * به صورت زمان واقعی اجرا می شود. (۵۵ فریم بر ثانیه) * دقت بسیار بالایی دارد.	* فقط توانایی ردیابی یک شی را دارد. * نیاز به تعیین جعبه محدود کننده شی به صورت دستی است.
Deep SORT	* توانایی ردیابی چندین شی به صورت همزمان را دارد. * به صورت زمان واقعی اجرا می شود. * دقت مناسبی دارد.	* در مواردی که نیاز به دقت فوق العاده ای است ممکن است انتظارات را برآورده نکند.
Tracktor++:	* توانایی ردیابی چندین شی به صورت همزمان را دارد. * از Faster R-CNN به عنوان آشکارساز استفاده می کند. * بسیار دقیق است.	* به صورت زمان واقعی اجرا نمی شود و به طور متوسط خروجی ۳ فریم بر ثانیه می دهد.

191 Occolusion

۶-۲-۳-۲- تشخیص فاصله خودرو نسبت به خودرو های دیگر

در این بخش، قصد داریم تا راه حل های مختلفی را که برای بخش تشخیص فاصله خودرو نسبت به خودرو های دیگر (به طور خاص خودروی جلویی) ارائه کردیم، با یکدیگر مقایسه کنیم و ضمن انطباق راه حل ها با چارچوب حل مسئله، مزایا و معایب راه حل ها را نیز بررسی نماییم و به جمع بندی در مورد روش‌های معرفی شده برسیم. یکی از روش هایی که ارائه شد، روش استفاده از پردازش تصویر کلاسیک بود که در آن با استفاده از تبدیل پرسپکتیو و استفاده از پیکسل های تصویر، فاصله ی حدودی تا خودروی رو به رویی را محاسبه می‌شود. از جمله مزایای این روش می توان به در دسترس بودن آن در همه شرایط و هر گونه تصویر، عدم نیاز به سخت افزار و سنسور خاص و همچنین ساده و قابل پیاده سازی بودن روش اشاره نمود.

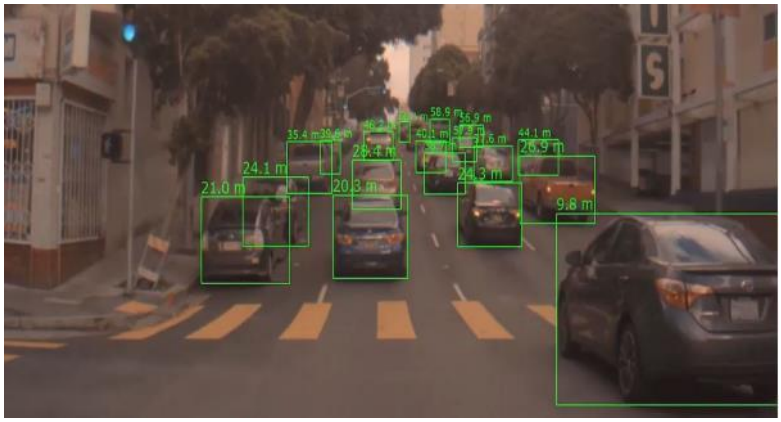
از معایب این روش هم می توان به این مورد اشاره نمود که در این روش، میزان دقت ما خیلی زیاد نیست و چون در این روش به جای استفاده از سنسور ها و سخت افزار های مرتبط، از تبدیلات پرسپکتیو و نسبت های پیکسل در هر اینچ استفاده می شود، فاصله ی محاسبه شده، دارای دقت خوب اما اندکی کمتر از روش های مبتنی بر استفاده از سنسور می باشد[۲۴۸].

شکل ۸۱-۲ نمونه ای از عملکرد روش مبتنی بر پردازش تصویر کلاسیک[۲۴۸]

اما روش دیگری که در این بخش به آن اشاره گردید، روش محاسبه فاصله با خودروی جلویی بر اساس شبکه های عصبی کانولوشنی و بازگشتی بود که در این روش، باید با استفاده از سنسورهایی همجون رادار و لیدار، اطلاعات دقیق از محیط اطراف خودرو جمع آوری گردد و در نهایت با استفاده از یک مدل آموزش داده شده روی این داده ها، فاصله تا خودروی رو به رویی را با دقت خوبی تخمین زد.

از مزایای این روش می توان به دقت بسیار بالا، سرعت زیاد و ضریب اطمینان مناسب اشاره نمود اما در برابر این مزایا، معایب مهمی همچون نیاز به استفاده از سنسور های گران قیمت و تجهیزات سخت‌افزاری دیگر نیز وجود

دارد، علاوه بر این، بروز یک اشتباه در آموزش دادن مدل (به عنوان مثال Overfitting) می تواند باعث خطای تخمین در تمام حالات و تصاویر گردد[۲۴۹].



شکل ۸۲-۲ نمونه ای از عملکرد روش مبتنی بر شبکه های عصبی و سنسور های رادار و لیدار[۲۴۹]

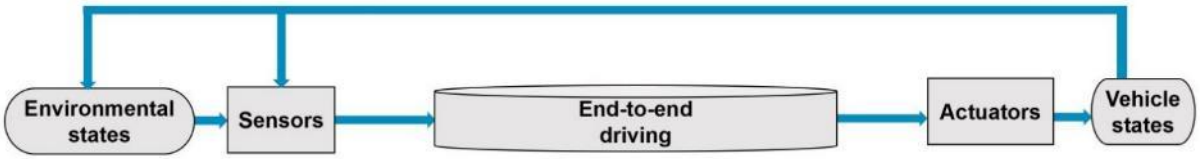
به عنوان جمع بندی و انتخاب نهایی بین روش های ارائه شده، با توجه به نوع مساله تعریف شده توسط شرکت طراح، عملا امکان استفاده از تجهیزاتی همچون لیدار و رادار جهت تهیه ی خوراک مورد نیاز مدل مبتنی بر شبکه های عصبی ما وجود ندارد و حتی اگر این امکان نیز وجود داشت، معایب دیگری همچون هزینه‌ی زیاد این تجهیزات و بروز مشکلاتی همچون نیاز به کالیبرایسیون سنسورها و امکان به وجود آمدن استهلاکات، ما را ملزم به استفاده از روش اول، یعنی استفاده از تبدیل پرسپکتیو و فاصله پیکسلی می نماید.

۷-۲-۳-۲- تقلید رفتاری

همانطور که توضیح داده شد، یکی از روش‌های کنترل خودروهای خودران استفاده از تقلید رفتاری انسان می‌باشد که خودرو حرکت خود را مستقیماً از ورودی های سنسور ایجاد می کند. مهندسان NVIDIA در سال ۲۰۱۶ با آموزش دادن یک شبکه ی کانولوشنی، موفق شدند که داده های دریافتی از یک دوربین را مستقیما به داده ی مورد نیاز برای چرخش فرمان خودرو تبدیل کنند[۲۴۹].

در تصویر ۸۳-۲ نحوه عملکرد این روش آورده شده که کنترل خودرو خودران با استفاده از داده‌های سنسور به صورت انتها به انتها[۱۹۲] صورت می گیرد.

192 End-to-end



شکل ۸۳-۲ ساختار عملکردی انتها به انتها یک خودرو خودران[۲۳۶]

روش تقلید رفتاری دارای چندین محدودیت است که کنترل یک خودرو رو را تحت تاثیر قرار می دهد. اولین مورد آن می توان به بایاس دیتاست اشاره کرد، با توجه به اینکه عملکرد این شبکه بر اساس تقلید رفتاری است و بیشتر رانندگی در دنیای واقعی شامل چند رفتار ساده یا یک واکنش سریع در برابر یک اتفاق نادر است، مجموعه داده‌گان می بایست تنوع کافی داشته باشند و ایجاد تنوع در مجموعه داده کار بسیار سخت و پیچیده ای است. علاوه بر آن مشکل وقوع overfit در آموزش شبکه نیاز بسیار محتمل است که باعث می شود خودرو در شرایط حساس مانند دور زدن هنگام رسیدن به یک مانع، به خوبی عمل نکند.

با توجه به چارچوب مسئله که در قالب یک ویدئو است، این روش نیازمند دسترسی کامل به عملگر یا فرمان خودرو است که بتوان کنترل خودرو را به صورت خودران انجام داد و این روش در چارچوب حل مسئله نمی-گنجد[۲۴۸].

۸-۲-۳-۲- مکان یابی

سیستم تشخیص موقعیت خودرو از موارد مهم در یک خودرو خودران است. این سیستم باید بتواند موقعیت خودرو نسبت به موانع ثابت و متحرک را تشخیص دهد. لکن با توجه به این مهم که تشخیص موقعیت خودرو چه در مقیاس بزرگ و چه در مقیاس کوچک از معیار های ارزیابی این پروژه نمی‌باشد و علاوه بر آن برای این بخش نیازمند اطلاعات سنسور است، استفاده از الگوریتم های تشخیص موقعیت خودرو اصلی و خودروهای اطراف که تا حد زیادی به بخش ردیابی خودرو های اطراف وابسته است مورد نیاز نبوده و به پیاده‌سازی این الگوریتم ها پرداخته نمی‌شود[۲۴۹].

۳- ادراک

خودرو خودران با استفاده از تجهیزات و سنسورهای مختلفی از جمله رادار، لیدار، GPS و همچنین با استفاده از تکنیک‌های پردازش تصویر با جهان بیرون ارتباط برقرار میکند و به اصطلاح آن را درک می کند.

با توجه به این که پروژه ی مسابقه در حیطه ی شبیه سازی تعریف شده است، عمده ی توجه ما روی بخش پردازش تصویر، شبکه های عصبی و بینایی ماشین است و بخش های کار با سنسور در این پروژه که در حوزه ی SIL[193] تعریف شده جایگاهی ندارد.

بینایی ماشین شامل روشهای مربوط به دستیابی تصاویر، پردازش، آنالیز و درک محتوای آن است. این سیستم تصاویر دنیای بیرون را به عنوان ورودی دریافت و دادههای عددی یا سمبلیک را به عنوان خروجی تولید مینماید. این سیستم با الگو برداری از سیستم بینایی انسان در رایانه شبیه‌سازی شده است.

به طور کلی حوزه ی ادراک محیط با بینایی ماشین در خودرو های خودران را می توان به بخش های مختلفی از جمله شناسایی علائم و چراغ های راهنمایی رانندگی، شناسایی اشیای ثابت و دنبال کردن اشیای متحرک، شناسایی لاین های جاده، تفکیک اشیا از هم و فاصله سنجی از اشیا و لبه ها تقسیم کرد و ضمنا اکثر بخش های این تقسیم بندی را می توان هم با روش های پردازش تصویر کلاسیک انجام داد و هم با روش های مبتنی بر شبکه های عصبی عمیق. لازم به ذکر است که تمام بخش های معرفی شده در حیطه ی پروژه تعریف شده در مسابقه نمی گنجند و تنها تعدادی از آن ها در بخش پیاده سازی مورد استفاده قرار می گیرند، اما در این بخش، جمع آوری اطلاعات درباره ی آن ها و بررسی و تحلیل آن ها خالی از لطف نیست.

در ادامه ضمن ارائه توضیحات و محتوای مرتبط با موضوع پروژه با استناد به مقالات و رساله های داخلی و خارجی، چارچوبی برای حل مساله ترسیم می کنیم و هم چنین در هر یک از بخش های این چارچوب، روش های مختلف را به دسته بندی های مختلف از جمله مرسوم ترین و کلاسیک ترین روش و جدید ترین روش تقسیم می کنیم.

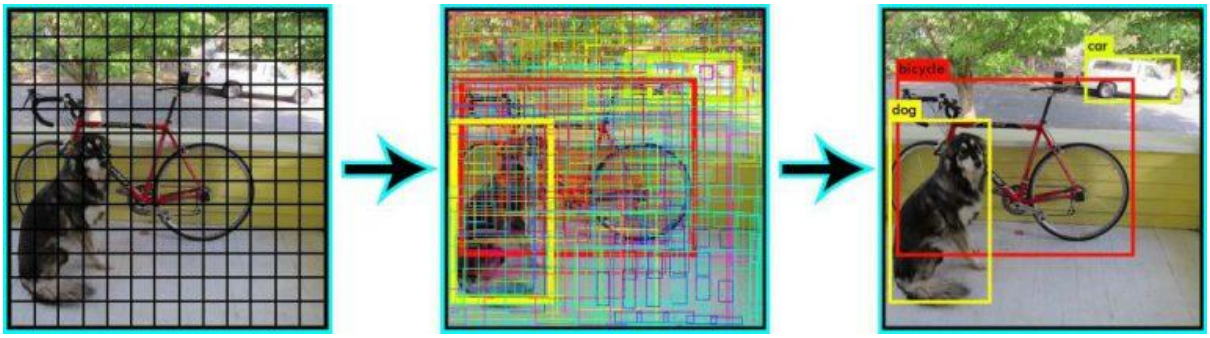
۱-۳- شناسایی عابر، وسایل نقلیه و چراغ راهنمایی و رانندگی

[193] Software In the Loop

تشخیص اشیا یک فناوری کامپیوتری است که در ارتباط با کامپیوتر ویژن و پردازش تصویر است که با شناسایی نمونه هایی از اشیاء معنایی یک کلاس خاص (مانند انسان ها، ساختمان ها و اتومبیل ها) در تصاویر و فیلم های دیجیتال مورد بررسی قرار می گیرد. در واقع در این فناوری ماشین تصویر را میبیند و پس از پردازش هایی مکان اشیا تعریف شده و کلاس یا دسته‌بندی آنها را نیز مشخص میکند.

شناسایی اشیا به شناسایی مکان و اندازه اشیا مورد نظر اشاره دارد. شناسایی اشیا چه در حالت ایستا که شامل چراغ های راهنمایی و رانندگی و نشانه های عبور از جاده و چه در حالت پویا که شامل وسایل نقلیه، عابرین پیاده و دوچرخه سواران می شود، همواره مورد بحث در علم هوش مصنوعی بوده است. در این بخش درباره پیشرفته‌ترین و به‌روزترین روش‌های پیشنهادی شناسایی اشیا بحث شده که این خود نقطه شروع چندین کار دیگر از جمله ردیابی اشیا و درک محیط می باشد[۲۵۰].

YOLO کامل‌ترین سیستم زمان واقعی (Real time) در یادگیری عمیق و حل مسائل تشخیص اشیا است. این الگوریتم ابتدا تصویر را به بخش‌های مختلف تقسیم می‌کند و هر بخش را علامت‌گذاری می‌کند، سپس الگوریتم شناسایی را به صورت موازی برای تمامی این بخش‌ها اجرا می‌کند تا ببیند هر بخش به کدام دسته‌بندی تعلق می‌گیرد. بعد از شناسایی کامل اشیا، به دور اشیا مورد نظر جعبه های محدود کننده رسم می‌کند.



شکل ۱-۳ نمایی از مراحل شناسایی اشیا توسط YOLO [۲۵۴]

همه این موارد ذکر شده به صورت موازی انجام می‌شوند؛ در نتیجه به صورت زمان واقعی است.

۱-۱-۳- تبیین روش مورد نظر

به طور کلی یک آشکارساز شی به منظور استخراج ویژگی های ورودی و ارسال این ویژگی ها به یک سیستم پیش‌بین طراحی شده است تا بتواند جعبه‌ای به دور شی تشخیص داده شده بیاندازند و کلاس آن را تشخیص دهد.

یولو اولین آشکارسازی است که این روند را به صورت انتها به انتها[۱۹۴] طی می کند.

YOLO شامل سه بخش است:

- **Backbone**: یک شبکه عصبی کانوولوشنی که وظیفه استخراج ویژگی را بر عهده دارد.
- **Neck**: به مجموعه ای از لایه‌هایی که به منظور ترکیب ویژگی‌های تصویر و انتقال آن‌ها به سمت پیش بینی کننده عمل می‌کنند گفته می‌شود.
- **Head**: ویژگی ها را از Neck دریافت می کند و مراحل پیش بینی مکان جعبه و کلاس را انجام می دهد.

YOLO دارای ورژن های متفاوتی از ۱ تا ۵ می باشد که شروع شکوه آن از ورژن سه بود که توانست در سال ۲۰۱۸ دقت بی نظیری را ارئه دهد. در ادامه، در سال ۲۰۲۰ دو ورژن بعدی آن یعنی چهار و پنج در فاصله کوتاهی از یکدیگر منتشر شدند. YOLOv4 با موارد جدید حیرت انگیز معرفی شد، با اختلاف زیاد از YOLOv3 بهتر عمل کرد و همچنین به مقدار قابل توجهی دقت متوسط آن افزایش یافت. با فاصله کمی از منتشر شدن YOLOv4، YOLOV5 منتشر شد و این نسخه توانست و عملکرد بهتری نسبت به تمام نسخه های قبلی داشته باشد.

در این بخش برای شناسایی عابر، وسایل نقلیه و چراغ راهنمایی و رانندگی از YOLOV5 استفاده شده است و در ادامه به تبیین دقیق‌تر این روش یادگیرنده و عمیق پرداخته شده است.

[194] End-to-end

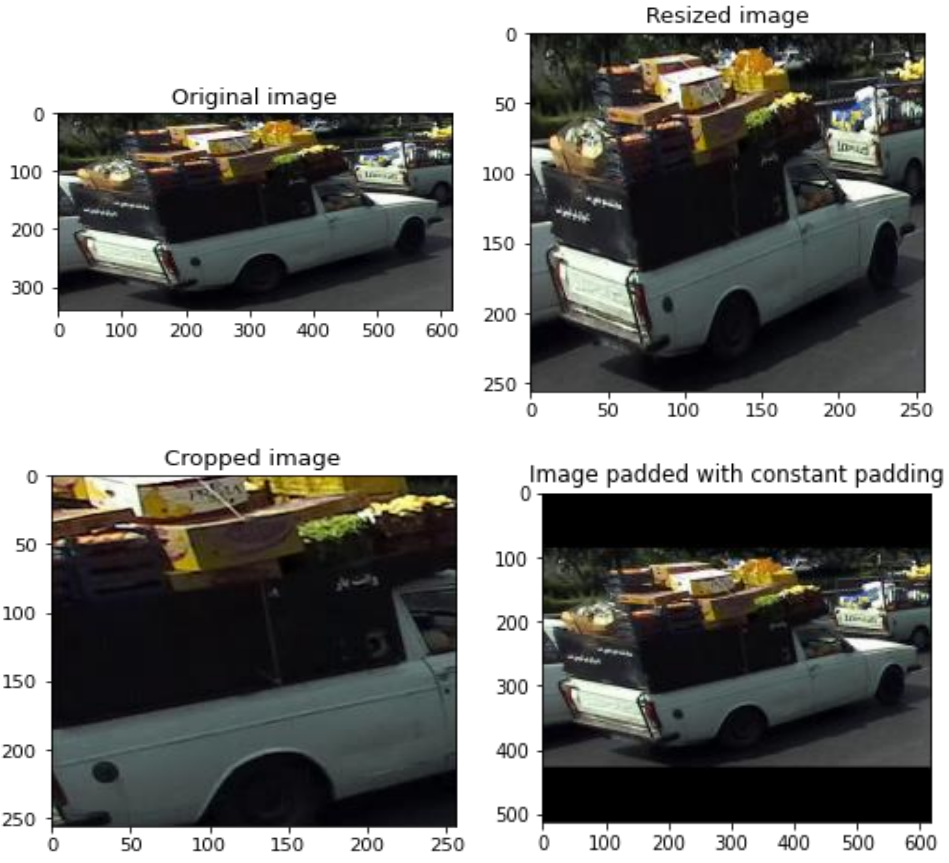
روند آموزش شبکه برای بررسی عملکرد نهایی سیستم بسیار مهم هست ک دو فرآیند تاثیرگذار و مهم آن به صورت زیر است:

- **داده افزایی:**

داده‌افزایی شامل مجموعه‌ای از تکنیک‌ها است که اندازه و کیفیت مجموعه داده‌های آموزشی را افزایش می‌دهد، به طوری که می‌توان با استفاده از آنها مدل های آموزش عمیق بهتری ایجاد کرد. الگوریتم‌های داده‌افزایی مورد بحث در این بررسی شامل تبدیل مقیاس تصویر، تنظیم فضای رنگ تصویر و داده‌افزایی موزاییکی می باشد[۲۵۱].

- **تبدیل مقیاس تصویر:**

در تصویر ۲-۳ نمونه‌ای از داده‌افزایی با استفاده از روش‌های تبدیل مقیاس تصویر آورده شده است.



شکل ۲-۳ نمونه‌ای از داده‌افزایی با استفاده از روش تبدیل مقیاس تصویر

- **تنظیم فضای رنگ تصویر:**

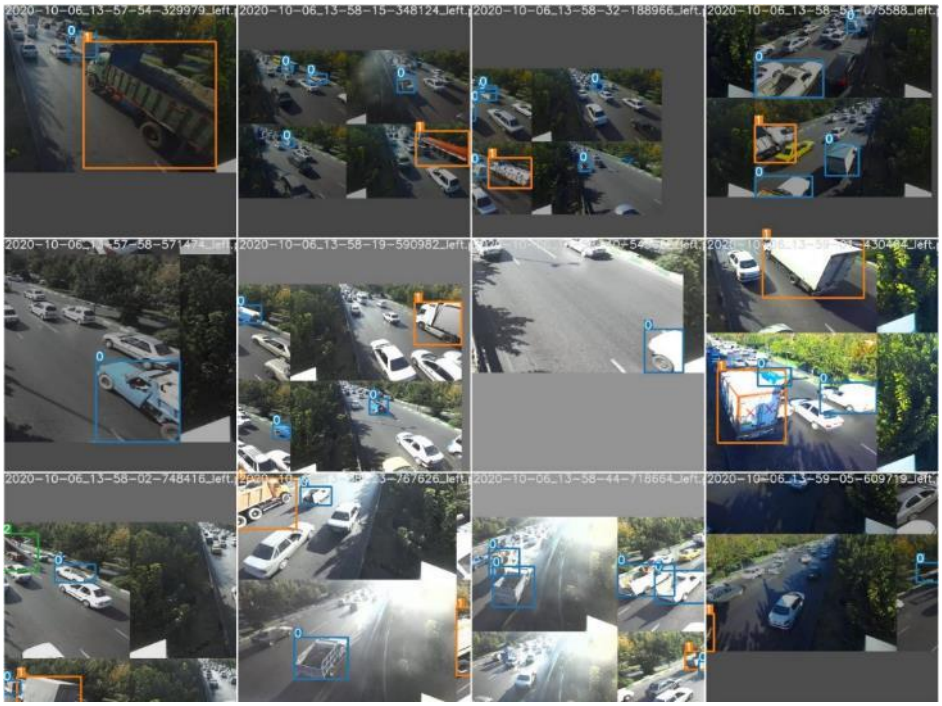
داده های تصویر در ۳ کانال انباشته شده تعریف می‌شوند. این کانال‌ها نشان دهنده مقادیر پیکسل برای یک مقدار رنگ RGB منفرد هستند. بایاس روشنایی از جمله چالشهایی است که اغلب در بروز مشکلات تشخیص تصویر وجود دارد. بنابراین، استفاده از روش تنظیم فضای رنگ تصویر برای جلوگیری از این بایاس قابل پیشبینی، بسیار کمک کننده است.

در تصویر ۳-۳ نمونه‌ای از داده‌افزایی با استفاده از روش تنظیم فضای رنگ تصویر آورده شده است.

شکل ۳-۳ نمونه‌ای از داده‌افزایی با استفاده از روش تبدیل تنظیم فضای رنگ

- **داده‌افزایی موزاییکی**

یکی از دلایل بهتر شدن نتایج YOLOv5 استفاده از ایده داده افزایی موزاییکی در روند آموزش داده‌ها است که نمونه ای از این روش در تصویر ۴-۳ آورده شده است.



شکل ۴-۳ نمونه‌ای از روش داده‌افزایی موزاییکی

همانطور که در تصویر ۴-۳ مشخص است، داده افزایی موزاییکی چهار تصویر را به چهار موزاییک با نسبت تصادفی ترکیب می کند که این خود باعث بهبود عملکرد آموزش شبکه می‌شود.

- **محاسبه تابع هزینه**

YOLO تابع هزینه را با استفاده از ترکیب خطی توابع هزینه GIoU، Objectness و Classification محاسبه می‌کند.

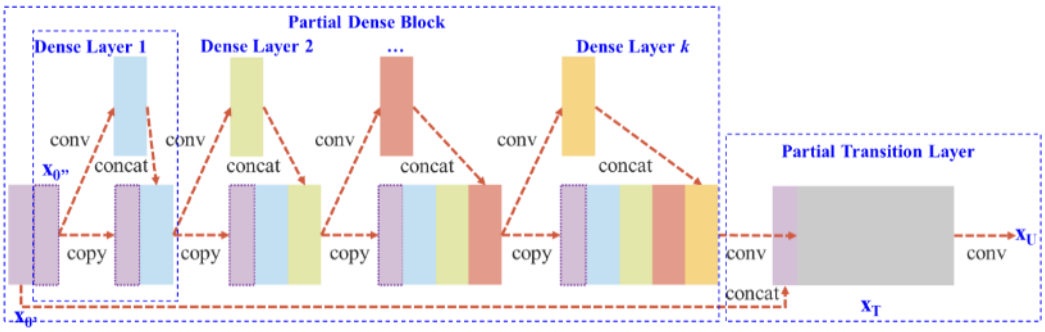
- GIoU: این تابع بیان می کند جعبه محدودکننده چقدر به لیبل نزدیک است.
- Objectness: این تابع برای دسته‌بندی کردن پس‌زمینه و پشت‌زمینه استفاده می‌شود.
- Classification: این تابع برای دسته‌بندی کردن کلاس‌های مختلف تحت آموزش مورد استفاده قرار می‌گیرد.

با مجموع توابع هزینه فوق، تابع هزینه کلی به صورت زیر تعریف می‌شود که شبکه در روند آموزش سعی در کاهش مقدار آن دارد.

loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{۳-۱}
$$

علاوه بر نکات گفته شده، شبکه YOLO5 از ساختار CSPNet برای قسمت Backbone خود استفاده می‌کند. هدف اصلی از طراحی CSPNet این است که این معماری بتواند ضمن کاهش میزان محاسبه، به همگرایی بهتری در محاسبه گرادیانت[۱۹۵] دست یابد. این هدف با تقسیم بردار ویژگی لایه اول به دو قسمت و سپس ادغام آنها از طریق یک سلسله مراتب مقطعی پیشنهادی حاصل می شود. مفهوم اصلی این است که با تقسیم جریان گرادیانت، جریان گرادیانت را از طریق مسیرهای مختلف شبکه منتشر کند و به این ترتیب CSPNet می تواند مقدار محاسبات را تا حد زیادی کاهش دهد و سرعت استنتاج را نیز بهبود بخشد.ساختارشبکه توضیح داده شده در تصویر ۵-۳ آورده شده است[۲۵۲].



شکل ۵-۳ ساختار شبکه CSPNet [۲۷].

بزرگترین سهم YOLOv5 ترجمه چارچوب Darknet به چارچوب PyTorch است. چارچوب Darknet به زبان C نوشته شده و امکان کنترل دقیق روی عملیات رمزگذاری شده در شبکه را فراهم می کند. از بسیاری جهات، کنترل زبان سطح پایین مزیت است، اما می تواند انتقال آن را تحقیقات جدید کندتر کند و نحوه گسترش و پیشرفت آن را سخت می سازد.

این شبکه در قسمت Neck خود از PANet و از ایده یادگیری خودکار جعبه انکرهای محدود کننده با استفاده از الگوریتم K-means و الگوریتم های ژنتیک استفاده می‌کند که باعث می‌شود در صورت وجود کلاسی با ابعاد بزرگ و کلاس دیگری با ابعاد کوچک، دقت شبکه در تشخیص اشیا کاهش نیابد. این تغییرات گفته شده سرعت،کارآمدی و بهینه بودن این آشکارساز را نسبت به ورژن‌های پیشین خود برتری می دهد[۳۰] .

شبکه YOLOv5 دارای پنج مدل است که مقایسه آن‌ها بر اساس زمان پردازش، دقت و تعداد پارامترها در جدول ۳-۱ مشاهده کرد.

[195] gradient

جدول ۱-۳ مقایسه ای از مدل های مختلف YOLOV5بر اساس سرعت، FLOPS، دقت و تعداد پارمترها[۲۵۳]

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}		params	FLOPS
YOLOv5s	37.0	37.0	56.2	**2.4ms**	**416**		7.5M	13.2B
YOLOv5m	44.3	44.3	63.2	3.4ms	294		21.8M	39.4B
YOLOv5l	47.7	47.7	66.5	4.4ms	227		47.8M	88.1B
YOLOv5x	**49.2**	**49.2**	**67.7**	6.9ms	145		89.0M	166.4B
YOLOv5x + TTA	**50.8**	**50.8**	**68.9**	25.5ms	39		89.0M	354.3B

با توجه به جدول ۱-۳، برای آموزش داده‌ها از YOLOv5s به‌دلیل زمان پردازش بسیار پایین در عین حال دقت بالا استفاده شده است که در بخش ۲-۱-۳ نحوه آموزش آن بر روی کلاس های

- Car
- Truck
- Bus
- Traffic Light
- Person

بیان شده است.

۲-۱-۳- چگونگی آموزش شبکه YOLOv5

در ادامه این بخش نحوه آموزش شبکه Yolov5 بر روی Custom Dataset توضیح داده شده است.

قبل از شروع آموزش شبکه، ابتدا مجموعه کد YOLOv5 را از آدرس گیت‌هاب[۲۹] به صورت زیر دانلود میکنیم.

```
$ git clone https://github.com/ultralytics/yolov5  # clone repo
```

سپس به پوشه دانلود شده رفته و کتابخانه های مورد نیاز که در فایل requirement.txt قرار دارند را برای آموزش این شبکه به صورت زیر نصب میکنیم:

```
$ cd yolov5
$ pip install -r requirements.txt  # install dependencies
```

برای آموزش شبکه YOLOv5، نیاز به نسخه پایتون بزرگتر از ۳.۸ و چارجوب پایتورچ بزرگتر ۱.۷ است.

۱. ساخت فایل Dataset.yaml

برای ساخت فایل Dataset.yaml می بایست به صورت زیر

۱- مسیر داده‌های آموزشی و ارزیابی مشخص شود،

۲- سپس تعداد کلاس ها مشخص شود،

۳- و در مرحله آخر لیستی از نام کلاس‌ها نوشته شود.

```
# train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3) list: [path1/images/, path2/images/]
train: ../Rahneshan/images/train/
val: ../Rahneshan/images/val/

# number of classes
nc: 5

# class names
names: ['person', 'car', 'bus', 'truck', 'traffic light']
```

۲. ساخت لیبل جعبه‌های محدود کننده هر تصویر متناسب با فرمت YOLOv5

برای آموزش این شبکه برای ۵ کلاس

- Car
- Truck
- Bus
- Traffic Light
- Person

از دیتاست COCO[۱۹۶] استفاده کردیم.

دیتاست COCO که مخفف اشیا معمول در میان سایر اشیا می‌باشد شامل ۳۳۰ هزار تصویر از صحنه‌ها و شرایط متفاوت می‌باشد که از این ۳۳۰ هزار تصویر بالغ بر ۲۰۰ هزار تصویر دارای لیبل‌های کامل می‌باشند. این دیتاست شامل ۸۰ کلاس مختلف از مواردی که ممکن است به صورت روزمره با آن‌ها برخورد داشت را دارا می‌باشد، برای مثال انسان، خودرو، کفش، لیوان و ... را می‌توان از کلاس‌های این دیتاست نام برد.

COCO از معروف‌ترین دیتاست‌های امروزی بوده که برای کاردبردهای مختلف استفاده می‌شود، و شامل انواع لیبل برای انجام موارد شناسایی اشیا[۱۹۷] کلاس‌بندی[۱۹۸]، بخش‌بندی[۱۹۹]، نقاط‌کلیدی[۲۰۰] و ... می‌باشد و بسیاری از مسابقات حوزه هوش‌مصنوعی بر مبنای استفاده از این دیتاست است.

فرمت بخش شناسایی اشیا در دیتاست COCO به صورت زیر می‌باشد:

```
1. x_min y_min x_max y_max
```

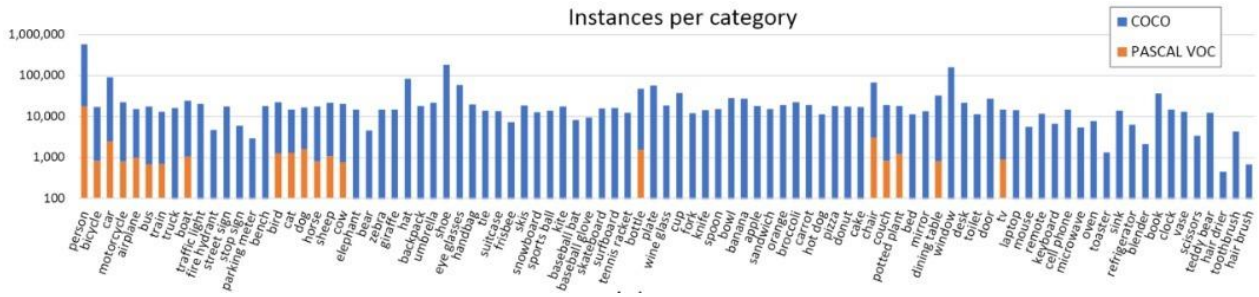
که نقاط بالا سمت چپ و پایین سمت راست باندینگ باکس در آن مشخص شده است. این اطلاعات علاوه تفاوت نوع اعداد با Yolo، دارای تفاوت ساختاری نیز می‌باشند و کل لیبل‌ها در یک فایل *json*. ذخیره شده است، برای استفاده از این دیتاست نیاز است که فرمت داده‌های موجود در آن به قالب مورد نیاز Yolo تبدیل شوند که این کار در ادامه انجام شده است.

[196] Common Objects in Context
[197] Object Detection
[198] Classification
[199] Segemntation
[200] Key-points

جدول ۲-۳ مشخصات دیتاست COCO

Dataset Name	COCO
تعداد تصاویر	**۳۳۰,۰۰۰**
لیبل زده شده	**۲۰۰,۰۰۰**
تعدا کلاس‌ها	**۸۰**
حجم دیتاست	**۱۷ گیگابایت**
ابعاد تصویر	**متفاوت**

که توزیع کلی داده‌های دیتاست COCO در تصویر ۶-۳ آورده شده است.



شکل ۶-۳ توزیع کلاس‌های مختلف دیتاست COCO

برای تبدیل به فرمت یولو می‌بایست برای هر تصویر یک فایل *.txt* ساخته شود که در هر ردیف لیبل های هر شی به فرمت YOLOv5 و کلاس آن قرار دارد. به طور کلی الگوریتم تبدیل لیبل‌های هر دیتاست به فرمت YOLOv5 به صورت مقابل می‌باشد:

در هر تصویر،

- یک ردیف برای هر شی،
- هر ردیف به فرمت *class <x_center> <y_center> <width> <height>* می باشد.
- مختصات هر جعبه محدود کننده باید به صورت نرمالایز شده یعنی عددی بین صفر و یک باشد. بدین صورت که *x_center* و *width* میبایست بر عرض تصویر تقسیم شوند و *y_center* و *height* بر طول تصویر.
- شماره کلاس‌ها نیز از صفر شروع می‌شود.

این روند را با استفاده از دو تابع زیر می‌توان انجام داد، که خروجی نهایی آن‌ها یک فایل *.txt* که حاوی مختصات نرمال شده هر جعبه محدود کننده و کلاس شی می‌باشد که نمونه ای از تصویر دیتاست و لیبل نهایی آن که به فرمت YOLO می باشد در تصویر ۷-۳ آورده شده است:

```
1.  def yolov5_format(f, img_shape, class_id, xmin, ymin, x_width, y_length):
2.      xmax = xmin + x_width
3.      ymax = ymin + y_length
4.      height, width = img_shape
5.      line ="{} {} {} {} {}".format(class_id, (xmin+xmax)/(2*width) , (ymin+ymax)/(2*height)
    , (xmax-xmin)/width, (ymax-ymin)/height)
6.      f.write(line)
7.      f.write('\n')
8.
9.  def imgdict_ann(img_dict , target_dir = 'Yoloformats/'):
10.     global counter
11.     if not os.path.exists(target_dir):
12.         os.mkdir(target_dir)
13.     img_id = img_dict['id']
14.     annIds = coco.getAnnIds(imgIds=img_id, catIds=all_desired_cat_ids, iscrowd=None)
15.     anns = coco.loadAnns(annIds)
16.     h,w = img_dict['height'] , img_dict['width']
17.     new_name = img_dict['file_name'].split('.')[0] + '.txt'
18.     f = open(target_dir + new_name , 'w')
19.     for ann in anns:
20.         counter += 1
21.         bbox = ann['bbox']
22.         cat  = ann['category_id']
23.         cat = all_desired_cat_ids.index(cat)
24.         yolov5_format(f, (h,w), cat, *bbox)
```

شکل ۷-۳ نمونه‌ای از تصویر لیبل زده شده موجود در دیتاست COCO [۲۵۳]

لیبل نهایی و به فرمت یولو تصویر ۷-۳ به صورت مقابل می‌باشد:

File Edit Format View Help

```
0 0.0469136 0.644907 0.0938272 0.347222
5 0.516667 0.456481 0.959259 0.522222
0 0.174691 0.599537 0.233333 0.467593
0 0.916667 0.580093 0.166667 0.460185
0 0.350617 0.588426 0.158025 0.417593
```

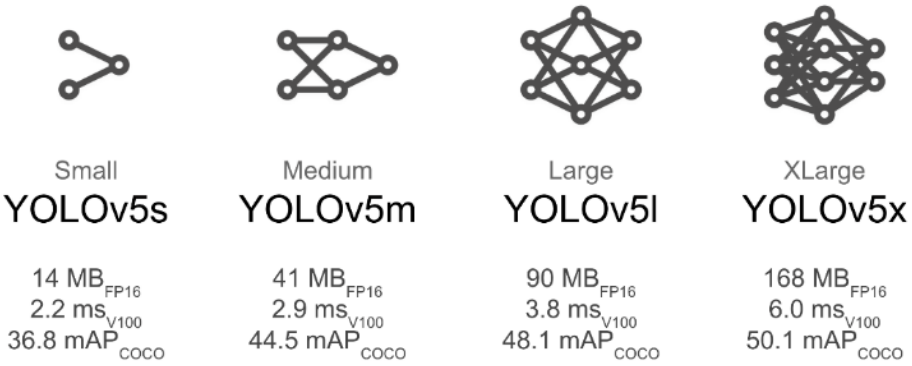
۳. سازمان‌دهی پوشه‌ها

نحوه پوشه بندی دیتاست به صورت مقابل در تصویر ۸-۳ آورده شده است و می‌بایست بدین فرمت باشد.

شکل ۸-۲ ساختار پوشه‌بندی دیتاست

۴. انتخاب مدل

از بین مدل‌های یولو که در تصویر ۹-۳ آورده شده، بدلیل زمان پردازش و در عین حال داشتن دقت بالا و روند آموزش سریع، مدل YOLOv5s را انتخاب شده است.



شکل ۹-۳ مدل‌های مختلف YOLOv5 و مشخصات آن‌ها [۲۵۳]

بعد از انتخاب مدل، می‌بایست مدل انتخاب شده را از پوشه yolov5/models انتخاب کرده و درون فایل *yolov5*.yaml* ، تعداد کلاس را متناسب با تعداد کلاس دیتاست خود تغییر داد.

۵. تنظیم پارامترهای داده‌افزایی

در فایل hyp.scratch.yaml در پوشه data، میتوان پارامترهای داده افزایی که در قسمت قبل توضیح داده شد، تنظیم کرد که البته مقادیر پیشفرض در نظر گرفته شده، بهترین مقادیر میباشند.

۶. آموزش

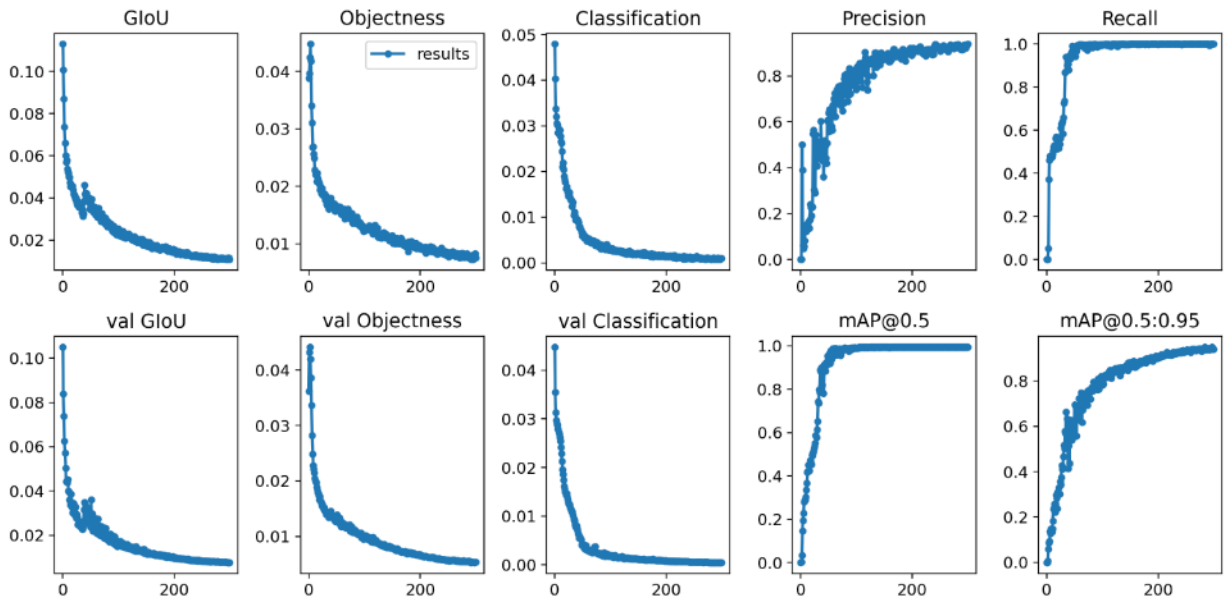
بعد از انجام مراحل فوق، با کد مقابل، شبکه YOLOv5s را می‌توان بر روی دیتاست خود آموزش داد:

```
1. $ python train.py --img 640 --batch 32 --epochs 300 --data dataset.yaml --
   weights yolov5s.pt
```

آموزش این شبکه برای ۵ کلاس تعریف شده در ۳۰۰ ایپاک، بتچ سایز ۳۲ و با وزن های از قبل آموزش داده شده بر روی دیتاست COCO و بر روی GPU T4 که توسط گوگل کولب در دسترس قرار گرفته است، آموزش داده شد.

۳-۱-۳- نتیجه آموزش و خروجی شبکه

نتایج آموزش آن را می توان در تصویر ۱۰-۳ مشاهده کرد که در ادامه تشریح هر کدام از معیارهای ارزیابی آورده شده است.



شکل ۱۰-۳ نتایج آموزش شبکه YOlOv5s

دسته بندی داده‌ها برای تحلیل:

۱. مثبت صحیح (True Positive) که یک معیار مثبت است. (درست شناسایی شده است)
۲. مثبت کاذب (False Positive) که یک معیار منفی است. (اشتباه شناسایی شده است)
۳. منفی صحیح (True Negative) که یک معیار مثبت است. (به درستی رد شده است)
۴. منفی کاذب (False Negative) که یک معیار منفی است. (اشتباه رد شده است)

Precision: 93.9%

وقتی که مدل نتیجه را مثبت[۲۰۱] پیش‌بینی می‌کند، این نتیجه تا چه اندازه درست است؟ زمانی که ارزش false positive بالا باشد، معیار صحت یا Percision، معیار مناسبی است. در واقع «حساسیت»

[201] Positive

معیاری است که مشخص می‌کند دسته‌بند، به چه اندازه در تشخیص کلاس‌های مثبت موفق بوده است. همانگونه که از رابطه فوق مشخص است، تعداد کلاس‌هایی که توسط دسته‌بند به اشتباه کلاس دیگری تشخیص داده شده‌اند، هیچ تاثیری در محاسبه این پارامتر ندارد و در واقع زمانی از این پارامتر به عنوان پارامتر ارزیابی استفاده می‌کنیم که هدف دستیابی به نهایت دقت در تشخیص نمونه‌های کلاس مثبت است[۲۵۵].

$$Percision = \frac{TP}{TP + FP} \qquad (۳-۲)$$

Recall: 100%

ممکن است در مواقعی دقت تشخیص کلاس منفی حائز اهمیت باشد.

زمانی که ارزش False Negatives بالا باشد، معیار Recall، معیار مناسبی خواهد بود.

$$\mathrm{Re}call = \frac{TP}{TP + FN} \qquad (۳-۳)$$

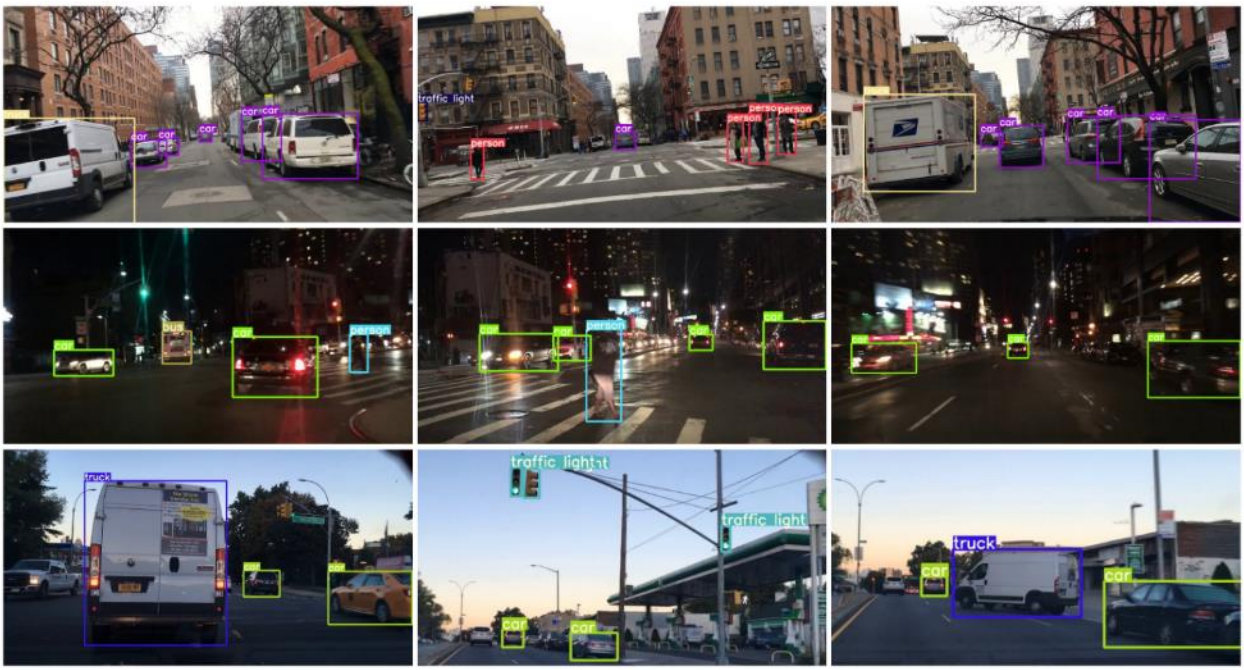
mAP@0.5: 99.5%

معیاری برای سنجش جعبه محدود کننده تشخیص داده شده است.

mAP@0.5:0.95: 93.91%

معیاری برای سنجش جعبه محدود کننده تشخیص داده شده است.

خروجی نتایج حاصل شده بر روی مجموع داده های تست در تصویر ۱۱-۳ آورده شده که همانطور که در تصویر مشاهده می‌شود، شبکه YOLOv5s توانسته کلاس های چراغ راهنمایی و رانندگی، اتوبوس، کامیون، سواری و عابر را برای هر ۳ ویدئو تست به خوبی تشخیص دهد.



شکل ۱۱-۳ نتیجه شبکه YOLOv5 بر روی داده‌های تست

۳-۱-۳- ساختار کد

در کد توسعه داده شده، ماژول yolo.py حاوی کلاس YOLO است که وظیفه شناسایی نوع کلاس‌های خودرو را بر عهده دارد. در main.py ابتدا این کلاس به ساختار اصلی اضافه شده و یک شیء[۲۰۲] از این کلاس ساخته می‌شود.

```
1. from elements.yolo import YOLO
2.
3. detector = YOLO('weights/yolo5s.pt')
4. yoloOutput = detector.detect('frame')
```

ابتدا مدل آموزش داده شده را Load می‌کنیم و سپس با فراخوانی تابع detect که در yolo.py است، کار تشخیص اشیا در هر فریم را انجام میدهیم.

خروجی yoloOutput به صورت مقابل خواهد بود که یکی دیکشنری از نام کلاس هر شی، مختصات باندینگ باکس آن، دقت شی تشخیص داده شده و شماره کلاس است.

```
1. item = {'label': label,
2.         'bbox' : [(xmin,ymin),(xmax,ymax)],
3.         'score': score,
4.      'cls'  : int(p(5))
5.         }
```

در بخش زیر نیز ورودی و خروجی ماژول yolo.py به اختصار آورده شده است:

```
1. """
2.     Input :
3.             BGR image
4.
5.
6.     Output:
7.     yolo return list of dict in format:
8.         {   label   :  str
9.             bbox    :  [(xmin,ymin),(xmax,ymax)]
10.            score   :  float
11.          Cls     :  int
12.        }
13. """
```

[202] Object

۲-۳- تشخیص علائم راهنمایی و رانندگی

شناسایی تابلوهای راهنمایی و رانندگی از موارد بسیار پراهمیت در درک محیط خودروهای خودران می‌باشد. زیرا قوانین و مقررات مربوط به مسیرهای رانندگی را مشخص می‌کنند و خودروها باید به آن‌ها پایبند باشند، در غیر این صورت سامانه‌ی رانندگی بر پایه‌ی مقررات راهنمایی و رانندگی نخواهد بود. علائم راهنمایی و رانندگی مورد بررسی در این کارویژه شامل تصاویر موجود در جدول ۳-۳ می‌باشد.

جدول ۳-۳ تابلوهای مورد بررسی در این کارویژه

عنوان تابلو	تصویر تابلو	عنوان تابلو	تصویر تابلو
ورود ممنوع		فقط عبور مستقیم مجاز است	
رعایت حق تقدم		گردش به چپ ممنوع	
دور زدن ممنوع		گردش به راست ممنوع	
توقف ممنوع		محدودیت سرعت	

در این بخش به معرفی یک روش برای تشخیص تابلوهای موجود جدول ۳-۳ پرداخت شده است.

۱-۲-۳- تبیین روش مورد نظر

با توجه به این که روش مورد نظر برای شناسایی تابلوها، باید علاوه بر پیدا کردن تابلوها[۲۰۳]، شناسایی تابلوی موردنظر را نیز انجام دهد، می‌توان دو رویکرد متفاوت در نظر گرفت، در رویکرد اول می‌توان تمام تابلوها را از یک نوع در نظر گرفت و مسئله‌ی «آیا در این تصویر تابلو وجود دارد؟ اگر وجود دارد کجای تصویر قرار دارد را حل کرد» و پس از تشخیص دادن تابلوها، به حل مسئله‌ی «نوع تابلوی پیدا شده چیست؟» را انجام داد، این روش نیازمند استفاده از دو مدل هوش مصنوعی، یکی برای پیدا کردن و دیگری برای شناسایی تابلوی پیدا شده می‌باشد،

[203] Detection

که به دلیل استفاده از دو مدل مختلف، بهینه نمی‌باشد. رویکرد دوم استفاده از یک مدل هم برای پیداکردن و هم تشخیص نوع تابلو می‌باشد، الگوریتم‌های جدیدتر، غالبا مبتنی بر رویکرد دوم هستند.

روش‌هایی که می‌تواند مورد استفاده قرار گیرد هم می‌تواند بر پایه‌ی مدل‌های کلاسیک و قدیمی‌تر باشد، و هم می‌تواند مبتنی بر روش‌های جدیدتر و عمیق[۲۰۴] باشند که در این کارویژه، روش مورد بررسی استفاده از یک الگوریتم مبتنی بر تشخیص[۲۰۵] و شناسایی[۲۰۶] می‌باشد. برای شناسایی تابلوهای راهنمایی و رانندگی، همانند بخش ۱-۳ از شبکه‌ی مبتنی بر یادگیری عمیق Yolo استفاده شده است. با توجه به یکسان بودن روش برای تشخیص تابلوها و تشخیص اشیا، تئوری کامل این قسمت در بخش ۱-۱-۳ توضیح داده شده‌است. در این بخش از مدل 5s استفاده شده است. مدل 5s سبک‌ترین مدل موجود برای الگوریتم Yolov5 می‌باشد که در اصل روی دیتاست COCO (با ۸۰ کلاس از اشیا مختلف آموزش دیده شده است. در این بخش، وزن‌ها[۲۰۷] این شبکه برای تشخیص تابلوهای مورد نظر بروزرسانی شده اند و شبکه مجددا از اول با استفاده از دیتاست مرتبط آموزش داده شده است.

چالش اصلی در آموزش مجدد Yolov5s، تهیه‌ی داده‌گان[۲۰۸] متناسب (با محوریت تابلوهای راهنمایی و رانندگی) می‌باشد. در این کارویژه، دیتاست‌های متعددی که از مجموعه تصاویر تابلوهای راهنمایی و رانندگی تشکیل شده‌اند جمع آوری و بررسی شده است و در نهایت از ترکیب بهترین دیتاست‌ها برای آموزش شبکه‌ی Yolov5s استفاده شده است.

۲-۲-۳- دیتاست‌های مورد استفاده

برای آموزش شبکه‌ی Yolov5s، به دیتاستی بزرگ شامل تابلوهای مورد نظر نیاز است که دیتاست مورد نظر باید شامل تصاویری از محیط‌های مختلف که تابلوهای مورد نظر در آن‌ها وجود دارند باشد. برای این کار می‌توان به صورت دستی به تهیه‌ی دیتاست پرداخت. برای این کار نیاز است که از محیط‌های مختلف عکس‌برداری شده و برچسب مورد نیاز شبکه‌ی Yolo برای هر تصویر ایجاد شود. این برچسب به صورت زیر می‌باشد.

```
1.  class x_normalized y_normalized h__normalized w_normalized
```

به این صورت که برای هر تصویر موجود در دیتاست، بایستی یک فایل txt. ایجاد شود که به تعداد تابلوهای موجود در آن تصویر، خط وجود دارد و در هر خط، اطلاعات مربوط به آن تابلو، اعم از نوش تابلو (کلاس)، نقطه‌ی وسط

[204] Deep
[205] Detection
[206] Recognition
[207] Weights
[208] Dataset

باندینگ باکس (x,y) به صورت نرمال نسبت به ابعاد تصویر و طول و عرض باندینگ باکس به صورت نرمال به ابعاد تصویر نیاز است که این کار را می‌توان با ابزارهایی مانند Labelme انجام داد. همان طور که مشخص است، انجام این کار به زمان، هزینه و نیروی کار زیادی نیاز دارد، به همین دلیل می‌توان از دیتاست‌های آماده از تصاویر تابلوهای مختلف استفاده کرد. دیتاست‌های متعددی برای تابلوهای راهنمایی و رانندگی وجود دارد که در زیر به برخی اشاره شده است. تمام دیتاست‌های توضیح داده شده بررسی شده و لیبل‌های آن‌ها در صورت مناسب بودن به لیبل صورت لیبل Yolo درآمده است. تمامی کدهای مربوطه برای دانلود، بررسی و تمیزسازی دیتاست در قالب یک جوپیتر نوت‌بوک[۲۰۹] به صورت ضمیمه موجود می‌باشد.

۱-۲-۲-۳- Swedish Dataset

این دیتاست[۲۵۶] شامل ۲۰۰۰۰ تصویر از ۳۵۰ کیلومتر از اتوبان‌های کشور سوییس می‌باشد تشکیل می‌شود، تصاویر توسط یک دوربین ۱.۳ مگاپیکسلی که برروی یک خودرو تعبیه شده‌است، تهیه شده است. این دیتاست شامل کلاس‌های منع‌کننده، سرعت و راهنمایی کننده می‌باشد. همان طور که مشخص است، این دیتاست به دلیل عدم وجود کلاس‌های مجزا قابل استفاده نمی‌باشد، منظور از عدم وجود کلاس‌های مجزا، عدم کلاس‌بندی کردن لیبل عنوان دار برای تمام تابلوهاست. دو نمونه از تصاویر این دیتاست در شکل‌های ۱۲-۳ و ۱۳-۳ نمایش داده شده است.

[209] Jupyter Notebook

شکل ۱۲-۳ نمونه‌ای از دیتاست Swedish [۲۵۶]

شکل ۱۳-۳ نمونه‌ای دوم از دیتاست Swedish [۲۵۶]

همان طور که از شکل ها ملاحظه می‌شود، ابعاد تابلوها بسیار ریز می‌باشد همچنین اطلاعات تکمیلی این دیتاست در جدول ۴-۳ ارائه شده است.

جدول ۴-۳ دیتاست Swedish

Dataset Name	Swedish Dataset
تعداد تصاویر	۲۰,۰۰۰
تصاویر لیبل زده شده	۲۰٪
تعدا تابلوها	۳۴۸۸
حجم دیتاست	۱.۵۳ گیگابایت
ابعاد تصویر	۱۲۸۰*۹۶۰
تعداد کلاس‌ها	۴
کلاس های جدا برای هر تابلو	خیر
اندازه‌ی مناسب تابلوها	خیر
قابل استفاده	خیر

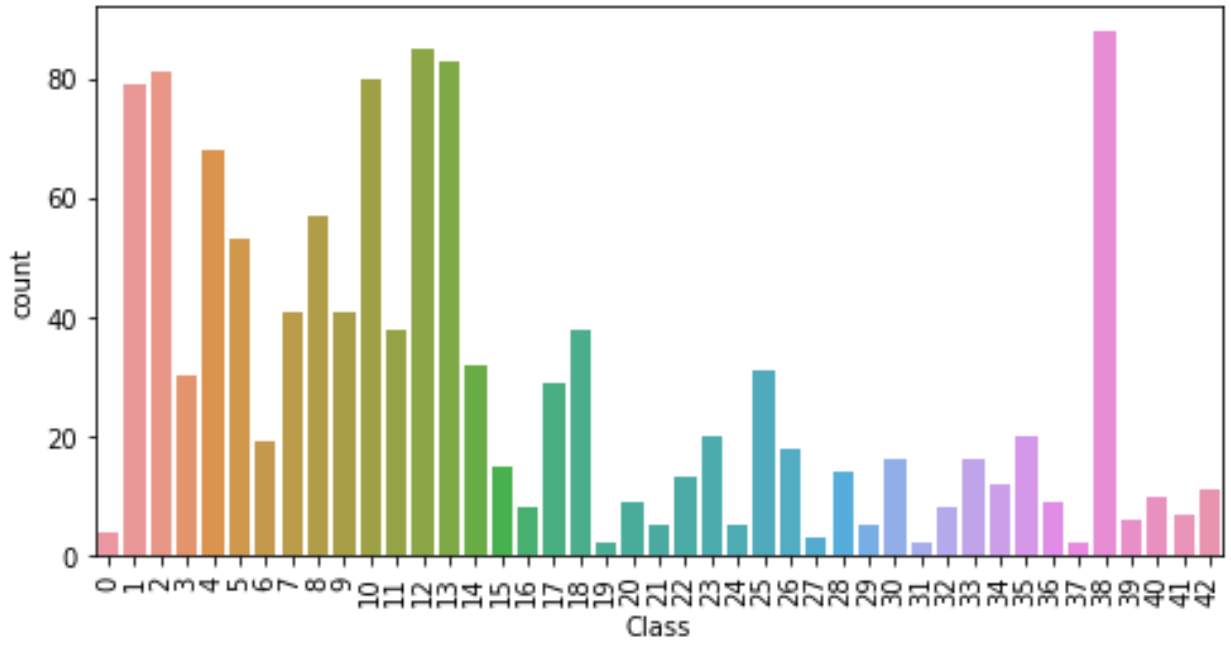
با مشاهده‌ی دیتاست می‌توان متوجه شد که برخی تابلوها بسیار ریز می‌باشند (کوچکتر از ۲۰ پیکسل در ۲۰ پیکسل) پس این دیتاست نمی‌تواند دیتاست مناسبی باشد.

German Traffic Sign Detection Benchmark GTSDB -۳-۲-۲-۲

این دیتاست[۲۵۷] که در مسابقات IJCNN سال ۲۰۱۳ ارائه شد، شامل مجوعه‌ای از ۹۰۰ تصویر که در هر کدام تابلوهای مختلف راهنمایی و رانندگی قرار دارد می‌باشد. کلاس‌های موجود در این دیتاست شامل ۴۳ کلاس از انواع تابلوهای راهنمایی و رانندگی و در ۳ موضوع[۲۱۰] ممنوع‌کننده[۲۱۱]، اجبارکننده[۲۱۲] و نشان‌دهنده‌ی خطر[۲۱۳] می‌باشد. در نگاه اول این دیتاست دیتاست مناسبی می‌باشد لکن مشکل اساسی که در این دیتاست وجود دارد،

[210] Category
[211] prohibitory
[212] mandatory
[213] Danger

تعداد کم تابلوهای متعلق به هر یک از ۴۳ کلاس است. نمودار شمارش هرکدام از کلاس‌ها در شکل ۱۴-۳ قابل ملاحظه است.



شکل ۱۴-۳ نمودار شمارش هر کدام از کلاس‌ها در دیتاست GDSDB

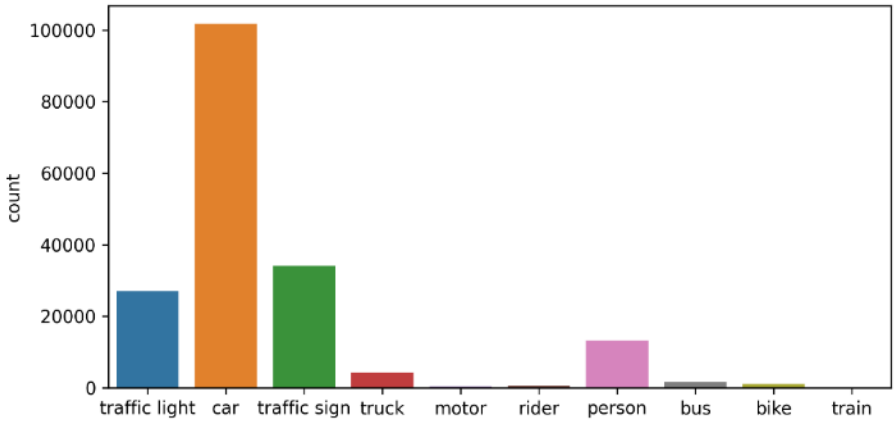
از مشکلات این دیتاست می‌توان به تعداد بسیار کم داده‌ها و عدم تناسب بین کلاس‌های مختلف است.

جدول ۵-۳ دیتاست GTSDB

Dataset Name	GTSDB
تعداد تصاویر	۹۰۰
تصاویر لیبل زده شده	۱۰۰٪
تعدا تابلوها	۱۲۱۳
حجم دیتاست	۱.۶ گیگابایت
ابعاد تصویر	۱۲۸۰*۹۶۰
تعداد کلاس‌ها	۴۳
کلاس های جدا برای هر تابلو	بله
اندازه‌ی مناسب تابلوها	بله
قابل استفاده	خیر

MakeML Cars and Traffic Signs Dataset -۳-۲-۲-۳

دیتاست[۲۵۸] شامل ۱۰,۰۰۰ تصویر از ۱۰ کلاس متفاوت می‌باشد که دارای لیبل‌های شناسایی اشیا که در خیابان موجود است می‌باشد، این دیتاست شامل کلاس‌های خودرو، وانت، انسان، تابلوهای راهنمایی و رانندگی، چراغ راهنمایی، دوچرخه، کامیون، موتورسیکلت و قطار می‌باشد. متاسفانه موضوع تابلوهای راهنمایی و رانندگی خود به صورت یک کلاس مجزا تعریف شده است و برای کلاس‌های مختلف تابلو، کلاسی در نظر گرفته نشده است.



شکل ۱۵-۳ تعداد اشیا[۲۱۴] هر کلاس در دیتاست MakeML

با توجه به توزیع داده‌ها و عدم وجود کلاس‌های مختلف برای تابلوهای راهنمایی و رانندگی، این دیتاست برای آموزش شبکه‌ی Yolo برای تشخیص تابلو مناسب نمی‌باشد، لکن از آن می‌توان برای آموزش شبکه برای تشخیص خودروها و ... استفاده کرد. در ادامه چند نمونه از تصاویر مربوط به این دیتاست مشاهده می‌شود. شایان ذکر است که لیبل‌های این دیتاست به فرمت مورد نیاز Yolo تبدیل شده و موجود می‌باشد. کد بررسی و تبدیل این دیتاست در پیوست آورده شده است.

[214] Object

شکل ۱۶-۳ نمونه‌ای از تصاویر دیتاست MakeML Cars and Traffic Signs Dataset [۲۵۸]

شکل ۱۷-۳ نمونه‌ای دیگر از تصاویر دیتاست MakeML Cars and Traffic Signs Dataset [۲۵۸]

جدول ۳-۶ دیتاست MakeML Cars and Traffic Signs Dataset

Dataset Name	MakeML Cars and Traffic Signs Dataset
تعداد تصاویر	۱۰,۰۰۰
تصاویر لیبل زده شده	۱۰,۰۰۰
تعداد تابلوها	۳۲۰۰
حجم دیتاست	۲.۳۷ گیگابایت
ابعاد تصویر	۶۰۰*۳۳۷
تعداد کلاس‌ها	۱۰
کلاس های جدا برای هر تابلو	خیر
اندازه‌ی مناسب تابلوها	بله
قابل استفاده	خیر

۳-۲-۲-۴ MakeML Road Signs Dataset

در دیتاست [۲۵۹] که توسط شرکت MakeML تهیه شده است تعداد ۸۷۷ تصویر را در ۴ کلاس چراغ راهنمایی، تابلوی‌محدودیت سرعت، تابلوی عبور پیاده، و توقف را در خودر جای داده است. فرمت این دیتاست نیز مشابه دیتاست MakeML Cars and Traffic Signs Dataset می‌باشد و فرمت مورد نیاز Yolo نیز برای آن به دست آمده است. تعدادی از تصاویر این دیتاست در ادامه مشاهده می‌شود.

شکل ۳-۱۸ نمونه‌ای از دیتاست MakeML Road Signs Dataset [۲۵۹]

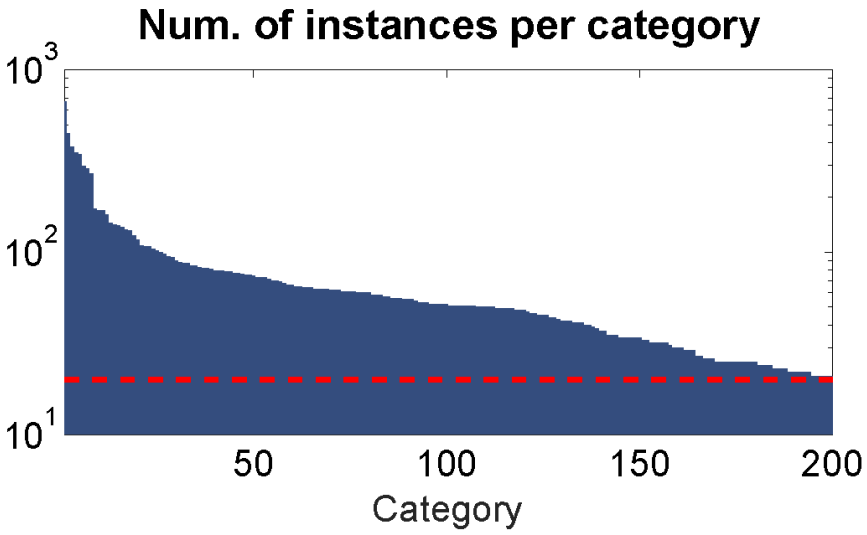
جدول ۷-۳ دیتاست MakeML Road Signs Dataset

Dataset Name	MakeML Road Signs Dataset
تعداد تصاویر	**۹۷۷**
تصاویر لیبل زده شده	**۹۷۷**
تعداد تابلوها	**۹۲۰**
حجم دیتاست	**۲۱۸ مگابایت**
ابعاد تصویر	**متغیر**
تعداد کلاس‌ها	**۴**
کلاس های جدا برای هر تابلو	**خیر**
اندازه‌ی مناسب تابلوها	**بله**
قابل استفاده	**خیر**

همان‌طور که از جدول ۷-۳ ملاحظه می‌شود، این دیتاست به دلیل عدم کلاس‌بندی تابلوها، برای آموزش شبکه‌ی تشخیص تابلوها مناسب نمی‌باشد، لکن برای سایر موارد می‌تواند مناسب باشد، پس این دیتاست نیز به فرمت مناسب Yolo تبدیل شده است. کد بررسی و تبدیل این دیتاست در پیوست آورده شده است.

Vicos/DFG -۳-۲-۲-۵

این دیتاست [۲۶۰] شامل ۷۰۰۰ تصویر با رزولوشن بالا می‌باشد که شامل ۲۰۰ کلاس از تابلوهای مختلف راهنمایی و رانندگی می‌باشد. این دیتاست با رانندگی در خیابان‌های اسلوونی به دست‌آمده است و همچنین توسط یک شرکت اسلوونیایی به‌نام DFG لیبل زده شده است به همین دلیل به اسم دیتاست DFG نیز شناخته می‌شود. همچنین علاوه بر دیتای واقعی، شامل مجموعه‌ای از دیتای مصنوعی می‌باشد که تقریبا برابر با تعداد داده‌های واقعی می‌باشد، این مهم حجم دیتاست را از ۷۰۰۰ تصویر به ۱۶۲۴۲ تصویر افزایش داده است. لیبل زنی این دیتاست نیز همانند دیتاست COCO می‌باشد و باید به فرمت مورد نیاز یولو تبدیل شود. توزیع کلاس‌های مختلف نیز در تصویر ۱۹-۳ آورده شده است، مشاهده می‌شود که در هر کلاس تابلو، حداقل ۲۰۰ تابلو وجود دارد. کلاس‌های پوشش داده شده در این دیتاست در تصویر ۲۰-۳ نمایش داده شده است.



شکل ۱۹-۳ توزیع تعداد تابلوهای مربوط به هر کلاس در دیتاست DFG [۲۶۰]

شکل ۲۰-۳ دیتاست Vicos یا DFG [۲۶۰]

اطلاعات مربوط به این دیتاست در جدول ۸-۳ آورده شده‌است.

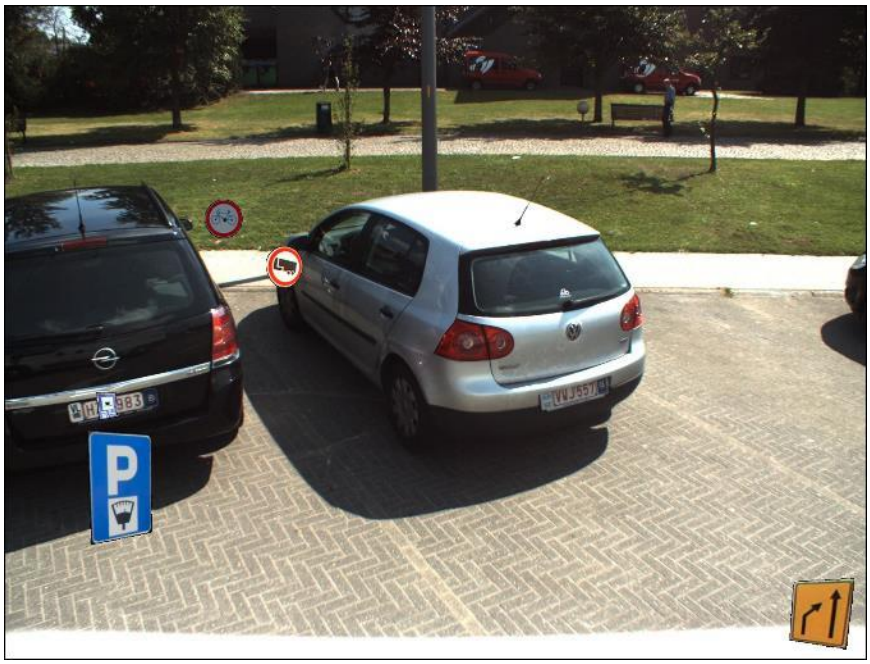
جدول ۸-۳ دیتاست Vicos/DFG Dataset

Dataset Name	DFG
تعداد تصاویر	۷۰۰۰
تصاویر لیبل زده شده	۷۰۰۰
تعداد تابلوها	۹۲۰
حجم دیتاست	۶.۹ گیگابایت
ابعاد تصویر	۱۹۲۰*۱۰۸۰
تعداد کلاس‌ها	۲۰
کلاس های جدا برای هر تابلو	بله
اندازه‌ی مناسب تابلوها	بله
قابل استفاده	بله

در تصاویر ۲۱-۳ الی ۲۳-۳، تعدادی از تصاویر موجود در این دیتاست مشاهده می‌شود.

تصویر ۲۱-۳ نمونه‌ی شماره ۱ از دیتاست DFG [۲۶۰]

تصویر ۲۲-۳ نمونه‌ی شماره ۲ از دیتاست DFG [۲۶۰]



تصویر ۲۳-۳ نمونه‌ی ای از دیتاست مصنوعی دیتاست DFG [۲۶۰]

متاسافانه شرکت DFG شماره‌ی کلاس مروبط به هر کدام از تابلوها را در فایل لیبل‌ها قرار نداده و درنتیجه با استفاده بخش *Finding Desired Images >>>Vicos-DFG* که در نوت‌بوک *Dataset.ipynb* قرار دارد، اقدام به دانلود یک نمونه از کدام کلاس‌ها می‌شود و فایل مربوطه به اسم golchined.zip ذخیره می‌شود. با دانلود این فایل، به صورت دستی به پیدا کردن شماره‌ی کلاس هر کدام از تابلوهای مورد نیاز پرداخته می‌شود. که کلاس‌های مورد نظر به صورت زیر می‌باشند:

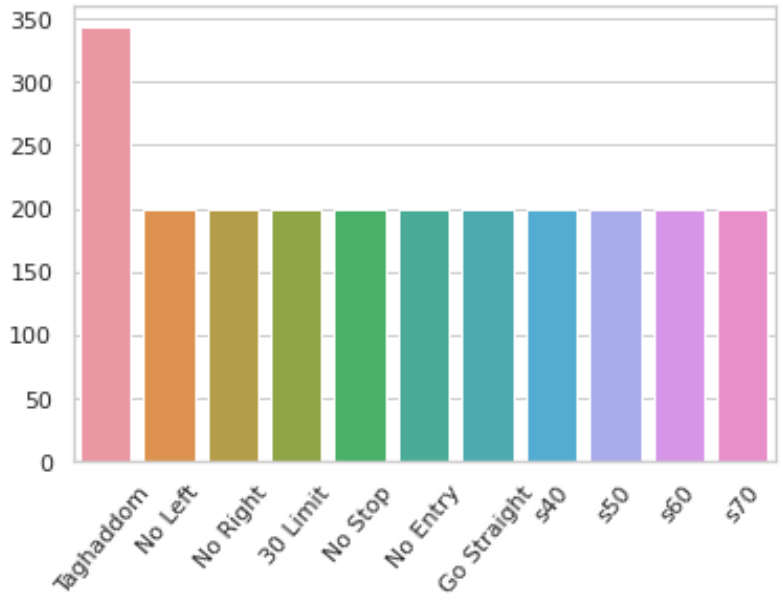
```
array([344, 200, 200, 200, 200, 200, 200, 199, 200, 200, 200])
```

که هر عدد به ترتیب بیانگر کلاس‌های زیر به ترتیب می‌باشد:

```
['Taghaddom', 'No Left', 'No Right', '30 Limit','No Stop' ,'No Entry', 'Go Straight','s
40','s50','s60','s70']
```

شایان ذکر است، کلاس‌هایی که با حرف انگلیسی s شروع می‌شوند، به معنی تابلوی محدودیت سرعت هستند.

سپس با استفاده از بخش *Vicos-DFG>>> Pictures with the desired signs in them*، تصاویر مورد نظر استخراج می‌شوند. در نهایت با استفاده از بخش *Cleaning for Yolo* در همان نوت‌بوک، دیتاست نهایی با فرمت مورد نیاز Yolo ساخته می‌شود. شایان ذکر است که تعداد تابلوهای مربوط به هر کلاس در شکل ۲۴-۳ نمایش داده شده است.



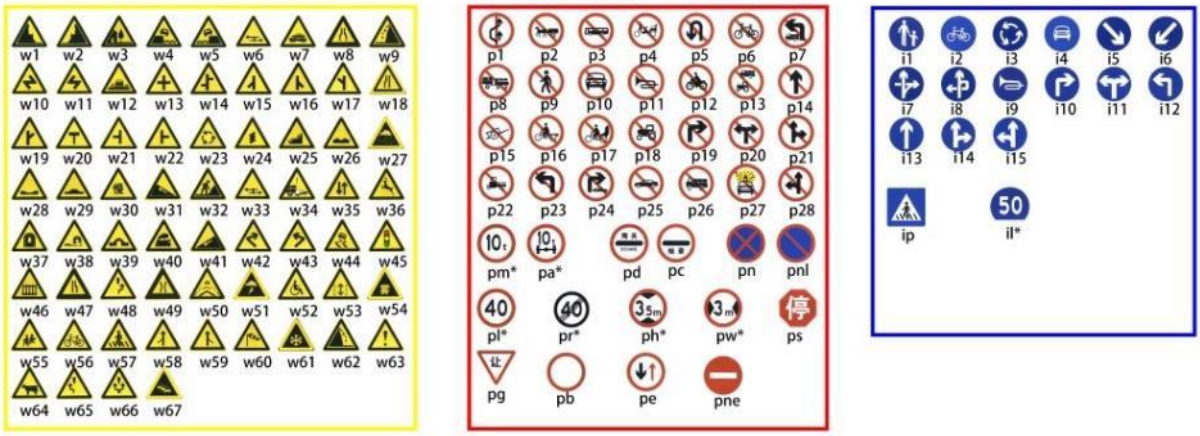
شکل ۲۴-۳ توزیع تابلوها در دیتاست DFG پس از استخراج تابلوهای مورد نظر

همان‌طور که مشاهده می‌شود، هم دیتاست متناسب[۲۱۵] می‌باشد و هم تعداد هر کدام از تابلوها مناسب می‌باشد پس این دیتاست می‌تواند از کاندیداهای نهایی برای آموزش شبکه باشد، تنها مشکلی که وجود دارد، عدم وجود تابلوی دوربرگردان-ممنوع در این دیتاست می‌باشد. پس می‌بایست از ترکیب این دیتاست و دیتاست‌های دیگر استفاده کرد.

[215] Balanced

Tsinghua Traffic Sign Detection and Classification in the Wild -۳-۲-۲-۶

این دیتاست[۲۶۱] که آخرین دیتاست مرتبط با تابلوهای راهنمایی و رانندگی است که در این کارویژه بررسی می‌شود، متعلق به دانشگاه شین‌خوا می‌باشد. در این دیتاست حدود ۱۰۰,۰۰۰ تصویر و بیش از ۳۰,۰۰۰ تابلوی راهنمایی و رانندگی وجود دارد. این دیتاست شامل حدود ۲۰۰ کلاس از تابلوهای مختلف راهنمایی و رانندگی می‌باشد که شمال تمامی تابلوهای موجود در جدول ۲۴-۳ می‌باشد. تمامی کلاس‌های تابلوی موجود در این دیتاست در تصویر ۲۵-۳ قابل مشاهده می‌باشد.



شکل ۲۵-۳ کلاس‌های موجود در دیتاست Tsinghua [۲۶۱]

از مشکلات این دیتاست، نیاز به دانلود حدود ۱۸ گیگابایت دیتاست و آپلود کردن در چندین گوگل درایو (به دلیل محدودیت فضای گوگل درایو) بوده است، دلیل دانلود دیتاست به صورت محلی[۲۱۶]، سرعت پایین سرورهای دانشگاه شین-خوا برای سرورهای متعلق به گوگل بوده است، برای تمام دیتاست‌های دیگر، فایل‌ها مستقیما از سرور سازنده دیتاست دانلود و به گوگل‌کولب[۲۱۷] انتقال داده شده‌اند. پس از پیدا کردن لیبل‌های کلاس‌های مورد نیاز با توجه به شکل ۲۵-۳ و جدا کردن تصاویر شامل تابلوهای موردنظر، داده‌ها به فرمت Yolo تبدیل شده و در گوگل درایو قرار داده شدند.

[216] Local
[217] Google Colab

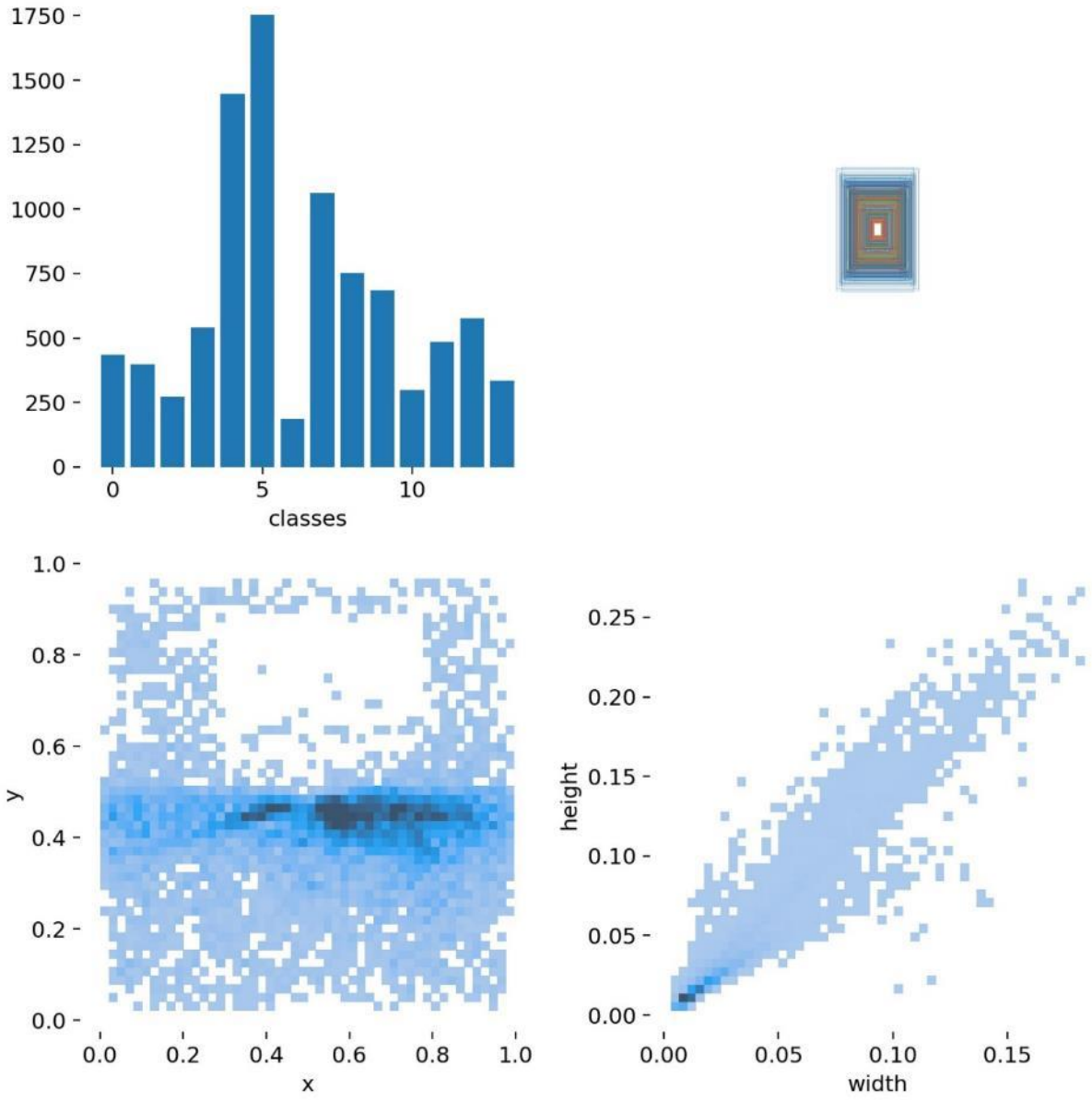
جدول ۹-۳ دیتاست Tsinghua

Dataset Name	Tsinghua
تعداد تصاویر	**۱۰۰,۰۰۰**
تصاویر لیبل زده شده	**۱۰۰,۰۰۰**
تعداد تابلوها	**۳۰,۰۰۰**
حجم دیتاست	**۱۸ گیگابایت**
ابعاد تصویر	**متغیر-رزولوشن بسیار بالا**
تعداد کلاس‌ها	**۲۰۰**
کلاس های جدا برای هر تابلو	**بله**
اندازه‌ی مناسب تابلوها	**بله**
قابل استفاده	**بله**

نمونه‌ای از تصاویر موجود در این دیتاست در ادامه نمایش داده‌شده است.

شکل ۲۶-۳ نمونه‌ای از دیتاست شین‌خوا [۲۶۱]

این دیتاست به همراه دیتاست موجود در بخش ۵-۲-۲-۳ تنها کاندیداهای بسیار مناسب برای آموزش شبکه‌ی Yolo می‌باشد، دلیل این انتخاب، متنوع بودن کلاس‌ها و تصاویر، بالا بودن تعداد تصاویر، کیفیت بالای تصاویر و تنوع تصاویر موجود در آن‌ها می‌باشد. با ترکیب این دو دیتاست، دیتاست مناسب ساخته میشود. در نهایت حجم دیتاست ساخته شده که Final_Datset نامیده می‌شود حدود ۱ گیگابایت می‌باشد.



شکل ۲۷-۳ توزیع تعداد و ابعاد داده ها در دیتاست نهایی

لازم به ذکر است است که این دیتاست شامل حدود ۷۰۰۰ تصویر می‌باشد، دلیل حجم نسبتا پایین این تصاویر، تغییر سایز دادن تمامی تصاویر موجود در دیتاست‌های مختلف به ابعاد مورد نیاز Yolo که دارای پهنای ۶۴۰ پیکسل با حفظ نسبت ابعاد می‌باشد، است.

۳-۲-۳- چگونگی آموزش و صحه‌گذاری

نحوه‌ی آموزش شبکه روی دیتاست‌های به دست‌آمده دقیقا مشابه بخش ۲-۱-۳ می‌باشد با این تفاوت که فایل Dataset.yaml باید مختص دیتاست finaldataset.zip ساخته شود که به صورت زیر می‌باشد:

```
# train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3) list: [path1/images/, path2/images/]
train: ../ finaldataset/images/train/
val: ../ finaldataset/images/val/

# number of classes
nc: 13

# class names
Names['Taghaddom', 'No Left', 'No Right', '30 Limit','No Stop' ,'No Entry', 'Go Straight', 's40','s50','s60','s70']
```

همچنین دیتاست نیز از قبل ایجاد شده است.

از تفاوت‌های دیگر، به نحوه‌ی داده افزایی شبکه باید اشاره کرد، در داده‌افزایی مطرح شده در بخش قبل، نصف داده‌ها حول محور عمودی آیینه[۲۱۸] می‌شوند. که این مهم در فایل yolov5/data/hyp.scratch.yaml در خط ۳۲ مشخص شده است

```
fliplr: 0.5  # image flip left-right (probability)
```

با توجه به این که، این نوع داده‌افزایی باعث گمراه شدن شبکه در تشخیص تابلوهای راهنمایی و رانندگی می‌شود، به این صورت که تابلوی گردش به راست ممنوع با آیینه شدن حول محول عمودی به تابلوی گردش به چپ ممنوع تبدیل می‌شود، این نوع داده‌افزایی باید غیرفعال شود. و این امر با تغییر خط ۳۲م فایل مذکور به صورت زیر حاصل می‌شود:

```
fliplr: 0.0  # image flip left-right (probability)
```

پس از طی مراحل ذکر شده و آموزش شبکه به مدت ۳۰۰ دوره[۲۱۹] به صورت زیر

```
!python train.py --img 640 --batch 32 --workers 8 --epochs 150\
 --data /content/yolov5/data/coco128.yaml \
 --cfg /content/yolov5/models/yolov5s.yaml \
```

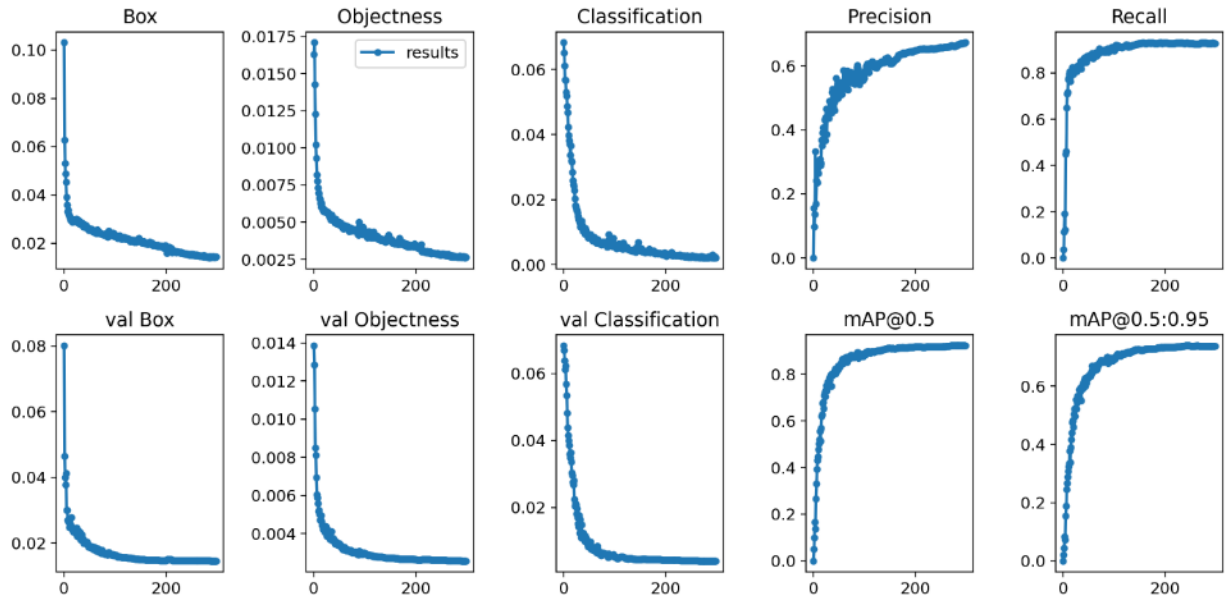
[218] Flip
[219] Epoch

```
4.   --project '/content/Yolo_RN_FDS/train'\
5.   --weights ''
6.   --hyp /content/yolov5/data/hyp.scratch.yaml
```

فایل خروجی best.pt که وزن‌های شبکه در بهترین دوره از ۳۰۰ دوره‌ی مذکور بودند به عنوان مدل نهایی شبکه استفاده می‌شود.

۴-۲-۳- نتیجه خروجی و صحه‌گذاری

نتایج اعتبار سنجی شبکه را روی دیتاست صحه‌گذاری[۲۲۰] که شامل ۱۰ درصد final_dataset بود را می‌توان در شکل ۲۸-۳ مشاهده کرد.



شکل ۲۸-۳ نتیجه‌ی معیارهای مختلف شبکه‌ی آموزش داده شده Yolov5s بر روی دیتاست صحه‌گذاری

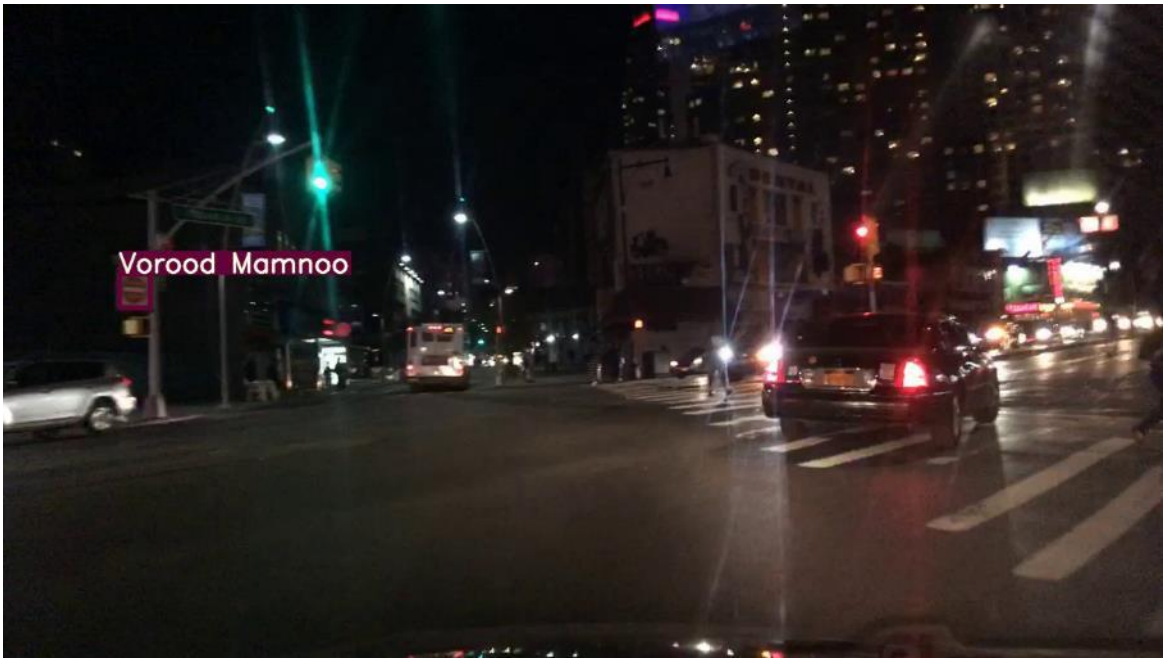
جدول ۱۰-۳ نتایج صحه‌گذاری شبکه Yolov5s

معیار	مقدار
Precision	**0.6727**
Recall	**0.9287**
mAP@0.5	**0.923**
mAP@0.5:0.95	**0.738**

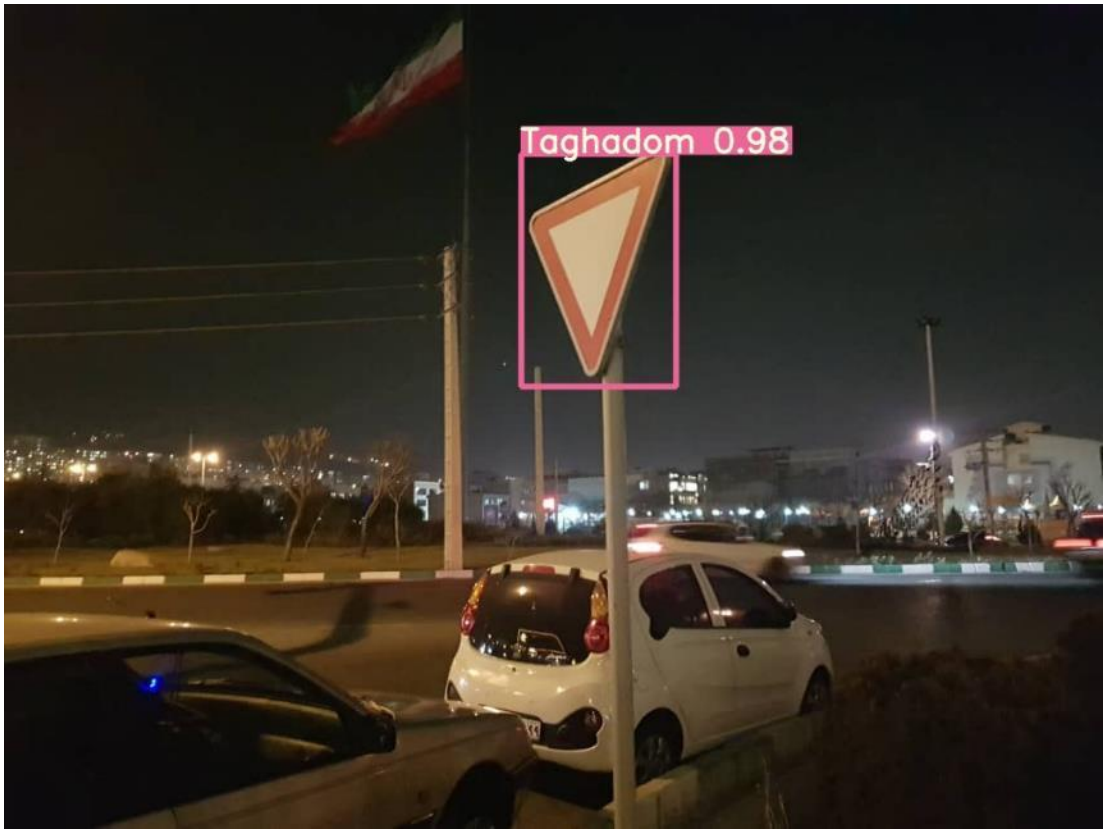
[220] Validation

حال به مشاهده‌ی خروجی شبکه‌ی آموزش داده شده روی داده‌های تست پرداخته می‌شود. تصاویر ۲۹-۳ و ۳۰-۳ به ترتیب از ویدئوهای ۱ و ۲ داده شده خروجی گرفته شده‌اند و تصویر ۳۱-۳ نیز تصویری در یکی از ار خیابان های تهران می‌باشد.

شکل ۲۹-۳ تابلو دور برگردان ممنوع شناسایی شده در ویدئو شماره ۱



شکل ۳۰-۳ تابلو ورود ممنوع شناسایی شده در ویدئو شماره ۲



شکل ۳۱-۳ تابلوی رعایت حق تقدم شناسایی‌شده در یکی از خیابان‌های تهران

شکل ۳۲-۳ تابلوهای متفاوت محدودیت سرعت که با برچسب‌های مختلف مشخص شده‌اند.

۵-۲-۳- تشریح نحوه پیاده سازی الگوریتم و ساختار کد

نحوه‌ی پیاده سازی و استفاده از شبکه در کد اصلی به صورتی می‌باشد که در ماژول main.py ابتدا ماژول YOLO_Sign باید وارد کد اصلی شود که این اتفاق به صورت زیر می‌باشد.

```
1.  from elements.yolo import YOLO_Sign
```

سپس باید مدل تشخیص تابلوها بارگذاری شود که با استفاده از کد زیر می‌باشد:

```
1.  sign_detector = YOLO_Sign(opt.weights_sign)
```

همچنین کلاس تابلوهای مورد نظر از قبل در کد مشخص شده‌اند:

```
1.  signs = ['Taghadom', 'Chap Mamnoo', 'Rast Mamnoo', 'SL30', 'Tavaghof Mamnoo',
2.           'Vorood Mamnoo', 'Mostaghom', 'SL40', 'SL50', 'SL60', 'SL70', 'SL80', 'SL100',
3.           'No U-Turn']
```

پس از آن، هر فریم وارد sign_detector شده و خروجی آن به دست می‌آید:

```
1.  signOutput = sign_detector.detect_sign(frame)
```

خروجی signOutput به صورت مقابل خواهد بود که یکی دیکشنری از نام کلاس هر شی، مختصات باندینگ باکس آن، دقت شی تشخیص داده شده و شماره کلاس است.

```
6.  item = {'label': label,
7.          'bbox' : [(xmin,ymin),(xmax,ymax)],
8.          'score': score,
9.          'cls'  : int(p(5))
10.         }
```

در بخش زیر نیز ورودی و خروجی ماژول yolo.py به اختصار آورده شده است:

```
14. """
15.     Input:
16.             BGR image
17.
18.     Output:
19.     yolo return list of dict in format:
20.         {   label   :  str
21.             bbox    :  [(xmin,ymin),(xmax,ymax)]
22.             score   :  float
23.           Cls     :  int
24.         }
25.  """
```

در نهایت با استفاده از کد زیر، باندینگ باکس هر تابلوی تشخیص‌داده شده به صورت زیر روی فریم اصلی چاپ می‌شوند:

```
for sign in signOutput:
            xyxy = [sign['bbox'][0][0], sign['bbox'][0][1], sign['bbox'][1][0],
            sign['bbox'][1][1]]
            plot_one_box(xyxy, frame, label=sign["label"],
            color=colors_signs[sign['cls']], line_thickness=3)
```

۳-۳- تشخیص خطوط جاده

شناسایی خطوط جاده یکی از بخش های مهم برای یک خودرو خودران است که با استفاده از آن، خودرو مسیرحرکت خود را مشخص می کند. خطوط جاده به دو دسته خطوط افقی و عمودی تقسیم می‌شوند که با توجه به نوع آن‌ها، نوع الگوریتم مورد استفاده برای تشخیص آن‌ها تعیین می‌شود.

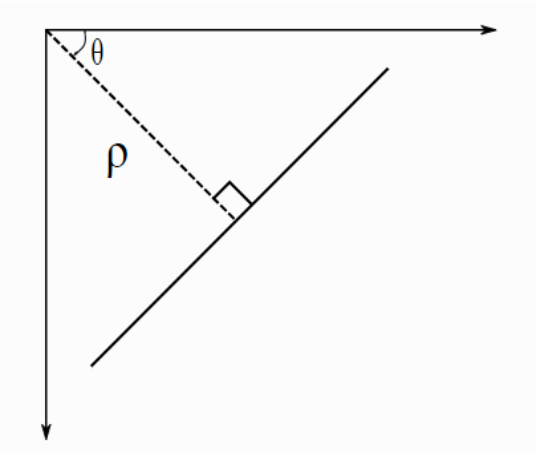
۱-۳-۳- تشخیص خطوط افقی

۱-۱-۳-۳- تبیین روش مورد نظر به همراه ریاضیات کامل

برای تشخیص خطوط افقی از روش های کلاسیک پردازش تصویر استفاده شده است. در مرحله ی اول برای تشخیص خطوط کلی موجود در تصویر از الگوریتم ترنسفورم هاف[۲۲۱] استفاده شده است[۲۶۲].

الگوریتم هاف، یک تکنیک محبوب برای تشخیص خطوط حاشیه ی هر نوع شکل هندسی است، حتی اگر شکل مورد نظر، به صورت خمیده یا دارای خطوط گسسته و شکسته باشد. در این قسمت، این مورد بررسی می گردد که از این الگوریتم چگونه می توان برای تشخیص خطوط افقی موجود در تصویر استفاده نمود.

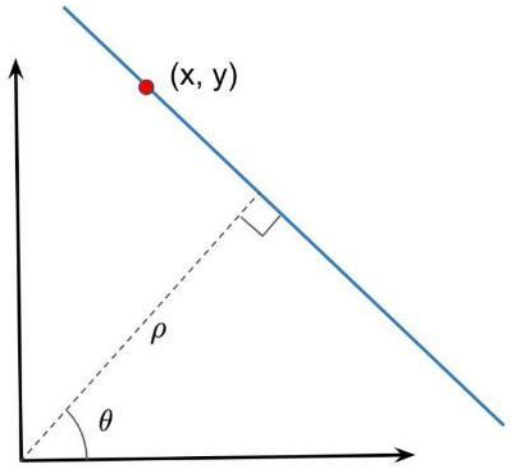
به صورت کلی یک خط را می توان به فرم معادلاتی $y = mx + c$ نمایش داد، اما فرمت متعارف دیگری نیز برای نمایش معادله ی یک خط وجود دارد که به صورت $\rho = xcos(\theta) + ysin(\theta)$ نشان داده می شود و به نمایش قطبی[۲۲۲] معروف است. در این معادله، ρ بیانگر فاصله ی عمودی خط از مبدا مختصات و θ زاویه ی خط ρ از محور افقی مختصات می باشد.



شکل ۳۳-۳ نمایش یک خط در فضای دو بعدی با معادله ی $\rho = xcos(\theta) + ysin(\theta)$

[221] Hough Line Transform
[222] Polar coordinate system

بنابراین در نمایش قطبی، اگر خط از زیر مبدا عبور کند ، ρ دارای علامت مثبت و θ کمتر از ۱۸۰ خواهد بود و اگر بالاتر از مبدا باشد، به جای این که زاویه را مقداری بیشتر از ۱۸۰ درجه در نظر بگیریم، کماکان θ کمتر از ۱۸۰در نظر میگریم، اما برای ρ علامت منفی در نظر می گیریم که این دو مورد از لحاظ ریاضیاتی برابر می باشند. همچنین خطوط عمودی دارای θ ۰ درجه و خطوط افقیθ ۹۰ درجه خواهند داشت.



شکل ۳۴-۳ نمایش یک خط بالاتر از مبدا با نمایش قطبی

بنابراین یک خط را در نمایش قطبی می توان با داشتن دو پارامتر ρ و θ از آن، ترسیم نمود. اما حال به تشریح چگونگی استفاده از این مفهوم در الگوریتم هاف برای تشخیص خطوط افقی موجود در تصویر پرداخته می گردد.

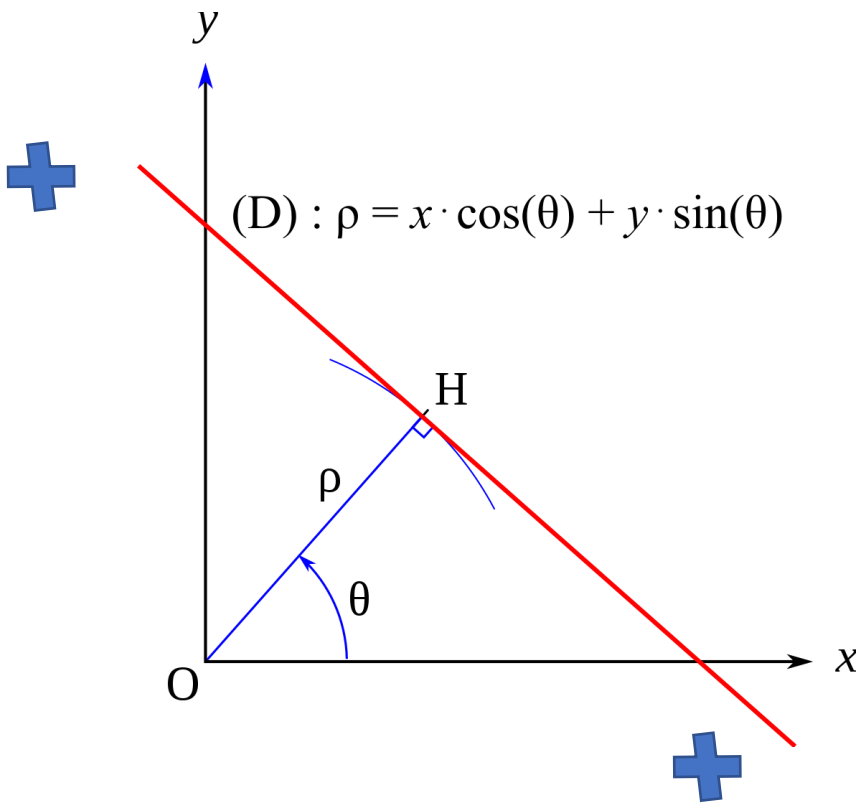
یک تصویر به ابعاد ۱۰۰ در ۱۰۰ در نظر بگیرید که یک خط افقی دقیقا در وسط آن موجود است. الگوریتم هاف، اولین نقطه از این خط را در نظر می گیرد که دارای مختصات (x, y) می باشد. حال در معادله ی به فرم قطبی این خط، برای θ، بسته به دقت انتخاب شده مقادیری را قرار می دهد، به عنوان مثال اگر دقت را یک درجه انتخاب کنیم، برای θ مقادیر ۰ تا ۱۸۰ را به ترتیب قرار می دهد و به ازای آن ها ρ را نسبت به مبدا مختصات به دست می آورد و جفت مقادیر θ و ρ را به ازای زوایای مختلف در حافظه ذخیره می کند.

حال، الگوریتم هاف نقطه ی دوم از خط را برداشته و همان عملیات بالا را روی آن تکرار می کند و به همین ترتیب این لیست از θ و ρ ها را برای تمام نقاط موجود در خط به دست می آورد. بنابراین در پایان عملیات، بررسی می کند که برای هر نقطه از خط، چه جفت مقادیری از θ و ρ به دست آمده و جفتی که در اکثریت نقاط خط موجود باشد، در نهایت به عنوان مشخصات خطی که از آن نقاط عبور کرده انتخاب می گردد. بدین ترتیب

معادله ی خطی که در صفحه وجود داشت، در نمایش قطبی به دست می آید و می توان آن را در صفحه ی دیگری ترسیم نمود[۲۶۳].

حال در صورتی که تصویر ما دارای تعداد بیشتری خط باشد، همگی آن ها توسط الگوریتم هاف شناسایی می گردد و مختصات آن ها به عنوان خروجی این الگوریتم قابل دستیابی می باشد.

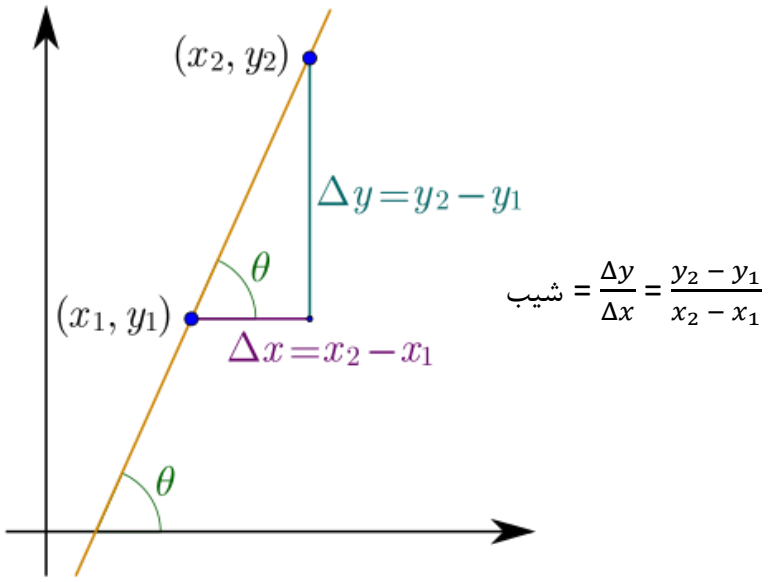
اما نوع خروجی داده شده توسط الگوریتم هاف، به صورت ارائه ی مختصات (x, y) دو نقطه ی ابتدایی و انتهایی خط می باشد.



شکل ۳۵-۳ دریافت نقاط ابتدایی و انتهایی خط تشخیص داده شده توسط الگوریتم هاف

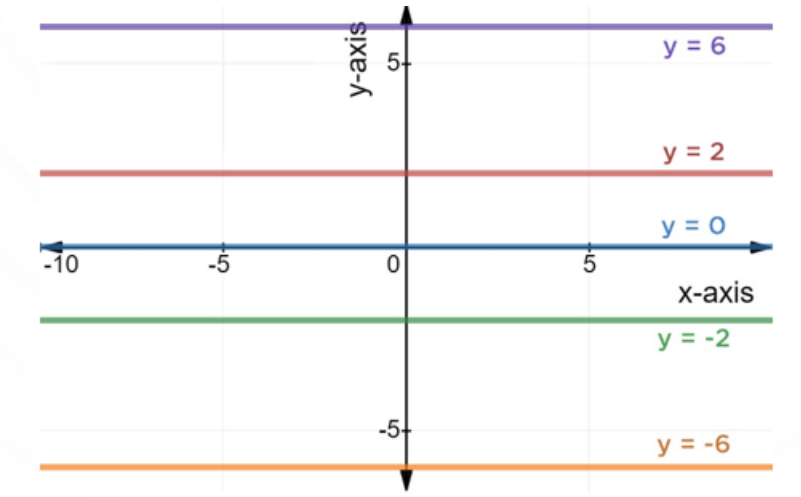
اما حالا که مختصات نقاط اول و آخر تمام خطوط موجود در تصویر به دست آمده، ابتدا برای به دست آوردن خطوط افقی از بین این خطوط، شیب همه خطوط را محاسبه گشته، و خطوطی که شیب آن ها از یک مقدار آستانه کمتر است جدا می شوند. علت انتخاب آستانه به این علت است که بسته به زاویه ی دوربین بسته شده روی خودرو، ممکن است شیب خطوط افقی دقیقا مقدار صفر نباشد و مقداری کمتر یا بیشتر باشد، اما با انتخاب آستانه، از این موضوع اطمینان حاصل می شود که تمام خطوط افقی موجود در تصویر، تشخیص داده شده باشند.

برای محاسبه ی شیب خطوط و جداسازی خطوط افقی، به این صورت عمل می گردد که از حاصل تقسیم تفاضل y خطوط نسبت به تفاصل x خطوط، شیب خط محاسبه می شود و در صورت کمتر بودن شیب از مقدار آستانه، آن خط به عنوان خط افقی ثبت می گردد.



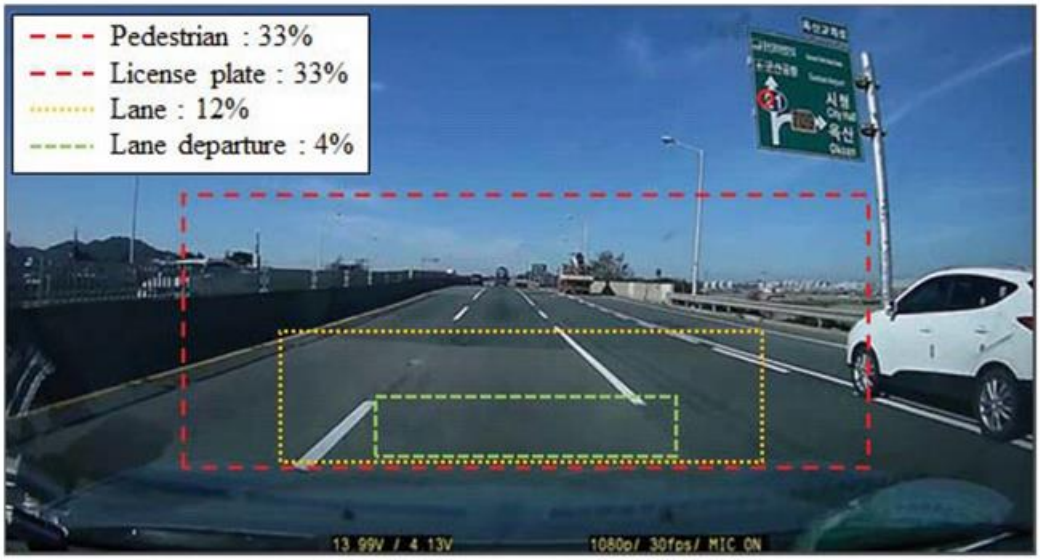
شکل ۳۶-۳ محاسبه ی شیب خطوط با نقاط دریافتی از الگوریتم هاف

در نهایت و پس از محاسبه ی شیب ها و جداسازی خطوط افقی، از مجموع خطوط افقی در یک ناحیه میانگین گیری انجام می شود و خط نهایی، با شیب و مختصات میانگین کلیه خطوط موجود در یک ناحیه، ترسیم می گردد.



شکل ۳۷-۳ خط آبی رنگ، خط میانگین ترسیم شده از تمام خطوط افقی موجود در صفحه

اما در مورد این که این میانگین گیری در کدام ناحیه باید انجام شود، مفهومی تحت عنوان منطقه مورد نظر[۲۲۳] تعریف می گردد که طبق آن، این میانگین گیری باید در ناحیه ی دید راننده ی خودرو انجام شود. این بدان معناست که سامانه ی راننده ی خودکار، تنها به تشخیص خطوط افقی در محلی از تصویر که به آن دید دارد نیاز دارد و این ناحیه ی دید، بسته به زاویه ی قرار گیری دوربین در خودرو مشخص می گردد[۲۶۴].



شکل ۳۸-۳ تعیین ناحیه ی دید مناسب برای سامانه، جهت میانگین گیری از خطوط افقی در آن[۲۶۴]

در این قسمت، توضیحات تئوری و ریاضیاتی نحوه ی تشخیص و ترسیم خطوط افقی موجود در جاده به پایان می رسد و در بخش بعد، نحوه ی پیاده سازی این موارد در برنامه، به تفصیل شرح داده می شود.

[223] Region of Interest

۲-۱-۳-۳- تشریح نحوه پیاده سازی الگوریتم، ساختار کد و ارائه ی نتایج

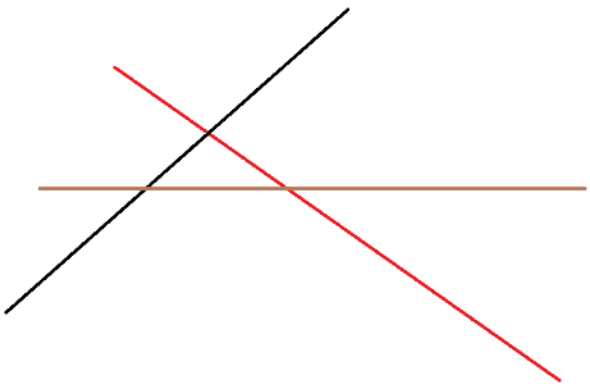
در این قسمت، به تشریح این موضوع پرداخته می شود که توضیحات تئوری و ریاضیات تبیین شده در بخش قبلی چگونه تبدیل به الگوریتم و در نهایت کدهای کامپیوتری می گردند.

تکه کد زیر، نحوه ی تشخیص خطوط کلی در تصویر را بیان می کند:

```
1.  def detect_lines(image):
2.      rho = 1
3.      angle = np.pi / 180
4.      min_threshold = 10
5.      lines = cv2.HoughLinesP(image, rho, angle, min_threshold, np.array([]),
6.      minLineLength=8,maxLineGap=4)
7.
8.      return lines
```

در قطعه کد بالا، تابعی به نام detect_lines تعریف شده که یک فریم از ویدئو را به عنوان ورودی دریافت می کند، در ادامه دقت محاسبه ی فاصله عمودی خطوط از مبدا مختصات یا همان ρ، یک پیکسل، دقت محاسبه ی زاویه ی خط عمود با محور افقی مختصات یا همان θ یک درجه و حداقل تعداد نقاطی که در صورت یکسان بودن جفت ρ و θ در آن ها، یک خط در نظر گرفته می شودند ۱۰ قرار داده شده است. در نهایت الگوریتم هاف تمام خطوط موجود در تصویر را با دقت های بیان شده و حداقل تعداد نقاط قابل قبول برای تشکیل خط، تشخیص می دهد و مختصات نقاط ابتدایی و انتهایی این خطوط، به عنوان خروجی از این تابع دریافت می گردد.

مشاهده ی عملکرد این قطعه از کد به شکل مجزا:



شکل ۳۹-۳ تصویر داده شده به تابع به عنوان ورودی

و اما خروجی داده شده توسط تابع:

```
Number of Detected Lines :  1580
[[[104 635 961 373]]

 [[312 256 961   7]]

 [[110 434 961  37]]

 ...

 [[636   4 649   6]]

 [[192 334 230 298]]

 [[  0 290 103 193]]]
```

در خروجی بالا، مشاهده می گردد که تمام خطوط موجود در تصویر با دقت یک پیکسل در فاصله و یک درجه در زاویه و همچنین با شرط وجود حداقل ده نقطه در خط، شناسایی شده و مختصات x و y نقاط ابتدایی و انتهایی آن ها به عنوان خروجی برگردانده شده است.

علت تعداد بالای خطوط شناسایی شده، مقادیری است که تحت عنوان دقت تشخیص در تابع ذخیره شده، به عنوان مثال، خطوط ترسیم شده در تصویری که به عنوان ورودی تابع در نظر گرفته شده، خود دارای ضخامت چندین پیکسل هستند، در حالی که دقت تشخیص تابع یک پیکسل می باشد و در واقع، پیکسل های تشکیل دهنده های یک خط، خود در جایگشت های مختلف، چند ده خط تشکیل می دهند که همگی آن ها توسط الگوریتم هاف شناسایی می گردد.

اما در تکه کد بعدی، نحوه ی تشکیل ناحیه دید راننده و میانگین گیری از خطوط افقی موجود در آن بر اساس شیب بیان شده است:

```
def horiz_lines(mask):
    roi = mask[560:, 230:1100]
    lines = detect_lines(roi)
    lines = lines.reshape(-1,2,2)
    slope = (lines[:,1,1]-lines[:,0,1]) / (lines[:,1,0]-lines[:,0,0])
    if (lines[np.where(abs(slope)<0.2)]).shape[0] > 60:
        xmin = lines[np.where(abs(slope)<0.2)].reshape(-1,4)[:,0].min(0)
        ymin = lines[np.where(abs(slope)<0.2)].reshape(-1,4)[:,1].min(0)
        xmax = lines[np.where(abs(slope)<0.2)].reshape(-1,4)[:,2].max(0)
        ymax = lines[np.where(abs(slope)<0.2)].reshape(-1,4)[:,3].max(0)
```

```
11.
12.         mean_slope = np.mean(slope[np.where(abs(slope)<0.2)])
13.         xy = np.array([xmin,ymin,xmax,ymax]) + [230,560,230,560]
14.         y0,x0 = np.mean(np.where(roi>0), axis=1) + [560,230]
15.         x = np.array([xy[0],xy[2]])
16.         y = mean_slope*(x - x0) + y0
17.         xmin,xmax = x
18.         ymin,ymax = y.astype(int)
19.
20.         return np.array([xmin,ymin,xmax,ymax])
21.     else:
22.         return None
```

در قطعه کد بالا، تابعی به نام horiz_lines مشاهده می شود که تصویر را به عنوان ورودی می گیرد و در خط دوم آن، ناحیه ی دید راننده تعریف گشته است. این ناحیه بر اساس پیکسل های تصویر و با تجربه و آزمون و خطا در ویدئو های متعدد و زوایای مختلف دوربین به دست آمده است.

در ادامه عملکرد قسمت تعیین ناحیه ی دید راننده به شکل مجزا نمایش داده است:

شکل ۴۰-۳ تصویر داده شده به تابع به عنوان ورودی جهت دریافت ناحیه ی دید راننده

شکل ۴۱-۳ ناحیه ی دید راننده که توسط تابع برگردانده شده است

پس از جدا کردن ناحیه ی دید از کل تصویر، در خط سوم، خطوط موجود در آن شناسایی شده و در خط چهارم، این خطوط به شکل جفت هایی از مختصات کارتزینی نقاط ابتدایی و انتهایی خطوط دسته بندی شده است.

در خط پنجم شیب تمامی این خطوط محاسبه شده و در حافظه ی موقت ذخیره شده است.

در خط ششم تا یازدهم این قطعه کد، ابتدا خطوط افقی، خطوطی با شیب کمتر از آستانه ۰.۲ در نظر گرفته شده و در ادامه شرطی گذاشته شده که در صورتی که تعداد این خطوط افقی بیشتر از ۶۰ بود، ماکزیمم و مینیمم مختصات خطوط به دست آورده شود تا از آن ها به عنوان نقاط ابتدایی و انتهایی برای رسم خط میانگین کلی استفاده شود.

شرط تعداد بیشتر از ۶۰ به این دلیل استفاده شده است که ممکن است چند نقطه با جفت ρ و θ های یکسان، با توجه به کمینه نقاط قابل قبول تعریف شده در کد، با هم یک خط تشکیل دهند اما این خط در تصویر موجود نباشد و تنها یک تطابق اتفاقی ρ و θ باشد. بنابراین، کمترین تعداد خط تشخیص داده شده جهت استفاده از آن ها در فرآیند میانگین گیری، ۶۰ در نظر گرفته شده است.

در خط دوازدهم، از شیب تمام خطوط افقی تشخیص داده شده در ناحیه، میانگین گرفته شده است.

در خط سیزدهم، x و y هایی که در خط ششم تا یازدهم به عنوان نقاط بیشینه و کمینه جهت استفاده به عنوان خطوط ابتدایی و انتهایی خط نهایی تشخیص داده شده بودند، با یک نسبت مختصات

بر اساس ناحیه ای که در خط دوم انتخاب شده بود جمع می شوند، چرا که این نقاط در ROI تشخیص داده شده‌اند و برای ترسیم صحیح در تصویر اصلی، باید به نسبت ROI انتخاب شده، به بالا منتقل شوند تا در تصویر اصلی در محل صحیح قرار بگیرند. در واقع مختصات x و y فعلی، مختصاتی است که بر اساس ابعاد ناحیه ی انتخاب شده به دست آمده و برای استفاده از آن ها در تصویر اصلی، باید این مختصات به محل صحیح خود در تصویر اصلی نگاشت شوند.

در خط چهاردهم، از تمام نقاط موجود در مختصات خطوط میانگین گیری می گردد تا به عنوان مختصات اختیاری در معادله ی خط مورد استفاده قرار گیرد.

در خط پانزدهم تا نوزدهم، بر اساس شیب میانگین به دست آمده و مختصات اختیاری به دست آمده از میانگین همه ی نقاط ابتدایی و انتهایی خط ها، مختصات x ابتدا و انتهای خط که از میانگین گیری به دست آمده بود به معادله ی خط داده می شود و مختصات y ابتدا و انتهای خط از این معادله دریافت می گردد.

در نهایت در ۳ خط آخر، تبیین می گردد که در صورت تشخیص خطوط افقی در ناحیه ی دید راننده، مختصات خط میانگین نهایی به عنوان خروجی تابع برگردانده شود و در غیر این صورت مقداری برگردانده نشود.

اما در ادامه، قطعه کد زیر که مربوط به بخش ترسیم است، بررسی می گردد:

```
1.  cap = cv2.VideoCapture("test1.mov")
2.  while(cap.isOpened()):
3.      try:
4.          ret , frame = cap.read()
5.          frame = cv2.rotate(frame,cv2.ROTATE_90_CLOCKWISE)
6.          mask = cv2.inRange(frame, np.array([70,70,90]), np.array([165,140,135]))
7.          out_points = horiz_lines(mask)
8.          for point in out_points:
9.              if out_points is not None:
10.                 frame=cv2.line(frame, (out_points[0],out_points[1]), (out_points[2],out_po
    ints[3]), [0,0,255], 5)
11.     except:
12.         pass
13.     cv2.imshow('output', frame)
14.     if cv2.waitKey(20) & 0xFF == ord('q'):
15.         break
16. cap.release()
17. cv2.destroyAllWindows()
```

در خطوط اول تا چهارم، ویدئو دریافت می گردد و هر فریم از این ویدئو به یک عکس تبدیل می گردد، همچنین برای دریافت خروجی مناسب از قطعه کد تست بالا، در خط پنجم، تصویر ۹۰ درجه چرخانده می شود، اما این

مورد در نهایت و در کد نهایی به شکل اتوماتیک بررسی می گردد و در این قسمت از گزارش صرفا برای نمایش خروجی اضافه گردیده است.

اما در خط ششم، محدوده ی رنگ خطوط جاده در تصویر انتخاب شده، بدین معنی که تنها قسمت‌هایی از تصویر که دارای رنگی مطابق با رنگ خطوط جاده که معمولا از گستره ی رنگ سفید هستند انتخاب گردد. خروجی خط ششم برای یک نمونه تصویر بدین ترتیب می باشد:

شکل ۴۲-۳ تصویر داده شده به قطعه کد مربوط به انتخاب گستره ی رنگ خطوط جاده

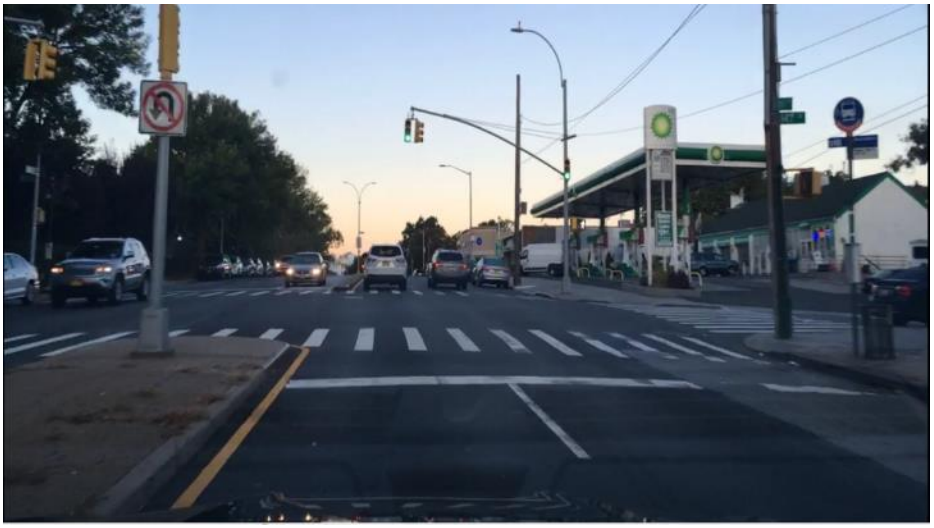
شکل ۴۳-۳ خروجی مربوط به قطعه کد مربوط به انتخاب گستره ی رنگ خطوط جاده

در خط هفتم، خط افقی میانگین در ناحیه ی دید راننده در تصویری که به گستره ی رنگ خطوط جاده برده شده، استخراج می گردند.

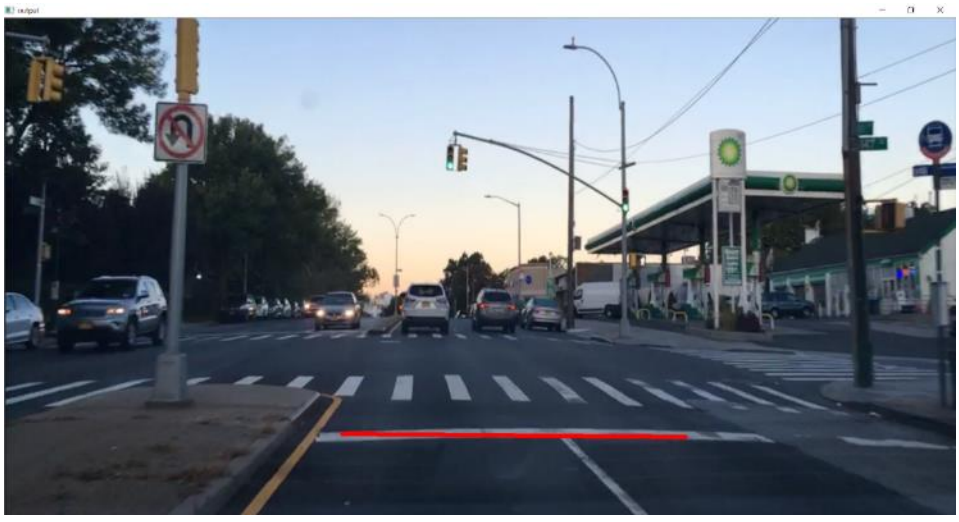
در خط هشتم تا دهم، تبیین می گردد که در صورتی که خط افقی میانگین شناسایی شده بود و این مقدار وجود داشت، این خط روی تصویر ترسیم گردد.

در نهایت و در خطوط باقی مانده ی کد، نحوه ی نمایش این تصاویر بیان شده است.

خروجی نهایی تشخیص خطوط افقی موجود در جاده:

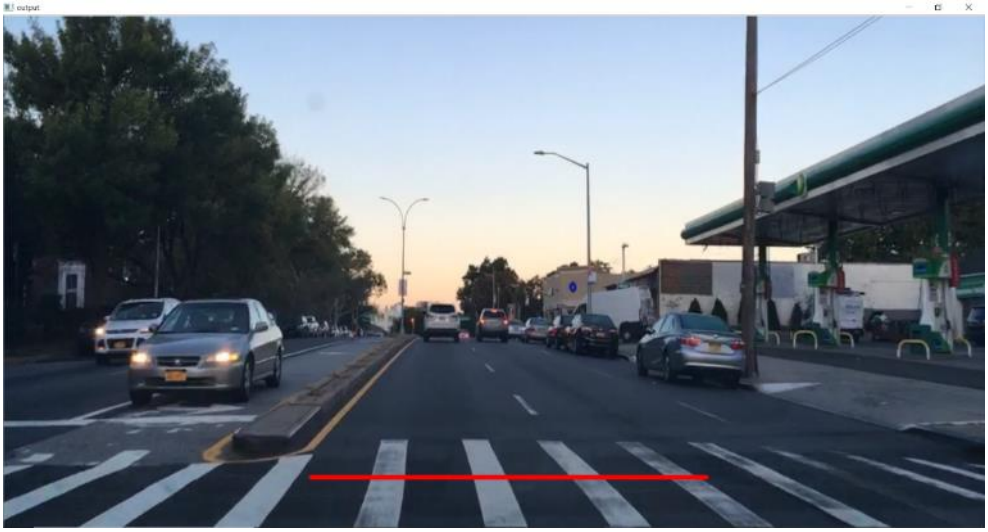


شکل ۴۴-۳ تصویر ورودی اول به قطعه کد تشخیص خطوط افقی



شکل ۴۵-۳ خروجی اول قطعه کد تشخیص خطوط افقی

شکل ۴۶-۳ تصویر ورودی دوم به قطعه کد تشخیص خطوط افقی



شکل ۴۷-۳ خروجی دوم قطعه کد تشخیص خطوط افقی

همچنین در خروجی های بالا دیده می شود که به علت ساختار دقیق میانگین گیری تعریف شده در کد، این قطعه کد توانایی این موضوع را نیز دارد که خطوط افقی موجود در خطوط عابر پیاده را تشخیص دهد و با میانگین گیری دقیق از شیب و ابعاد آن ها، خط افقی میانگین را ترسیم کند. در واقع، این قسمت از کد، در این تصویر تشخیص داده که چیدمان و نحوه ی قرار گیری خطوط عابر پیاده در کنار هم به منظور تشکیل یک خط افق فرضی در میان خیابان بوده و این خط را به درستی شناسایی و ترسیم نموده است.

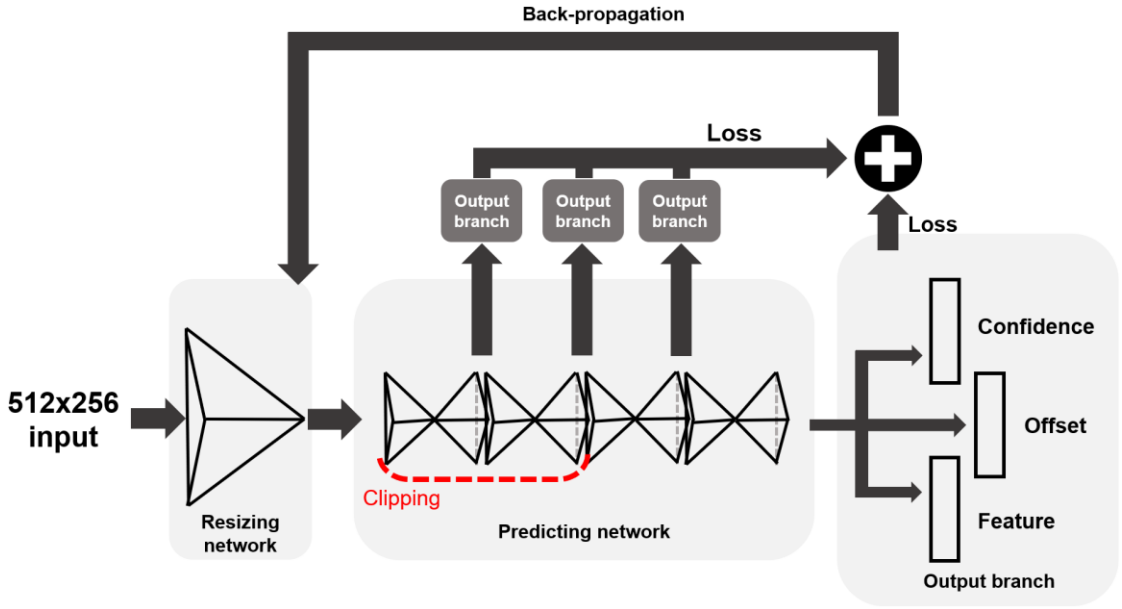
۲-۳-۳- تشخیص خطوط عمودی

۱-۲-۳-۳- تبیین شبکه PINet [۲۶۵]

تکنیک‌های درک محیط برای رانندگی خودکار باید با شرایط مختلف سازگار باشد. در مورد تشخیص خط ترافیک بسیاری از شرایط باید در نظر گرفته شود، مانند تعداد خطوط ترافیک و قدرت محاسباتی سیستم هدف. برای رفع این مشکلات، در این مقاله روشی جدید پیشنهاد شده است که بر اساس تخمین نقاط کلیدی[۲۲۴] و رویکرد تقسیم نمونه[۲۲۵] عمل می کند.

این شبکه شامل چندین شبکه ساعت شنی پشته ای[۲۲۶] است که به طور همزمان آموزش می بینند. بنابراین می توان اندازه مدل های آموزش دیده را با توجه به توان محاسباتی محیط هدف انتخاب کرد. این شبکه در مجموعه داده های CULane، TuSimple و CurveLanes که مجموعه داده های عمومی محبوبی برای تشخیص خطوط هستند، به دقت بالایی دست یافته است.

تصویر ۴۸-۳ ساختار شبکه PINet برای تشخیص خطوط ترافیکی را نشان می‌دهد.



شکل ۴۸-۳ ساختار شبکه PINet [۲۶۵]

[224] Keypoint estimation
[225] Instance Segmentation
[226] Stacked hourglass networks

با توجه به تصویر ۳-۴۸، ساختار شبکه دارای سه قسمت اصلی است. ابتدا تصویر ورودی به سایز 256 × 512 تبدیل می‌شود و سپس تصویر فشرده شده به شبکه پیش‌بینی کننده که شامل چهار ماژول ساعت شنی است، تغذیه می‌شود. سه شاخه خروجی در انتهای هر بلوک ساعت شنی اعمال می شود. آنها دقت[۲۲۷] ، میزان جبران[۲۲۸] و بردار ویژگی را پیش بینی می کنند. تابع هزینه را می‌بایست از خروجی های هر بلوک ساعت شنی محاسبه کرد.

ایده اصلی این مقاله در سه مورد زیر خلاصه می‌شود که به تفصیل به آن خواهیم پرداخت:

۱- در این مقاله با استفاده از روش تخمین نقاط کلیدی، یک روش جدید برای تشخیص خطوط ترافیکی پیشنهاد داده است. این شبکه خروجی به مراتب بهتری نسبت به شبکه‌هایی که مبتنی بر تقسیم بندی معنایی هستند تولید می کند.

۲- این ساختار از چندین ماژول ساعت شنی تشکیل شده است و می‌توان از هر ماژول مدل‌هایی با اندازه‌های متفاوتی بدست آورد. همچنین می‌توان با تغییر تعداد ماژول ها توان محاسباتی این شبکه را تنظیم کرد.

۳- این روش را می توان در صحنه های مختلف ترافیکی مانند خطوط ترافیکی عمودی یا افقی و تعداد دلخواه خطوط ترافیکی در هر جاده، اعمال کرد.

با توجه به تصویر ۳-۴۸، ساختار PINet دارای دو شبکه است.

۱- Resizing Network:

این شبکه سایز تصویر ورودی را کاهش می‌دهد تا از نظر حافظه و زمان پردازش بهینه باشد. ابتدا تصویری به ابعاد 256 × 512 به شبکه وارد می‌شود. این شبکه دارای ۳ لایه کانوولوشنی است که مشخصات هر لایه کانوولوشنی در جدول ۳-۱۱ آورده شده است. در نهایت خروجی که این شبکه می‌دهد، خروجی تغییر سایز شده به ابعاد 256 × 512 می‌باشد.

جدول۳-۱۱ جزئیات شبکه Resizing [۲۶۵]

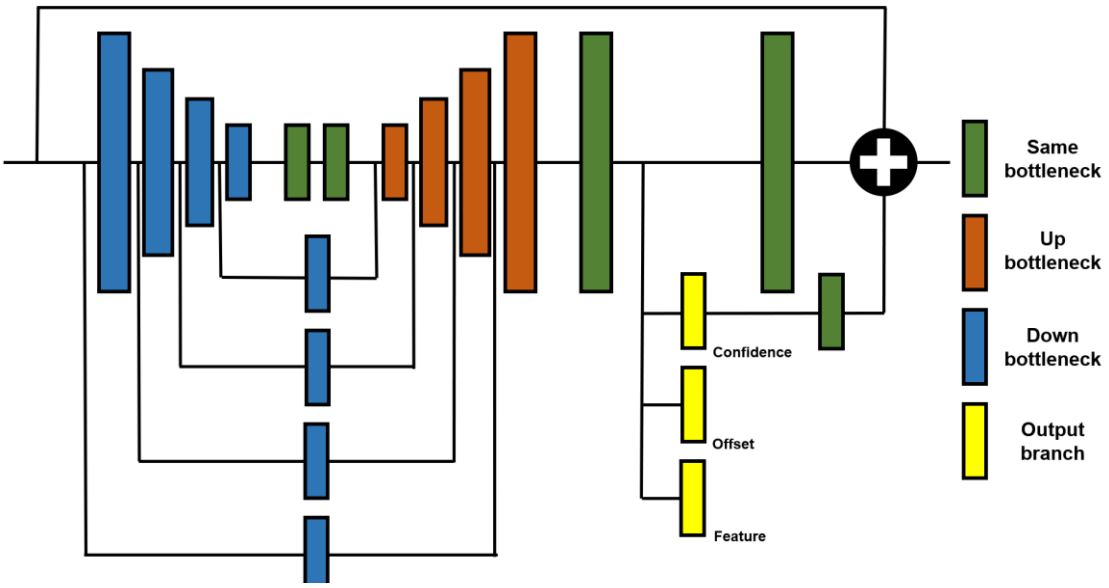
Layer	Size	Stride	Output Size
Input data			**3 × 512 × 256**
Conv+Prelu+bn	**3** × 3	2	**3 × 256 × 128**
Conv+Prelu+bn	3 × 3	2	**64 × 128 × 64**
Conv+Prelu+bn	3 × 3	2	**128 × 64 × 32**

[227] Confidence
[228] Offset

۲- Predicting Network:

خروجی شبکه Resizing به شبکه پیش‌بینی کننده وارد می‌شود. این قسمت نقاط دقیق خطوط ترافیکی و ویژگی های تعبیه شده برای تقسیم بندی را پیش بینی می کند. این شبکه از چندین ماژول ساعت شنی تشکیل شده است که هر کدام شامل یک انکودر[۲۲۹]، دیکودر[۲۳۰] و سه شاخه خروجی است، همانطور که در شکل ۴۹-۳ نشان داده شده است.

برخی از اتصالات پرشی[۲۳۱] ، اطلاعات مقیاس های مختلف را به لایه های عمیق تر منتقل می کنند.



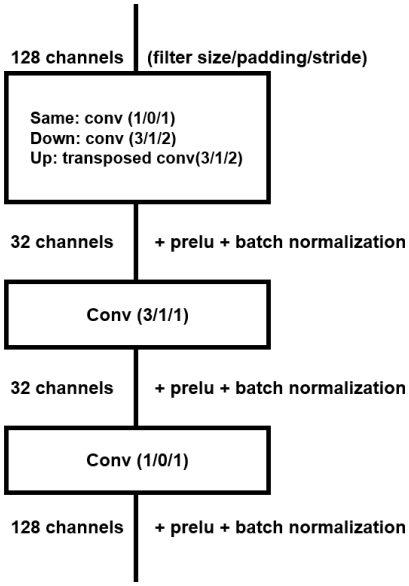
شکل ۴۹-۳ جزئیات بلوک ساعت شنی [۲۶۵]

هر بلوک رنگی در شکل ۴۹-۳ یک ماژول Bottle-Neck است. این ماژول های BottleNeck در شکل ۵۰-۳ توضیح داده شده است. سه نوع BottleNeck وجود دارد: BottleNeck یکسان ، پایین و بالا.

- **same BottleNeck** خروجی به همان اندازه ورودی تولید می‌کند.
- **Down BottleNeck** برای نمونه برداری در انکودر اعمال می شود. اولین لایه آن با توجه به شکل ۵۰-۳ با یک لایه کانولوشن با اندازه فیلتر سه، گام دو جایگزین می شود.

[229] Encoder
[230] Decoder
[231] Skip-Connections

- **Up BottleNeck** در لایه اول آن کانولوشن ترانهاده شده با اندازه فیلتر سه، گام دو برای نمونه برداری بالا[۲۳۲] استفاده می شود.



شکل ۵۰-۳ ماژول BottleNeck [۲۶۵]

هر شاخه خروجی دارای سه لایه کانولوشن است و یک شبکه به سایز 32×64 تولید می کند. مقادیر اطمینان در مورد وجود نقطه کلیدی، میزان جابجایی و بردار ویژگی هر سلول در شبکه خروجی توسط شاخه های خروجی پیش بینی می شود. جدول ۱۲-۳ جزئیات شبکه پیش بینی را نشان می دهد. همچنین تابع هزینه با توجه به هدف هر شاخه خروجی اعمال می شود.

[232] Up-sampling

جدول ۱۲-۳ جزئیات شبکه پیش بین [۲۶۵]

	Layer	Size/Stride	Output size
	Input data		128*64*32
Encoder	Bottle-neck(down)		128*32*16
	Bottle-neck(down)		128*16*8
	Bottle-neck(down)		128*8*4
	Bottle-neck(down)		128*4*2
	Bottle-neck		128*4*2
(Distillation layer)	Bottle-neck		128*4*2
	Bottle-neck		128*4*2
	Bottle-neck		128*4*2
Decoder	Bottle-neck(up)		128*8*4
	Bottle-neck(up)		128*16*8
	Bottle-neck(up)		128*32*16
	Bottle-neck(up)		128*64*32
Output branch	Conv+Prelu+bn	3/1	64*64*32
	Conv+Prelu+bn	3/1	32*64*32
	Conv	1/1	C*64*32

برای آموزش شبکه، چهار تابع هزینه به هر شاخه خروجی از شبکه های ساعت شنی اعمال می شود. همانطور که در جدول ۱۲-۳ مشاهده می‌شود، شاخه خروجی ۶۴ شبکه ایجاد می کند و هر سلول در شبکه خروجی از مقادیر پیش بینی شده ۷ کانال شامل مقدار اطمینان (۱ کانال) ، مقدار آفست (۲ کانال) و ویژگی تعبیه شده (۴) تشکیل شده است.

مقدار اطمینان تعیین می کند که آیا نقاط اصلی خط ترافیک وجود دارد یا خیر. مقدار آفست موقعیت دقیق نقاط کلیدی پیش بینی شده توسط مقدار اطمینان را محلی می کند و از ویژگی تعبیه شده برای تفکیک نقاط کلیدی به عنوان نمونه منفرد استفاده می شود.

۱- **Confidence Loss**: شاخه خروجی اطمینان مقدار اطمینان هر سلول را پیش بینی می کند. اگر نقطه کلیدی در سلول وجود داشت، مقدار اطمینان نزدیک به ۱ خواهد بود، در غیر این صورت مقدار صفر خواهد بود.

این تابع هزینه شامل دو بخش است،

- existence loss: به سلول‌هایی اعمال می‌شود که دارای نقاط کلیدی باشند.

$$L_{exist} = \frac{1}{N_e} \sum_{C_c \in G_e} (C_c^* - C_c)^2 \qquad (۳-۳)$$

- non-existence loss: برای کاهش میزان اطمینان هر سلول پس زمینه استفاده می شود.

$$L_{non\_exist} = \frac{1}{N_n} \sum_{\substack{C_c \in G_n \\ C_c > 0.01}} \left(C_c^{\,*} - C_c \right)^2 + 0.0001. \sum_{C_c \in G_n} C_c^{\,2} \qquad (۳-۴)$$

۲- **Offset Loss**: از شاخه آفست، PINet مکان دقیق نقاط اصلی را برای هر سلول خروجی پیش بینی می کند. خروجی هر سلول مقداری بین صفر و یک دارد. این مقدار موقعیت مربوط به سلول مربوطه را نشان می دهد. شاخه آفست دارای دو کانال برای پیش بینی مختصات x و محور y نقطه کلیدی است.

$$L_{offset} = \frac{1}{N_e} \sum_{C_x \in G_e} \left(C_x^{\,*} - C_x \right)^2 + \frac{1}{N_e} \sum_{C_y \in G_e} \left(C_y^{\,*} - C_y \right)^2 \qquad (۳-۵)$$

۳- **Embedding Feature**: هنگامی که شبکه آموزش می یابد، این تابع هزینه زمانی که هر سلول به یک نمونه مشخص تعلق دارد ویژگی ها را نزدیکتر می‌کند. این ویژگی ها را هنگامی که سلول ها به نمونه‌های مختلف تعلق دارند توزیع می کند. در اینجا می‌توان با استفاده از تکنیک خوشه بندی ساده مبتنی بر فاصله، نقاط کلیدی را به صورت نمونه مجزا تشخیص داد. بدین صورت که اگر ویژگی های تعبیه شده در برخی از نقاط اصلی پیش بینی شده در یک فاصله مشخص باشد، ما در نظر می گیریم که نقاط از یک نمونه هستند.

$$\boldsymbol{L_{feature} = \frac{1}{{N_e}^2} \sum_{i}^{N_e} \sum_{j}^{N_e} l(i,j)} \qquad (۳-۶)$$

$$\boldsymbol{l(i,j) = \begin{cases} ||F_i - F_j||_2 if I_{ij} = 1 \\ max(0, K - ||F_i - F_j||_2) if I_{ij} = 0 \end{cases}}$$

۴- **Distillation Loss**:

هر چه تعداد ماژول های ساعت شنی بیشتر شود، نتیجه بهتر خواهد شد.

با استفاده از این تکنیک می‌توان با بریدن قسمتی از ماژول‌های ساعت شنی، قسمت بریده شده عملکرد بهتری نسبت به مدل اصلی خود ارائه دهد.

$$L_{distillation} = \sum_{m}^{M} D(F(A_M) - F(A_m)), \tag{۳-۷}$$

$$F(A_m) = S(G(A_M)), S: spatialsoftmax,$$

$$G(A_m) = \sum_{i=1}^{C} |A_{mi}|, G: R^{C\times H\times W} \rightarrow R^{H\times W}$$

در نهایت با جمع توابع هزینه فوق، تابع هزینه مورد نیاز جهت آموزش شبکه بدست می‌آید[۲۶۵].

$$L_{total} = \lambda_e L_{exist} + \lambda_n L_{non-exist} + \lambda_o L_{offset} + \lambda_f L_{feature} + \lambda_d L_{distillation} \tag{۳-۸}$$

$$\lambda_o = 0.2, \lambda_f = 0.5, \lambda_d = 0.1$$

$$\lambda_e = 1.0, \lambda_n = 1.0 \xrightarrow{at the last 40 epochs} 2.5$$

۲-۲-۳-۳- چگونگی آموزش شبکه PINet

در این بخش نحوه آموزش شبکه PINet بر روی دیتاست CULane توضیح داده شده است.

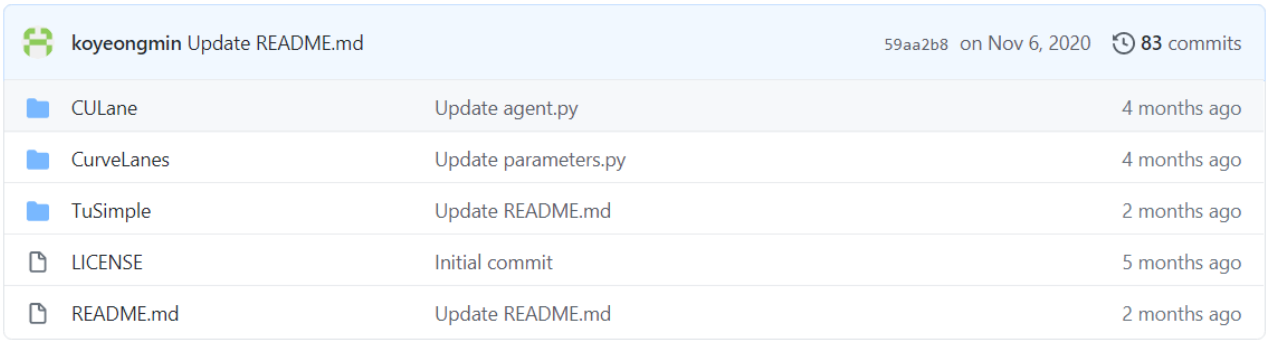
قبل از شروع آموزش شبکه، ابتدا مجموعه کد PINet را از آدرس گیت‌هاب PINet_new [۲۶۶] به صورت زیر دانلود میکنیم.

```
$ https://github.com/koyeongmin/PINet_new.git
```

پکیج‌های مورد نیاز جهت نصب:

- $Python \geq 3.6$
- $PyTorch \geq 1.1.0$
- Opencv
- Numpy
- Sklearn (به منظور ارزیابی)
- Ujon (به منظور ارزیابی)

سورس اصلی کد این شبکه به صورت تصویر ۵۱-۳ می‌باشد:

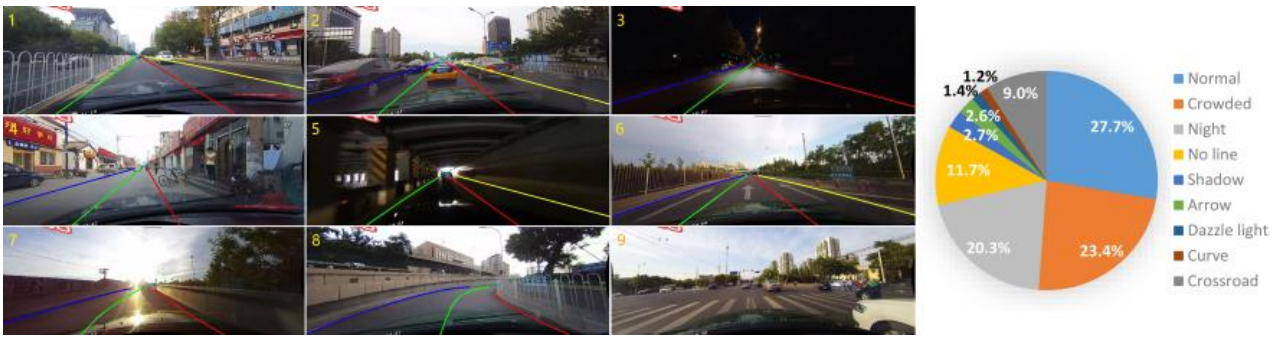


شکل ۵۱-۳ سورس کد شبکه PINet [۲۶۶]

این شبکه بر روی سه دیتاست CULane، CurveLanes و TuSimple به صورت جداگانه آموزش داده شده به همین منظور بدلیل تفاوت در ساختار دیتاست‌ها، برای هر عنوان، سورس کدی به صورت جداگانه قرار داده شده است.

آموزش شبکه PINet بر روی دیتاست CULane [۲۶۷]:

CULaneیک مجموعه داده چالش برانگیز در مقیاس بزرگ برای تحقیقات دانشگاهی در مورد تشخیص خط ترافیک است. این مجموعه داده توسط دوربین های نصب شده بر روی شش وسیله نقلیه مختلفی که توسط رانندگان مختلف در پکن رانندگی می‌کردند، جمع آوری شده است. بیش از ۵۵ ساعت فیلم جمع آوری و ۱۳۳،۲۳۵ فریم استخراج شد. نمونه ای از این دیتاست در تصویر ۵۲-۳ آورده شده است.



شکل ۵۲-۳ (الف) مثال‌هایی از مجموعه داده برای سناریوهای مختلف. (ب) درصد داده سناریو های مختلف در این دیتاست[۲۶۷].

این مجموعه داده به ۸۸۸۸۰ برای مجموعه داده آموزش، ۹۶۷۵ برای مجموعه داده ارزیابی و ۳۴۶۸۰ برای مجموعه تست تقسیم شده است. همچنین مجموعه تست به ۲ دسته عادی و ۸ چالش برانگیز تقسیم می شود[۲۶۷].

برای آموزش شبکه ابتدا دیتاست CULane را می‌بایست از سایت مقابل دانلود کرد :

https://xingangpan.github.io/projects/CULane.html

فرمت داده‌های دیتاست CULane به صورت مقابل می‌باشد:

```
dataset
  |
  |----train_set/               # training root
  |------|
  |------|----driver_23_30frame/
  |------|----driver_161_90frame/
  |------|----driver_182_30frame/
  |
  |----test_set/               # testing root
  |------|
  |------|----driver_37_30frame/
  |------|----driver_100_30frame/
  |------|----driver_193_90frame/
  |
  |----list/               # testing root
  |------|
  |------|----test_split/
  |------|----test.txt
  |------|----train.txt
  |------|----train_gt.txt
  |------|----val.txt
  |------|----val_gt.txt
```

به منظور آموزش این شبکه ابتدا با دستور مقابل به پوشه CULane رفته و سپس Parameters.py را باز می‌کنیم.

```
$ cd CULane
```

در فایل Parameters.py، پارامترهای مهم شبکه در آن قرار دارد.

به منظور آموزش این شبکه می بایست ابتدا مسیر داده‌های آموزش و تست مجموعه داده CULane در خط ۴۹ و ۵۰ فایل Parameters.py مشخص کرد.

```
train_root_url="/dataset/CULane_dataset/train_set/"
test_root_url ="/dataset/CULane_dataset/test_set/"
```

در مرحله بعد خط ۱۳ Parameters.py که در آن آدرس مدل می‌بایست قرار داد، به صورت خالی قرار می‌دهیم:

```
13. model_path = ""
```

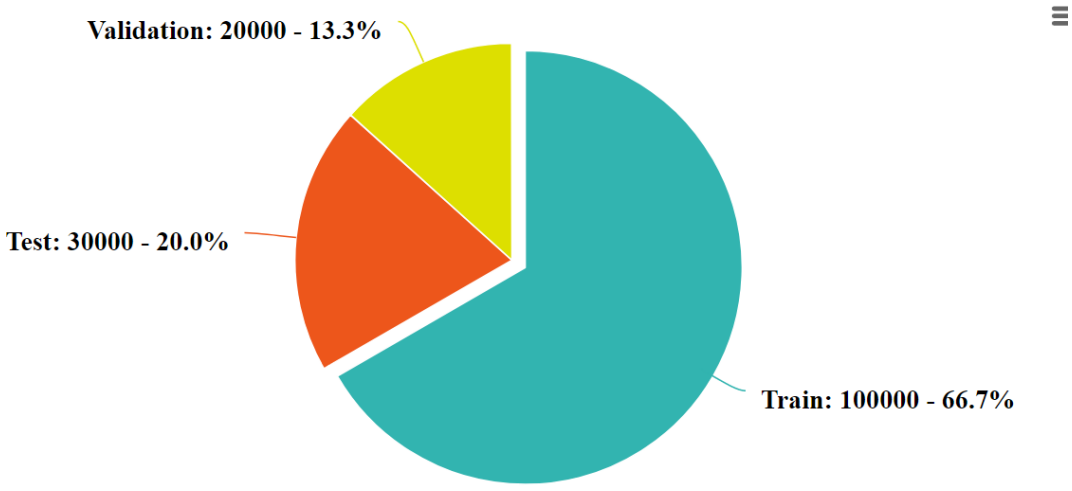
در نهایت بعد از انجام مراحل فوق با اجرای دستور مقابل روند آموزش شبکه PINet بر روی مجموعه داده CULane آغاز می‌شود.

```
1. $ train.py
```

نتیجه داده‌های تست در پوشه test_result و مدل آموزش داده شده در پوشه savefile ذخیره می‌شود.

آموزش شبکه PINet بر روی دیتاست CurveLanes [۲۶۸]:

CurveLanes یک مجموعه داده برای تشخیص خطوط جاده با ۱۵۰ هزار تصویر برای سناریوهای دشوار مانند منحنی ها و جاده های چند لاین در تشخیص خطوط ترافیکی است. این مجموعه داده در سناریوهای واقعی شهری و بزرگراهی در چندین شهر چین جمع آوری شده است. این بزرگترین مجموعه داده تشخیص خط تاکنون است و وضوح بیشتر تصاویر به صورت ۲۶۵۰ در ۱۴۵۰ می‌باشد. مجموعه داده‌گان این دیتاست به صورت تصویر ۵۳-۳ تقسیم می‌شوند[۲۶۸]:



شکل ۵۳-۳ تقسیم‌بندی مجموعه داده CurveLanes

نمونه ای از تصاویر لیبل زده شده این دیتاست در تصویر ۵۴-۳ آورده شده است.

شکل ۵۴-۳ نمونه‌هایی از مجموعه‌داده CurveLanes [۲۶۸]

برای آموزش شبکه ابتدا دیتاست CurveLanes را می‌بایست از سایت مقابل دانلود کرد :

https://reposhub.com/python/deep-learning/xbjxh-CurveLanes.html

مجموعه داده CurveLanes شامل سه بخش باشد:

- **آموزش:** شامل تصاویر، لیبل‌ها و فایل train.txt می‌باشد.
- **اعتبارسنج:** شامل تصاویر، لیبل‌ها و valid.txt می‌باشد.
- **تست:** شامل تصاویر و test.txt می‌باشد.

فرمت داده‌های دیتاست CurveLanes به صورت مقابل می‌باشد:

```
|
├─test
|  └─images
|          0001bca638957d305a25dd6be2fd5224.jpg
|          ...
|
├─train
|  |  train.txt
|  |
|  ├─images
|  |      00007f3230d35a893230c1b3ef8c52fd.jpg
|  |      ...
|  |
|  └─labels
|          00007f3230d35a893230c1b3ef8c52fd.lines.json
|          ...
|
└─valid
    |  valid.txt
    |
    ├─images
    |      00022953ff37d3174cff99833df8799e.jpg
    |      ...
    |
    └─labels
           00022953ff37d3174cff99833df8799e.lines.json
```

برای هر تصویر، یک فایل برچسب lines.json وجود دارد که در آن مختصات x، y نقاط هر خط به صورت مقابل علامت گذاری شده است.

```
{
  "Lines":[
    # A lane marking
    [
      # The x, y coordinates for key points of a lane marking that has at least t
wo key points.
      {
        "y":"1439.0",
        "x":"2079.41"
      },
      {
        "y":"1438.08",
        "x":"2078.19"
```

```
      },
      ...
    ]
    ...
  ]
}
```

به منظور آموزش این شبکه ابتدا با دستور مقابل به پوشه CurveLanes رفته و سپس Parameters.py را باز می‌کنیم.

```
$ cd CurveLanes
```

در فایل Parameters.py، پارامترهای مهم شبکه در آن قرار دارد.

به منظور آموزش این شبکه می بایست ابتدا مسیر داده‌های آموزش و تست مجموعه داده CurveLanes در خط ۵۵ و ۵۶ فایل Parameters.py مشخص کرد.

```
train_root_url= "/dataset/CurveLanes/train/"
test_root_url = "/dataset/CurveLanes/valid/"
```

در مرحله بعد خط ۱۵ Parameters.py که در آن آدرس مدل می‌بایست قرار داد، به صورت خالی قرار می‌دهیم:

```
model_path = ""
```

در نهایت بعد از انجام مراحل فوق با اجرای دستور مقابل روند آموزش شبکه PINet بر روی مجموعه داده CurveLanes آغاز می‌شود.

```
$ train.py
```

نتیجه داده‌های تست در پوشه test_result و مدل آموزش داده شده در پوشه savefile ذخیره می‌شود.

آموزش شبکه PINet بر روی دیتاست TuSimple [۲۶۹]:

این مجموعه داده به عنوان بخشی از چالش تشخیص خط Tusimple منتشر شد. این شامل ۳۶۲۶ کلیپ ویدئویی با مدت زمان یک ثانیه است. هر یک از این کلیپ های ویدئویی شامل ۲۰ فریم است که آخرین فریم حاوی لیبل است. این فیلم ها با نصب دوربین ها روی داشبورد خودرو ثبت شده‌اند [۲۶۹].

نمونه‌ای از تصویر دیتاست TuSimple در تصویر ۵۵-۳ آورده شده است.

شکل ۵۵-۳ نمونه‌هایی از مجموعه‌داده Tusimple [۲۶۹]

برای آموزش شبکه ابتدا دیتاست TuSimple را می‌بایست از سایت مقابل دانلود کرد :

https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection

فرمت داده‌های دیتاست TuSimple به صورت مقابل می‌باشد:

```
dataset
  |
  |----train_set/               # training root
  |------|
  |------|----clips/            # video clips, 3626 clips
  |------|------|
  |------|------|----some_clip/
  |------|------|----...
  |
  |------|----label_data_0313.json      # Label data for lanes
  |------|----label_data_0531.json      # Label data for lanes
  |------|----label_data_0601.json      # Label data for lanes
  |
  |----test_set/               # testing root
  |------|
  |------|----clips/
  |------|------|
  |------|------|----some_clip/
  |------|------|----...
  |
  |------|----test_label.json           # Test Submission Template
  |------|----test_tasks_0627.json      # Test Submission Template
```

به منظور آموزش این شبکه ابتدا با دستور مقابل به پوشه TuSimple رفته و سپس Parameters.py را باز می‌کنیم.

```
$ cd TuSimple
```

در فایل Parameters.py، پارامترهای مهم شبکه در آن قرار دارد.

به منظور آموزش این شبکه می بایست ابتدا مسیر داده‌های آموزش و تست مجموعه داده TuSimple در خط ۵۴ و ۵۵ فایل Parameters.py مشخص کرد.

```
54. train_root_url="/dataset/TuSimple_dataset/train_set/"
55. test_root_url="/ dataset/TuSimple_dataset/test_set/"
```

بعد از مشخص کردن آدرس مجموعه‌داده، با اجرا کردن کد "fix_dataset.py"، مجموعه‌داده به فرمت مورد نیاز تولید می‌شود و داده ها در پوشه "dataset" ذخیره می‌شوند.

در مرحله بعد خط ۱۳ Parameters.py که در آن آدرس مدل می‌بایست قرار داد، به صورت خالی قرار می‌دهیم:

```
13. model_path = ""
```

در نهایت بعد از انجام مراحل فوق با اجرای دستور مقابل روند آموزش شبکه PINet بر روی مجموعه داده TuSimple آغاز می‌شود.

```
1. $ train.py
```

نتیجه داده‌های تست در پوشه test_result و مدل آموزش داده شده در پوشه savefile ذخیره می‌شود.

۴-۲-۳-۳- نتیجه آموزش و خروجی شبکه

همانطور که در بخش ۳-۲-۳-۳ گفته شده،

دیتاست های نام برده بسیار حجیم بوده و آموزش دوباره آن‌ها نیازمند سخت افزار مناسب و زمان کافی می‌باشد و آموزش دوباره این دیتاست‌ها صرفا دوباره تولید کردن نتایج است. به همین منظور ما از مدل‌های از پیش آموزش داده شده که در سایت گیت‌هاب شبکه PINet [۲۶۶] قرار گرفته است، استفاده کردیم.

نتایج مدل آموزش داده شده بر روی دیتاست TuSimple در جدول ۳-۱۳ آورده شده است :

جدول ۳-۱۳ نتایج شبکه PINet بر روی دیتاست TuSimple

Accuracy	False Positive	False Negative
96.75%	0.0310	**0.0250**

دقت یا Accuracy:

به طور کلی، دقت به این معناست که مدل تا چه اندازه خروجی را درست پیش‌بینی می‌کند. با نگاه کردن به دقت، بلافاصله می‌توان دریافت که آیا مدل درست آموزش دیده است یا خیر و کارآیی آن به طور کلی چگونه است.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Examples} \qquad (۳-۹)$$

همچنین نتایج مدل آموزش داده شده بر روی دیتاست CULane در جدول ۱۴-۳ آورده شده است:

جدول ۱۴-۳ نتایج شبکه PINet بر روی دیتاست CULane [۲۶۶]

Category	F1-measure
Normal	90.3
Crowded	72.3
HLight	66.3
Shadow	68.4
No line	49.8
Arrow	83.7
Curve	65.6
Crossroad	1427 （FP measure）
Night	67.7
Total	74.4

F1-Measue:

معیار F1، یک معیار مناسب برای ارزیابی دقت یک آزمایش است. این معیار Precision و Recall را با هم در نظر می‌گیرد.

$$F1 = 2 \times \frac{Percision \times \mathrm{Re}call}{Percision + \mathrm{Re}call} \qquad (۳-۱۰)$$

در ادامه به بررسی نتایج مدل‌های مختلف از پیش آموزش داده شده شبکه PINet بر روی دیتاست های مختلف خواهیم پرداخت.

نتایج مدل آموزش داده شده بر روی دیتاست TuSimple:

نتایج ویدئو های تست در اختیار گذاشته شده بر روی این مدل در تصاویر ۵۶-۳، ۵۷-۳ آورده شده است.

شکل ۵۶-۳ نتیجه تست توسط مدل TuSimple

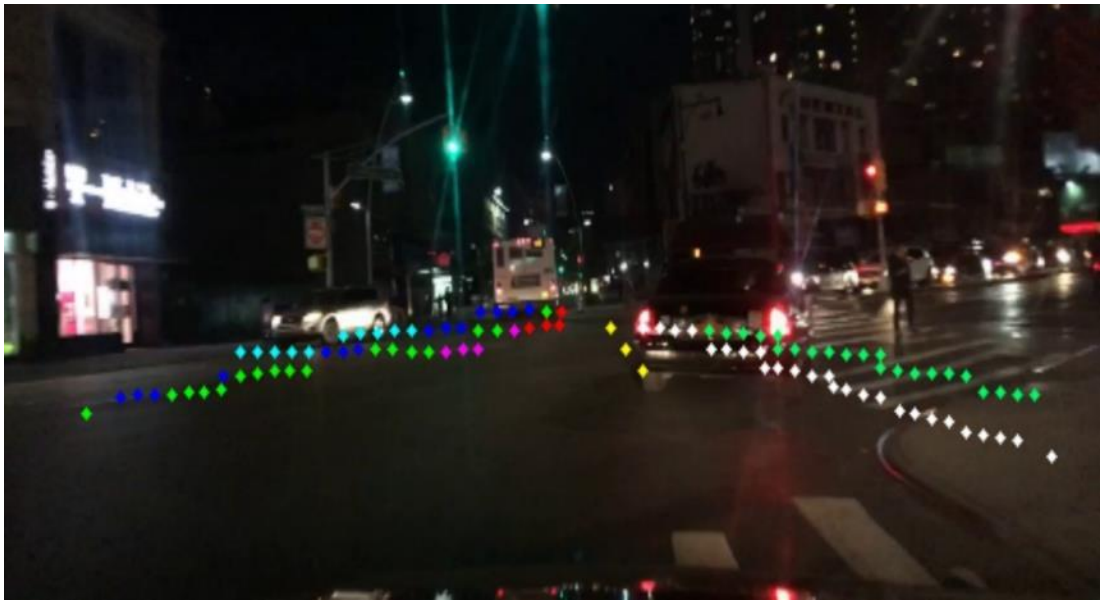
شکل ۵۷-۳ نتیجه تست دو توسط مدل TuSimple

همانطور که از نتایج مشخص است این مدل از پیش آموزش داده شده بر روی دیتاست TuSimple به دلیل سادگی داده‌های این دیتاست، نتایج قابل قبولی بر روی ویدئوهای تست حاصل نکرده است.

نتایج مدل آموزش داده شده بر روی دیتاست CurveLanes:

نتایج ویدئو های تست در اختیار گذاشته شده بر روی این مدل در تصاویر ۵۸-۳، ۵۹-۳، ۶۰-۳ آورده شده است.

شکل ۵۸-۳ نتیجه تست یک توسط مدل CurveLanes



شکل ۵۹-۳ نتیجه تست دو توسط مدل CurveLanes

شکل ۶۰-۳ نتیجه تست سه توسط مدل CurveLanes

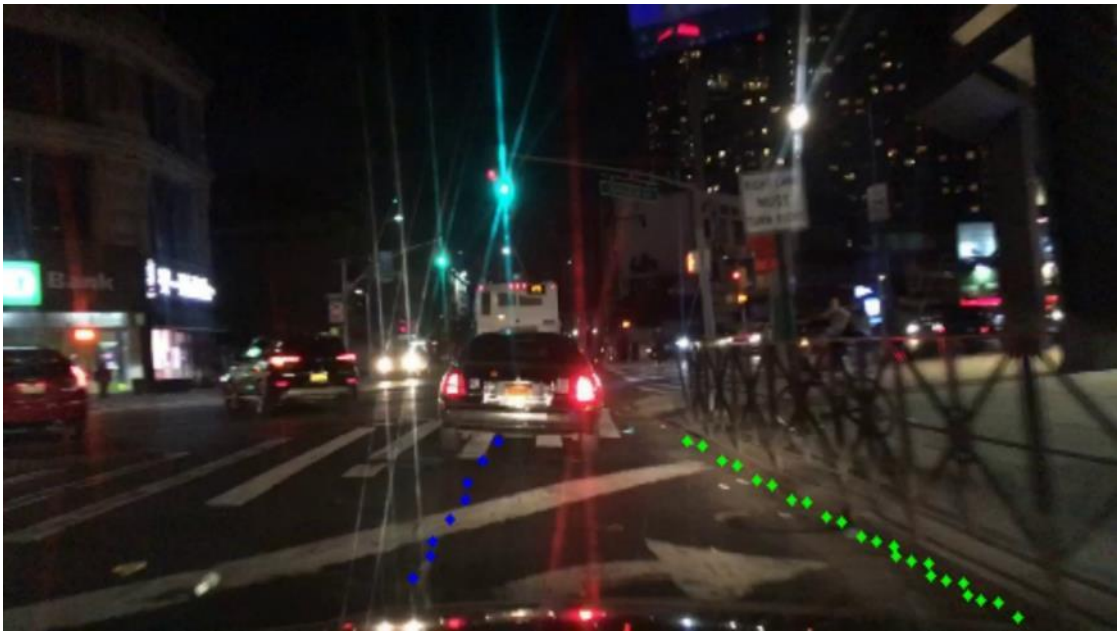
همانطور که از نتایج مشخص است این مدل از پیش آموزش داده شده بر روی دیتاست CurveLanes به دلیل پیچیدگی داده‌های این دیتاست و همینطور وجود داده‌های بسیار زیاد با تعداد زیادی خط ترافیکی در دیتاست خود، سعی در یافتن خطوط ترافیکی بیش از مقدار نیاز دارد. به همین دلیل با اینکه این مدل نسبت به مدل آموزش داده شده بر روی TuSimple بهتر عمل کرده است ولی نتوانسته انتظارات را به خصوص در شب برآورده کند.

نتایج مدل آموزش داده شده بر روی دیتاست CuLane:

نتایج ویدئو های تست در اختیار گذاشته شده بر روی این مدل در تصاویر ۳-۶۱، ۳-۶۲، ۳-۶۳ ، ۳-۶۴ و ۳-۶۵ آورده شده است.

شکل ۳-۶۱ نتیجه ویدئو یک

شکل ۳-۶۲ نتیجه ویدئو دو



شکل ۳-۶۳ نتیجه ویدئو سه

شکل ۳-۶۴ نتیجه ویدئو چهار

شکل ۳-۶۵ نتیجه ویدئو پنج

با توجه به تصاویر فوق، این مدل از پیش آموزش داده شده بر روی دیتاست CuLane به خوبی توانسته بر پیچیدگی و چالش‌های ویدئو تست غلبه کند و نتایج قابل قبولی بویژه در شب ارائه داده است و انتخاب ما برای تشخیص خطوط ترافیکی برای نرم‌افزار انتهایی خود، این مدل می‌باشد.

۵-۲-۳-۳- تشریح نحوه پیاده سازی الگوریتم و ساختار کد

با توجه به نتایج گرفته شده در قسمت ۴-۲-۳-۳، دو مدل آموزش داده شده بر روی دو دیتاست CULane و CurveLanes را در ساختار نرم افزار خود قرار دادیم که کاربر بتواند مدل دلخواه خود را بر اساس پیچیدگی ویدئو، برای شناسایی خطوط ترافیکی انتخاب کند.

پیشنهاد ما استفاده از مدل CULane است.

در کد توسعه داده شده، ماژول PINet.py حاوی کلاس LaneDetection است که وظیفه شناسایی خطوط ترافیکی و رسم آن را بر عهده دارد. در main.py ابتدا این کلاس به ساختار اصلی اضافه شده و یک شیء[۲۳۳] از این کلاس ساخته می‌شود.

```
1. if opt.lane_detector_type == 'culane':
2.     lane_detector = LaneDetection(opt.culane_model)
3.     print("CULane model loaded!")
4. if opt.lane_detector_type == 'curvelane':
5.     lane_detector = CurveLane(opt.curvelane_model)
6.     print("Curvelane model loaded!")
```

اگر مدل انتخابی، مدل آموزش داده شده بر روی دیتاست CuLane باشد، مدل CuLane لود[۲۳۴] می‌شود و اگر مدل انتخابی، مدل آموزش داده شده بر روی دیتاست Curvelanes باشد، مدل Curvelanes لود می‌شود. البته مدل انتخابی به صورت پیش‌فرض بر روی CuLane تنظیم شده است.

با فراخوانی تابع Testing که در PINet.py است، کار تشخیص خطوط ترافیکی و رسم آن در هر فریم انجام می‌گیرد.

[233] Object
[234] Load

ساختار کد PINet.py در ادامه به صورت کامل توضیح داده شده است.

ابتدا کتابخانه های مورد نیاز را ایموپرت [۲۳۵]یا وارد می کنیم.

ابتدا کتابخانه های مورد نیاز را لود می کنیم.

```
1. import cv2
2. import json
3. import torch
4. from CULane.agent import *
5. from CULane.util import *
6. import numpy as np
7. from copy import deepcopy
8. import time
9. from CULane.parameters import Parameters
10. import os
```

با فراخوانی کلاس Parameters، پارامترهای مورد نیاز فراخوانی می‌شود.

```
1. p = Parameters()
```

متغیر device مشخص می‌کند آیا سیستم شما به GPU متصل است یا خیر. اگر متصل باشد، مدل بر روی GPU شما Load می‌شود در غیر این صورت بر روی همان CPU اجرا می‌شود.

یکی از ویژگی های مهم این نرم‌افزار، نوشتن کدها بر پایه منطق شی‌گرایی است. در برنامه‌نویسی شی‌گرا ویژگی های مربوط به هر شی در محدوده و کلاس مربوط به خود شی قرار می گیرد. اشیا دیگر قدرت دسترسی و ایجاد تغییرات در داده های کلاس را ندارند و فقط می توانند به لیستی از توابع کلاس که بصورت عمومی تعریف شده‌اند دسترسی داشته باشند. این ویژگی در برنامه‌نویسی شی گرا باعث بالا رفتن امنیت و جلوگیری از فساد ناخواسته اطلاعات شده است.

[235] import

در کلاس LaneDetection، ساختار توابع به صورت مقابل نوشته شده است:

```
class LaneDetection():
    def __init__(self,model_path):

    def Testing(self, frame, mask):

    ##############################################################################
    ## test on the input test image
    ##############################################################################
    def test(self, lane_agent, test_images, mask, thresh = p.threshold_point, index= -
1):

    ##############################################################################
    ## eliminate result that has fewer points than threshold
    ##############################################################################
    def eliminate_fewer_points(self, x, y):

    ##############################################################################
    ## generate raw output
    ##############################################################################
    def generate_result(self, confidance, offsets,instance, thresh):
```

ابتدا در قسمت __init__، مدل را Load می‌کنیم. این تابع یک ورودی به عنوان آدرس مدل می‌گیرد. ویژگی این قسمت این است که فقط یک‌بار اجرا می‌شود و کار Load کردن مدل در هر فریم انجام نمی‌شود.

```
def __init__(self,model_path):
    self.lane_agent = Agent()
    self.lane_agent.load_weights(model_path)
```

در تابع Testing:

ورودی‌ها:

- Frame: تصویر ورودی
- Mask: تصویر ROI

خروجی:

- تصویری که روی آن خطوط ترافیکی شناسایی شده، رسم شده است.

```
1. def Testing(self, frame, mask):
2.
3.         if torch.cuda.is_available():
4.             self.lane_agent.cuda()
5.             torch.cuda.synchronize()
6.
7.         frame = cv2.resize(frame, (512,256))/255.0
8.         mask = cv2.resize(mask, (512,256))
9.
10.        frame = np.rollaxis(frame, axis=2, start=0)
11.        _, _, ti = self.test(self.lane_agent, np.array([frame]), mask)
12.        ti[0] = cv2.resize(ti[0], (1280,720))
13.
14.        return ti[0]
```

تابع test در تابع Testing، فراخوانی شده است.

ورودی:

- Lane_agent: مدل لود شده می‌باشد.
- Test_images: فریم اصلی تصویر.
- Mask: ماسک تصویر.
- Thresh: تعیین آستانه برای مقادیر نقاط تخمین زده شده برای خطوط ترافیکی

خروجی:

مختصات x و y نقاط تخمین زده شده به همراه تصویر خروجی.

```
def test(self, lane_agent, test_images, mask, thresh = p.threshold_point, index= -1):

    result = lane_agent.predict_lanes_test(test_images)
    if torch.cuda.is_available():
        torch.cuda.synchronize()
    confidences, offsets, instances = result[index]

    num_batch = len(test_images)

    out_x = []
    out_y = []
    out_images = []

    for i in range(num_batch):
        # test on test data set
        image = deepcopy(test_images[i])
        image =  np.rollaxis(image, axis=2, start=0)
        image =  np.rollaxis(image, axis=2, start=0)*255.0
        image = image.astype(np.uint8).copy()

        confidence = confidences[i].view(p.grid_y, p.grid_x).cpu().data.numpy()

        offset = offsets[i].cpu().data.numpy()
        offset = np.rollaxis(offset, axis=2, start=0)
        offset = np.rollaxis(offset, axis=2, start=0)

        instance = instances[i].cpu().data.numpy()
        instance = np.rollaxis(instance, axis=2, start=0)
        instance = np.rollaxis(instance, axis=2, start=0)

        # generate point and cluster
        raw_x, raw_y = self.generate_result(confidence, offset, instance, thresh)

        # eliminate fewer points
        in_x, in_y = self.eliminate_fewer_points(raw_x, raw_y)

        # sort points along y
        in_x, in_y = sort_along_y(in_x, in_y)

        result_image = draw_points(in_x, in_y, deepcopy(image), mask)

        out_x.append(in_x)
        out_y.append(in_y)
        out_images.append(result_image)

    return out_x, out_y, out_images
```

* در کد فوق تابع generate_result، نقاط تخمین شده برای خطوط ترافیکی را تعیین می‌کند.

* تابع eliminate_fewer_points، نقاطی که دارای تعداد کمتری نسبت به آستانه است حذف می‌کند.

* در انتها تابع draw_points، وظیفه رسم نقاط را بر روی تصویر اصلی برعهده دارد.

۳-۴- تعیین فاصله نسبت به خودروهای اطراف

در این بخش به تعیین فاصله نسبت به خودروهای اطراف با دو روش کلاسیک و یادگیری عمیق پرداخته شده است.

۱-۴-۳- فاصله سنج کلاسیک

۱-۱-۴-۳- تبیین روش مورد نظر به همراه ریاضیات کامل

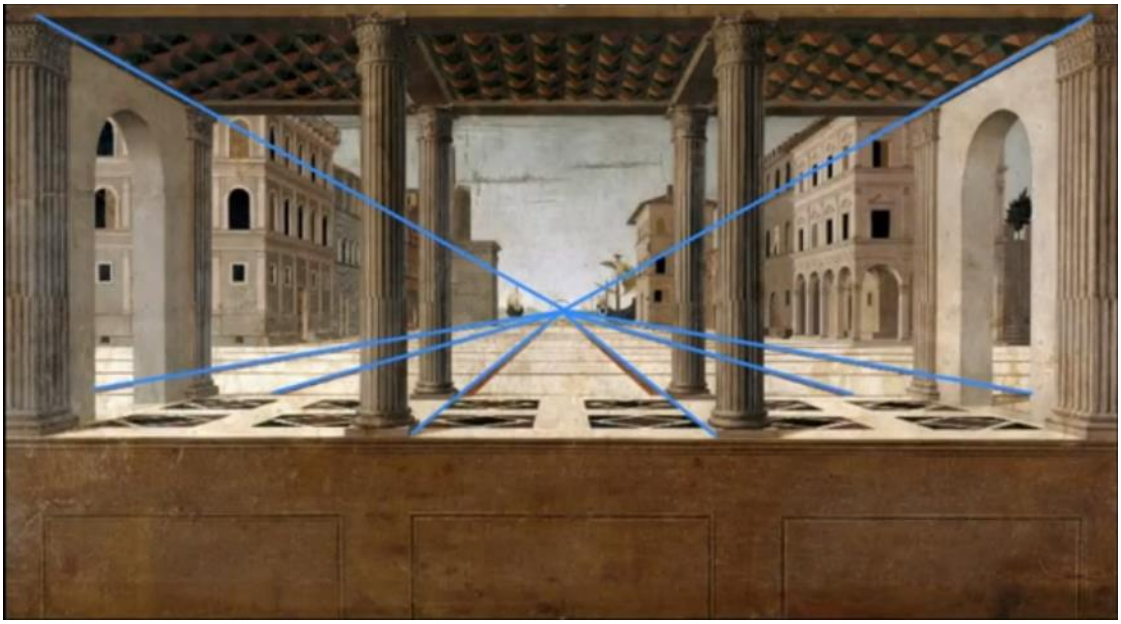
در این قسمت از گزارش، تمام ابعاد تئوری و ریاضیاتی مرتبط با فاصله سنجی کلاسیک بررسی می گردد.

برای محاسبه ی فاصله از خودرو های اطراف در یک تصویر دو بعدی، ابتدا لازم است تا محدوده ای که فاصله در آن محاسبه می گردد از بالا و در واقع از نگاه چشم یک پرنده[۲۳۶] نگاه شود. در ادامه با داشتن نرخ تبدیل پیکسل به متر مناسب و محاسبه ی تعداد پیکسل ها، فاصله بر حسب متر به دست خواهد آمد.

مقدمات دیدن یک تصویر از بالا و از نگاه چشم پرنده، با مفهوم ماتریس تبدیل پرسپکتیو[۲۳۷] آغاز می شود. در واقع هدف از این بخش، آن است که به تصویری دست پیدا گردد که در آن، همان تصویری که توسط دوربین از رو به رو ضبط شده، از زاویه ی بالا قابل مشاهده باشد به نحوی که انگار دوربین دقیقا در بالای جاده قرار گرفته است. برای این کار، باید به ماتریس نگاشتی دست پیدا کرد که با ضرب آن در تک تک پیکسل های تصویر فعلی، پیکسل به محل جدید خود منتقل گردد و تصویری از بالای جاده به دست آید.

برای تبیین نحوه ی تعریف ماتریس نگاشت مطلوب یا همان ماتریس تبدیل پرسپکتیو، ابتدا تصویری از نقاشی معروف نقاش ایتالیایی، فرانچسکو دی جورجیو مارتینی بررسی می گردد[۲۷۰]:

[236] Bird Eye View
[237] Perspective Transform



شکل ۶۶-۳ اثر نقاشی نقاش ایتالیایی و مشاهده ی پرسپکتیو در آن [۲۷۰]

همان گونه که در تصویر بالا مشاهده می گردد، پرسپکتیو یا همان دور و نزدیک بودن اشیا و ساختمان ها به خوبی در تصویر بالا ترسیم گشته است. مشاهده می شود که تمامی خطوطی که از گوشه های نقاشی رسم شده، در یک نقطه ی واحد به هم برخورد می کنند. به این نقطه در مفاهیم مرتبط با پرسپکتیو و دور و نزدیکی اشیا در تصویر، نقطه ی محو شدن[۲۳۸] گفته می شود.

نکته ی جالب توجه دیگر در این نقاشی، توجه بی نظیر نقاش به مبانی پرسپکتیو می باشد که نمود آن را در طراحی کاشی های کف خیابان می بینیم که به صورت یک ذوزنقه با ضلع پایینی بزرگتر و ضلع بالایی کوچکتر نقاشی شده و اگر اضلاع جانبی آن را امتداد دهیم، به نقطه ی محو شدن می رسیم.

بنابراین با الهام از مبانی طراحی پرسپکتیو در هنر، هدف ما دقیقا یافتن نقطه ی محو شدن در تصویر و انتخاب یک محوطه ی ذوزنقه ای در جلوی خودرو می باشد که طبیعتا از بالا به شکل یک مربع یا مستطیل منتظم دیده می گردد[۲۷۱].

این ذوزنقه به شکل دلخواه انتخاب می گردد و محدودیتی در طول اضلاع افقی یا زوایای اضلاع جانبی آن وجود ندارد.

[238] vanishing point

در این الگوریتم، دو نقطه ی بالایی ذوزنقه از مربع کشیده شده دور خودرو از بخش شناسایی خودرو دریافت می شود و دو نقطه ی پایین ذوزنقه نیز دو نقطه ی از بخش جلویی خودرو انتخاب می گردد.

شکل ۶۷-۳ انتخاب ذوزنقه ی مناسب جهت اعمال تبدیل پرسپکتیو

اما جهت دریافت ماتریس پرسپکتیو و مشاهده ی ذوزنقه ی نشان داده شده از زاویه ی بالا، از الگوریتم دریافت تبدیل پرسپکتیو[۲۳۹] کتابخانه ی معروف اپن سی وی[۲۴۰] استفاده می شود.

این الگوریتم چهار نقطه ی ذوزنقه ی انتخاب شده و هم چنین چهار نقطه ای که قرار است این نقاط به آن منتقل شوند را دریافت می کند و ماتریس نگاشت مناسب برای این موضوع را ارائه می دهد.

نحوه ی ساخت این ماتریس توسط این الگوریتم بدین شکل می باشد که الگوریتم به دنبال نگاشتی می گردد تا نقطه ای با مختصات (u, v) را به نقطه ی جدید با مختصات (u_w, v_w) منتقل کند. از لحاظ روابط ماتریسی این مختصات جدید به شکل زیر به دست می آید:

$$\begin{bmatrix} u_w \\ v_w \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad (۳-۱۱)$$

[239] getPerspectiveTransform
[240] OpenCV

که در رابطه ی بالا، H ماتریس هموگرافی[241] می باشد که چیزی جز یک ماتریس ۳ در ۳ نیست که وظیفه ی انتقال و نگاشت نقاط مبدا به مقصد را به عهده دارد.

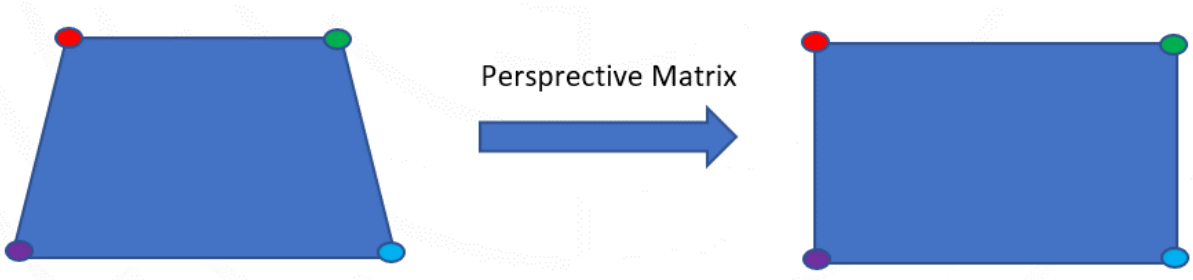
$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \qquad (۳-۱۲)$$

بدین ترتیب انتقال نقطه ای با مختصات (x_2, y_2) را به نقطه ی جدید با مختصات (x_1, y_1) توسط ماتریس هموگرافی به شکل زیر خواهد بود:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \qquad (۳-۱۳)$$

این همان الگوریتمی است که توسط دستور دریافت تبدیل پرسپکتیو کتابخانه ی OpenCV انجام می گردد و این ماتریس به عنوان خروجی ایجاد می شود.

در بخش های قبلی گفته شد که برای محاسبه ی این ماتریس به ۴ نقطه ی مبدا و ۴ نقطه ی مقصد احتیاج است و در مورد ۴ نقطه ی مبدا که نقاط گوشه ی ذوزنقه هستند نیز توضیحات کافی ارائه شد. حال کافیست تا ۴ نقطه ی مقصد، ۴ گوشه ی مربعی که قصد نمایش تصویر از بالا در آن داریم انتخاب گردد تا با مشخص شدن نقاط مبدا و مقصد، ماتریس تبدیل به دست آید.



شکل ۶۸-۳ نحوه تبدیل پرسپکتیو

[241] homography matrix

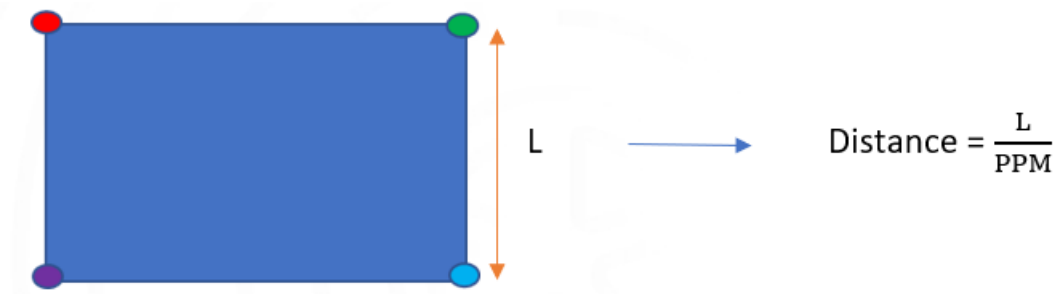
در تصویر بالا به خوبی مشاهده می گردد که ۴ نقطه‌ی مقصد، ۴ گوشه از فریم نمایش تصویر نهایی هستند و ۴ نقطه‌ی مبدا نیز ۴ گوشه ی ذوزنقه‌ی موجود در تصویر اولیه که توسط ماتریس پرسپکتیو به هم نگاشت می گردند [۲۷۱].

حال با داشتن ماتریس تبدیل پرسپکتیو و با استفاده از الگوریتم تصویر تاب خورده[۲۴۲]، تک تک پیکسل های موجود درون ذوزنقه ی ابتدایی را به مربع انتهایی نگاشت کنیم[۲۷۲].

این الگوریتم، با دریافت یک تصویر و ماتریس پرسپکتیو، تک تک نقاط تصویر را تحت ماتریس داده شده به فضای جدید نگاشت می کند و تصویر جدید را به عنوان خروجی ارائه می دهد.

در نهایت کافی است تا طول مربع نهایی را بر حسب پیکسل به دست آوریم و بر یک نرخ پیکسل بر متر مناسب تقسیم کنیم تا مقدار فاصله بر حسب متر به دست آید.

نرخ پیکسل بر متر، بسته به زاویه و نوع دوربین مورد استفاده، مقداری بین ۴۰ تا ۷۰ خواهد بود که با صحیح و خطا به راحتی قابل محاسبه می باشد.



شکل ۶۹-۳ نحوه محاسبه فاصله

در این قسمت، توضیحات تئوری و ریاضیاتی محاسبه ی فاصله به روش کلایسک به پایان می رسد و در بخش بعد، نحوه ی پیاده سازی این مورد در برنامه، به تفصیل شرح داده می شود.

[242] warpPerspective

۲-۱-۴-۳- تشریح نحوه پیاده سازی الگوریتم، ساختار کد و ارائه ی نتایج

در این قسمت، به تشریح این موضوع پرداخته می شود که توضیحات تئوری و ریاضیات تبیین شده در بخش قبلی چگونه تبدیل به الگوریتم و در نهایت کدهای کامپیوتری می گردند.

تکه کد زیر، نحوه ی محاسبه ی ماتریس پرسپکتیو و به دست آوردن تصویر انتقال یافته به دید از بالا را بیان می کند:

```
1.  def Classic_Distance(image, pts):
2.      pts = pts.astype(dtype = "float32")
3.      (tl, tr, br, bl) = pts
4.      widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
5.      widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
6.      maxWidth = max(int(widthA), int(widthB))
7.      heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
8.      heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
9.      maxHeight = max(int(heightA), int(heightB))
10.     dst = np.array([
11.         [0, 0],
12.         [maxWidth - 1, 0],
13.         [maxWidth - 1, maxHeight - 1],
14.         [0, maxHeight - 1]], dtype = "float32")
15.     M = cv2.getPerspectiveTransform(pts, dst)
16.     warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
17.     return warped
```

در قطعه کد بالا، تابع `Classic_Distance` مشاهده می گردد که تصویر و ۴ نقطه ی ذوزنقه را به عنوان ورودی دریافت می کند و تصویر دید از بالا را به عنوان خروجی برمی گرداند.

در خط دوم و سوم، نقاط دریافت شده از ورودی تابع درون ۴ متغیر مجزا ذخیره می گردد. این متغیر ها به ترتیب بیانگر نقطه ی بالا چپ یا top left، بالا راست یا top right ، پایین راست یا bottom right و پایین چپ یا bottom left می باشند.

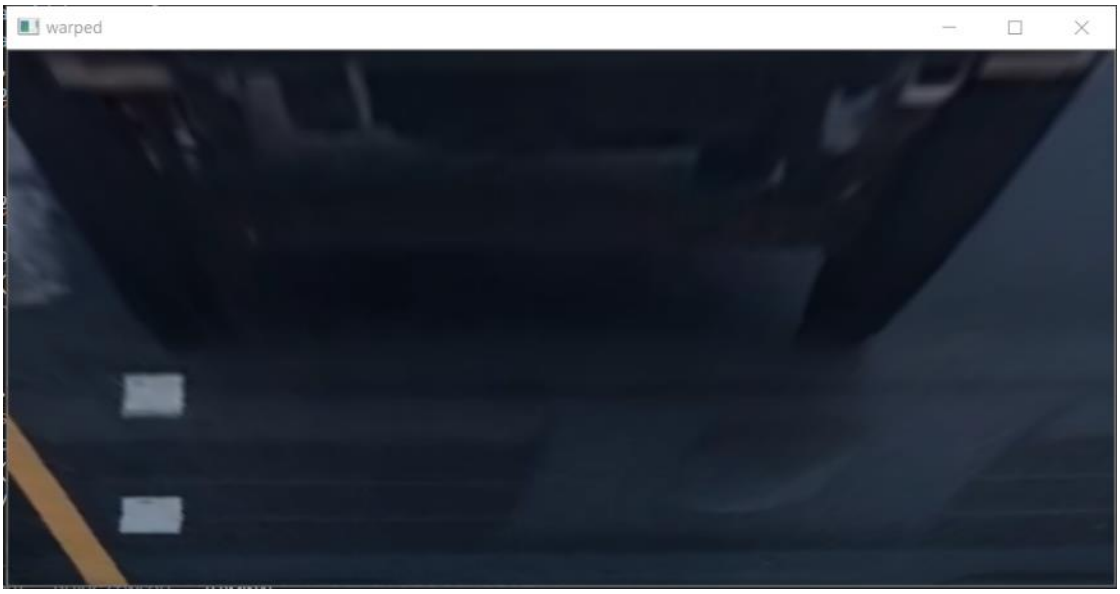
در خط چهارم تا نهم، ماکزیمم طول و عرض این ذوزنقه محاسبه می گردد. این محاسبات بدین صورت انجام می گردد که برای هر کدام از دو مختصات x و y برای هر ۴ نقطه، طول و عرض به صورت دو به دو و به شکل اقلیدسی محاسبه می گردد و در نهایت ماکزیمم آن ها محاسبه می گردد. عرض ها از محاسبه ی فاصله ی جفت جفت بین نقاط بالا و پایین و طول ها از محاسبه ی فاصله ی جفت جفت بین نقاط راست و چپ به دست می آیند.

در ادامه و در خطوط دهم تا چهاردهم، ۴ نقطه ی مقصد تعریف می شود که این نقاط، دقیقا همان ابعاد ذوزنقه ی مبدا می باشند. در خط پانزدهم، ماتریس پرسپکتیو بر اساس نقاط مبدا و مقصد محاسبه می گردد و در نهایت در خط شانزدهم، تصویر دید از بالا بر اساس ماتریس به دست آمده استخراج می گردد.

خروجی های مربوط به این قطعه کد به شکل زیر می باشند:

شکل ۷۰-۳ تصویر ورودی قطعه کد مربوط به بخش به دست آوردن تصویر از بالا

شکل ۷۱-۳ ذوزنقه‌ی انتخاب شده بر اساس خروجی بخش شناسایی خودرو



شکل ۷۲-۳ تصویر ترنسفورم شده به دید از بالا

همچنین ماتریس پرسپکتیو برای این تصویر به مقدار زیر محاسبه گشته است:

```
Distance Matrix :
 [[-8.32318608e-01 -7.81976757e-01  9.19165014e+02]
 [-1.02052817e-16 -1.26688066e+00  7.25922616e+02]
 [-3.20835936e-19 -2.23311289e-03  1.00000000e+00]]
```

در ادامه قطعه کد زیر اجرا می گردد:

```
image = cv2.imread(frame)
pts = np.array([(566,573),(828,573),(1114,821),(333,821)])
warped = Clasic_Distance(image, pts)
ppm=100
distance = warped[0].shape[0]/ppm
font = cv2.FONT_HERSHEY_SIMPLEX
char = "Distance ="+str(round(distance,2))
cv2.putText(image, char, (400,100), font, 3, (0, 0, 255), 2, cv2.LINE_AA)
cv2.fillPoly(image, pts =[pts], color=(0,255,0))
cv2.imshow("Original", image)
cv2.waitKey(0)
```

در قطعه کد بالا، در خط اول که تصویر خوانده می شود.

در خط دوم، نقاط ذوزنقه‌ی ابتدایی به شکل دستی به کد داده شده. در این بخش این نکته قابل توجه است که فرآیند دریافت این ۴ نقطه در کد اصلی به صورت اتوماتیک انجام می شود بدین صورت که ۲ نقطه ی بالایی از خروجی بخش تشخیص خودرو دریافت می شود و ۲ نقطه ی پایینی نیز همیشه ثابت هستند و از مختصات

خودرویی که دوربین روی آن نصب است برداشته می شود. اما برای تست قطعه کد به شکل مجزا، این نقاط به صورت دستی به تابع داده می شود.

در خط سوم، تابع روی تصویر اعمال می شود و پس از محاسبه ی ماتریس پرسپکتیو، تصویر ترنسفورم شده به دید از بالا به عنوان خروجی برگردانده می شود که درون حافظه ذخیره می گردد.

در خط چهارم نرخ تبدیل پیکسل به متر تعریف شده که با آزمون خطا، این مقدار ۱۰۰ به دست آمده است.

در خط پنجم تا هشتم، فاصله تا خودرو بر اساس طول پیکسلی فاصله از تصویر دید از بالا و تبدیل آن به متر با نرخ تبدیل پیکسل به متر، محاسبه شده و روی تصویر درج شده است.

در خط نهم، ذوزنقه ی استفاده شده برای محاسبه ی ماتریس پرسپکتیو نیز ترسیم شده که تنها برای ایجاد شهود بهتر می باشد و این قسمت در کد اصلی حذف شده است.

و در نهایت در خطوط ده و یازده، تصویر پردازش شده به نمایش گذاشته شده است.

در ادامه تعدادی از خروجی های این قطعه کد ارائه می گردد:

شکل ۳-۷۳ خروجی اول قطعه کد محاسبه فاصله به روش کلاسیک

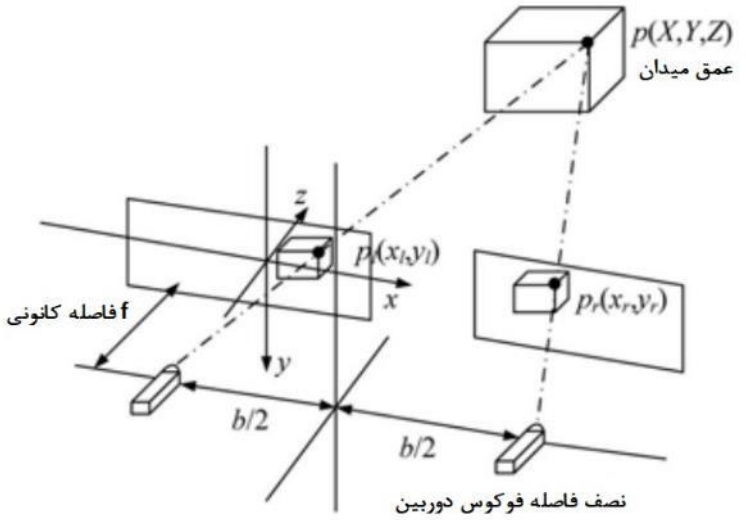
شکل ۳-۷۴ خروجی دوم قطعه کد محاسبه فاصله به روش کلاسیک

شکل ۳-۷۵ خروجی سوم قطعه کد محاسبه فاصله به روش کلاسیک

۲-۴-۳- فاصله سنج با رویکرد شبکه‌های عصبی

۱-۲-۴-۳- تبیین روش مورد نظر

ابتدا چشم چپ خود را با دست راست نگه دارید و نقطه ای در نظر بگیرید، حال چشم چپ خود را ببندید و آن نقطه را با چشم راست خود ببینید، اختلافی میان نقطه‌ی دیده شده با چشم چپ و راست وجود دارد که به آن دیسپریتی[۲۴۳] گفته می‌شود. با استفاده از این مفهوم می‌توان درکی از عمق داشت و فاصله اجسام را نسبت به خود محاسبه کرد. یکی از روش های ادراک بصری، استفاده از سیستم بینایی دو چشمی یا استریو[۲۴۴] است که براساس هندسه استریو، اطلاعات سه بعدی محیط را در اختیار سیستم قرار می‌دهد. نمایی از این سیستم بینایی در شکل ۳-۷۶ نشان داده شده است.



شکل ۳-۷۶ نمای کلی از سیستم استریو

سیستم بینایی دو چشمی برای بدست آوردن اطلاعات سه بعدی از چشم انسان تقلید می کند. این یک روش مناسب و مهم در زمینه دید رایانه است. همانطور که در شکل ۳-۷۶ نشان داده شده است، دو دوربین به صورت افقی قرار گرفته‌اند. $p_R(x_R, y_R), p_L(x_L, y_L)$ مختصات پیکسل های تصویر در دوربین های چپ و راست هستند. با استفاده از تصاویر چپ و راست، نقشه عمق تصویر و در نهایت ابر نقاط به دست می آید. $p(X, Y, Z)$ مختصات سه بعدی نقاط محیط در دستگاه مختصات دوربین است. با این روش می توان ابر نقاط محیط را بدست آورد و با استفاده از ماتریس تبدیل مختصات، دستگاه مختصات نقاط را تغییر داد. از دیگر روش‌های موجود برای

[243] Disparity
[244] Stereo

دستیابی به درکی از عمق، سنسور لیدار است اما مزیت استفاده از سیستم دید دوچشمی (استریو) ارزان بودن آن نسبت به این حسگر گران قیمت است.

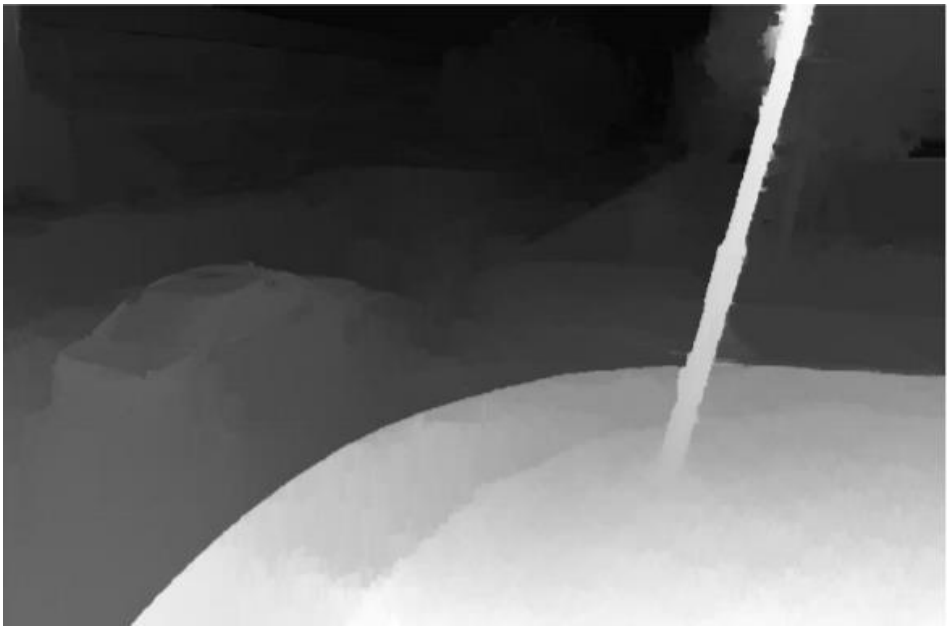
دوربین‌های استریو با الگو گیری از چشم انسان ساخته شده‌اند. بدین صورت که یک تصویر سمت چپ و یک تصویر سمت راست به ما می دهند که می توان اطلاعات بسیار زیادی را از این تصاویر استخراج کرد. نمونه ای از دوربین استریو مربوط به شرکت Zed در تصویر ۷۷-۳ آورده شده است.

شکل ۷۷-۳ دوربین استریو Zed

دوربین استریو مانند چشم انسان از دو دوربین چپ و راست تشکیل شده است. نکته مهم در استفاده از دو دوربین در کاربرد های استریو همزمانی آن ها می باشد تا محاسبات دیسپریتی تحت تاثیر حرکت اجسام قرار نگیرد. دو تصویر آورده شده در شکل ۷۸-۳، تصویر های چپ و راست دوربین Zed می‌باشند.

شکل ۷۸-۳، تصاویر چپ و راست حاصل شده از دوربین Zed

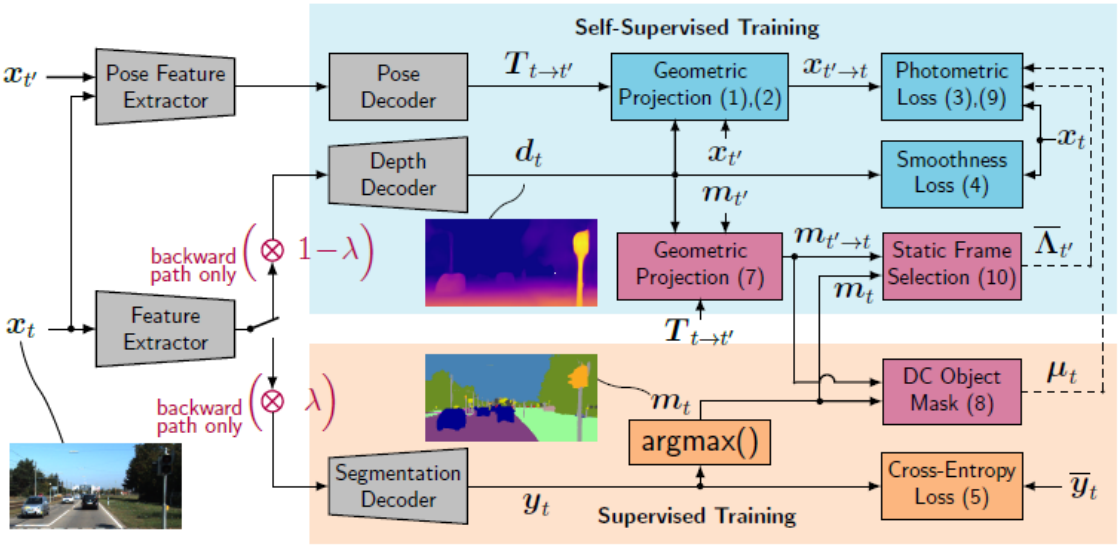
در تصویر ۷۹-۳ دیسپریتی محاسبه شده از روی دو تصویر چپ و راست تصویر ۷۸-۳ با الگوریتم SGBM آورده شده است. همانطور که مشخص است، هر چه اجسام به دوربین نزدیکتر باشند مقدار دیسپریتی بزرگتر خواهد بود و در عین حال رنگ آن به سفید نزدیکتر می‌شود. الگوریتم SGBM بدین صورت کار می‌کند که بلوک هایی با ابعاد مشخص در تصویر سمت سمت چپ، بلوک های متناظر با خود را در تصویر سمت راست پیدا می‌کند و مقدار دیسپریتی تصویر با توجه به تعداد پیکسل هایی که بلوک در نظر گرفته شده در تصویر سمت چپ و راست جابجا شده است به دست می آید.



شکل ۷۹-۳ دیسپریتی حاصل شده از دو تصویر چپ و راست دوربین استریو Zed با الگوریتم SGBM

حال برآورد دقیق اطلاعات عمق از یک صحنه برای کاربردهایی که به یک مدل محیط سه بعدی مانند رانندگی خودکار نیاز دارند، ضروری است. الگوریتم های مبتنی بر مدل کلاسیک می توانند عمق را از تصاویر استریو یا از توالی های تصویر (فیلم ها) پیش بینی کنند، که به کیفیت مدل محدود می شود.

با توجه به اینکه در این پروژه اطلاعات سنسوری و دوربین استریو در دسترس نیست و می‌بایست فاصله را با استفاده از تصاویر دو بعدی بدست آورد، ایده جدیدی بر گرفته از مقاله SGDepth [۲۷۴] برای محاسبه عمق در پیش گرفتیم. در این مقاله روشی با رویکرد نظارت بر خود[۲۴۵] ارائه شده که صرفاً به مدل های تجسم هندسی تصویر متکی است و با به حداقل رساندن خطاهای فوتومتریک و بدون نیاز به برچسب، عمق را بهینه می کند. این روشهای تخمین عمق یک چشمی که با رویکرد نظارت بر خود عمل می‌کنند، فقط به یک تصویر واحد RGB به عنوان ورودی احتیاج دارند و به تصویر چپ و راست و یا هرگونه سنسور خارجی نیاز ندارند.



شکل ۸۰-۳ ساختار شبکه SGDepth [۲۷۴]

ساختار شبکه SGDepth که در تصویر ۸۰-۳ آورده شده است، از سه رویکرد استفاده می‌کند:

۱- رویکرد خود نظارتی در قسمت بدست آوردن عمق، با بلوک‌های آبی مشخص شده است.

۲- رویکرد با نظارت در قسمت یافتن تقسیم‌بندی معنایی تصویر، با بلوک های نارنجی مشخص شده است.

۳- نحوه ارتباط این دو قسمت با یکدیگر، با بلوک‌های قرمز مشخص شده است.

با توجه به تصویر ۸۰-۳، ابتدا ماتریس نگاشت پیکسل‌های تصویر ورودی در زمان t به زمان t' توسط بلوک Pose Encoder بدست می‌آید. در این بلوک موقعیت نسبی در زمان t' نسبت به t توسط یک ماتریس تبدیل تعیین

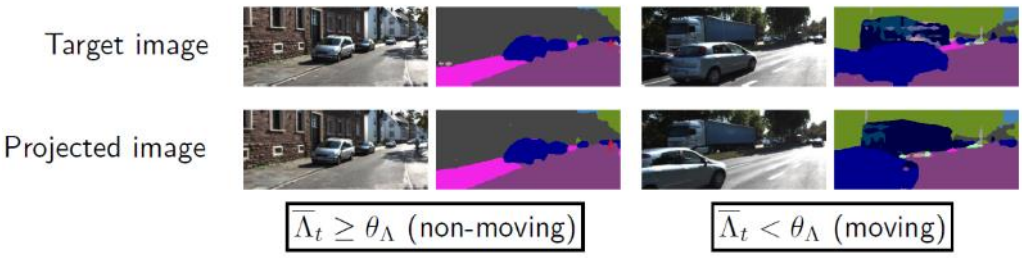
[245] Self-supervised

می‌شود. به منظور یافتن مختصات جدید پیکسل‌ها در زمان t' از روی پیکسل های زمان t به مختصات $u_{t,i} = (h_i;w_i; 1)^T$، می‌بایست ابتدا تصویر موجود در زمان t را بوسیله بلوک Geometric Projection به فضای سه بعدی نگاشت کرد و سپس با اعمال تبدیل $T_{t\to t'}$ مختصات تغییر یافته سیستم بدست می‌آید و در نهایت می‌بایست پیکسل‌های تغییر یافته در فضای سه بعدی را به فضای دو بعدی نگاشت کرد که این‌کار توسط یک Reprojection Loss که یک تابع هزینه به منظور کمینه کردن خطای نگاشت به فضای دو بعدی است، انجام می‌شود.

$$\boldsymbol{u}_{t\to t',i} = \underbrace{[\boldsymbol{K}|\boldsymbol{0}]\,\boldsymbol{T}_{t\to t'}}_{\substack{\text{transformation}\\ \text{to frame } t'}} \underbrace{\left[\begin{pmatrix} d_{t,i}\boldsymbol{K}^{-1}\boldsymbol{u}_{t,i} \\ 1 \end{pmatrix}\right]}_{\substack{\text{projection to}\\ \text{3D point cloud}}} \qquad (۳-۱۴)$$

مختصات جدید پیکسل‌ها همانطور که توضیح داده شد، با فرمول بالا بیان می‌شود که در آن K مختصات ثابت دوربین دیتاست تحت آموزش می‌باشد. در بلوک Depth decoder عمق تصویر تخمین زده می‌شود و با ترکیب آن با نتایج حاصل از بلوک Geometric Projection سعی در بهبود این تخمین دارد. علاوه بر آن بلوک Smoothness Loss وظیفه دارد پیکسل‌هایی که در نزدیک یکدیگر هستند را ترغیب کند تا عمق مشابه داشته باشند. بعد از انجام مراحل گفته شده تصویرِ Disparity حاصل می‌شود.

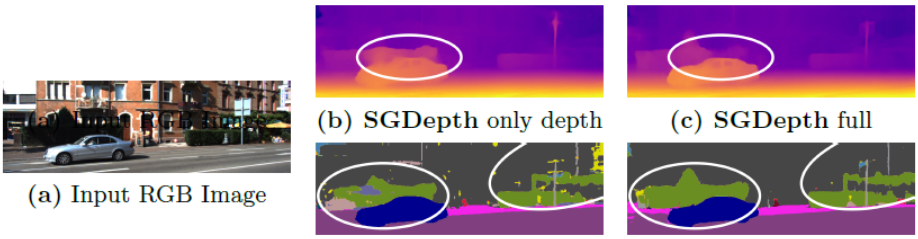
ایده جدیدی که در این مقاله معرفی شد، بهبود نتایج Disparity توسط تقسیم‌بندی معنایی است. با استفاده از تقسیم بندی معنایی سعی می‌کنند اشیایی که در تصویر در حال حرکت هستند و ثابت هستند را تشخیص دهند. بدین صورت که اگر در هنگام آموزش IoU قسمت تقسیم‌بندی شده در تصویر $t-1$ از تصویر t از یک حد معین بیشتر باشد، شی به عنوان یکی شی در حال حرکت شناسایی می‌شود. نمونه‌ای از قسمت تقسیم‌بندی معنایی توضیح داده شده در تصویر ۳-۸۱ آورده شده است.



شکل ۳-۸۱ تقسیم‌بندی معنایی تصویر به منظور تشخیص اشیا در حال حرکت [۲۷۴]

در نهایت با تشخیص اشیا در حال حرکت و ثابت و تاثیر دو بخش تقسیم‌بندی شده و دیسپریتی بر روی یکدیگر،

باعث می‌شود دقت هر دو بخش همانطور که در تصویر ۸۲-۳ نشان داده شده است بهبود یابد.



(a) Input RGB Image

(b) **SGDepth** only depth

(c) **SGDepth** full

(d) **SGDepth** only seg.

(e) **SGDepth** full

شکل ۸۲-۳ (a) تصویر ورودی، (b) نتیجه تصویر دیسپریتی به تنهایی، (c) تصویر دیسپریتی بعد از انجام آموزش کامل، (d) نتیجه تصویر تقسیم‌بندی شده به تنهایی، (e) تصویر تقسیم‌بندی شده بعد از انجام آموزش کامل[۲۷۴].

۲-۲-۴-۳- چگونگی آموزش

در این بخش نحوه آموزش شبکه SGDepth بر روی دو دیتاست Kitti و Cityscapes توضیح داده شده است.

قبل از شروع آموزش شبکه، ابتدا مجموعه کد SGDepth را از آدرس گیت‌هاب[۲۷۵] به صورت زیر دانلود میکنیم.

```
1. $ https://github.com/ifnspaml/SGDepth.git
```

پکیج‌های مورد نیاز جهت نصب:

- $numpy \geq 1.18.1$
- $pillow \geq 6.1.0$
- $torchvision \geq 0.3.0$
- $tensorboardx \geq 2.0$
- $matplotlib \geq 3.1.3$
- $pyyaml \geq 5.3$
- $cudatoolkit = 10.0$
- $cudnn = 7.5.1$
- $pytorch == 1.1.0$

با دستور مقابل به پوشه پروژه رفته:

```
1. $ cd SGDepth
```

سپس مراحل زیر را به ترتیب انجام دهید:

۱- دستور مقابل را اجرا کنید:

```
$ export IFN_DIR_CHECKPOINT=/path/to/folder/Checkpoints
```

۲- دیتاست Cityscapes را از سایت **https://www.cityscapes-dataset.com/** دانلود کنید و در پوشه‌ای به نام "Dataset" قرار دهید.

۳- دیتاست Kitti را از سایت **http://www.cvlibs.net/datasets/kitti/** دانلود کنید و در همان پوشه "Dataset" قرار دهید. به منظور اطمینان جهت یکسان بودن ساختار دیتاست با سورس اصلی نویسنده، دستور مقابل را اجرا کنید:

```
$ python dataloader\data_preprocessing\download_kitti.py
```

۴- پوشه "Dataset" را به صورت مقابل آماده کنید:

- ابتدا دستور مقابل را اجرا کنید:

```
$ export IFN_DIR_DATASET=/path/to/folder/Dataset
```

- فایل های **json** را دانلود کرده و در پوشه "cityscapes" قرار دهید.
- فایل های **json** را دانلود کرده و در پوشه "kitti" قرار دهید.
- فایل های **json** را دانلود کرده و در پوشه "kitti_zhou_split" قرار دهید.
- فایل های **json** را دانلود کرده و در پوشه "kitti_kitti_split" قرار دهید.
- فایل های **json** را دانلود کرده و در پوشه "kitti_2015" قرار دهید.

۵- در نهایت با اجرای کد مقابل روند آموزش شروع می‌شود:

```
python3 train.py \
        --model-name zhou_full \
        --depth-training-loaders "kitti_zhou_train" \
        --train-batches-per-epoch 7293 \
        --masking-enable \
        --masking-from-epoch 15 \
        --masking-linear-increase
```

۳-۲-۴-۳- نتیجه آموزش و خروجی شبکه

دیتاست های Kitti و Cityscapes بسیار حجیم بوده و آموزش دوباره آن‌ها نیازمند سخت افزار مناسب و زمان کافی می‌باشد و آموزش دوباره این دیتاست‌ها صرفا باز تولید نتایج است. به همین منظور ما از مدل‌های از پیش آموزش داده شده که در سایت گیت‌هاب شبکه SGDepth قرار گرفته است، استفاده کردیم.

نتایج حاصل از آموزش شبکه SGDepth در تصویر ۸۳-۳ آورده شده است و بهترین نتایج به صورت پررنگ مشخص شده است.

- منظور از K دیتاست Kitti می‌باشد.
- منظور از (CS) دیتاست Cityscapes می‌باشد.

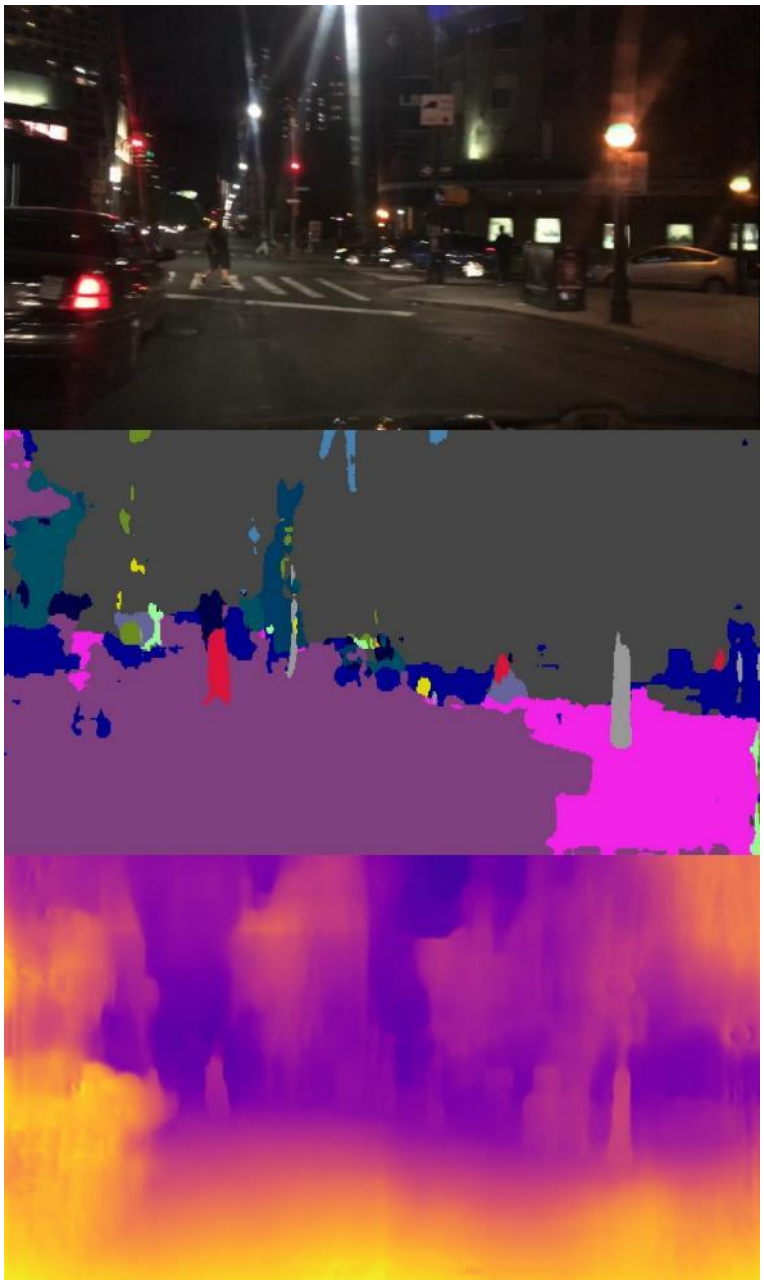
Method	Resolution	Dataset	Lower is better				Higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
SGDepth only depth	640×192	K	0.117	0.907	4.844	0.196	0.875	0.958	0.980
SGDepth add multi-task training	640×192	(CS) + K	0.117	0.918	4.777	0.193	0.872	0.960	**0.982**
SGDepth add scaled gradients	640×192	(CS) + K	**0.113**	**0.817**	**4.671**	0.191	0.877	**0.961**	**0.982**
SGDepth add semantic mask	640×192	(CS) + K	0.116	0.917	4.726	**0.189**	0.874	**0.961**	**0.982**
SGDepth add threshold	640×192	(CS) + K	**0.113**	0.861	4.724	0.191	**0.879**	0.960	0.981
SGDepth full	640×192	(CS) + K	**0.113**	0.835	4.693	0.191	**0.879**	**0.961**	0.981

شکل ۸۳-۳ نتایج شبکه SGDepth آموزش داده شده بر روی دو دیتاست Kitti و Cityscapes [۲۷۴]

در تصویر ۸۴-۳، تصویر تست ویدئو ۱ و نتایج خروجی دیسپریتی و دسته‌بندی معنایی حاصل از شبکه SGDepth را مشاهده می‌کنید:

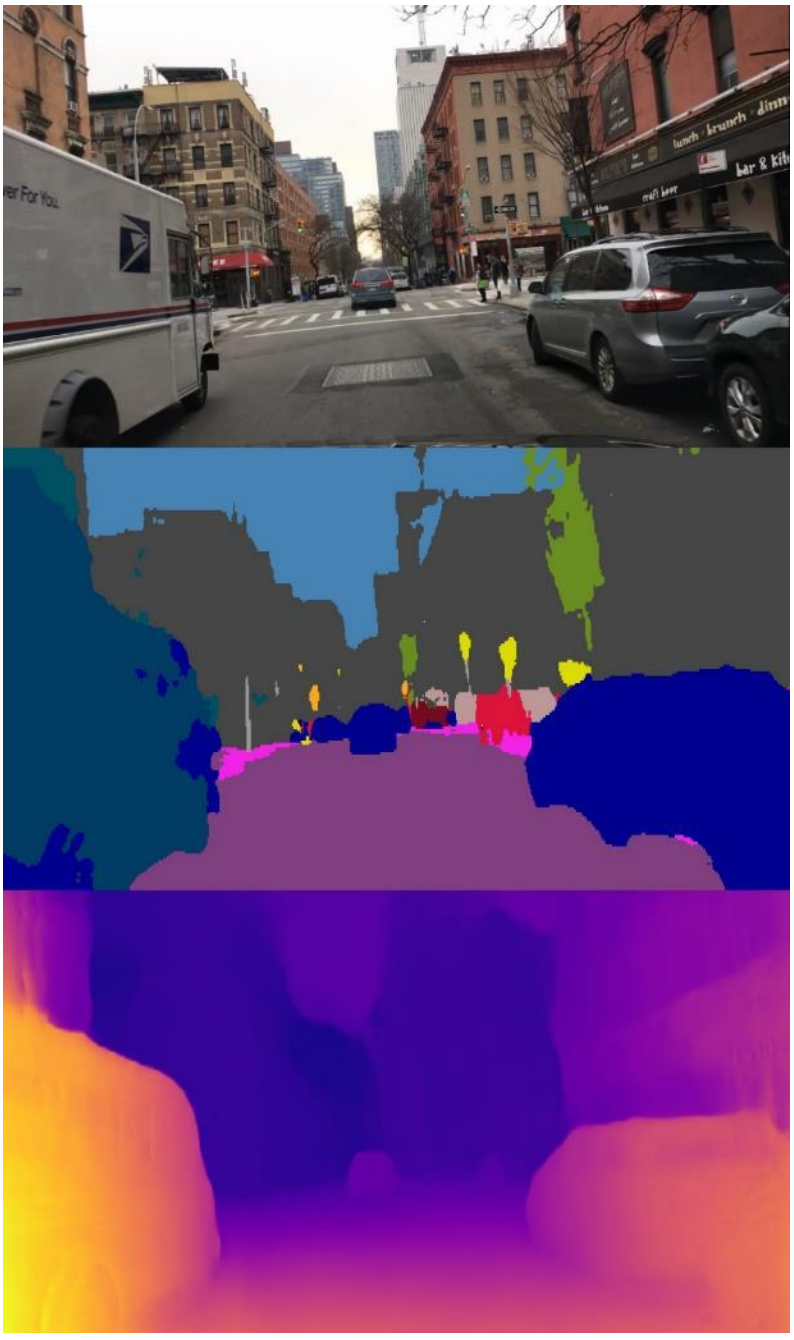
شکل ۸۴-۳ نتایج خروجی دیسپریتی و دسته بندی معنایی ویدئو تست ۱

در تصویر ۸۵-۳، تصویر تست ویدئو ۲ و نتایج خروجی دیسپریتی و دسته‌بندی معنایی حاصل از شبکه SGDepth را مشاهده می‌کنید:



شکل ۸۵-۳ نتایج خروجی دیسپریتی و دسته بندی معنایی ویدئو تست ۲

در تصویر ۸۶-۳، تصویر تست ویدئو ۳ و نتایج خروجی دیسپریتی و دسته‌بندی معنایی حاصل از شبکه SGDepth را مشاهده می‌کنید:



شکل ۸۶-۳ نتایج خروجی دیسپریتی و دسته بندی معنایی ویدئو تست۳

تصویر ۸۷-۳، هندسه دوربین استریو را نشان می‌دهد که در آن:

C1: دوربین سمت چپ.

C2: دوربین سمت راست.

Baseline (b): به فاصله دوربین چپ از راست گفته می‌شود.

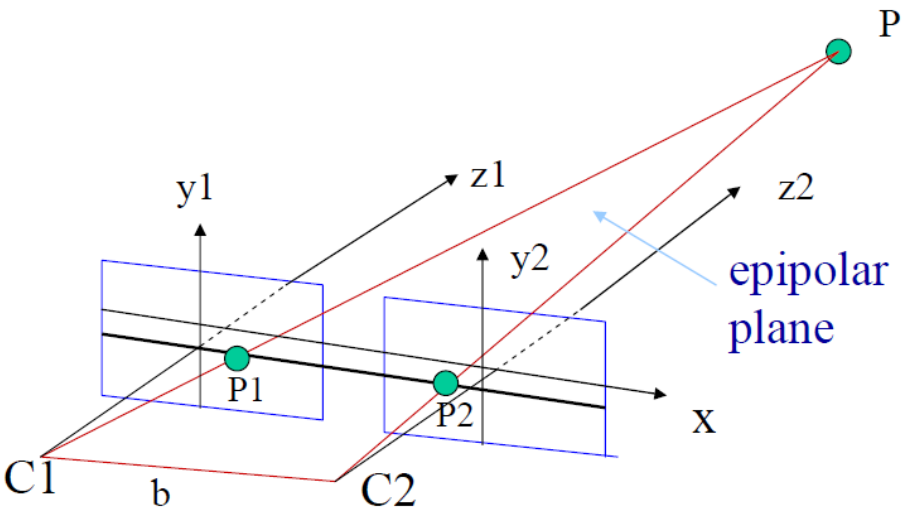
P1: مختصات نقطه مورد نظر در تصویر سمت چپ.

P2: مختصات نقطه مورد نظر در تصویر سمت راست.

P: مختصات نقطه مورد نظر در فضای سه بعدی.

Disparity (d): به فاصله P1 و P2 از یکدیگر گفته می‌شود.

Focal length (f): به فاصله کانونی دوربین گفته می‌شود.



شکل ۸۷-۳ هندسه دوربین استریو [۲۷۶]

با استفاده از هندسه دوربین استریو که در تصویر ۸۷-۳ آورده شد، می‌توان مختصات سه بعدی نقطه مورد نظر را به صورت مقابل محاسبه کرد:

$$Depth: \quad Z = \frac{f \times b}{d}$$
$$X = \frac{X_l \times z}{f}$$
$$\mathrm{Y} = \frac{Y_l \times z}{f} \qquad (۳-۱۵)$$

برای بدست آوردن مقادیر فوق نیازمند داشتن مشخصات دوربین دیتاستی که روی آن آموزش انجام شده است هستیم. مشخصات دوربین دیتاست Kitti و Cityscapes در جدول ۱۵-۳ آورده شده است.

جدول ۱۵-۳ مشخصات دوربین Kitti و Cityscapes

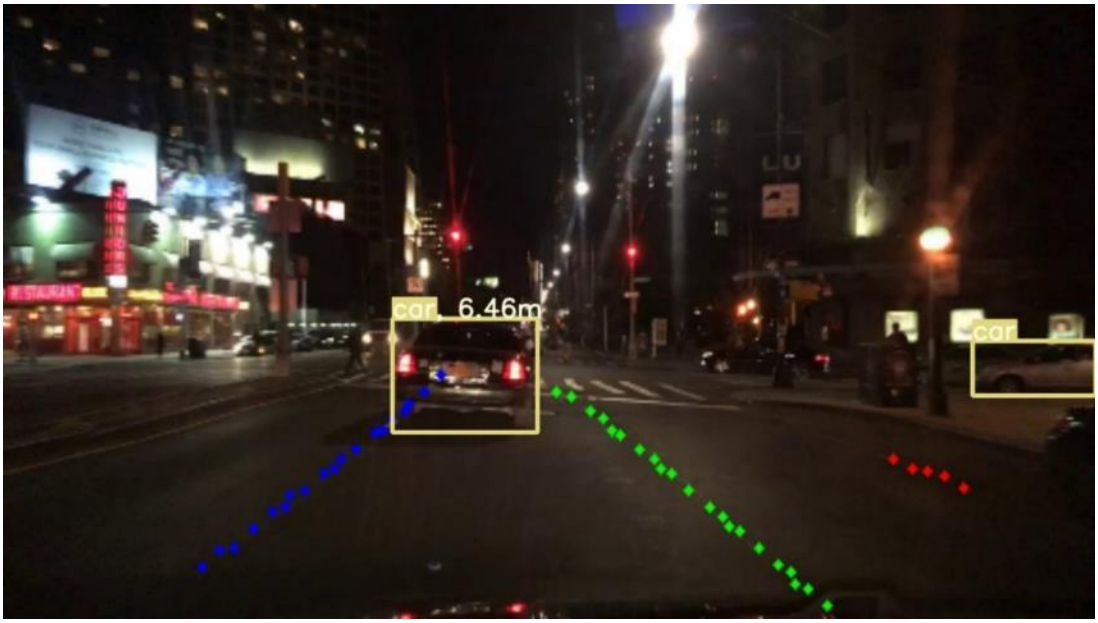
	فاصله کانونی (f)	baseline
Kitti	654.24 (pixel)	**0.5707 (m)**
Cityscapes	2262 (pixel)	**0.22 (m)**

برای محاسبه فاصله می‌بایست مشخصات دوربین آخرین دیتاستی که روی آن آموزش انجام شده است در نظر گرفت. به همین دلیل دیتاست Kitti را معیار اندازه‌گیری فاصله در نظر می‌گیریم.

در تصویر ۸۸-۳، نتیجه خروجی اندازه گیری فاصله نسبت به خودروهای دیگر بر روی تصویر تست ویدئو ۱ آورده شده:

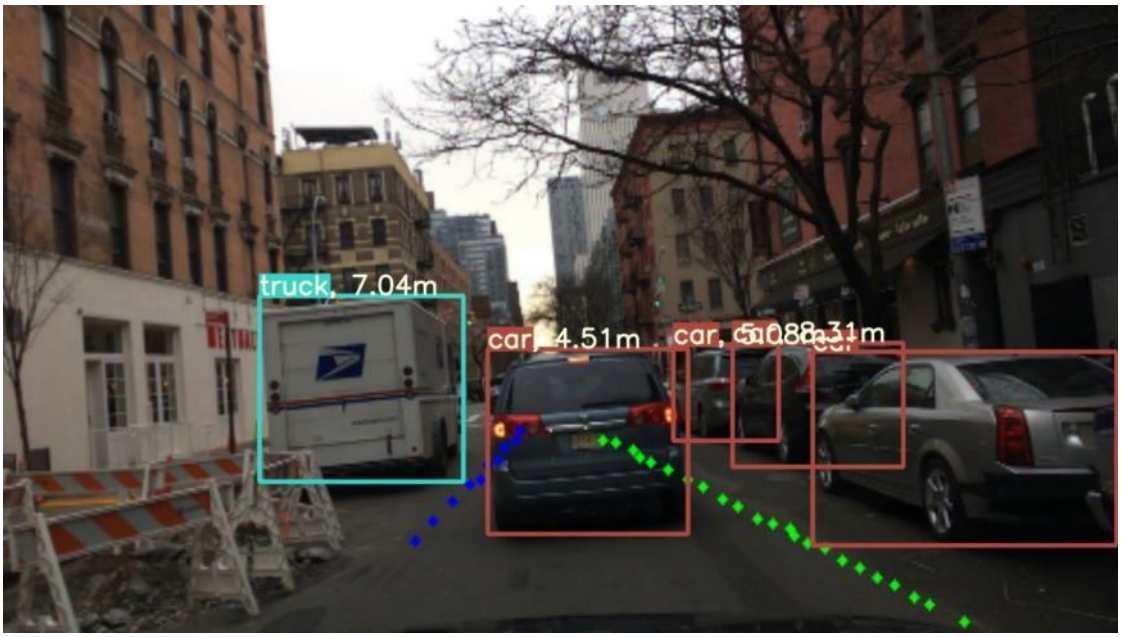
شکل ۸۸-۳ نتیجه خروجی اول اندازه گیری فاصله نسبت به خودروهای دیگر

در تصویر ۸۹-۳، نتیجه خروجی اندازه گیری فاصله نسبت به خودروهای دیگر بر روی تصویر تست ویدئو ۲ آورده شده:



شکل ۸۹-۳ نتیجه خروجی دوم اندازه گیری فاصله نسبت به خودروهای دیگر

در تصویر ۹۰-۳، نتیجه خروجی اندازه گیری فاصله نسبت به خودروهای دیگر بر روی تصویر تست ویدئو ۳ آورده شده:



شکل ۹۰-۳ نتیجه خروجی سوم اندازه گیری فاصله نسبت به خودروهای دیگر

خروجی نتایج فوق با شرط اندازه‌گیری فاصله کمتر از ۱۰ متر و ROI یا منطقه محدود کننده محدود به جاده جلویی خودرو بدست آمده است.

۴-۲-۴-۳- تشریح نحوه پیاده سازی الگوریتم و ساختار کد

در کد توسعه داده شده، ماژول SGD.py که در پوشه elements قرار دارد، حاوی کلاس Inference است که وظیفه بدست آوردن تصویر دیسپریتی و تقسیم‌بندی را بر عهده دارد. در main.py ابتدا این کلاس به ساختار اصلی اضافه شده و یک شیء[۲۴۶] از این کلاس ساخته می‌شود.

```
1. disparity_detector = Inference(opt.disp_detector)
```

با فراخوانی تابع inference که در SGD.py است، کار بدست آوردن دیسپریتی و تقسیم‌بندی در هر فریم انجام می‌گیرد.

ساختار کد SGD.py در ادامه به صورت کامل توضیح داده شده است.

ابتدا کتابخانه های مورد نیاز را ایموپرت [۲۴۷]یا وارد می کنیم.

```
1.  from elements.yolo import YOLO, YOLO_Sign
2.  from elements.PINet import LaneDetection
3.  from elements.SGD import Inference
4.  from elements.Curvlane import CurveLane
5.  from elements.asset import cityscape_xyz, kitti_xyz, apply_mask, ROI, kitti_xyz_dist, city
    scape_xyz_dist
6.  from utils.plots import plot_one_box
7.  import matplotlib.pyplot as plt
8.  from elements.asset import horiz_lines, detect_lines
9.  import numpy as np
10. import os
11. import cv2
12. from time import time as t
13. import datetime
14. import random
15. import sys
16. from datetime import timedelta
17. from SGDepth.arguments import InferenceEvaluationArguments
```

با فراخوانی کلاس InferenceEvaluationArguments، پارامترهای مورد نیاز فراخوانی می‌شود.

```
1.  opt = InferenceEvaluationArguments().parse()
```

[246] Object
[247] import

یکی از ویژگی های مهم این نرم‌افزار، نوشتن کدها بر پایه منطق شی‌گرایی است. در برنامه‌نویسی شی‌گرا ویژگی های مربوط به هر شی در محدوده و کلاس مربوط به خود شی قرار می گیرد. اشیا دیگر قدرت دسترسی و ایجاد تغییرات در داده های کلاس را ندارند و فقط می توانند به لیستی از توابع کلاس که بصورت عمومی تعریف شده‌اند دسترسی داشته باشند. این ویژگی در برنامه‌نویسی شی گرا باعث بالا رفتن امنیت و جلوگیری از فساد ناخواسته اطلاعات شده است.

در کلاس Inference، ساختار توابع به صورت مقابل نوشته شده است:

```
class Inference:
    def __init__(self, model_path):
    def load_image(self, frame):
    def normalize(self, tensor):
    def inference(self, frame):
    def save_pred_to_disk(self, segs_pred, depth_pred):
    def scale_depth(self, disp):
```

ابتدا در قسمت __init__، مدل را Load می‌کنیم. این تابع یک ورودی به عنوان آدرس مدل می‌گیرد. ویژگی این قسمت این است که فقط یک‌بار اجرا می‌شود و کار Load کردن مدل در هر فریم انجام نمی‌شود.

فرآیند اصلی کار در تابع inference انجام می‌شود.

```
def inference(self, frame):

    # load image and transform it in necessary batch format
    self.load_image(frame)

    with torch.no_grad():
        output = self.model(self.batch) # forward pictures

    disps_pred = output[0]["disp", 0] # depth results
    segs_pred = output[0]['segmentation_logits', 0] # seg results

    segs_pred = segs_pred.exp().cpu()
    segs_pred = segs_pred.numpy()  # transform preds to np array
    segs_pred = segs_pred.argmax(1)  # get the highest score for classes per pixel

    depth_pred, seg_img = self.save_pred_to_disk(segs_pred, disps_pred)

    return depth_pred, seg_img
```

در این تابع،

ورودی:

- frame: تصویر ورودی

خروجی:

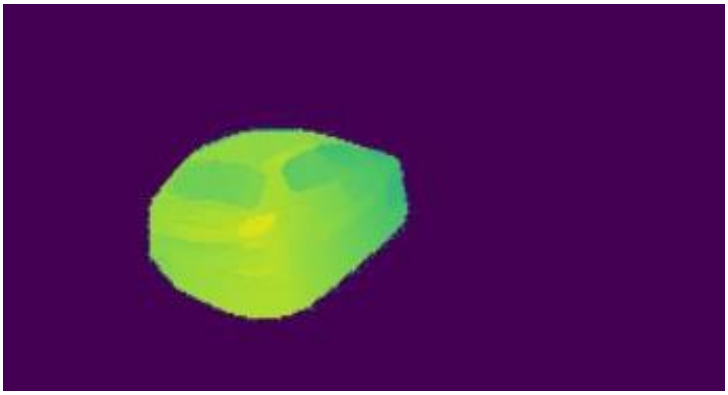
- Seg_img: خروجی تصویر تقسیم‌بندی شده
- Depth_pred: خروجی تصویر دیسپریتی

در تابع inference، ابتدا تصویر را به فرمت مورد نیاز تبدیل و سپس با استفاده از تابع normalize، تصویر را جهت خروجی گرفتن از شبکه نرمالایز می‌کنیم.

در قسمت تقسیم‌بندی، شبکه برای هر پیکسل مقدار متناظری متناسب با کلاس در نظر می‌گیرد که در تابع save_pred_to_disk از هر کلاس پیکسلی، یک تصویر رنگی از آن‌ها ایجاد می‌کند.

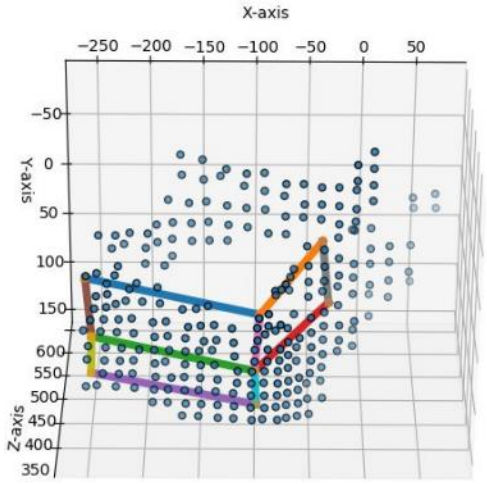
۵-۲-۴-۳- ایده

شبکه SGDepth قادر تولید دیسپریتی (عمق تصویر)، با تنها یک تصویر سه کاناله است. برای افزایش دقت فاصله سنجی، می‌توان از اطلاعات موجود در فضای سه بعدی نیز کمک گرفت. شبکه RFNet نیز با ترکیب دیسپرتی به‌دست آمده، ماسک خودرو را خروجی می‌دهد. ایده اصلی در ترکیب خروجی‌های این دو شبکه شکل می‌گیرد. به کمک ماسک خودرو و اعمال آن روی دیسپریتی، نویزهای اطراف خودرو و دیسپرتی سایر نقاط به غیر از خودرو حذف شده و تنها اطلاعات سه بعدی خودرو مقابل باقی می‌ماند.



شکل ۹۱-۳ جداسازی خودرو از روی تصویر دیسپریتی

نقاط خودرو را می‌توان به کمک هندسه استریو، به فضای سه‌بعدی انتقال داد و پیدا کردن صفحه پشت خودرو در فضای سه‌بعدی، تاثیر نویز و نقاط اطراف را در پیدا کردن فاصله کاهش داد. درنهایت به کمک فاصله صفحه پشت خودرو و دوربین، فاصله خودرو مقابل با دقت بالایی به دست می‌آید.



شکل ۹۲-۳ ابرنقاط خودرو بدست آورده شده از تصویر ۸۷-۳

۳-۵- تشخیص پیاده‌رو

۳-۵-۱- تبیین روش مورد نظر

برای تشخیص پیاده‌رو نیازمند داشتن تصویر تقسیم‌بندی شده به صورت معنایی هستیم. با توجه به اینکه شبکه SGDepth به طور همزمان خروجی تقسیم‌بندی معنایی و دیسپریتری ارائه می‌کند و منبع اصلی آن نیز مقاله Self-Supervised Monocular Depth Estimationاست، جزئیات دقیق این مقاله برای این بخش در قسمت ۳-۴-۲-۱ ارائه شده است.

۳-۵-۲- چگونگی آموزش و صحه‌گذاری

با توجه به اینکه آموزش دو قسمت دیسپریتی و تقسیم‌بندی معنایی تحت عنوان یک شبکه واحد به نام SGDepth صورت می‌گیرد، نحوه آموزش این شبکه به همراه نتایج آن به صورت کامل در قسمت ۳-۴-۲-۲ توضیح داده شده است.

۳-۵-۳- نتیجه آموزش، تشریح نحوه پیاده سازی الگوریتم و ساختار کد

ایده جالب استفاده شده در این مقاله، وجود یک ترید آفی بین خروجی دیسپریتی و تقسیم‌بندی معنایی است. هر دو قسمت با تاثیر بر روی یکدیگر سعی در بهبود نتیجه خود دارند. همانطور که در قسمت ۳-۴-۲-۳ گفته شد، دیتاست های Kitti و Cityscapes بسیار حجیم بوده و آموزش دوباره آن‌ها نیازمند سخت افزار مناسب و زمان کافی می‌باشد و آموزش دوباره این دیتاست‌ها صرفا باز تولید نتایج است. به همین دلیل ما از مدل‌های از پیش آموزش داده شده که در سایت گیت‌هاب شبکه SGDepth قرار گرفته است استفاده کردیم.

در تصویر ۳-۹۳ و ۳-۹۴ نتیجه خروجی دسته‌بندی معنایی کلاس پیاده‌رو حاصل از شبکه SGDepth بعد از ایده استفاده شده از کانتورها را مشاهده می‌کنید:

شکل ۹۳-۳ نتیجه اول خروجی دسته‌بندی معنایی کلاس پیاده‌رو

شکل ۹۴-۳ نتیجه دوم خروجی دسته‌بندی معنایی کلاس پیاده‌رو

کلاس اشیا دسته‌بندی شده به صورت مقابل می‌باشد.

```
self.labels = (('CLS_ROAD', (128, 64, 128)),
               ('CLS_SIDEWALK', (244, 35, 232)),
               ('CLS_BUILDING', (70, 70, 70)),
               ('CLS_WALL', (102, 102, 156)),
               ('CLS_FENCE', (190, 153, 153)),
               ('CLS_POLE', (153, 153, 153)),
               ('CLS_TRLIGHT', (250, 170, 30)),
               ('CLS_TRSIGN', (220, 220, 0)),
               ('CLS_VEGT', (107, 142, 35)),
               ('CLS_TERR', (152, 251, 152)),
               ('CLS_SKY', (70, 130, 180)),
               ('CLS_PERSON', (220, 20, 60)),
               ('CLS_RIDER', (255, 0, 0)),
               ('CLS_CAR', (0, 0, 142)),
               ('CLS_TRUCK', (0, 0, 70)),
               ('CLS_BUS', (0, 60, 100)),
               ('CLS_TRAIN', (0, 80, 100)),
               ('CLS_MCYCLE', (0, 0, 230)),
               ('CLS_BCYCLE', (119, 11, 32)),
               )
```

برای این بخش فقط نیازمند پیاده‌رو یا کلاس 'CLS_ROAD' که با رنگ به فرمت

RGB= (244, 35, 232) مشخص شده است، هستیم.

برای جداسازی کلاس پیاده‌رو و اعمال آن بر روی تصویر اصلی با استفاده از تابع apply_mask این کار را انجام می‌دهیم:

این تابع، ورودی:

- Image: تصویر اصلی
- Seg_img: تصویر تقسیم‌بندی شده

دریافت می‌کند و خروجی اعمال شده تقسیم‌بندی پیاده رو بر روی تصویر اصلی را با توجه به ROI باز می‌گرداند.

```
1.  def apply_mask(image, seg_img, masked_image, color = [244, 35, 232], alpha=0.5):
2.      try:
3.          img = image.copy()
4.          np.save('seg.npy',seg_img)
5.          seg_img = np.load('seg.npy')
6.
7.          mask = (seg_img == np.array([244, 35, 232]))[...,1].astype('uint8')
8.          contours, _ = cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
9.
10.         canvas = np.zeros_like(seg_img, np.uint8)
11.
12.         cnts = sorted(contours, key = cv2.contourArea, reverse=True)
13.         n = len(cnts)
14.
15.         if n < 5:
16.             for i in range(2):
17.                 cv2.drawContours(canvas, cnts, i,  (0,255,0), -1, cv2.LINE_AA)
18.         else:
19.             for i in range(2):
20.                 cv2.drawContours(canvas, cnts, i,  (0,255,0), -1, cv2.LINE_AA)
21.
22.         mask_new = (canvas == np.array([0, 255, 0]))[...,1].astype('uint8')
23.         mask_new = (cv2.resize(mask_new, (img.shape[1], img.shape[0])))
24.         masked_image = cv2.bitwise_and(masked_image, masked_image, mask =mask_new)
25.         mask = (masked_image > 0)[...,1].astype('uint8')
26.
27.         for c in range(3):
28.             img[:,:,c]= np.where(mask == 1, img[:,:,c]*(1 - alpha)+alpha*color[c],img[:,:,
    c])
29.     except:
30.         pass
31.     return img
```

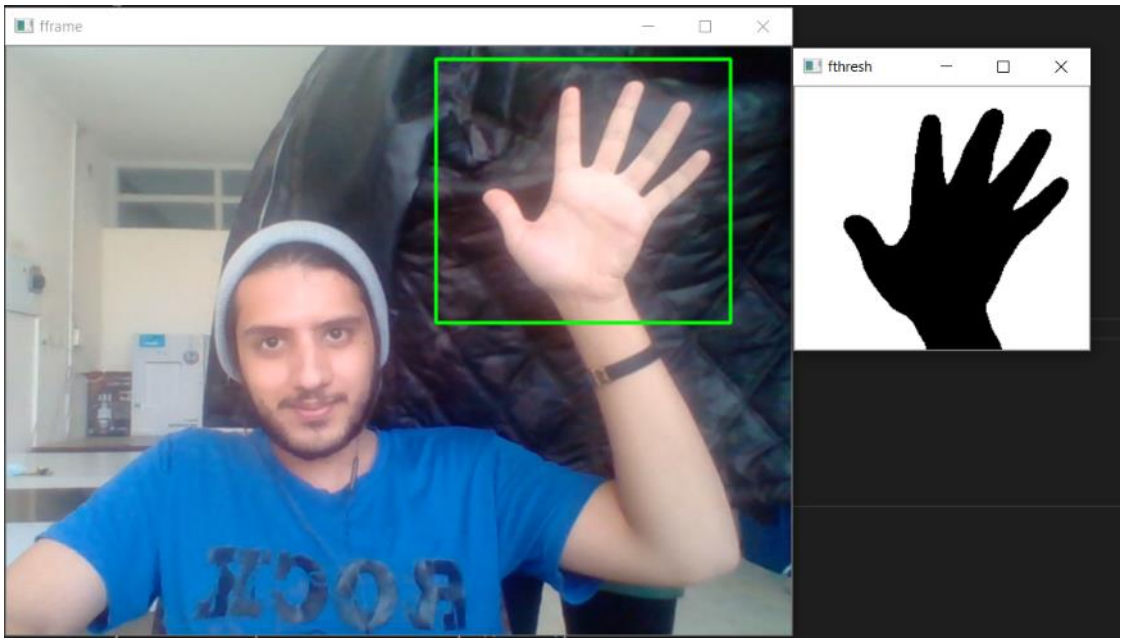
برای افزایش دقت تقسیم‌بندی پیاده‌رو از مفهوم کانتور ها استفاده می‌کنیم، بدین صورت که با در نظر گرفتن دو کانتور ماکزیمم، سعی می‌شود قسمت‌های تقسیم بندی شده کوچک که بیشتر به دلیل نویز ایجاد شده‌اند حذف شود.

۶-۳- تشخیص خطوط عابر پیاده

۱-۱-۶-۳- تبیین روش مورد نظر به همراه ریاضیات کامل

برای خطوط عابر پیاده در این پروژه از مفهوم کانتور[۲۴۸] ها استفاده شده است.

کانتور ها را می توان به صورت منحنی هایی تعریف کرد که تمام نقاط به هم پیوسته در یک تصویر را در بر می گیرند. نقاط به هم پیوسته را نیز می توان نقاطی از تصویر با شدت رنگ[۲۴۹] در یک محدوده ی یکسان تعریف نمود. در کل کانتور ها مفید ترین ابزار برای تجزیه و تحلیل اشکال موجود در یک تصویر و تشخیص اشیایی که محدوده شدت رنگ یکسان دارند، می باشند[۲۷۷].



شکل ۹۵-۳ نمایش مفهوم تشخیص نقاط به هم پیوسته و یک رنگ در تصویر توسط کانتور ها

در کتابخانه ی اپن سی وی، تئوریات و ریاضیات استفاده شده در پیدا کردن کانتور ها، دقیقا همچون تئوریات و ریاضیات پیدا کردن یک جسم سفید با یک پس زمینه ی سیاه می باشد. بنابراین باید دقت داشت که ناحیه ای که قصد شناسایی و جداسازی آن را داریم، باید نسبت به پس زمینه ی خود روشن تر باشد و شدت رنگ بیشتری داشته باشد. به عبارت دیگر الگوریتم یافتن کانتور برای تمایز بین دو جسم با شدت نور تقریبا یکسان، عملکرد

[248] Contours
[249] Intensity

مناسبی از خود نشان نمی دهد اما برای یافتن تمایز بین دو جسم یا دو ناحیه با شدن رنگ های مخالف یا تیره و روشن، بسیار خوب عمل می کند[۲۷۸].

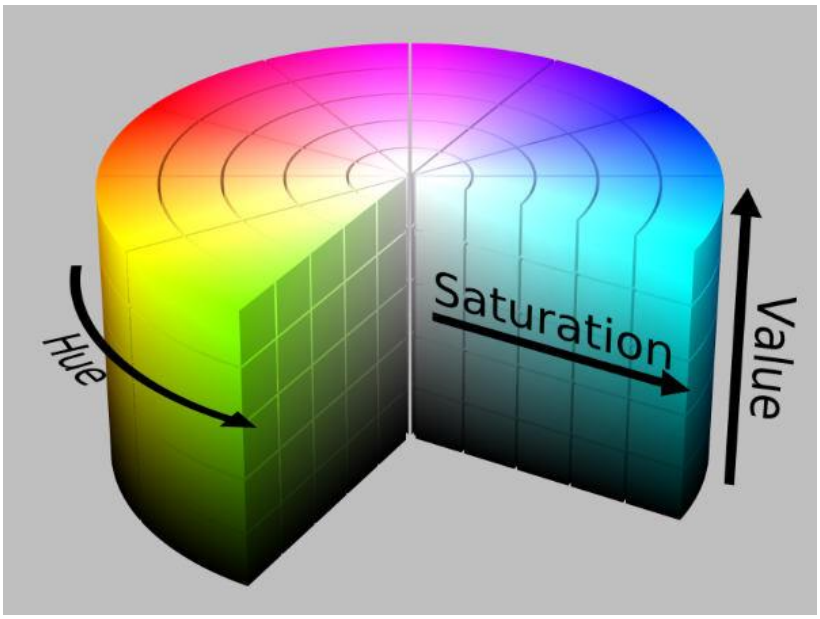
فرآیند تشخیص ناحیه ی نقاط پیوسته در یک محدوده از تصویر یا همان یافتن کانتور، مسیر نسبتا مشخص و قابل فهمی را دنبال می کند. ماتریسی از مشخصات پیکسل ها و شدت رنگ آن ها تشکیل می گردد و بین آن ها شباهت سنجی می شود. در نهایت پیکسل هایی که از لحاظ شدت رنگ مشابه هم باشند و روشن تر از مابقی پیکسل ها باشند، به عنوان ناحیه ی کانتور انتخاب می گردند.

برای یافتن شدت رنگ هر تصویر، الگوریتم پیدا کردن کانتور، ابتدا تصویر را به فضای HSV منتقل می کند. این فضا از اجزای Hue (رنگدانه)، Saturation (اشباع) و Value (روشنایی) تشکیل شده است.

رنگدانه، قسمت رنگی پیکسل است و به عنوان یک عدد از ۰ تا ۳۶۰ درجه بیان می شود. به عنوان مثال مقادیر ۰ تا ۶۰ نشان دهنده ی رنگ قرمز و مقدار ۶۰ تا ۱۲۰ نشان دهنده ی رنگ زرد می باشد.

اشباع، مقدار خاکستری بودن و در واقع غلظت رنگ است و مقداری بین ۰ تا ۱۰۰ درصد را در بر می گیرد. کاهش فاکتور اشباع به سمت ۰ درصد می تواند منجر به محو شدن پیکسل شود.

روشنایی نیز در ارتباط با شدت رنگ است و در محدوده ی ۰ تا ۱ بیان می گردد.



شکل ۹۶-۳ فضای HSV

الگوریتم پیدا کردن کانتور با بردن تصویر به فضای HSV، پیکسل هایی که در یک محدود ی خاص به هم شباهت دارند و نسبت به پس زمینه دارای شدت رنگ کمتری هستند را به عنوان نقاط پیوسته در نظر می گیرد و نقاط حاشیه و مرزی این این نقاط را به عنوان خروجی ارائه می دهد[۲۷۹].

در نهایت با ترسیم این نقاط حاشیه ای روی تصویر اصلی و پر کردن آن ها با یک رنگ خاص، می توان ناحیه ی کانتور را به طور کامل روی یک تصویر به نمایش گذاشت. همچنین امکان محاسبه ی مساحت کانتور وجود دارد که می توان از محاسبه ی مساحت کانتور جهت عدم رسم کانتورهایی که مساحتشان از حد خاصی کمتر باشد استفاده نمود.

۲-۱-۶-۳- تشریح نحوه پیاده سازی الگوریتم، ساختار کد و ارائه ی نتایج

در قطعه کد زیر، نحوه ی تشخیص کانتور خطوط عابر پیاده و ترسیم آن ها ارائه شده است:

```
1.  frame = cv2.imread(frame.png')
2.  mask = cv2.inRange(frame, np.array([70,70,90]), np.array([165,140,135]))
3.  test_img = np.zeros_like(mask)
4.  test_img[560:, 230:1100] = mask[560:, 230:1100]
5.  points,_ = cv2.findContours(test_img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
6.  for point in points:
7.      if cv2.contourArea(point)>2500:
8.          cv2.fillPoly(frame, pts =[point], color=(0,255,0))
9.  cv2.imshow('output', frame)
10. cv2.waitKey(0)
```

در خط اول، تصویر مورد نظر خوانده می شود.

در خط دوم، این تصویر همانند بخش شناسایی خط افقی، بخش سفید تصویر جدا شده، تا عملیات شناسایی خطوط عابر پیاده راحت تر روی آن انجام گردد.

در خط سوم و چهارم، ناحیه ی دید راننده از تصویر جدا شده تا عملیات یافتن کانتور های خط عابر پیاده در این قسمت انجام شود. تعیین ناحیه ی دید راننده همانند بخش تشخیص خطوط افقی انجام خواهد شد و از همان ناحیه ی دیدی استفاده می شود که در بخش های قبل استفاده شده است.

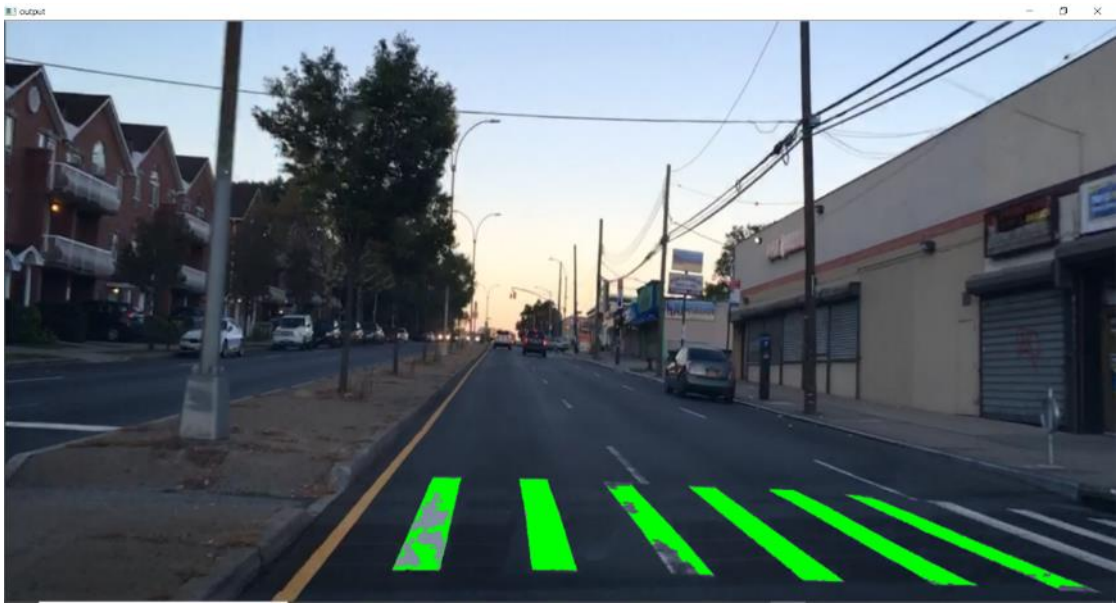
در خط پنجم با استفاده از دستور cv2.findContours عملیات یافتن کانتور در ناحیه ی دید راننده انجام می شود.

در خط ششم تا هشتم، این شرط قید شده که در صورتی که مساحت کانتور یافت شده از میزان مشخصی که برابر با اندازه ی معمول یک خط عابر پیاده است با صحیح و خطا به دست آمده کمتر باشد، این کانتور رسم نشود، در غیر این صورت، این کانتور و محیط داخلی آن با رنگ سبز مشخص گردند.

و در نهایت در خط نهم و دهم این تصویر به نمایش گذاشته شده است.

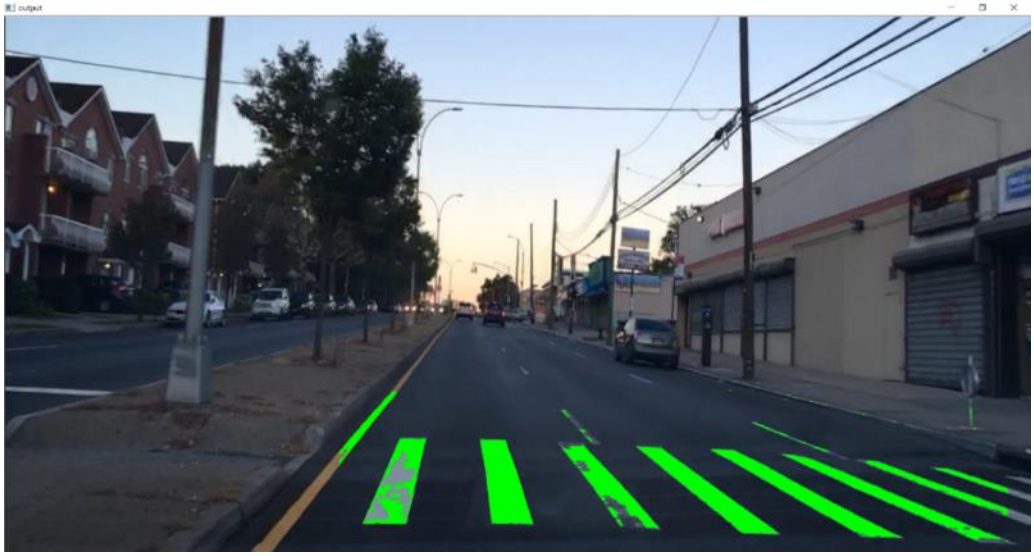
نمونه ی خروجی قطعه کد برای تصویر ورودی ۹۷-۳ در تصویر ۹۸-۳ آورده شده است.

شکل ۹۷-۳ تصویر داده شده به عنوان ورودی به قطعه کد تشخیص عابر پیاده



شکل ۹۸-۳ خروجی اول قطعه کد تشخیص عابر پیاده

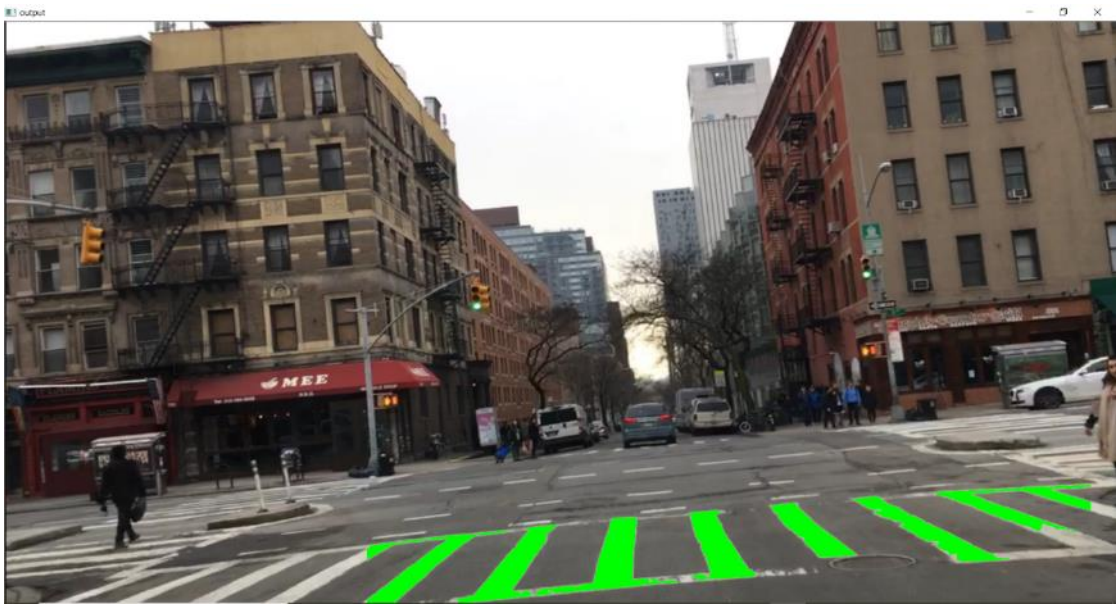
حال این مورد بررسی می گردد که اگر آستانه و آستانه‌ی مربوط به مساحت کانتور ها در نظر گرفته نمی‌شد، چه اتفاقی می افتاد:



شکل ۹۹-۳ خروجی قطعه کد تشخیص عابر پیاده بدون شرط مساحت

می بینیم که در تصویر، قطعات دیگری نیز به عنوان خطوط به هم پیوسته شناسایی شده‌اند و به رنگ سبز در آمده‌اند که این مورد کاملا هم صحیح است، اما قطعات شناسایی شده اصلا در ابعاد و اندازه ی خطوط عابر پیاده نیستند، بنابراین با گذاشتن شرط مساحت کانتور، این موضوع به خوبی حل و فصل می گردد. در ادامه این بررسی رو یک ویدئوی دیگر نیز انجام می گردد:

شکل ۱۰۰-۳ تصویر داده شده به عنوان ورودی به قطعه کد تشخیص عابر پیاده



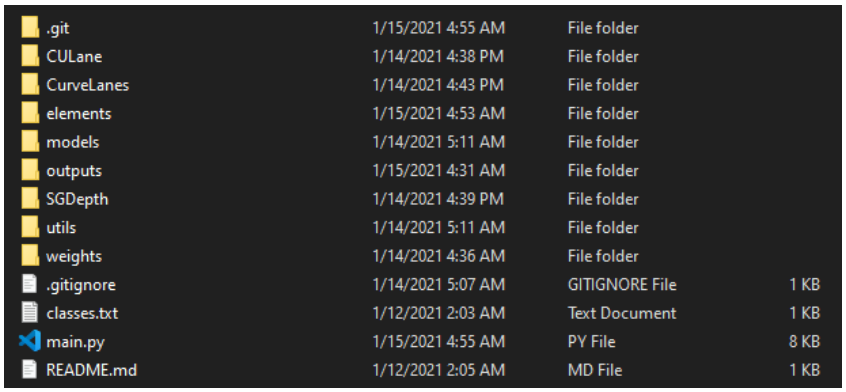
شکل ۱۰۱-۳ خروجی دوم قطعه کد تشخیص عابر پیاده

۷-۳- ارائه یک مثال حل شده

حال که بخش‌های تئوری و اساسی قسمت ادراک در قسمت‌های قبل تشریح شد، بایستی تمامی بخش‌ها را در قالب یک کد اصلی کنار هم قرار داد و شاهد خروجی نرم‌افزار اصلی بود. زبان برنامه‌نویسی برای تمامی بخش‌های ذکر شده در قسمت ادراک، زبان برنامه نویسی پایتون بوده که دارای نوشتار[۲۵۰] قابل فهم و سرعت بالا بوده و تمامی ماژول‌ها و کتاب‌خانه‌های مورد استفاده در این زبان دارای جامعه[۲۵۱] و مستندات[۲۵۲] مناسب می‌باشد. در ادامه، قسمت‌های اصلی کد main.py و نحوه‌ی استفاده از آن برای دریافت خروجی‌های مورد نظر آوزده شده است.

۱-۷-۳- تشریح بستر نرم افزاری (کد اصلی)

پوشه‌ای اصلی نرم افزار، شامل پوشه‌ها و کدهای متنوعی می‌باشد که در شکل ۱۰۲-۳ مشاهده می‌شود.



شکل ۱۰۲-۳ پوشه‌ی اصلی نرم‌افزار

همان طور که مشخص است، این پوشه دارای فایل‌ها و پوشه‌های متنوعی می‌باشد، در این بخش به بررسی کد اصلی یا main.py پرداخته می‌شود که برای خروجی گرفتن مورد نیاز است. در داخل سایر پوشه‌ها، اطلاعات مربوط به مدل ها و ابزارهای مورد استفاده موجود است که در بخش‌های گذشته به توضیح هر کدام به صورت مجزا پرداخته شده است. در این بخش فایل main.py به صورت کامل مورد بررسی قرار می‌گیرد.

در قسمت اول کد، کتابخانه‌ها و ابزارها وارد نرم افزار می‌شوند[۲۵۳] که به صورت زیر می‌باشد:

```
1.  from elements.yolo import YOLO, YOLO_Sign
```

[250] Syntax
[251] Community
[252] Documentation
[253] Import

```
from elements.PINet import LaneDetection
from elements.SGD import Inference
from elements.Curvlane import CurveLane
from elements.asset import cityscape_xyz, kitti_xyz, apply_mask, ROI, kitti_xyz_dist, city
scape_xyz_dist
from utils.plots import plot_one_box
import matplotlib.pyplot as plt
from elements.asset import horiz_lines, detect_lines
import numpy as np
import os
import cv2
from time import time as t
import datetime
import random
import sys
from datetime import timedelta
from SGDepth.arguments import InferenceEvaluationArguments
```

سپس آرگمان‌های مورد نیاز به برنامه معرفی می‌شوند، در بخش ۲-۷-۳، آرگمانها به صورت کامل توضیح داده شده و نحوه‌ی استفاده از آنها نیز آورده شده است.

```
opt = InferenceEvaluationArguments().parse()
```

نحوه‌ی خروجی گرفتن از برنامه به این صورت می‌باشد که یا خروجی به صورت مستقیم نمایش داده‌شود و یا این‌که خروجی به صورت ویدئو یا تصویر ذخیر شود، در ادامه‌ی برنامه چک می‌شود که حداقل یکی از دو نوع خروجی فعال باشد و در صورتی که هیچ‌کدام فعال نباشند، کاربر یک پیام با این مضمون که هیچ خروجی‌ای دریافت نمی‌شود دریافت می‌کند و برنامه متوقف می‌شود.

```
if opt.noshow and not opt.save:
    print("You're not getting any outputs!!\nExit")
    sys.exit()
```

سپس، در صورتی که یکی از انواع خروجی گرفتن فعال باشد، برنامه به کار خود ادامه می‌دهد. ابتدا وزن‌های مربوط به مدل تشخیص اشیا که همان مدل yolo می‌باشد بارگذاری می‌شود و آبجکت آن ساخته می‌شود.

```
detector = YOLO(opt.weights_detector)
```

سپس نوع مدل تشخیص‌خطوط جاده بر اساس یکی از دو مدل culane و یا curvelane که کاربر مشخص کرده‌است بارگذاری و وارد برنامه شده و آبجکت آن ساخته می‌شود.

```
if opt.lane_detector_type == 'culane':
    lane_detector = LaneDetection(opt.culane_model)
```

```
    print("CULane model loaded!")
if opt.lane_detector_type == 'curvelane':
    lane_detector = CurveLane(opt.curvelane_model)
    print("Curvelane model loaded!")
```

سپس مدل تشخیص دیسپریتی مدل تشخیص تابلوهای راهنمایی و رانندگی وارد برنامه می‌شود:

```
disparity_detector = Inference(opt.disp_detector)
sign_detector = YOLO_Sign(opt.weights_sign)
```

در ادامه، ویدئوی مورد بررسی که توسط کاربر مشخص شده است و در آرگمان opt.video قرار گرفته است بارگذاری شده و در خط بعدی تعداد فریم‌های موجود در ویدئو شمارش شده و در متغیر frame_count قرار می‌گیرد.

```
cap = cv2.VideoCapture(opt.video)
frame_count = cap.get(cv2.CAP_PROP_FRAME_COUNT)
```

سپس، درصورتی که کاربر هنگام اجرای برنامه درخواست ذخیره‌سازی خروجی را داده‌باشد، ابتدا پوشه‌ی outputs در صورتی که موجود نباشد ساخته می‌شود، و یک پوشه به اسم فایل خروجی که کاربر برای فایل خروجی در نظر گرفته است و در آرگمان opt.output_name قرار گرفته است ساخته می‌شود. سپس یک فایل خروجی در پوشه‌ای که ساخته شد قرار گرفته که ابعاد آن برابر با ابعاد ویدئوی اصلی می‌باشد. همچنین در صورتی که کاربر آرگمان opt.save_frames را انتخاب کرده‌باشد، در پوشه‌ی خروجی، یک پوشه به نام frames ایجاد می‌شود. مطالب ذکر شده تا اینجا در کد زیر نمایش داده شده است.

```
if opt.save:
    if len(opt.output_name.split('.'))==1:
        opt.output_name += '.mp4'
    output_video_folder = os.path.join('outputs/', opt.output_name.split('.')[0])
    if opt.save_frames:
        output_frames_folder = os.path.join(output_video_folder, 'frames')
        os.makedirs(output_frames_folder, exist_ok=True)
    output_video_name = os.path.join(output_video_folder, opt.output_name)
    os.makedirs(output_video_folder, exist_ok = True)

    w = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
    h = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)

    out = cv2.VideoWriter(output_video_name,
                          cv2.VideoWriter_fourcc(*'mp4v'),
                          opt.outputfps, (int(h), int(w)))
```

در ادامه، برای هر کدام از کلاس‌های تشخیص داده‌شده‌ی Yolo هم برای تشخیص تشخیص انسان‌ها و وسایل نقلیه و چراغ راهنمایی و هم برای خروجی های Yolo_sign، اسامی کلاس ها و رنگ باندینگ باکس هر کدام از کلاس‌ها مشخص می‌شود.

```
names = {
        'person': 0,
        'car' : 1,
        'bus': 2,
        'truck' : 3,
        'traffic light' : 4}
colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

signs = ['Taghadom', 'Chap Mamnoo', 'Rast Mamnoo', 'SL30', 'Tavaghof Mamnoo',
         'Vorood Mamnoo', 'Mostaghom', 'SL40', 'SL50', 'SL60', 'SL70', 'SL80', 'SL100',
         'No U-Turn']
colors_signs = [[random.randint(0, 255) for _ in range(3)] for _ in signs]
```

سپس دو متغیر avg_fps و frame_num با مقدار اولیه صفر برای محاسبه‌ی تعداد فریم‌های پردازش شده در ثانیه و شماره‌ی فریم فعلی تعریف می‌شود که در ادامه آپدیت می‌شوند.

```
avg_fps = 0 #Average FPS
frame_num = 0
```

سپس حلقه‌ی پردازش فریم‌ها که یک حلقه‌ی while می‌باشد شروع می‌شود. شرط خروج از حلقه پایان یافتن ویدئو می‌باشد. به این وصرت که تا زمان‌که ویدئو روشن باشد، این حلقه ادامه می‌یابد. سپس یک فریم از ویدئوی ورودی خوانده شده و در متغیر frame قرار می‌گیرد. همچنین frame_num یک واحد اضافه می‌شود. در صورتی که باقی‌مانده‌ی شماره‌ی فریم به آرگمان opt.framedrop صفر نشود، آن فریم نادیده گرفته شده و پردازشی روی آن انجام نمی‌شود و فریم بعدی خوانده می‌شود، در صورت صفر شدن این باقی‌مانده، فریم مذکور مورد پردازش قرار می‌گیرد.

در ادامه‌ی حلقه، در صورتی که فریم خوانده شده موجود بوده باشد، یک متغیر به اسم t1 ایجاد شده که مقدار زمان آن لحظه در سیستم قرار داده می‌شود. سپس فریم ورودی به صورت پادساعتگرد به میزان ۹۰ درجه میچرخد.

```
if ret:
    t1 = t() #Start Time
    frame = cv2.rotate(frame, cv2.ROTATE_90_COUNTERCLOCKWISE)
    # frame = cv2.rotate(frame, cv2.ROTATE_90_CLOCKWISE)
```

سپس، از فریم خوانده شده یک کپی گرفته شده و سپس ابتدا شبکه‌ی Yolo اولی و سپس Yolo دومی که مربوط به تابلوها می‌باشد و پس از آن شبکه‌ی مربوط به دیسپریتی عمل می‌کنند و خروجی هرکدام در متغیرهای مربوطه ذخیره می‌شوند.

```
main_frame = frame.copy()
yoloOutput = detector.detect(frame)
signOutput = sign_detector.detect_sign(frame)
disparity, seg_img = disparity_detector.inference(frame)
```

سپس با استفاده از تابع ROI، قسمتی از فریم که مورد نیاز برای پردازش نمی‌باشد استخراج شده و در متغیر masked_image قرار می‌گیرد. این ماسک شامل یک قسمت ذوذنقه‌ای از فریم اصلی می‌باشد. سپس خطوط جاده تشخیص داده می‌شوند. و در نهایت ماسک ذکر شده روی تصویر قرار می‌گیرد.

```
masked_image = ROI(main_frame)
        frame = lane_detector.Testing(frame, masked_image)

        frame = apply_mask(frame, seg_img)
```

در ادامه، به ازای هرکدام از اشیا شناسایی شده توسط شبکه‌ی Yolo اولی، باندینگ باکس مربوطه برای هر کدام از اشیا ساخته می‌شود، سپس در صورتی که لیبل شی تشخیص داده شده خودرو، اتوبوس و یا وانت باشد، نقطه‌ی بالا سمت چپ و پایین سمت راست باندینگ باکس به دست می‌آید. سپس شرط آن‌که نقطه‌ی مرکزی باندینگ باکس داخل ماسک تصویر باشد چک می‌شود. سپس در صورت ارضا نمودن شرط، تصویر و دیسپریتی به دست آمده که در داخل باندینگ‌باکس قرار میگیرد برش می‌یابد. برای قسمت‌هایی از تصویر که سیاه نیستند و داخل باندینگ باکس قرار می‌گیرند، دیسپیریتی به در داخل یک لیست ذخیره می‌شود. سپس با توجه به معیار محاسبه‌ی عمق که می‌تواند *kitti* و یا *cityscape* باشد، فاصله‌ی شی محاسبه شده و در نهایت در صورتی که این فاصله کمتر از ۱۰ متر باشد برروی باندینگ باکس چاپ شده و در تصویر نمایش داده می‌شود.

```
for obj in yoloOutput:
    xyxy = [obj['bbox'][0][0], obj['bbox'][0][1], obj['bbox'][1][0], obj['bbox'][1][1]]
    depth = []
    if obj['label'] =='car' or obj['label'] == 'truck' or obj['label'] == 'bus':
        x_pts = (obj['bbox'][0][0]+obj['bbox'][1][0])/2
        y_pts = (obj['bbox'][0][1]+obj['bbox'][1][1])/2

        #Distance Measurement
        if np.dot(masked_image[int(y_pts), int(x_pts)], main_frame[int(y_pts), int(x_pts)]) != 0:

            Ry = 192/720
```

```
            Rx = 640/1280
            x_new, y_new =(Rx * x_pts, Ry * y_pts)

            cropped_img = main_frame[xyxy[1]:xyxy[3], xyxy[0]:xyxy[2]]
            cropped_disp = np.array(disparity[int(xyxy[1]*Ry):int(xyxy[3]*Ry), int(xyxy[0]*Rx):int(xyxy[2]*Rx)])
            cropped_img = cv2.resize(cropped_img, (cropped_disp.shape[1], cropped_disp.shape[0]))

            cropped_img = cropped_img[int(cropped_img.shape[0]/2 - 20): int(cropped_img.shape[0]/2 + 20),
                                      int(cropped_img.shape[1]/2 - 20): int(cropped_img.shape[1]/2 + 20)]

            indices = np.where(cropped_img!= [0])
            coordinates = zip(indices[0], indices[1])

            for x,y in coordinates:
                try:
                    depth.append([x, y, cropped_disp[y,x]])
                except:
                    pass

            if opt.depth_mode == 'kitti':
                distance = kitti_xyz_dist(depth)
            else :
                distance = cityscape_xyz_dist(depth)

            printed_distance = np.mean(np.array(sorted(distance)[:15]))

            if  printed_distance < 10:
                plot_one_box(xyxy, frame, printed_distance, label=obj['label'], color=colors[names[obj['label']]], line_thickness=3)
            else:
                plot_one_box(xyxy, frame, label=obj['label'], color=colors[names[obj['label']]], line_thickness=3)
        else:
                plot_one_box(xyxy, frame, label=obj['label'], color=colors[names[obj['label']]], line_thickness=3)
    else:
        plot_one_box(xyxy, frame, label=obj['label'], color=colors[names[obj['label']]], line_thickness=3)
```

در ادامه، به ازای هر تابلوی تشخیص داده‌شده توسط شبکه‌ی Yolo مورد استفاده برای تشخیص تابلو، مختصات باندینگ باکس مروبط به هر تابلو به دست‌آمده به همراه لیبل بر روی تصویر رسم می‌شوند.

```
for sign in signOutput:
            xyxy = [sign['bbox'][0][0], sign['bbox'][0][1], sign['bbox'][1][0], sign['bbox'][1][1]]
```

```
3.            plot_one_box(xyxy, frame, label=sign["label"],  color=colors_signs[sign['cls'
              ]], line_thickness=3)
```

پس از این مرحله، حال که پردازش فریم به اتمام رسیده است، مجددا زمان سیستم در متغیر دیگری به اسم t2 ذخیره می‌شود. پس از این، زمان پردازش که مربوط به انجام تمام مراحل ذکر شده برای تک فریم می‌باشد از رابطه‌ی $t_2 - t_1$ به دست می‌آید. حال باید فهمید که در هر ثانیه چه تعداد فریم پردازش می‌شود که برای این مهم از رابطه‌ی $fps = \frac{1}{t2-t1}$ استفاده می‌شود و تا ۳ رقم اعشار گرد می‌شود.

```
1.  t2 = t() #End of frame time
2.  fps = np.round(1 / (t2-t1) , 3)   #Running FPS
```

پس از آن، نیاز است که میانگین *fps* های گذشته محاسبه شود. برای این کار، از میانگین متحرک[۲۵۴] *fps* های گذشته استفاده می‌شود که برای هر *fps* یک وزن در نظر می‌گیرد. به این صورت که هر *fps* جدید به اندازه‌ی ۰.۰۵ کل میانگین تاثیر خواهد داشت و رابطه‌ی نهایی به صورت رابطه‌ی زیر خواهد بود.

$$fps_{avg}(t) = 0.95 \times fps_{avg}(t-1) + 0.05 fps(t)$$

کدهای این بخش به صورت زیر می‌باشد.

```
1.  t2 = t() #End of frame time
2.  fps = np.round(1 / (t2-t1) , 3)   #Running FPS
3.  avg_fps = fps * 0.05 + 0.95 * avg_fps
```

پس از این، باید زمان پردازش باقی مانده تخمین زده شود، برای این‌کار می‌بایست تعداد کل فریم‌های پردازش شده تا به حال که در متغیر *frame_num* ذخیره شده است از تعداد کل فریم‌های ویدئو که در متغیر *frame_count* کم شده و بر میانگین فریم پردازش شده در ثانیه (*fps*) تقسیم شود. سپس می‌بایست این مقدار که بر حسب ثانیه است به قالب ثانیه:دقیقه:ساعت تبدیل شود که این مهم با استفاده از تابع *timedelta* از کتابخانه‌ی *datetime* انجام می‌شود، در نهایت کل این زمان به دست آمده به فرمت رشته[۲۵۵] تبدیل می‌شود.

```
1.  estimated_time = (frame_count - frame_num) / avg_fps
2.  estimated_time = str(timedelta(seconds=estimated_time)).split('.')[0]
3.  s = "FPS : "+ str(fps)
```

[254] Moving Average
[255] String

سپس در صورتی که کاربر آرگمان opt.fps را فعال کرده باشد، در گوشه‌ی بالا سمت چپ تصویر مقدار *fps* چاپ می‌شود.

```
if opt.fps:
    cv2.putText(frame, s, (40, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), thickness= 2)
```

در ادامه، با استفاده از تابع *horiz_lines* که در ماژول *assets.py* نوشته شده است، خطوط عمودی جاده به دست آمده و برروی فریم چاپ می‌شوند. شایان ذکر است که نیاز است این عملیات لازم است روی فریم بدون تغییر انجام شود، پس ورودی تابع *main_frame* می‌باشد، همچنین ورودی دوم *frame* است که تابلوها و خط کشی جاده و فواصل اشیا بر روی آن مشخص شده است.

```
#Cross Walk Lines
frame = horiz_lines(main_frame, frame, mode = opt.mode)
# Saving the output
if opt.save:
    out.write(frame)
    if opt.save_frames:
        cv2.imwrite(os.path.join(output_frames_folder , '{0:04d}.jpg'.format(int(frame_num
        ))) , frame)
```

در نهایت چک می‌شود که تابع آرگمان *opt.noshow* فعال کرده است یا خیر، در صورت فعال بودن، خروجی ای کاربر نمایش داده نمی‌شود. در غیر این صورت، خروجی به صورت همرمان با پردازش نمایش داده می‌شود.

```
if not opt.noshow:
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

پس از این مراحل، مشخصات برنامه که شامل نام ویدئوی مورد پردازش، تعداد فریم‌های پردازش شده و تعداد کل فریم‌ها، *fps*، و زمان تخمینی باقی‌مانده نمایش داده شود که در کد زیر انجام می‌شود:

```
sys.stdout.write(
    "\r[Input Video : %s] [%d/%d Frames Processed] [FPS : %f] [ET : %s]"
    % (
        opt.video,
```

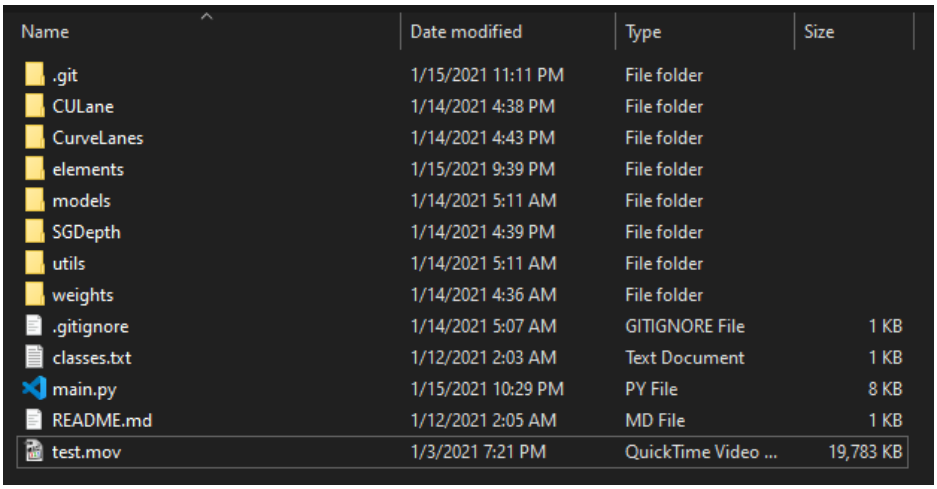
```
        frame_num,
        frame_count,
        fps,
        estimated_time
    )
```

در نهایت در صورتی که کل فریم‌های ویدئو پردازش شده بود، ویدئو بسته شده و برنامه به اتمام می‌رسد.

```
cap.release()
os.remove('seg.npy')
if not opt.noshow:
    cv2.destroyAllWindows()
```

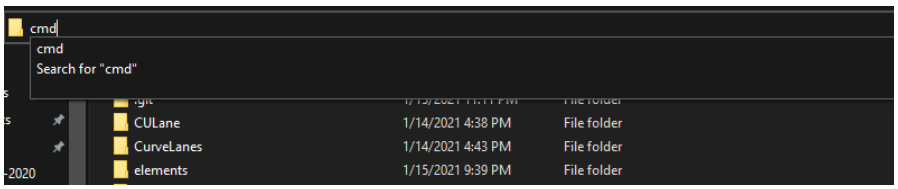
حال که کد اصلی برنامه به صورت کامل شرح داده‌شد، نحوه‌ی خروجی گرفتن از برنامه توضیح داده می‌شود.

ابتدا بایستی ویدئوی مورد نظر در پوشه‌ی اصلی نرم‌افزار قرار گیرد، برای مثال در تصویر ۱۰۳-۳، یک فایل به اسم *test.mov* اضافه شده است.



شکل ۱۰۳-۳ پوشه‌ی اصلی نرم‌افزار پس از اضافه کردن ویدئو

حال می‌بایست با کلیک کردن روی قسمت آدرس پوشه، در داخل آدرس عبارت cmd را تایپ کرده و دکمه‌ی Enter بر روی صفحه کلید زده شود.



شکل ۱۰۴-۳ تایپ cmd بر بخش آدرس پوشه‌ی اصلی

پس از این، یک پنجره‌ی خط‌فرمان[۲۵۶] در داخل پوشه باز می‌شود که می‌توان عبارات دستوری برنامه را با استفاده از آن اجرا کرد.

شکل ۱۰۵-۲ محیط خط فرمان باز شده در پوشه‌ی اصلی

ابتدا در داخل این محیط عبارت زیر بایستی تایپ شود که کتابخانه‌های مورد نیاز نصب شوند.

```
1.  pip install -r requirements.txt
```

سپس در داخل این محیط می‌بایست عبارت زیر را تایپ نمود که خط فرمان متوجه شود که برنامه‌ای تحت زبان پایتون به نام *main.py* قرار است اجرا شود:

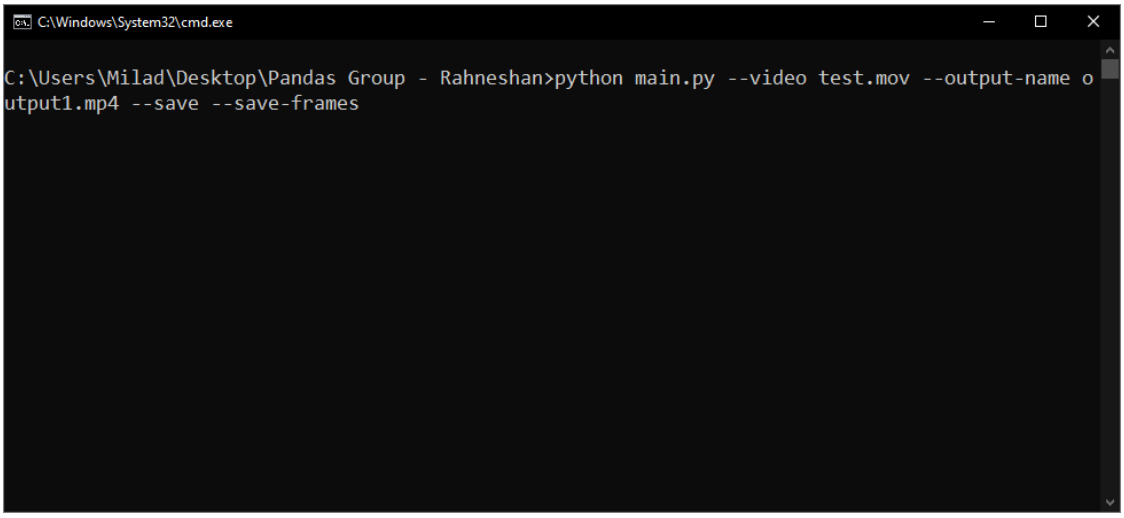
```
2.  python main.py
```

سپس، در صورت نیاز، می‌توان آرگمان‌های مختلف مورد نیاز به خط کد بالا اضافه شود، شایان ذکر است که آرگمان‌ها مقادیر پیش‌فرض دارند لکن برای استفاده‌ی عمومی، نیاز است که مشخص شوند. برای نمونه بایستی آرگمان اسم ویدئوی ورودی با عبارت *video--* که اسم ورودی که همان ویدئوی شماره ۱ رهنشان می‌باشد مشخص شود. همچنین مطلوب است که اسم خروجی به عنوان *output1.mp4* باشد و به علاوه ویدئوی خروجی ذخیره شده و فریم‌های آن نیز ذخیره شوند. پس کد بالا به صورت زیر تغییر خواهد کرد.

[256] Command Line

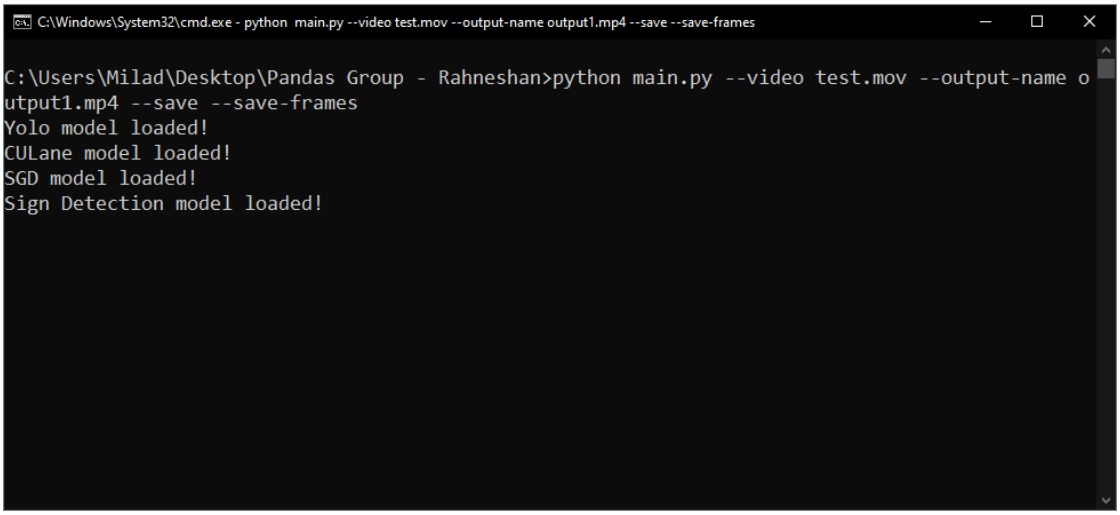
```
1.  python main.py --video test.mov --output-name output1.mp4 --save --save-frames
```

حال در صورتی که کلید Enter زده شود، برنامه شروع به اجرا شدن خواهد کرد.



شکل ۳-۱۰۶ عبارت مورد نیاز در خط فرمان ویندوز

پس از شروع به اجرا کردن نرم افزار، ابتدا به کاربر اطلاع داده می‌شود که مدل‌های مختلف در داخل برنامه بارگذاری شده‌اند.

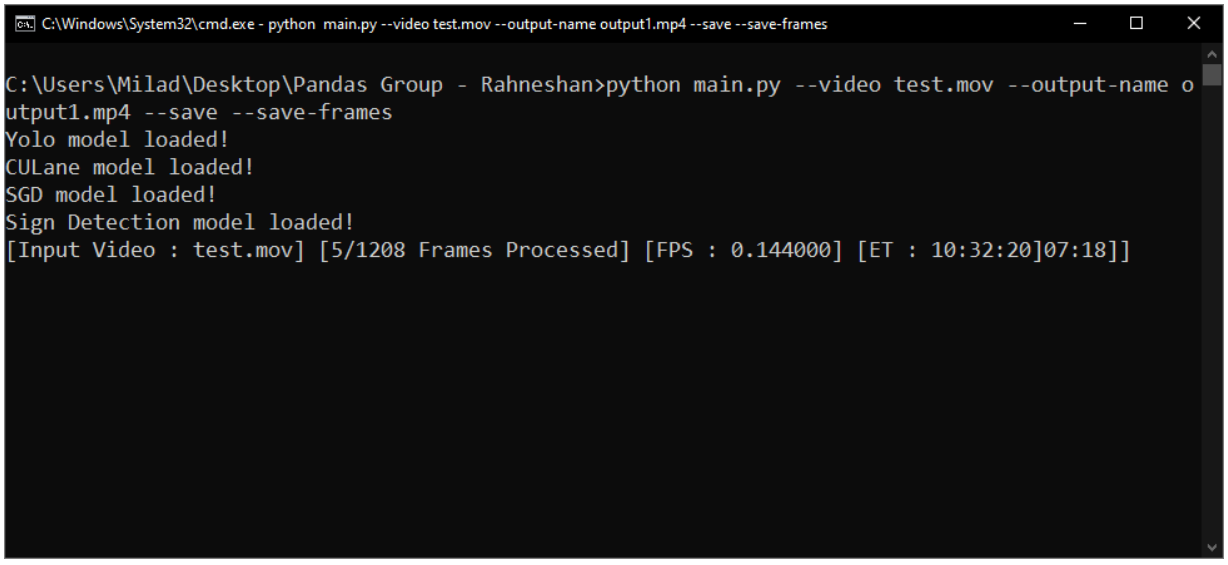


شکل ۳-۱۰۷ بارگذاری مدل‌های مختلف

پس از چند ثانیه، در یک پنجره‌ی جدید به اسم *frame* خروجی کامل ویدئو شروع به نمایش می‌کند.

شکل ۱۰۸-۳ خروجی ویدئو در حال اجرای برنامه

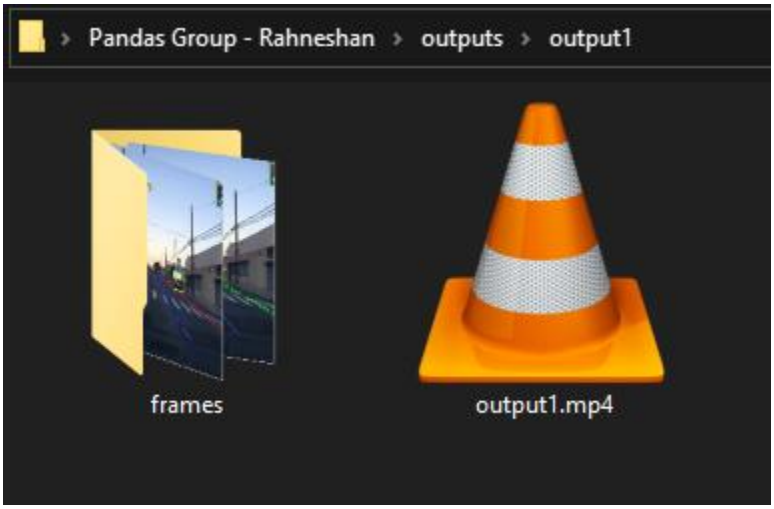
به طور همزمان، در داخل محیط خط فرمان، اطلاعات مفیدی از نام ویدئوی در حال پردازش، *fps*، تعداد فریم-های پردازش‌شده، زمان تخمینی باقی‌مانده به صورت زنده[۲۵۷] نمایش داده می‌شود.



شکل ۱۰۹-۳ جزئیات پردازش ویدئو

[257] Real Time

در پوشه‌ی outputs یک پوشه به اسم output1 که همان خروجی مورد نظر بود ایجاد شده است، که در داخل آن، یک پوشه به اسم frames و یک ویدئو به نام output1.mp4 ذخیره شده است که ویدئو همان ویدئوی خروجی ذخیره شده می‌باشد و در داخل پوشه‌ی frames تک تک فریم‌های خروجی نیز ذخیره می‌شوند.



شکل ۱۱۰-۳ پوشه‌ی خروجی ویدئوی پردازش شده

در ادامه به بررسی زمان پردازش ویدئوها توسط نرم‌افزار پرداخته می‌شود. معیار اندازه‌گیری، مقدار *fps* می‌باشد که بیانگر تعداد فریم‌های پردازش شده در واحد زمان می‌باشد. همان‌طور که واضح است، این نرم‌افزار علاوه بر مدل‌ها و الگوریتم‌های کلاسیک، از مدل‌های مبتنی بر یادگیری‌عمیق[۲۵۸] نیز استفاده می‌کند. مدل‌های پردازش تصویر مبتنی بریادگیری عمیق، به دلیل قابل انجام بودن مراحل انجام کار به صورت موازی[۲۵۹]، در صورت اجرا شدن توسط یک پردازنده‌ی گرافیکی[۲۶۰]، سرعت بسیار بیشتری نسبت به اجرای همان الگوریتم برروی یک پردازنده‌ی مرکز[۲۶۱] را دارا خواهند بود. اجرای این نرم افزار و نمونه‌های ارائه شده برروی یک سیستم با پردازنده‌ی مرکزی Intel® Xeon®CPU@2.3GHz و پردازنده‌ی گرافیکی Nvidia Tesla T4 که یک پردازنده‌گرافیکی میان‌رده محسوب می‌شود و زمان‌های مربوط به هر پردازش هر فریم توسط هر شبکه در جدول زیر قابل ملاحظه است.

[258] Deep Leaning
[259] Parallel
[260] GPU
[261] CPU

جدول ۱۶-۳ زمان مورد نیاز پردازش هر فریم توسط هر بخش از نرم افزار

سخت‌افزار مورد استفاده	نام مدل	*fps*	بخش نرم‌افزار
GPU	Yolov5x	38fps	تشخیص اشیا برروی خیابان به جز تابلوها
GPU	Yolov5s	95fps	تشخیص تابلوها
GPU	PINET	20fps	تشخیص خطوط جاده
CPU	-	1000fps	تشخیص فاصله‌ی کلاسیک
GPU/CPU	SGDepth	3fps	تشخیص عمق، سگمنت‌کردن پیاده‌رو و در نهایت تشخیص فاصله
GPU/CPU		6.4fps	پردازش کلی برروی سخت افزار ذکرشده

همان‌طور که از جدول ۱۶-۳ مشاهده می‌شود، توان پردازشی اصلی در بخش تشخیص عمق با استفاده از دوربین تک‌چشم[۲۶۲] می‌باشد. در خودروهای خودران این مهم با استفاده از سنسورهای لیدار[۲۶۳] و یا استفاده از دوربین‌های دوچشمی[۲۶۴] تحت عنوان استریو[۲۶۵] انجام می‌شود. در واقع با استفاده از یک تصویر خالی نمی‌توان عمق تصویر را به صورت دقیق ارزیابی کرد و صرفا می‌توان با استفاده از روش‌های مختلف (در اینجا استفاده از شبکه‌ی عصبی SGDepth) به پیدا کردن یک تخمین از عمق تصویر مربوطه دست پیدا کرد. همان‌طور که در جدول مشاهده می‌شود، چالش اصلی زمانی، همان استفاده از شبکه‌ی SGDepth می‌باشد و در صورت عدم استفاده از شبکه‌ی مذکور و استفاده از روش‌های کلاسیک، تعداد فریم‌های پردازش‌شده به حدود ۱۰-۱۲ فریم بر ثانیه نیز می‌رسد که روش‌هایی با دقت پایین‌تر از روش عمیق می‌باشند. پس با جایگزین کردن تشخیص عمق با روش‌اصلی آن (لیدار و یا دوربین استریو) می‌توان علاوه بر افزایش دقت کلی نرم‌افزار، سرعت پردازش را نیز تا حد بسیار خوبی افزایش داد.

۲-۷-۳- تشریح آرگمانها

در اجرای برنامه‌ی اصلی، یک سری متغیرها در داخل کد وجود دارد که در شرایط مختلف باید تغییر کنند. برای مثال، نام ویدئوی ورودی به برنامه و یا نام ویدئوی خروجی از این قبیل پارامترها می‌باشند. در صورتی که به‌صورت

[262] Molocular
[263] Lidar
[264] Binocular
[265] Stereo

دستی تغییر داده‌شوند، هم ممکن است زمان زیادی طول بکشد و هم ممکن است در صورت عدم تسلط به کد برنامه، موجب خرابی کد شود. همچنین گاها تعداد تغییرات مورد نظر زیاد است و باید در قسمت‌های مختلفی از برنامه اعمال شود. در این موارد، از مفهومی به نام آرگمان استفاده می‌شود که در هنگام اجرای کد اصلی به بخش خط فرمان اضافه می‌شود.

نرم‌افزار بخش ادراک نیز از این قاعده مستثنی نبوده و بخش‌های مختلفی دارد که نیاز است به صورت آرگمان تعریف شود، این مهم در فایل *arguments.py* در پوشه‌ی *SGDepth* می‌باشد. در قسمت اولیه‌ی این فایل، آرگمان‌های مورد نیاز کاربر حین اجرا کردن برنامه اضافه شده است که در ادامه به توضیح هر کدام پرداخته شده است.

```
1.  self.ap.add_argument('--weights-
    detector', type=str, default='weights/yolov5m.pt', help='weights for the main yolo detecto
    r')
2.  self.ap.add_argument('--weights-
    sign'    , type=str, default='weights/best_sign2.pt', help='sign detector weights')
3.  self.ap.add_argument('--disp-
    detector', type=str, default='weights/model_full.pth', help='disparity model weights')
4.  self.ap.add_argument('--lane-detector-
    type', type=str, default='culane', help='Choose between culane or curvelane')
5.  self.ap.add_argument('--culane-
    model', type=str, default='weights/culane_model.pkl', help='Culane model')
6.  self.ap.add_argument('--curvelane-
    model', type=str, default='weights/curvelane_model.pkl', help='Curvelane model')
7.  self.ap.add_argument('--video', type=str, default='test2.mp4', help = 'The input video')
8.  self.ap.add_argument('--save', action= 'store_true', help = 'Saving the output video')
9.  self.ap.add_argument('--
    noshow', action= 'store_true', help =  'Do not Show the output frames')
10. self.ap.add_argument('--frame-
    drop', type = int, default = 1 , help =  'Frame Drop for processing')
11. self.ap.add_argument('--
    outputfps', type = int, default = 30 , help =  'Output Video FPS')
12. self.ap.add_argument('--fps', action = 'store_true' , help =  'Show fps')
13. self.ap.add_argument('--output-
    name', type = str ,default = 'output.mov' , help =  'Outputput video address')
14. self.ap.add_argument('--depth-
    mode', type = str ,default = 'kitti' , help =  'Choosing depth mode (kitti or cityscape)')

15. self.ap.add_argument('--
    mode', type = int, default = 1, help = 'Choose theprocessing model (1,2,3)')
16. self.ap.add_argument('--save-
    frames', action = 'store_true' , help =  'Saves individual Frames')
```

هر کدام از این آرگمان‌ها یا دستور خاصی را به نرم افزار می‌دهند و یا آدرس و یا حالت خاصی را مشخص می‌کند. شایان ذکر است که هر کدام از آرگمان‌های بالا یک مقدار پیشرفت داشته که در صورت اضافه نشدن به خط‌فرمان،

همان مقدار پیش‌فرض برای آرگمان مذکور در نظر گرفته می‌شود. برای استفاده از هر کدام از آرگمان‌ها، بایستی اسم آرگمان را پس از دو خط‌فاصله‌ی[۲۶۶] متوالی (--) در خط فرمان نوشت، و در صورتی که آرگمان مذکور متغیر خاصی را مشخص می‌کند به صورت دودویی[۲۶۷] (به صورت خاموش یا روشن) نمی‌باشد، باید مقدار آن متغیر در ادامه وارد شود.

۱. آرگمان weights-detector که وزن‌های مرتبط با شبکه‌ی تشخیص اشیا Yolo1 را مشخص می‌کند، مقدار پیش‌فرض این آرگمان `'weights/yolov5m.pt'` بوده که که در صورت نیاز می‌توان آن را تغییر داد و در صورتی که اسمی از آرگمان weights-detector در هنگام اجرای برنامه آورده نشود، همان مقدار پیش‌فرض در نظر گرفته می‌شود.
۲. آرگمان weights-sign که این آرگمان آدرس وزن‌های شبکه‌ی Yolo2 که مربوط به تشخیص تابلوها بود را تعیین می‌کند، و مقدار پیش فرض آن نیز `'weights/best_sign2.pt'` می‌باشد.
۳. آرگمان disp-detector که شامل وزن‌های مدل مورد نظر شبکه‌ی تشخیص دیسپریتی می‌باشد و مقدار پیش‌فرض آن نیز `'weights/model_full.pth'` می‌باشد.
۴. آرگمان lane-detector که نام شبکه‌ی تشخیص خطوط خیابان را مشخص می‌کند، مقدار پیش-فرض آن `'culane'` می‌باشد و در صورت نیاز به نوع‌شبکه‌ی `curvelane` می‌توان آن عبارت را وارد آرگمان کرد.
۵. آرگمان culane-model که آدرس وزن‌های شبکه‌ی culane را دارا می‌باشد و با اضافه کردن آدرس می‌توان وزن‌های دلخواه را برای شبکه تعیین کرد.
۶. آرگمان curvelane-model که آدرس وزن‌های شبکه‌ی curvelane را دارا می‌باشد و با اضافه کردن آدرس می‌توان وزن‌های دلخواه را برای شبکه تعیین کرد.
۷. آرگمان video که از مهم‌ترین آرگمان‌ها می‌باشد، آدرس فایل ویدئوی ورودی به نرم‌افزار اصلی را مشخص می‌کند. مقدار پیشفرض این آرگمان test.mov می‌باشد و در صورت نیاز به پردازش ویدئوهای دیگر، باید اسم آن ویدئو را در ادامه‌ی آرگمان video حین اجرا کردن نرم‌افزار اضافه کرد.

[266] Dash
[267] Binary

۸. آرگمان save در صورت وارد شدن ویدئوی خروجی را ذخیره می‌نماید. برای ذخیره‌سازی ویدئوی خروجی کافی است عبارت save-- به خط فرمان اضافه شود.

۹. آرگمان noshow : در صورت فعال‌سازی، خروجی پردازش شده نمایش‌داده نمی‌شود. این حالت برای وقتی مناسب است که فقط نیاز است ویدئوی خروجی ذخیره شود و نیازی به دیدن همزمان خروجی نمی‌باشد.

۱۰. آرگمان frame-drop تعداد فریم‌هایی که پس از هر پردازش چشم‌پوشی می‌شوند را تعیین می‌کند. این آرگمان زمانی مفید است که نخواهیم تمامی فریم‌های وردئو پردازش شوند.

۱۱. آرگمان output-fps تعداد فریم بر ثانیه‌ی ویدئوی خروجی را مشخص می‌کند.

۱۲. آرگمان fps در صورت فعال بودن، یک متن که fps در حال اجرا شدن برنامه را در خود دارد در گوشه‌ی بالا سمت چپ ویدئو نمایش می‌دهد.

۱۳.آرگمان output-name نام ویدئوی خروجی را مشخص می‌کندو مقدار پیش‌فرض این آرگمان output.mov می‌باشد که بهتر است با توجه به نسبت به ویدئوی ورودی تعیین شود.

۱۴.آرگمان depth-mode روش تشخیص عمق تصاویر را مشخص می‌کند که بر مبنای kitti و یا cityscape باشد که مقدار پیش‌فرض بر روی kitti قرار داده شده است.

۱۵. آرگمان mode حالت اصلی نرم‌افزار را مشخص می‌کند، حالت های ۱، ۲ و۳ به ترتیب متعلق به روز، شب و حالت شلوغی می‌باشد و حالت پیش‌فرض ۱ در نظر گرفته شده است.

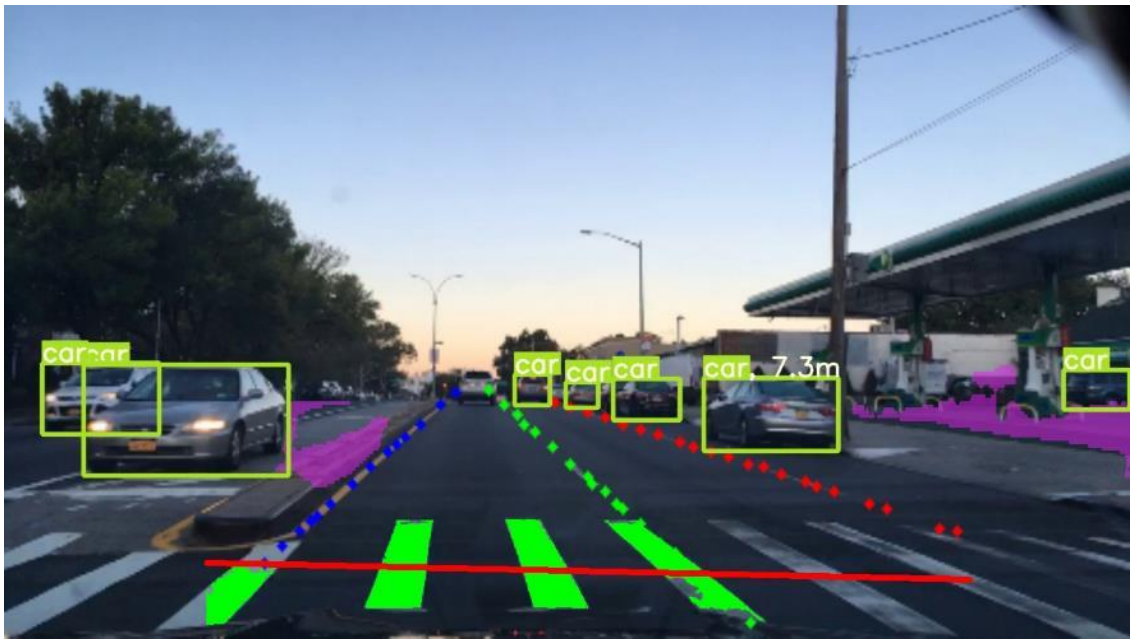
۱۶. آرگمان save-frames در صورت فعال بودن تک تک فریم‌های خروجی را در پوشه‌ای به اسم خروجی که در آرگمان ۱۳ مشخص شد ذخیره می‌کند.

۳-۷-۳- خروجی نرم افزار برای ویدئو تست اول

این ویدئو که در حالت عادی روز تصویربرداری شده است، دارای mode یک می باشد و به صورت پیش‌فرض می‌توان از آن خروج گرفت. چند نمونه از خروجی‌های این ویدئو را می‌توان در ادامه مشاهده کرد.

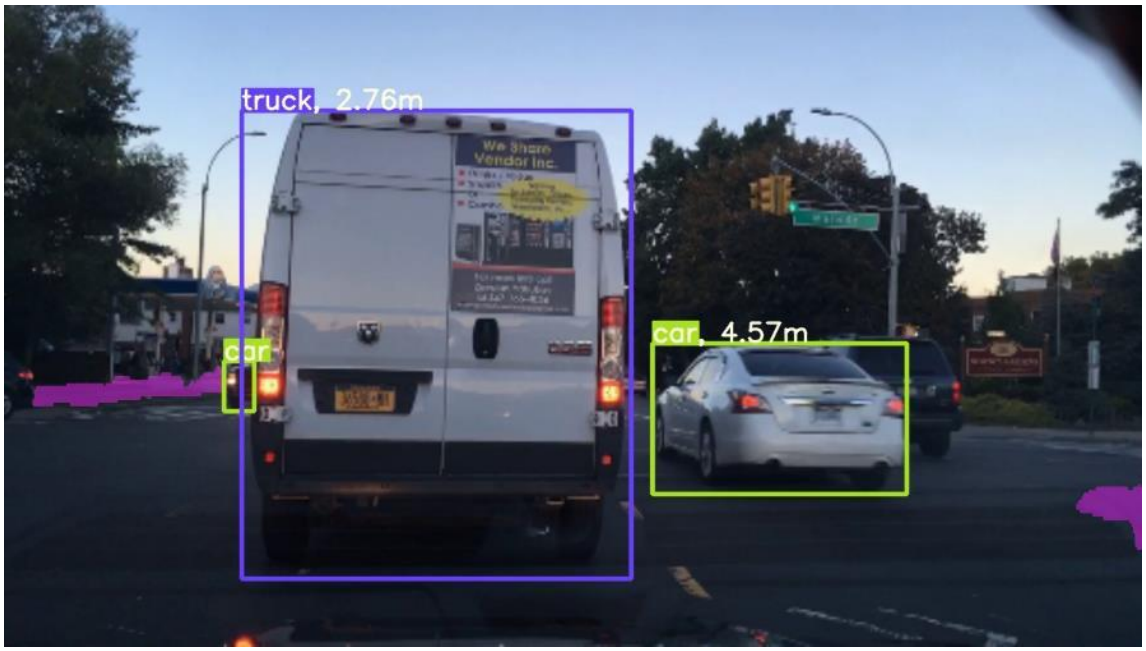
شکل ۱۱۱-۳ نمونه‌ی شماره‌ی اول از خروجی ویدئوی اول

شکل ۱۱۲-۳ نمونه‌ی شماره دوم از خروجی ویدئوی اول



شکل ۳-۱۱۳ نمونه‌ی شماره سوم از خروجی ویدئوی اول

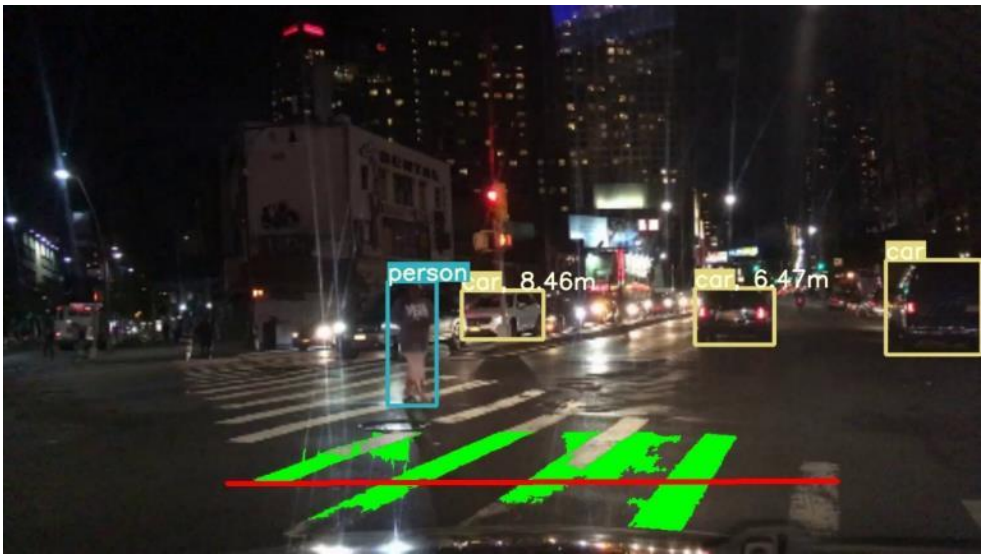
شکل ۳-۱۱۴ نمونه‌ی شماره چهارم از خروجی ویدئوی اول



شکل ۱۱۵-۳ نمونه‌ی شماره پنجم از خروجی ویدئوی اول

۴-۷-۳- خروجی نرم افزار برای ویدئو تست دوم

برای تهیه‌ی خروجی این ویدئو، لازم است که به کد اصلی داخل خط فرمان، آرگمان mode 2 -- اضافه شود، زیرا ویدئو در حالت شب می‌باشد. در ادامه دو مورد از خروجی‌های مربوط به ویدئوی شماره ۲ مشاهده می‌شود.

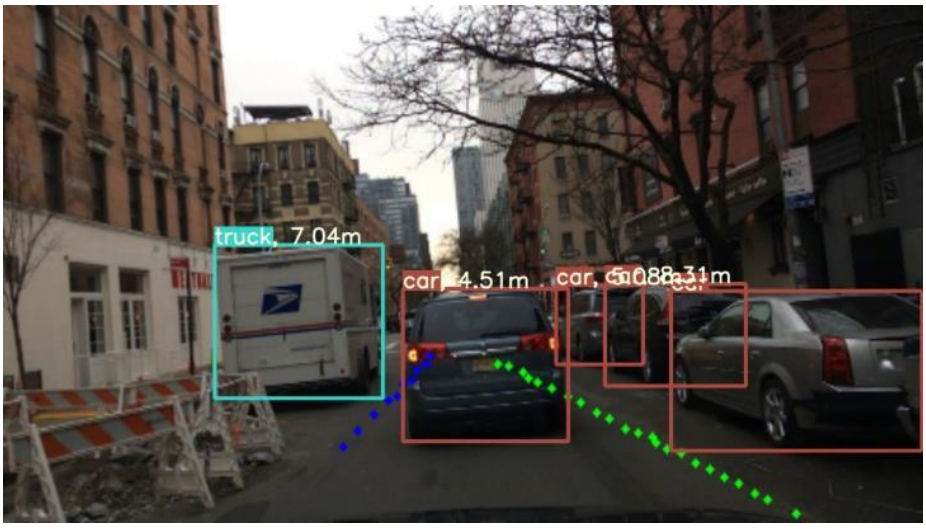


شکل ۳-۱۱۶ نمونه‌ی شماره اول از خروجی ویدئوی دوم

شکل ۳-۱۱۷ نمونه‌ی شماره دوم از خروجی ویدئوی دوم

۵-۷-۳- خروجی نرم افزار برای ویدئو تست سوم

برای تهیه‌ی خروجی این ویدئو، لازم است که به کد اصلی داخل خط فرمان، آرگمان mode 3 -- اضافه شود که پارامترهای برنامه متناسب با حالت شلوغی تعیین شوند. در ادامه چند مورد از خروجی‌های مربوط به ویدئوی شماره ۲ مشاهده می‌شود.



شکل ۳-۱۱۸ نمونه‌ی شماره اول از خروجی ویدئوی سوم

شکل ۳-۱۱۹ نمونه‌ی شماره دوم از خروجی ویدئوی سوم

۴- پارک خودکار

۴-۱ دینامیک و سینماتیک خودرو

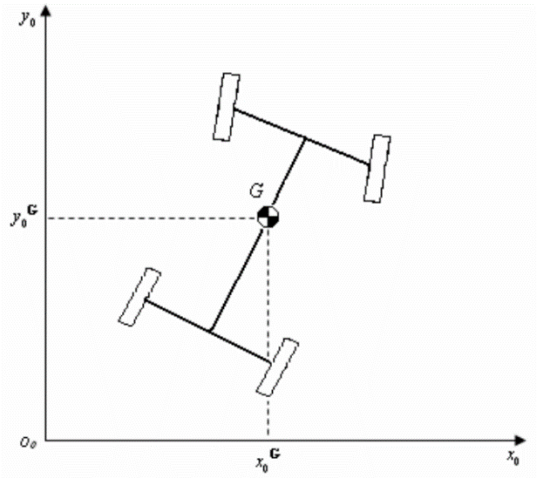
در این بخش، مدل‌سازی سینماتیک و دینامیک خودرو انجام شده است. در بخش اول، سینماتیک خودرو تشریح گردیده است. در بخش دوم، معادلات دینامیکی خودرو و شبیه‌سازی رفتار آن در دو جهت طولی و عرضی آورده شده است.

۴-۱-۱- سینماتیک خودرو

معادلات سینماتیکی، معادلاتی هستند که حرکت وسیله نقلیه را بیان می‌کنند. در این معادلات عوامل ایجاد کننده حرکت، شامل نیروها و گشتاورها بیان نمی‌شود. در ابتدا دستگاه‌های مختصاتی که برای بیان حرکت مورد نیاز است، ارائه شده است. پس از آن تشریح معادلات سینماتیکی حرکت خودرو آورده شده است. محورها و دستگاه‌های مختصات مورد استفاده در این بخش بر اساس استاندارد SAE J670e بیان شده است[۲۸۰].

۴-۱-۱-۱- دستگاه مختصات ثابت به زمین

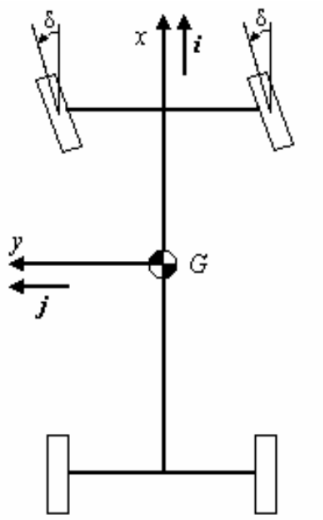
این دستگاه مختصات، متصل به زمین و به صورت ثابت است. مبدا مختصات با O_0 و محورهای اصلی در آن با مولفه‌های $[x_0, y_0, z_0]$ نشان داده می‌شود. این محورهای اصلی با نام محورهای زمین شناخته می‌شود. در شکل ۱-۳، صفحه افقی این دستگاه مختصات نشان داده شده است. محور z_0 به صورت عمود بر صفحه و با رعایت قانون دست راست تعیین می‌شود[۲۸۱].



شکل ۱-۴ صفحه افقی دستگاه مختصات زمین[۲۸۱]

۲-۱-۱-۴- دستگاه مختصات بدنی

این دستگاه مختصات نیز دستگاه راست‌گردی است که بر روی مرکز ثقل جسم متحرک سوار بوده و محور طولی آن در جهت حرکت رو به جلو خودرو است. محور ارتفاع آن در جهت رو به بالا بوده و محور عرضی آن با توجه به قانون دست راست به سمت چپ وسیله خواهد بود. بردارهای یکه این دستگاه با [i j k] تعیین می‌شود. در شکل ۲-۱، صفحه افقی این دستگاه نشان داده شده است. محور سوم آن، عمود بر صفحه و به سمت بیرون خواهد بود[۲۸۱].



شکل ۲-۴ صفحه افقی دستگاه مختصات زمین[۲۸۱]

۳-۱-۱-۴- تبدیل دستگاه‌های مختصات

با توجه به دستگاه‌های مختصات تعریف شده، معادلع ماتریس انتقال از دستگاه بدنی به دستگاه زمین در رابطه ۱-۳ آورده شده است[۲۸۲].

$$\mathbf{R_b^e} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (۴-۱)$$

۴-۱-۱-۴- معادلات سینماتیک خودرو

مدل سینماتکی با توجه به توضیحات گفته شده، روابط سینماتیکی خودرو در معادلات ۲-۴ تا ۴-۴ آورده شده است[۲۸۳].

$$\dot{X}_G = u\cos\psi - v\sin\psi \quad (۴-۲)$$

$$\dot{Y}_G = u\sin\psi + v\cos\psi \quad (۴-۳)$$

$$\dot{\psi}_G = r \quad (۴-۴)$$

این معادلات شامل مولفه‌های زیر است:

u: سرعت در راستای طولی بیان شده در دستگاه بدنی

v: سرعت در راستای عرضی بیان شده در دستگاه بدنی

r: سرعت دورانی در راستای ارتفاعی و بیان شده در دستگاه بدنی

$\dot{X}_G$: مولفه طولی موقعیت مرکز جرم بیان شده در دستگاه زمین

$\dot{Y}_G$: مولفه عرضی موقعیت مرکز جرم بیان شده در دستگاه زمین

$\dot{\psi}_G$: مولفه زاویه موقعیت مرکز جرم بیان شده در دستگاه زمین

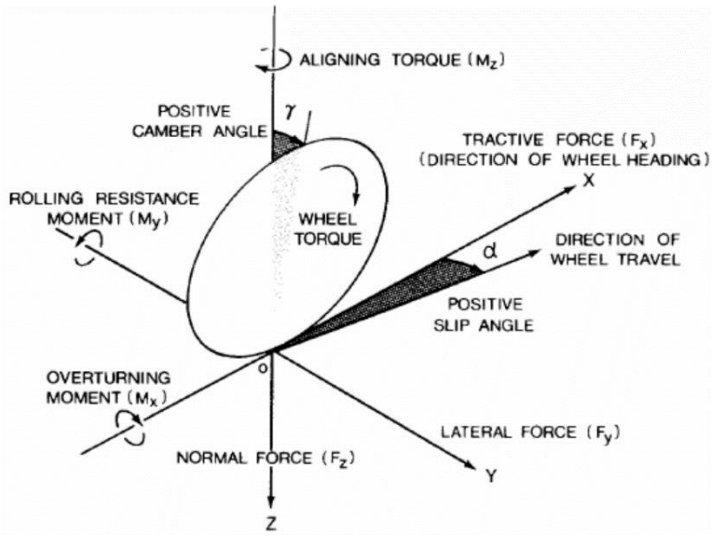
۲-۱-۴- نیروهای خارجی وارد بر خودرو

نیروهای خارجی وارد بر خودرو شامل نیروهای وارد بر چرخ، نیروی مقاومت هوا و ... است. در این بخش، این نیروها به صورت مختصر توضیح داده شده است.

۱-۲-۱-۴- نیروهای وارد بر چرخ

وسایل نقلیه موتوری، اغلب دارای تایرهای پنوماتیکی هستند. این تایرها مسئولیت انتقال قدرت تولید شده در موتور به زمین را دارند. نیروهای خارجی وارد بر تایر و دستگاه مختصات بدنی آن در شکل ۳-۴ نشان داده شده است. جهت این نیروها و گشتاورها بر اساس استاندارد انجمن مهندسان خودرو[۲۶۸] است[۲۸۴].

[268] SAE

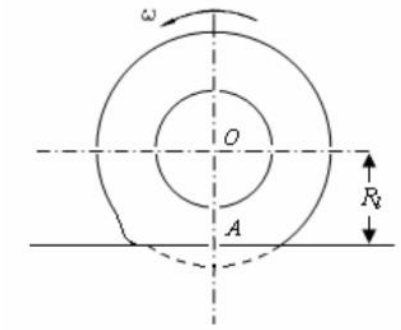


شکل ۳-۴ نیروها و گشتاورهای وارد بر تایر[۲۸۴]

مبدا مختصات دستگاه بدنی تایر، بر مرکز تماس تایر با زمین منطبق است. محور x آن نقطه تلاقی صفحه چرخ و صفحه زمین و جهت مثبت آن، به سمت حرکت رو به جلو است. محور z عمود بر زمین و به سمت بیرون است. محور y نیز بر اساس قاعده دست راست بدست آمده است. گشتاور تولید شده به وسیله موتور با نام گشتاور چرخ نشان داده شده است. اختلاف زاویه بین محور افقی دستگاه مختصات تایر با جهت سرعت مرکز تایر با α نشان داده شده است. این زاویه زاویه لغزش نامیده می‌شود. هم‌چنین، زاویه تحدب که با γ نشان داده می‌شود، زاویه بین صفحه تایر و صفحه xz است. در تمامی بخش‌های بعدی از زاویه تحدب تایر صرف نظر شده است[۲۸۵].

۱-۱-۲-۱-۴- شعاع دوران

با توجه به آن‌که، تایر پنوماتیکی رفتاری متفاوت از چرخ صلب دارد، مرکز دوران تایر، دقیقا در نقطه تماس تایر با زمین قرار ندارد. در شکل ۴-۴ نقطه تماس با زمین با A نشان داده شده است.



شکل ۴-۴ مرکز دوران تایر[۲۸۳]

هم‌چنین، بر اساس مکانیزم عملکردی تایر پنوماتیکی، سرعت دورانی تایر کمتر از سرعت دورانی چرخ صلب با بار یکسان است. به همین دلیل شعاع دوران در تایر پنوماتیکی، بیشتر از فاصله مرکز تایر تا سطح زمین و کمتر از شعاع تایر خواهد بود. شعاع دوران تایر به عنوان نسبت سرعت خطی و دورانی تایر به تعریف می‌شود. این مفهوم در رابطه ۵-۴ نشان داده شده است[۲۸۵].

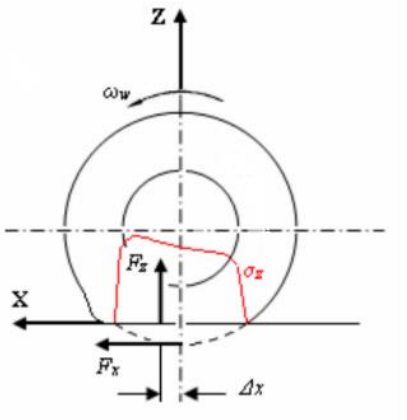
$$u_{wheel} = R_r \omega_{wheel} \Rightarrow R_l < R_r < R \qquad (۴-۵)$$

۲-۱-۲-۱-۴- نیروی مقاوم دوران

در صورت صلب بودن تایر و زمین، بدون اعمال نیروی خارجی، تایر وضعیت قبلی خود را حفظ می‌کند. در حالی که در حالت حرکت واقعی سیستم، رفتار مکانیکی تایر به صورت ایده‌آل الاستیک نیست و دارای مقدار کمی کرنش پلاستیک است. به همین دلیل، در هنگام این کرنش پلاستیک، نیاز به صرف انرژی وجود دارد. این عامل، باعث به وجود آمدن نیروی مقاوم در برابر دوران می‌گردد. به صورت خلاصه، این نیروی مقاوم به صورت ضریبی از نیروی وارد به چرخ در راستای عمود بر سطح زمین خواهد بود که در رابطه ۶-۳ نشان داده شده است[۲۸۳].

$$F_{rolling} = f_r F_z \qquad (۴-۶)$$

این رابطه بر اساس استاندارد SAEJ670e بدست آمده است. هم‌چنین، دلیل این امر، جابه‌جایی نیروی وارد به چرخ در راستای عمود بر چرخ بر اساس توزیع بار وارد بر آن و لزوم حفظ تعادل چرخ است. این رفتار در شکل ۵-۴ نشان داده شده است[۲۸۳].



شکل ۵-۴ توزیع بار در تایر[۲۸۳]

ضریب f_r به صورت تجربی تعیین می‌شود. این ضریب به عوامل مختلفی مانند سرعت افقی تایر، فشار هوا، نیروهای عمودی و... وابسته است. مهم‌ترین عامل در تعیین این ضریب، سرعت افقی تایر است. با صرف نظر کردن از باقی عوامل، تقریب این ضریب در رابطه ۴-۷ آورده شده است[۲۸۵].

$$f_r = f_0 + K\,u^2 \qquad (۴-۷)$$

این رابطه دارای مولفه‌های زیر است:

u: سرعت خودرو در راستای طولی بیان شده در دستگاه بدنی

f_0, K: ضرایب تجربی ثابت

با توجه به موارد بیان شده، محل وارد شدن نیروی عمودی بر تایر از معادله تعادل تایر به دست می‌آید[۲۸۳]:

$$\sum M_y = I_{wheel}\,\ddot{\theta} = 0 \;\Rightarrow\; F_{rolling} R_r + F_z\,\Delta x = 0 \qquad (۴-۸)$$

$$\Delta x = f_r\,R_r \qquad (۴-۹)$$

۳-۱-۲-۱-۴- نیروی مجانبی تایرها

برای تحلیل دینامیک خودرو، نیاز به تعریف رفتار تایرها در راستای عرضی وجود دارد. نیروی وارد بر تایر در راستای عرضی، تابع عوامل مختلفی است. در رابطه ۳-۱۰، این عوامل آورده شده است[۲۸۶].

$$\mathrm{F_y} = \mathrm{Y_p}(\alpha, \gamma, \mathrm{F_x}, \mathrm{F_z}) \qquad (۴-۱۰)$$

این رابطه شامل مولفه‌های زیر است:

α: زاویه لغزش تایر

γ: زاویه تحدب تایر

$\mathrm{F_x}$: نیروی افقی وارد بر تایر

$\mathrm{F_z}$: نیروی عمودی وارد بر تایر

مدل‌های مختلفی برای بیان رفتار تایر در راستای مجانبی وجود دارد. در این پژوهش، از مدل خطی برای بیان رفتار تایرها استفاده شده است[۲۸۰]. رابطه ۴-۱۱ نیروی عمودی وارد بر تایرها را نشان می‌دهد:

$$F_y = C_i \, \alpha_i \qquad (۴-۱۱)$$

این رابطه شامل مولفه‌های زیر است:

α: زاویه لغزش تایر

C: ضریب ثابت تجربی

i: دارای دو مقدار ۱ و ۲ به ترتیب برای نشان دادن محورها جلو و عقب خودرو است

مدل‌های دیگری برای شبیه‌سازی رفتار تایر از جمله مدل خطی به همراه خلاصی و مدل غیر خطی وجود دارد که در این پژوهش مورد استفاده قرار نگرفته است[۲۸۶].

۴-۱-۲-۲- نیروی مقاومت هوا

نیروی مقاومت هوا، نیرویی است که در جهت مخالف حرکت خودرو وارد می‌شود. این نیرو، با استفاده از رابطه ۴-۱۲ تعریف می‌شود[۲۸۳].

$$F_{aero} = \frac{1}{2} S C_x \, p_{air} u^2 \qquad (۴-۱۲)$$

این رابطه شامل مولفه‌های زیر است:

S: سطح مقطع خودرو

C_x: ضریب درگ هوا

p_{air}: چگالی هوا

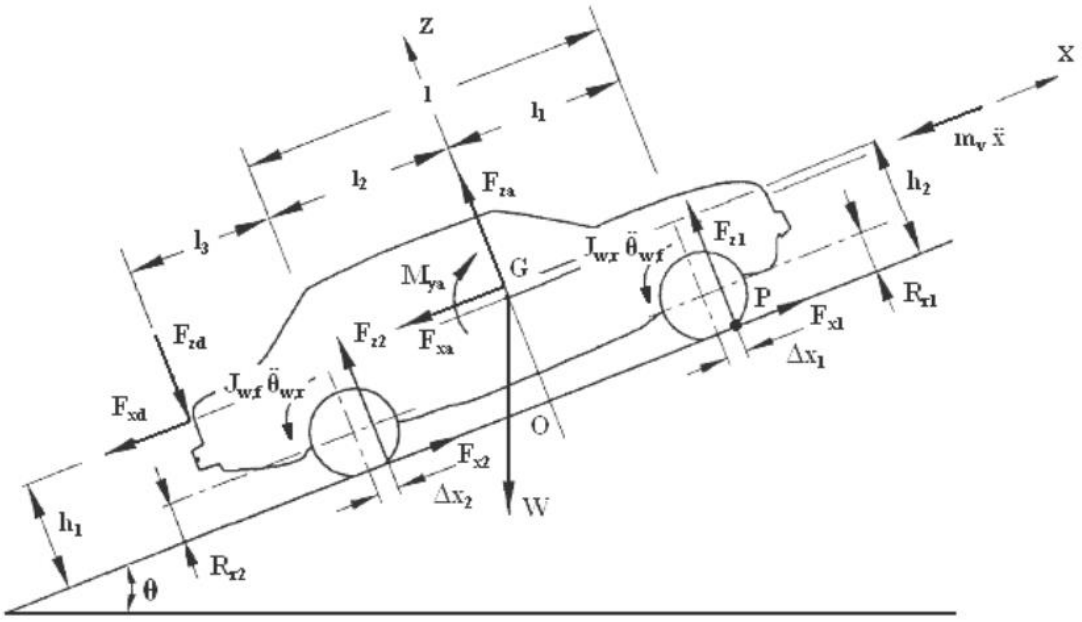
u: سرعت افقی خودرو

۳-۱-۴- مدل‌سازی دینامیک در راستای طولی

در این بخش مدل‌سازی دینامیک خودرو در راستای طول انجام می‌شود. اولین قدم برای درک بهتر رفتار دینامیکی خودرو در راستای طولی، ساخت یک مدل ریاضی با تقریب مناسب از رفتار سیستم فیزیکی است. در ابتدا معادلات دینامیکی حرکت در راستای طولی تبیین می‌شود. پس از آن مدل‌سازی از سیستم تولید گشتاور در موتور و سیستم انتقال آن به چرخ‌ها تشریح می‌گردد.

۱-۳-۱-۴- معادلات دینامیکی حرکت در راستای طولی

به طور کلی شکل ۶-۴، نیروی‌ها و گشتاورهای وارد شده بر خودرو در راستای طولی را نشان می‌دهد:



شکل ۶-۴ نیروها و گشتاورهای وارد بر خودرو در راستای طولی[۲۸۳]

مولفه‌های نشان داده شده در شکل ۶-۴ به شرح زیر است:

W: وزن خودرو

γ: زاویه سطح شیب‌دار

m_v: جرم خودرو

J_{wf} ، J_{wr} : ممان اینرسی چرخ های عقب و جلو

$\ddot{x}$: شتاب طولی خودرو

$\ddot{\theta}$: شتاب دورانی خودرو در راستای محور عرضی

l: فاصله‌ی محور جلو و عقب

l_1: فاصله محور جلو از مرکز جرم

l_2: فاصله محور عقب از مرکز جرم

l_3: فاصله کشنده از محور عقب

F_{xa}, F_{za}, M_{ya}: نیروها و گشتاورهای مقاومت هوا به ترتیب در راستای افقی، عمودی و عرضی

h_1, h_2: فاصله‌ی نقطه اعمال نیروهای اینرسی و مقاومت هوا تا سطح زمین

$\Delta x_1, \Delta x_2$: فاصله‌ی نقطه اعمال نیروهای عمودی چرخ و مرکز آن

R_{r1}, R_{r2}: شعاع دوران چرخ‌های جلو و عقب

F_{x1}, F_{x2}: نیروی افقی وارد بر چرخ های جلو و عقب

F_{z1}, F_{z2}: نیروی عمودی وارد بر چرخ‌های جلو و عقب

F_{xd}, F_{zd}: نیروهای وارد بر خودرو از طرف کشنده

معادله دینامیکی تعادل گشتاورها در جهت عمود بر صفحه و در نقطه P که در شکل ۶-۴ نشان داده شده، به صورت زیر خواهد بود[۲۸۳]:

$$\begin{aligned} &m_v \ddot{x} h_1 + 2J_{wf}\ddot{\theta}_{wf} + 2J_{wr}\ddot{\theta}_{wr} - 2F_{z2}(l - \Delta x_1 + \Delta x_2) + \\ &F_{zd}(l + \Delta x_1 + l_3) + F_{xd}h_1 + F_{xa}h_2 - F_{za}(l_1 + \Delta x_1) \\ &- M_{ya} + W\sin(\gamma)h_1 + W\cos(\gamma)(l_1 + \Delta x_1) = 0 \end{aligned} \qquad (۴-۱۳)$$

بر اساس معادله ۱۳-۴، فرضیات ساده کننده زیر در نظر گرفته شده است:

$$F_{za} = M_{ya} = F_{zd} = F_{xd} = 0 \qquad (۴-۱۴)$$

$$J_{wf} = J_{wr} = J_w \qquad (۴-۱۵)$$

$$\Delta x_1 = \Delta x_2 = U \qquad (۴-۱۶)$$

$$\theta_{wf} = \theta_{wr} = \theta_w \quad (۴-۱۷)$$

$$W\sin(\gamma) = W_x \quad (۴-۱۸)$$

$$W\cos(\gamma) = W_z \quad (۴-۱۹)$$

$$R_{r1} = R_{r2} = R_r \quad (۴-۲۰)$$

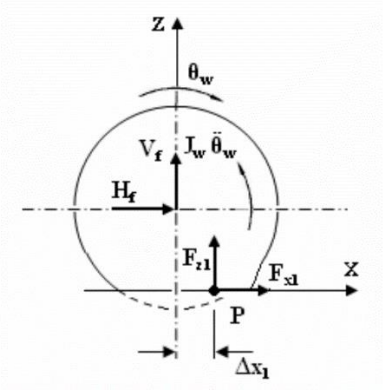
با در نظر گرفتن فرضیات ساده کننده و مرتب کردن معادله ۴-۱۳ برای نیروی عمودی وارد بر چرخ‌های محور عقب خواهیم داشت[۲۸۳]:

$$F_{z2} = \frac{1}{2l}\left[m_v\ddot{x}h_1 + 4J_w\ddot{\theta}_w + F_{xa}h_2 + W_x h_1 + W_z(l_1 + U)\right] \quad (۴-۲۱)$$

با نوشتن معادله تعادل نیروها در راستای z مرتب کردن آن برای نیروی عمودی وارد بر چرخ جلو خواهیم داشت[۲۸۳]:

$$F_{z1} = \frac{W_z - 2F_{z2}}{2} \quad (۴-۲۲)$$

در معادله‌ی ۴-۲۲ نیز، فرضیات ساده‌سازی رعایت شده است. در شکل ۴-۷ دیاگرام آزاد چرخ جلو آورده شده است .



شکل ۴-۷ دیاگرام آزاد نیروها در چرخ جلوی خودرو[283]

معادله تعادل گشتاورها در راستای عمود بر صفحه و برای چرخ جلو که متحرک است، در رابطه ۲۳-۴ آورده شده است [۲۸۳].

$$J_w\ddot{\theta}_w + = F_{z1}U + F_{x1}R_r = 0 \qquad (۴-۲۳)$$

با مرتب کردن این رابطه برای نیروی افقی وارد بر چرخ جلو خواهیم داشت:

$$F_{x1} = -\frac{J_w\ddot{\theta}_w + F_{z1}U}{R_r} \qquad (۴-۲۴)$$

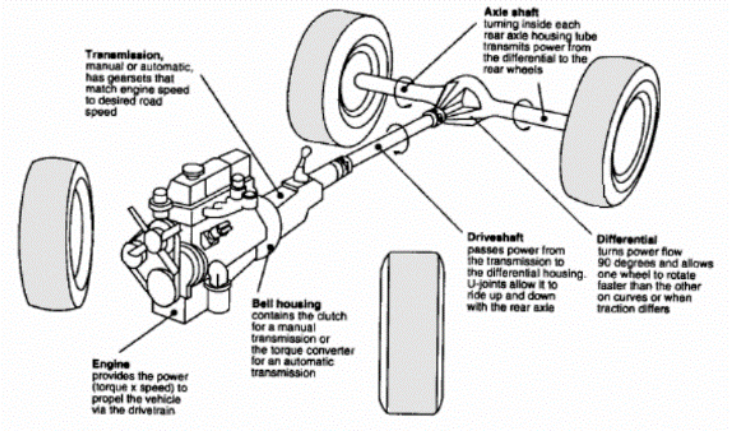
رابطه ۲۵-۳ معادله دینامیکی حرکت در راستای افقی را نشان می‌دهد. با مرتب کردن برای نیروی افقی وارد بر چرخ عقب به معادله ۲۶-۴ خواهیم رسید[۲۸۳].

$$m_v\ddot{x} - 2F_{x1} - 2F_{x2} + F_{xa} - F_{xd} + W_x = 0 \qquad (۴-۲۵)$$

$$F_{x2} = \frac{1}{2}(m_v\ddot{x} - 2F_{x1} + F_{xa} + W_x) \qquad (۴-۲۶)$$

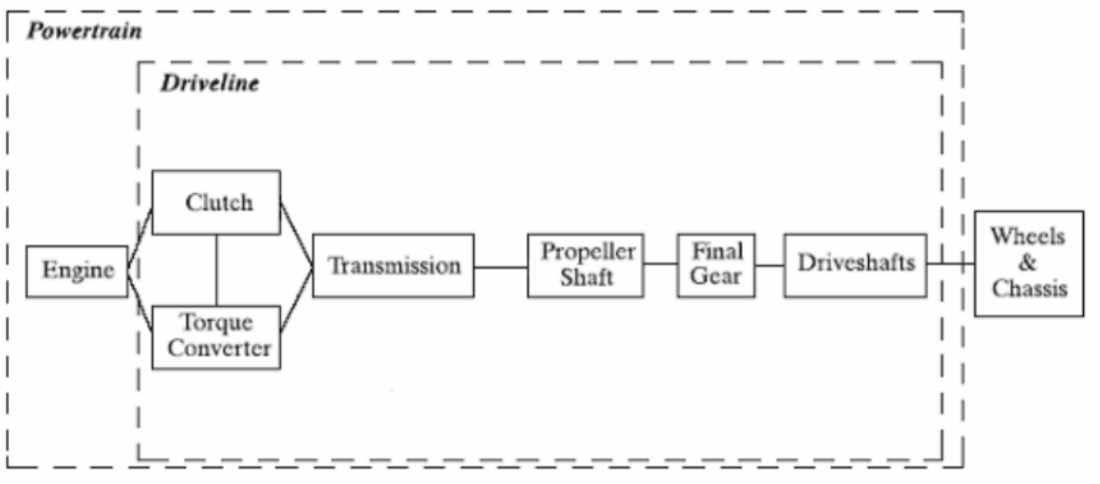
۲-۳-۱-۴- مدل‌سازی سیستم تولید و انتقال قدرت

برای مدل‌سازی رفتار کلی دینامیکی سیستم قدرت خودرو که شامل بخش‌های مختلفی از جمله سیستم تولید قدرت، انتقال قدرت، شاسی و... است، روش‌های مختلفی وجود دارد. در این پژوهش، رفتار دینامیکی سیستم قدرت خودرو بر اساس سرعت خودرو و میزان فشرده شده پدال گاز انجام می‌شود. در شکل ۸-۴، نمای کلی از سیستم قدرت خودرو نشان داده شده است.



شکل ۸-۴ نمای کلی از سیستم قدرت خودرو[۲۸۴]

با توجه به شکل ۸-۴، دیاگرام سیستم انتقال قدرت خودرو به صورت زیر خواهد بود:



شکل ۹-۴ اجزای سیستم قدرت خودرو[۲۸۷]

برای عملکرد خودرو دو مولفه محدود کننده وجود دارد. مولفه اول حداکثر نیروی قابل انتقال به جاده از طریق تایرها بدون لغزش است. مولفه دوم حداکثر نیروی قابل ایجاد توسط سیستم تولید و انتقال قدرت با گشتاور موتور است. هدف از مدل‌سازی‌های انجام گرفته در این بخش، شناسایی و یافتن مهم‌ترین مولفه‌های موثر در سیستم قدرت خودرو است. بر همین اساس تمامی اجزای این بخش به صورت صلب در نظر گرفته شده است. ورودی مدل ساخته شده، میزان فشردگی پدال گاز و خروجی آن سرعت دورانی چرخ‌ها خواهد بود[۲۸۳].

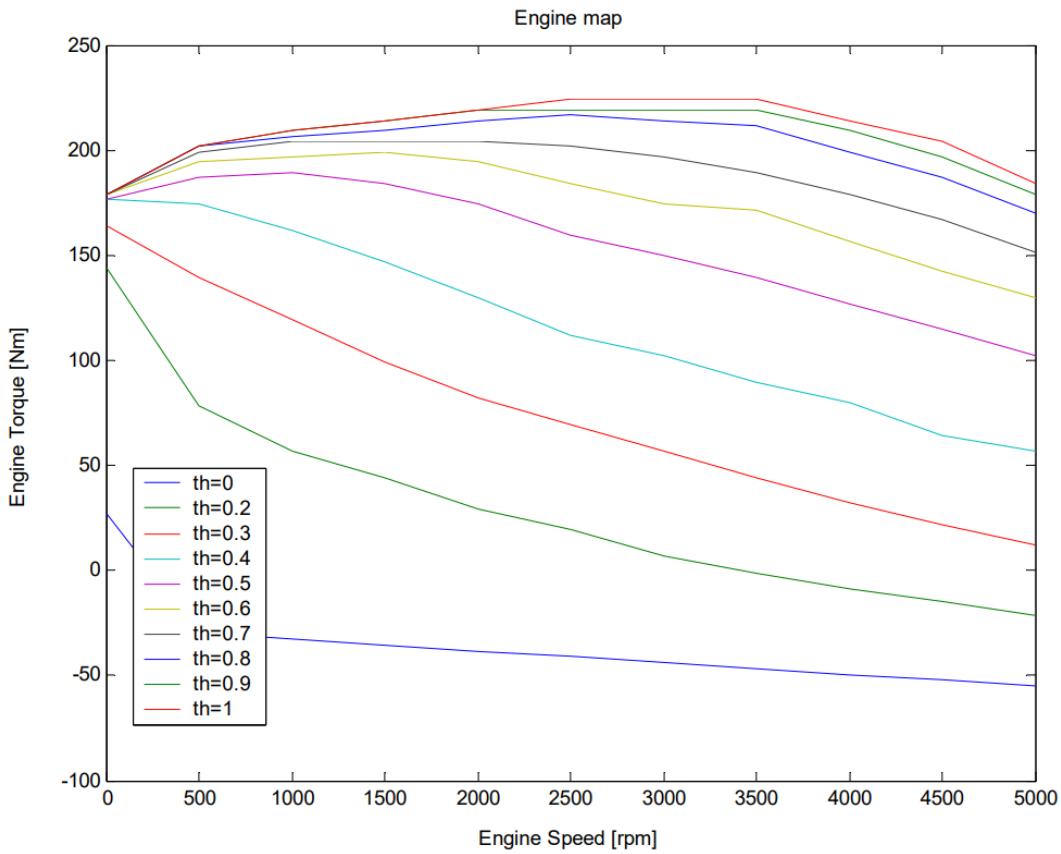
مدل ساخته شده شامل دو بخش خواهد بود:

- مدل‌سازی موتور
- مدل‌سازی خط انتقال قدرت

در ادامه به بررسی مدل‌سازی ریاضی هر یک از این بخش‌ها پرداخته شده است.

۱-۲-۳-۱-۴- مدل‌سازی موتور خودرو

جدای از کنترل پدال گاز، قدرت تولید شده به وسیله موتور احتراق داخلی، وابسته به سرعت دورانی آن است. به همین‌دلیل، عملکرد موتور احتراق داخلی به وسیله یک دسته از منحنی‌ها شامل محور افقی سرعت دورانی موتور و محور عمودی گشتاور تولید شده توسط موتور بیان می‌شود. هر یک از منحنی‌ها، برای یک میزان فشرده شدن پدال گاز رسم می‌شود. در شکل ۱۰-۴، نمونه‌ای از این منحنی‌ها رسم شده است[۲۸۳].



شکل ۱۰-۴ نمونه‌ای از منحنی‌های مشخصه موتور[۲۸۳]

میزان فشرده شدن پدال در منحنی‌های بالا به دارای مقادیر بین صفر(کاملا آزاد) و یک (کاملا فشرده) است. مولفه‌های منحنی‌های مشخصه شکل ۱۰-۴ به صورت مستقل نبوده و این مولفه‌های سرعت و گشتاور به وسیله رابطه ۲۷-۳ که قدرت تولیدی توسط موتور را نشان می‌دهد، به هم وابسته هستند[۲۸۳].

$$P_e = T_e \omega_e \qquad (۴-۲۷)$$

موتور به وسیله سیستم انتقال قدرت به چرخ‌ها متصل می‌گردد. رابطه ۲۸-۴ ارتباط بین سرعت موتور و سرعت دورانی چرخ‌ها را نشان می‌دهد[۲۸۳]:

$$\omega_{enigne} = \tau_c \, \tau_d \, \omega_{wheel} \qquad (۴-۲۷)$$

مولفه‌های τ_d و τ_c نسبت‌های انتقال قدرت چرخ‌دنده‌های گیربکس هستند. منحنی مشخصه موتور، گشتاور تولیدی به وسیله موتور را به عنوان تابعی از سرعت موتور و میزان فشرده بودن پدال از بیان می‌کند. در حالت عملکرد مناطق میانی منحنی‌ها، می‌توان میزان گشتاور تولیدی توسط موتور را با رابطه خطی ۲۸-۴ برحسب میزان فشردگی پدال گاز نشان داد[۲۸۸].

$$T_{enigne} = T_{emax}\,\alpha + T_{emin}\,(1-\alpha) \qquad (۴-۲۸)$$

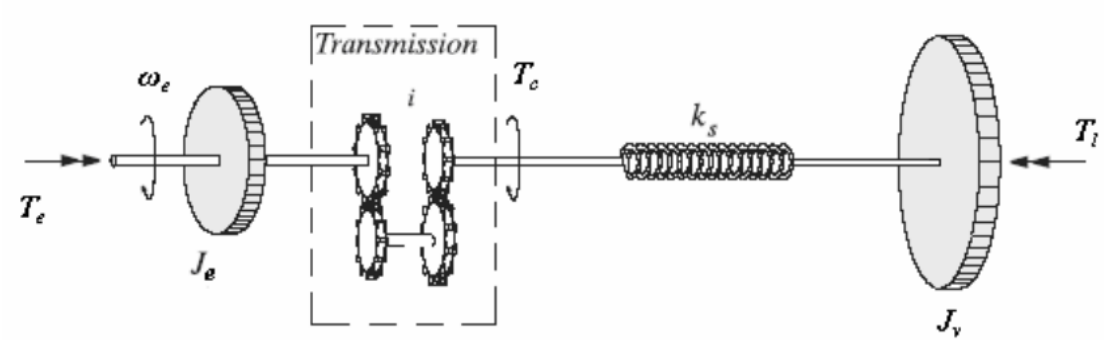
در رابطه بالا T_{emax} و T_{emin} به ترتیب حداکثر و حداقل گشتاور تولیدی موتور را نشان می‌دهند.

۴-۱-۳-۲-۲- مدل‌سازی رفتار راننده

رانندگان واقعی سرعت خودرو را با استفاده از فشردن پدال گاز جهت شتاب‌گیری مثبت و فشردن پدال ترمز برای شتاب‌گیری منفی انجام می‌دهند. در این پژوهش، مدل‌سازی پدال ترمز انجام نشده است. به همین دلیل، شتاب منفی، با کاهش فشردگی پدال گاز در نظر گرفته شده است.

۴-۱-۳-۲-۳- مدل‌سازی سیستم انتقال قدرت

در این قسمت، هدف شبیه‌سازی سیستم انتقال قدرت مکانیکی خودرو به صورت یک سیستم کاهیده است. به همین منظور از قانون کار و انرژی و قانون انرژی جنبشی معادل استفاده شده است. سیستم انتقال قدرت در شکل ۴-۱۱ نشان داده شده است.

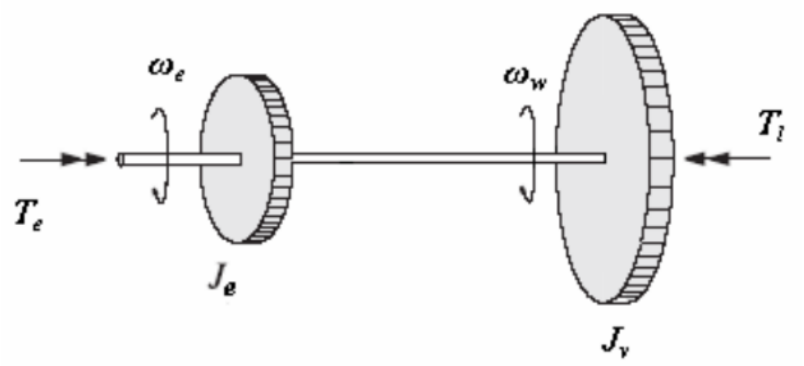


شکل ۴-۱۱ شماتیک سیستم انتقال قدرت خودرو در هنگام درگیری چرخ‌دنده‌ها[۲۸۳]

در شبیه‌سازی سیستم انتقال قدرت از وجود کلاچ و تاثیرات آن صرف نظر شده است. ورودی سیستم شکل ۴-۱۱ گشتاور و سرعت دورانی موتور و خروجی آن سرعت دورانی چرخ ها و گشتاور آن‌ها یا گشتاور بار است. برای ساده‌سازی مدل نظری، فرضیات زیر در نظر گرفته شده است[۲۸۳]:

- صرف نظر کردن از کلاچ و عملکرد آن
- صلب بودن تمامی اجزای سیستم انتقال قدرت
- درگیری و عدم درگیری چرخ‌دنده‌ها به صورت پایدار و صرف نظر از عملکرد حالت گذرای آن‌ها

با توجه به فرضیات مذکور، می‌توان سیستم انتقال قدرت در هنگام درگیری یک دنده خاص را به صورت شکل ۴-۱۲ معادل‌سازی کرد.



شکل ۱۲-۴ سیستم معادل کاهیده انتقال قدرت[۲۸۳]

بر همین اساس، می‌توان با استفاده از رابطه ۲۹-۴، رابطه اساسی بین گشتاور تولیدی موتور و انرژی جنبشی خودرو را با استفاده از قانون دوم نیوتن بیان کرد[۲۸۸-۲۸۹].

$$P_e - P_l = \frac{d\tau}{dt} \qquad (۴-۲۹)$$

رابطه بالا شامل مولفه‌های زیر است:

P_e توان موتور:

P_e توان بار:

τ انرژی جنبشی خودرو:

انرژی جنبشی سیستم کاهیده معادل خودرو در رابطه ۳۰-۴ آورده شده است[۲۸۳].

$$\tau = \frac{1}{2} I_{eq}\, \omega^2{}_{engine} \qquad (۴-۳۰)$$

با فرض عدم تغییر ممان اینرسی معادل در رابطه بالا، میزان تغییرات انرژی جنبشی به صورت رابطه ۳۱-۴ است. هم‌چنین طرف چپ معادله ۲۹-۴ را می‌توانبا استفاده از رابطه توان با رابطه ۳۲-۴ جایگزین کرد. با استفاده از روابط ۳۱-۴ و ۳۲-۴، فرم نهایی مدل‌سازی سیستم انتقال قدرت معادل در رابطه ۳۳-۴ حاصل می‌شود[۲۸۳].

$$\frac{d\tau}{dt} = I_{eq}\, \omega_{engine} \frac{d\omega_{engine}}{dt} \qquad (۴-۳۱)$$

$$P_e - P_l = (T_e - T_l)\, \omega_{engine} \qquad (۴-۳۲)$$

$$I_{eq}\ \frac{d\omega_{engine}}{dt}\ = T_e - \ T_l \qquad (۴-۳۳)$$

معادله ۳۳-۴ شامل پارامترهای زیر است:

گشتاور موتور: T_e

گشتاور بار: T_l

ممان اینرسی معادل سیستم کاهیده : I_{eq}

سرعت دورانی موتور: ω_{engine}

مولفه‌های گشتاور بار شامل گشتاور معادل نیروی مقاوم هوا، نیروی مقاوم دوران و نیروی شیب جاده است. این مولفه‌ها در رابطه ۳۴-۴ نشان داده شده است. هم‌چنین ممان اینرسی معادل، شامل ممان اینرسی معادل موتور، شاسی و چرخ‌ها خواهد بود. این مولفه‌ها در رابطه ۳۵-۴ نشان داده شده است[۲۸۳].

$$T_l = \ T_{eq,aero} + \ T_{eq,rolling} + \ T_{eq,slope} \qquad (۴-۳۴)$$

$$I_{eq} \ = \ I_{engine} + \ I_{eq,chassi} + \ 4\ I_{eq,wheel} \qquad (۴-۳۵)$$

۴-۲-۳-۱-۴- سیستم دینامیکی معادل

برای بیان سیستم دینامیکی معادل با استفاده از رابطه ۳۳-۴، نیاز به تشریح گشتاورهای بار و جرم‌های معادل با استفاده از اصل دالامبر وجود دارد. بر اساس این اصل، نیروی F_e وارد بر یک نقطه از سیستم، معادل نیروی یا گشتاور کاهیده F_r در سیستم معادل کاهیده خواهد بود، در صورتی که میزان کار انجام شده توسط دو نیرو برابر باشد[۲۸۸-۲۸۹-۲۹۰]. بر همین اساس رابطه ۳۶-۴ شکل می‌گیرد.

$$F_e ds_e = \ F_r ds_r \qquad (۴-۳۶)$$

در این رابطه ds_r و ds_e به ترتیب جابه‌جایی نیرو معادل کاهیده و جابه‌جایی نیروی اصلی است. به طور مشابه برای گشتاور کاهیده T_r نیز خواهیم داشت:

$$F_e ds_e = \ T_r d\theta_r \qquad (۴-۳۷)$$

بر همین اساس، با تقسیم دو طرف رابطه بالا به زمان می‌توان تمامی نیروهای وارد بر خودرو را با استفاده از رابطه ۴-۳۹ به گشتاورهای معادل کاهیده تبدیل کرد.

$$F_e ds_e = T_r d\theta_r \Rightarrow F_e ds_e/dt = T_r d\theta_r/dt \quad (۴-۳۸)$$

$$F_e u_e = T_r w_r \Rightarrow T_r = F_e u_e/w_r \quad (۴-۳۹)$$

در رابطه ۴-۳۹ u_e سرعت سیستم واقعی و w_r سرعت دورانی معادل در سیستم کاهیده است. با توجه به سیستم کاهیده ۴-۱۲، می‌توان تمامی نیروهای خارجی وارد بر سیستم را به صورت گشتاورهای معادل کاهیده برای استفاده در معادله ۴-۳۳ درآورد.

همانند گشتاورهای کاهیده معادل نیروهای خارجی وارد بر خودرو، با استفاده از نظریه انرژی جنبشی، ممان اینرسی‌های معادل در سیستم کاهیده را محاسبه کرد[۲۸۸-۲۸۹-۲۹۰]. بر اساس قانون پایستاری انرژی، انرژی جنبشی سیستم معادل کاهیده با سیستم اصلی برابر است. این قانون در رابطه ۴۰-۳ نشان داده شده است. بر همین اساس با مرتب کردن این رابطه می‌توان جرم معادل کاهیده را به دست آورد. هم‌چنین، با استفاده از روابط ۴-۴۱ الی ۴-۴۳ می‌توان ممان اینرسی معادل کاهیده جرم‌های اجزای سیستم را به دست آورد.

$$\frac{1}{2} m_e u_e^2 = \frac{1}{2} m_r u_r^2 \Rightarrow m_r = m_e \frac{u_e^2}{u_r^2} \quad (۴-۴۰)$$

$$\frac{1}{2} m_e u_e^2 = \frac{1}{2} m_r {r_r}^2 \omega_r^2 \quad (۴-۴۱)$$

$$I_r = m_r {r_r}^2 \quad (۴-۴۲)$$

$$\frac{1}{2} m_e u_e^2 = \frac{1}{2} I_r \omega_r^2 \Rightarrow I_r = m_e \frac{u_e^2}{\omega_r^2} \Rightarrow I_r = m_e \frac{{r_e}^2 \omega_e^2}{\omega_r^2} \quad (۴-۴۳)$$

روابط بالا دارای مولفه‌های زیر است:

m_{e} جرم سیستم اصلی:

u_{e} سرعت خطی سیستم اصلی:

r_{e} شعاع ژیراسیون سیستم اصلی :

m_{r} جرم سیستم کاهیده:

u_{r} سرعت خطی سیستم کاهیده:

شعاع ژیراسیون سیستم کاهیده : r_{r}

ممان اینرسی معادل کاهیده : I_{r}

با استفاده از معادله ۴۳-۴ می‌توان ممان اینرسی معادل کاهیده اجزای سیستم انتقال قدرت را محاسبه کرد.

۱-۴-۲-۳-۱-۴- گشتاورهای معادل کاهیده وسیله نقلیه

با توجه به رابطه ۳۹-۳ گشتاور معادل کاهیده نیروهای مقاومت هوا، مقاوم دوران و مولفه افقی وزن خودرو به ترتیب با استفاده از روابط ۱۲-۴ ، ۶-۴ و ۱۸-۴ و هم چنین با به کارگیری رابطه ۲۷-۴ بدست می‌آید. این روابط در ادامه آورده شده است[۲۸۳]:

$$\mathrm{T_{eq,aero}} = \mathrm{F_{aero}}\ \frac{\mathrm{u}}{\omega_{\mathrm{engine}}} = \mathrm{F_{aero}}\ \mathrm{R_r}\frac{\eta_{\mathrm{c}}\eta_{\mathrm{d}}}{\tau_{\mathrm{c}}\tau_{\mathrm{d}}} \qquad (۴-۴۴)$$

$$\mathrm{T_{eq,rolling}} = \mathrm{F_{rolling}}\ \frac{\mathrm{u}}{\omega_{\mathrm{engine}}} = \mathrm{F_{rolling}}\ \mathrm{R_r}\frac{\eta_{\mathrm{c}}\eta_{\mathrm{d}}}{\tau_{\mathrm{c}}\tau_{\mathrm{d}}} \qquad (۴-۴۵)$$

$$\mathrm{T_{eq,slope}} = \mathrm{W_x}\ \frac{\mathrm{u}}{\omega_{\mathrm{engine}}} = \mathrm{W_x}\ \mathrm{R_r}\frac{\eta_{\mathrm{c}}\eta_{\mathrm{d}}}{\tau_{\mathrm{c}}\tau_{\mathrm{d}}} \qquad (۴-۴۶)$$

در روابط بالا $\eta_{\mathrm{c}}\eta_{\mathrm{d}}$ نشان دهنده بازده سیستم گیربکس است که در انتقال گشتاور بین بار و موتور ظاهر می‌شود.

۲-۴-۲-۳-۱-۴- ممان اینرسی‌های معادل کاهیده اجزای انتقال قدرت

با توجه به رابطه ۴۳-۴ می‌توان ممان اینرسی معادل کاهیده اجزای سیستم انتقال قدرت را به دست آورد. بر همین اساس، ممان اینرسی موتور تنها با در نظر گرفتن اجزای متحرک آن و به صورت صریح بیان می‌شود. رابطه ۴۷-۳ ممان اینرسی موتور را نشان داده است[۲۸۶].

$$I_{engine} = I_{cgi} + I_{fw} = (m_c + m_{cr}){R_c}^2 n_{cyl} + I_{fw} \qquad (۴-۴۷)$$

این معادله شامل مولفه‌های زیر است:

ممان اینرسی چرخ دنده کرنک : I_{cgi}

ممان اینرسی چرخ طیار : I_{fw}

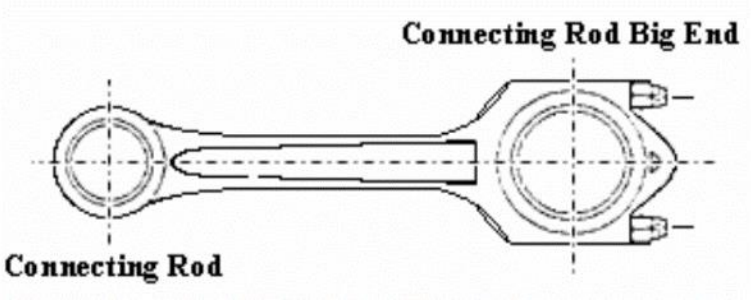
جرم کرنک : m_c

m_{cr} : جرم راد متصل کننده

R_c : شعاع ژیراسیون کرنک

n_{cyl} : تعداد سیلندرها

در شکل ۱۳-۴ نمونه‌ای از راد متصل کننده موتور نشان داده شده است.



شکل ۱۳-۴ راد اتصال در موتور احتراق داخلی[۲۸۶]

ممان اینرسی معادل کاهیده جرم خودرو (شاسی) با استفاده از رابطه ۴۳-۴ به صورت زیر خواهد بود[۲۸۳]:

$$I_{eq,chassis} = m_e R_r^2 \left(\frac{\eta_c \eta_d}{\tau_c \tau_d}\right)^2 \qquad (۴-۴۸)$$

به همین ترتیب ممان اینرسی معادل چرخ‌ها نیز با استفاده از رابطه ۴۹-۳ به دست می‌آید[۲۸۳].

$$I_{eq,wheel} = J_{wheel} \left(\frac{\eta_c \eta_d}{\tau_c \tau_d}\right)^2 \qquad (۴-۴۹)$$

۴-۱-۴- مدل‌سازی دینامیک عرضی

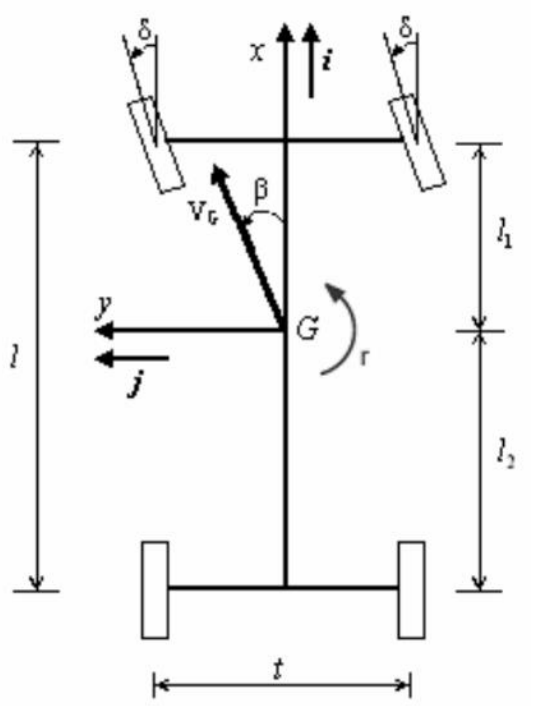
درجات آزادی در معادلات دینامیکی متغیر هستند. تعداد این درجات آزادی با توجه به هدف پژوهش تعیین می‌شود. برای مدل‌سازی دینامیک عرضی خودرو، می‌توان به مدل دو درجه آزادی موسوم به مدل دوچرخه و مدل چهار درجه آزادی اشاره کرد. در این پژوهش از مدل دو درجه آزادی استفاده شده است[۲۸۱].

برای مدل‌سازی دینامیک عرضی خودرو، فرضیات ساده‌کننده زیر در نظر گرفته شده است:

- صلب بودن خودرو
- در نظر نگرفتن سیستم تعلیق

- در نظر نگرفتن اثرات زاویه تحدب تایرها
- صلب بودن زاویه فرمان و منطبق بودن زاویه چرخ‌ها با زاویه فرمان
- عدم استفاده از قانون آکرمن و یکسان بودن زاویه فرمان چرخ‌های محور.
- در نظر نگرفتن وزن چرخ‌ها
- عدم تغییر مکان مرکز جرم خودرو در طی حرکت
- عدم در نظر گرفتن ممان اینرسی‌های فرعی

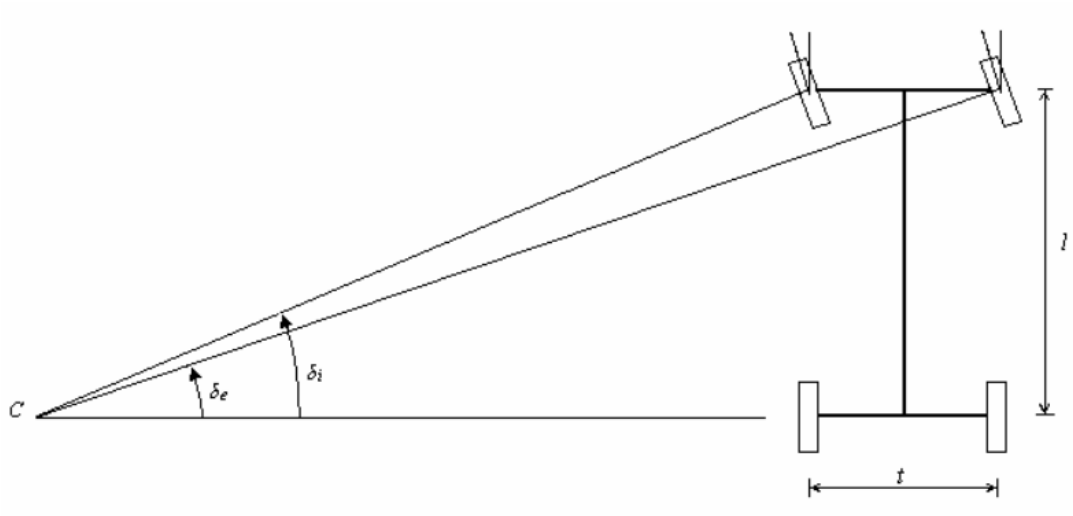
با توجه به آنکه، مدل خودرو برای عملیات پارک خودکار در سرعت پایین مورد استفاده قرار می‌گیرد، فرضیات در نظر گرفته شده خطای محسوسی به سیستم دینامیکی تحمیل نخواهد کرد. هم‌چنین، کنترل فرمان به وسیله محور جلو و محور عقب، محور محرک است. برای بیان معادلات از دستگاه بدنی استفاده می‌شود که در شکل ۱۴-۴، آورده شده است[۲۸۳].



شکل ۱۴-۴ دستگاه مختصات بدنی با مبدا مرکز جرم[۲۸۳]

با توجه به تقارن تقریبی خودروها در راستای محور طولی، محور عرضی به عنوان محور اصلی در نظر گرفته می‌شود. به همین‌دلیل، ممان اینرسی‌های فرعی این محور برابر صفر خواهند بود. برای سادگی مدل، از ممان اینرسی فرعی دیگر محورها صرف نظر شده است.

در شکل ۱۵-۴، نحوه چرخش فرمان در چرخ‌های جلو نشان داده شده است. با توجه به این شکل، رابطه ۵۰-۴ به دست می‌آید.



شکل ۱۵-۴ زاویه فرمان چرخ‌های محور جلو[۲۸۳]

$$\frac{t}{l} = \frac{1}{\tan \delta_e} - \frac{1}{\tan \delta_i} \qquad (۴-۵۰)$$

رابطه بالا شامل مولفه‌های زیر است:

t : فاصله چرخ‌های محور

l : فاصله محور جلو و عقب

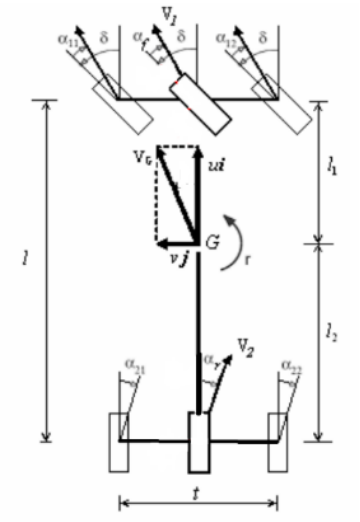
δ_e : زاویه فرمان چرخ شعاع بیرونی چرخش

δ_i : زاویه فرمان چرخ شعاع داخلی چرخش

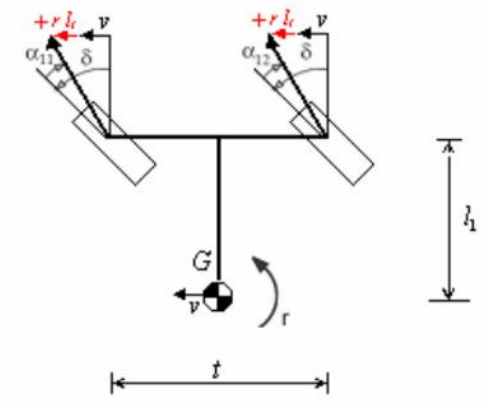
با توجه به فرضیات ساده کننده، زاویه فرمان در چرخ‌ها برابر در نظر گرفته می‌شود. به همین دلیل، می‌توان مدل خودرو را با مدل دو درجه آزادی دوچرخه جایگزین کرد. این مدل در شکل ۱۶-۴ نشان داده شده است. در این مدل چرخ‌ها جلو با چرخ مرکزی محور جلو و چرخ‌های عقب با چرخ مرکزی محور عقب جایگزین می‌شوند. در ادامه به استخراج زوایای لغزش پرداخته می‌شود[۲۸۳].

۱-۴-۱-۴- استخراج زوایای چرخش تایرها

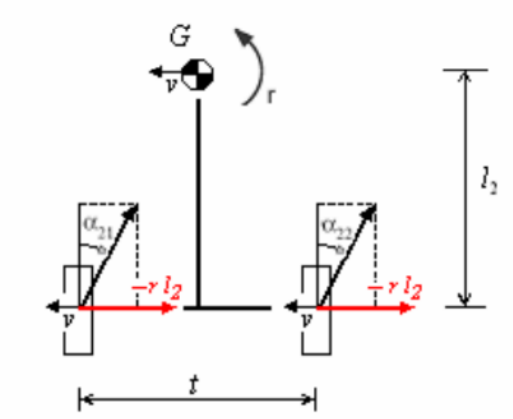
با توجه به رابطه ۱۱-۴، برای تعیین نیروی عرضی تایرها، نیاز به استخراج زاویه لغزش تایرها وجود دارد. از دیدگاه حرکت عرضی، سرعت چرخ مرکزی محور جلو و عقب به ترتیب برابر $v + rl_1$ و $v - rl_2$ خواهد بود[۲۸۳]. شکل‌های ۱۷-۴ و ۱۸-۴ نشان دهنده این موضوع هستند.



شکل۱۶-۴ مدل دوچرخه از خودرو[۲۸۳]



شکل۱۷-۴ سرعت عرضی چرخ‌های محور جلو[۲۸۳]



شکل۱۸-۴ سرعت عرضی چرخ‌های محور عقب [۲۸۳]

با توجه به استفاده از مدل دوچرخه، سرعت طولی چرخ‌های جلو و عقب برابر u خواهد بود. بر همین اساس برای زوایای لغزش چرخ‌های مرکزی محور جلو و عقب روابط ۵۱-۴ و ۵۲-۴ به دست می‌آید. این روابط بر اساس تعریف زاویه لغزش به دست می‌آید. با مرتب کردن این روابط، معادله‌ی زوایای لغزش به دست می‌آید[۲۸۳].

$$\tan(\delta_f - \alpha_f) = \frac{v + rl_1}{u} \Rightarrow \alpha_f = \delta_f - \tan^{-1}(\frac{v + rl_1}{u}) \qquad (۴-۵۱)$$

$$\tan(\alpha_r) = \frac{v - rl_2}{u} \Rightarrow \alpha_r = -\tan^{-1}(\frac{v - rl_2}{u}) \qquad (۴-۵۲)$$

بر همین اساس، زاویه لغزش چرخ مرکزی محور جلو و محور عقب، دارای اختلاف کوچکی خواهند بود. برای ساده‌سازی روابط می‌توان فرض کرد که سرعت طولی خودرو همواره بزرگتر یا برابر با سرعت عرضی آن است. بنابراین، می‌توان روابط را به صورت زیر تقریب زد[۲۸۳]:

$$\alpha_f = \delta_f - \frac{v + rl_1}{u} \qquad (۴-۵۳)$$

$$\alpha_r = -\frac{v - rl_2}{u} \qquad (۴-۵۴)$$

۲-۴-۱-۴- معادلات دینامیکی در راستای عرضی

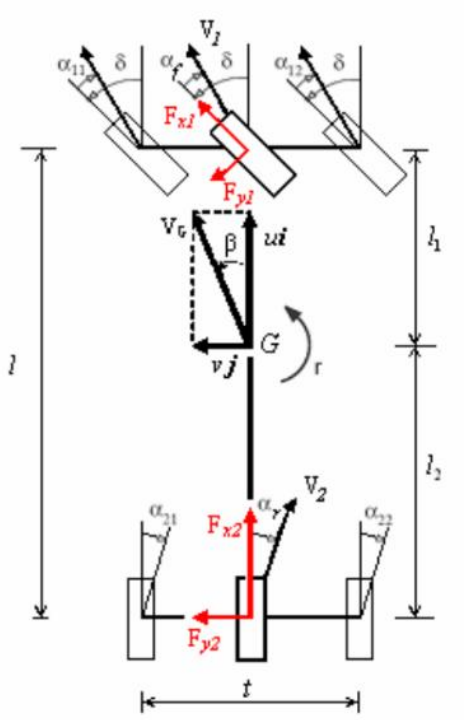
با در نظر گرفتن معادلات تعادل در صفحه زمین، می‌توان به معادلات دینامیکی در راستای عرضی دست پیدا کرد. در همین راستا معادلات ۵۵-۴ الی ۵۸-۴ با توجه به شکل ۱۹-۴ به دست می‌آید[۲۸۳].

$$\beta = \tan^{-1}(\frac{v}{u}) \qquad (۴-۵۵)$$

$$\sum f_x = ma_x \quad (۴-۵۶)$$

$$\sum f_y = ma_y \quad (۴-۵۷)$$

$$\sum M_z = J\dot{r} \quad (۴-۵۸)$$



شکل ۱۹-۴ نیروهای خارجی وارد بر خودرو از دیدگاه مجانبی[۲۸۳]

با توجه به دوران دستگاه مختصات بدنی در راستای عمود بر صفحه و حول مرکز جرم ، برای به دست آوردن شتاب خطی مرکز جرم خودرو از معادله سرعت مشتق گرفته می‌شود. در همین راستا معادلات ۴-۶۰ و ۴-۶۱ برای شتاب خطی در دو راستای طولی و عرضی به دست می‌آید[۲۸۳].

$$V = u\,\hat{\imath} + v\,\hat{\jmath} \Rightarrow a_G = \frac{dV}{dt} = \dot{u}\,\hat{\imath} + +ur\,\hat{\jmath} + \dot{v}\,\hat{\jmath} - vr\,\hat{\imath} \quad (۴-۵۹)$$

$$\alpha_{Gx} = \dot{u} - vr \quad (۴-۶۰)$$

$$\alpha_{Gy} = \dot{v} + ur \quad (۴-۶۱)$$

با در نظر گرفتن تمامی نیروها، معادلات ۵۶-۴ الی ۵۸-۴ به شکل در خواهد آمد:

$$\dot{u} = vr + \frac{1}{m}\left(F_{x1}\cos\delta - F_{y1}\sin\delta + F_{x2} - F_{xa}\cos\beta - W_x\right) \qquad (۴-۶۲)$$

$$\dot{v} = -ur + \frac{1}{m}\left(F_{x1}\sin\delta + F_{y1}\cos\delta + F_{y2} - F_{xa}\sin\beta\right) \qquad (۴-۶۳)$$

$$\dot{r} = \frac{1}{J}\left[(l_1 + \Delta x)\left(F_{x1}\sin\delta + F_{y1}\cos\delta\right) - (l_2 - \Delta x)F_{y2}\right] \qquad (۴-۶۴)$$

روابط بالا شامل مولفه‌های زیر است:

F_{x1} : مجموع نیروی وارد بر چرخ‌های جلو در راستای طولی

F_{x2} : موع نیروی وارد بر چرخ‌های عقب در راستای طولی

F_{y1} : مجموع نیروی وارد بر چرخ‌های جلو در راستای عرضی

F_{y2} : مجموع نیروی وارد بر چرخ‌های عقب در راستای عرضی

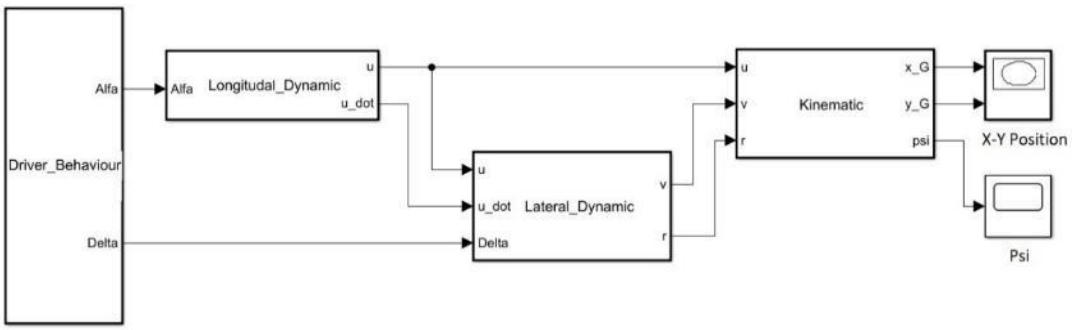
F_{xa} : نیروی مقاومت هوا در راستای طولی حرکت

W_x : نیروی وزن در راستای طولی با توجه به شیب جاده

باقی مولفه‌ها در قسمت‌های قبل تعریف گردیده است.

۵-۱-۴- پیاده‌سازی در بستر نرم‌افزاری

در این قسمت، به ارزیابی شبیه‌سازی سیستم دینامیکی خودرو در محیط نرم‌افزار متلب پرداخته شده است. در شکل ۲۰-۴ نمای کلی از سیستم دینامیکی شبیه‌سازی شده مشاهده می‌گردد.



شکل ۲۰-۴ دیاگرام کلی مدل‌سازی دینامیکی

در قسمت اول، با وارد کردن فشردگی پدال گاز و زاویه فرمان، رفتار راننده شبیه‌سازی شده است. با استفاده از فشردگی پدال گاز و مدل‌سازی سیستم قدرت خودرو در قسمت دینامیک طولی، سرعت طولی و تغییرات آن بر حسب زمان به دست می‌آید. این مولفه‌ها به همراه زاویه فرمان، وارد قسمت دینامیک عرضی خودرو شده و با محاسبه‌ی نیروهای وارد بر چرخ‌ها، مقدار سرعت عرضی و دورانی خودرو محاسبه می‌گردد. در انتها با توجه به سرعت‌های خودرو در محورهای مختلف و با استفاده از معادلات سینماتیکی، موقعیت مرکز جرم خودرو در صفحه زمین و زاویه قرارگیری آن نسبت به محور افقی تعیین می‌گردد. لازم به ذکر است تمامی مدل‌سازی در محیط متلب به صورت کد نوشته شده است و از محیط سیمولینک استفاده نگردیده است. در ادامه به صورت خلاصه، کد نوشته شده تشریح می‌گردد. برای مدل‌سازی سیستم دینامیکی از ضرایب ماشین مگان کوپه ۱۶ وی با قدرت ۱۵۰ اسب بخار که در شکل ۲۱-۴ نشان داده شده، استفاده شده است[۲۸۳].

شکل ۲۱-۴ مدل ماشین مورد استفاده در شبیه‌سازی دینامیکی

برای مدل‌سازی سیستم دینامیکی، ۶ متغیر حالت در نظر گرفته شده است که در ادامه آورده شده است:

$$States = [x_G \; y_G \; \psi \; u \; v \; r]^T \qquad (۴-۶۵)$$

مولفه‌های این معادله عبارتند از:

x_G : موقعیت طولی مرکز جرم توصیف شده در دستگاه زمین

y_G : موقعیت عرضی مرکز جرم توصیف شده در دستگاه زمین

ψ : موقعیت زاویه‌ای مرکز جرم توصیف شده در دستگاه زمین

u : سرعت طولی خودرو بیان شده در دستگاه بدنی

سرعت عرضی خودرو بیان شده در دستگاه بدنی : v

سرعت دورانی خودرو بیان شده در دستگاه بدنی : r

با توجه به توضیحات گفته شده، ابتدا ضرایب خودرو، میزان زمان شبیه‌سازی، گام زمانی شبیه‌سازی و مقدار اولیه متغیرهای حالت در کد زیر آورده شده است:

```
clc;clear all;close all;
global Parameters
run('Specification')

%% Initializing

% Time
T0 = 0;      % [sec] Initial Time
Ts = 0.001;  % [sec] Time Step
Tf =100;     % [sec] Final Time
t  = T0:Ts:Tf;
Nt = numel(t);

% Initial Value of States
x0   = 0;
y0   = 0;
psi0 = 0;
u0   = 0;
v0   = 0;
r0   = 0;
```

با توجه به کد بالا، میزان زمان شبیه‌سازی برابر ۱۰۰ ثانیه، گام زمانی برابر ۰/۰۰۱ ثانیه و مقدار اولیه متغیرهای حالت برابر صفر در نظر گرفته شده است. پس از آن ورودی‌های سیستم شامل میزان فشردگی پدال گاز در بازه [۰/۱ ۱] و زاویه فرمان در بازه [۶۰- ۶۰] درجه به سیستم داده می‌شود. علاوه بر این، شماره دنده ماشین و میزان شیب جاده به درجه داده می‌شود. به دلیل کاربرد دینامیک در پارک خودکار تنها دنده ۱ و دنده معکوس خودرو در نظر گرفته شده که به ترتیب با اعداد ۱ و -۱ قابل اعمال به مدل دینامیکی خودرو است.

```
% System Inputs
Theta0 = 0.8; % [%] [0.1 1] Accelerator
Delta0 = 0;   % [Deg] [-60 60]
Dande  = 1;   % 1 = First Gear , -1 = Reverse

Theta = Theta0 * ones(Nt,1);
Delta = Delta0 * (pi/180) * ones(Nt,1);

% Others
Gama = 0 * pi/180;     % [Deg] Slope Angle of Earth
```

پس از دادن ورودی‌های سیستم، متغیرهای حالت و باقی پارامترها به داخل تابع دینامیک سیستم منتقل می‌شود. در ابتدا میزان متغیرهای حالت به متغیرهای تعریف شده انتقال داده شده و سرعت‌های دورانی و خطی تایرها با استفاده از روابط محاسبه می‌گردد.

```
x_g0 = States(1);
y_g0 = States(2);
psi0 = States(3);
u0   = States(4);
v0   = States(5);
r0   = States(6);

W_wheel0 = u0 /  Parameters.R_r ;
W_engine0 = sign(Dande)*(Parameters.Tau_c * Parameters.Tau_d) * W_wheel0;
```

در مرحله بعدی با استفاده از ورودی میزان فشردگی پدال گاز سرعت خطی خودرو و تغییرات آن در واحد زمان محاسبه می‌گردد. این کار با استفاده از معادلات شبیه‌سازی موتور در دینامیک طولی خودرو انجام می‌شود.

```
%% Longitudal Dynamic
T_engine = Parameters.T_e_max * Theta + Parameters.T_e_min * (1 - Theta); % Engine Torque


T_aero    = 0.5 * Parameters.rho_air * Parameters.S * Parameters.C_x * (((Parameters.Eta_c
 * Parameters.Eta_d)/(Parameters.Tau_c * Parameters.Tau_d))^3) * (Parameters.R_r^3) * W_en
gine0^2; % Equivalent Torque of Aerodynamic Force
F_r = Parameters.F0 + Parameters.K * u0^2;

if u0 == 0
    F_r = 0;
end

T_rolling = Parameters.m_v * Parameters.g * cos(Gama) * F_r * ((Parameters.Eta_c * Paramet
ers.Eta_d)/(Parameters.Tau_c * Parameters.Tau_d)) * Parameters.R_r ;
                  % Equivalent Torque of Rolling Resistance Force
T_slope   = sign(u0)* Parameters.m_v * Parameters.g * sin(Gama) * Parameters.R_r * ((Param
eters.Eta_c * Parameters.Eta_d)/(Parameters.Tau_c * Parameters.Tau_d));
                % Equivalent Torque of Slope Weight Force

T_load = T_aero + T_rolling + T_slope;

if T_engine <= T_load
    T_engine = T_load;
end

W_engine1 = ((T_engine - T_load) / Parameters.I_eq) * Ts + W_engine0;

W_wheel1 = sign(Dande) * ( 1 / (Parameters.Tau_c * Parameters.Tau_d) )* W_engine1;


u1 = Parameters.R_r * W_wheel1;

u_dot = (u1 - u0) / Ts;

W_wheel_dot = (W_wheel1 - W_wheel0) / Ts;
```

در مرحله بعدی، با استفاده از معادلات نیروها در قسمت دینامیک عرضی، سرعت‌های عرضی و دورانی خودرو محاسبه می‌شود.

```
%% Lateral Dynamic

delta_x = F_r * Parameters.R_r;

Beta = atan( v0 /u1 ); % [rad] Slide Slip Angle
if isnan(Beta)
    Beta = 0;
end

alfa_f = Delta - atan( (v0 + r0 * (Parameters.l1) ) / u1 );
if isnan(alfa_f)
    alfa_f = 0;
end

alfa_r = - atan( ( v0 - r0 * (Parameters.l2) )/ u1 );
if isnan(alfa_r)
    alfa_r = 0;
end

F_y_1 = sign(u1) * Parameters.C1 * alfa_f;
F_y_2 = sign(u1) * Parameters.C2 * alfa_r;


F_xa = 0.5 * Parameters.rho_air * Parameters.S * Parameters.C_x * u1^2;

F_z_2 = ( Parameters.m_v * (u_dot - v0 * r0) * Parameters.h1 + 2 * Parameters.I_each_wheel * (W_wheel_dot) +...
        2 * Parameters.I_each_wheel * (W_wheel_dot) * cos(Delta) + sign(u1) * F_xa * cos(Beta) * Parameters.h2 + Parameters.m_v * Parameters.g * sin(Gama) * Parameters.h1 + ...
        Parameters.m_v * Parameters.g * cos(Gama) * (Parameters.l1 + sign(u1)* delta_x))/(2*Parameters.l);

F_z_1 = (Parameters.m_v * Parameters.g * cos(Gama) - 2 * F_z_2)/2;

F_x_1 = -(Parameters.I_each_wheel * W_wheel_dot + F_z_1 * sign(u1) * delta_x ) / Parameters.R_r;

v_dot = -u1 * r0 + (1/Parameters.m_v) * ( 2 * F_x_1 * sin(Delta) + 2 * F_y_1 * cos(Delta) + ...
        2 * F_y_2 -  sign(u1) * F_xa * sin(Beta));

r_dot = (1/Parameters.j_z) * ( (Parameters.l1 + sign(u1) * delta_x) * ( 2 * F_x_1 * sin(Delta) + 2 * F_y_1 * cos(Delta)) - ( Parameters.l2 - sign(u1) * delta_x ) * 2 * F_y_2);


v1 = v_dot * Ts + v0;
r1 = r_dot * Ts + r0;
```

در انتها با استفاده از معادلات سینماتیکی، موقعیت مرکز جرم خودرو پس از طی یک گام زمانی بدست می‌آید.

```
%% Kinematic
```

```
2.  psi1 = r1 * Ts + psi0;
3.  x_g1 = (u1 * cos(psi1) - v1 * sin(psi1)) * Ts + x_g0;
4.  y_g1 = (u1 * sin(psi1) + v1 * cos(psi1)) * Ts + y_g0;
```

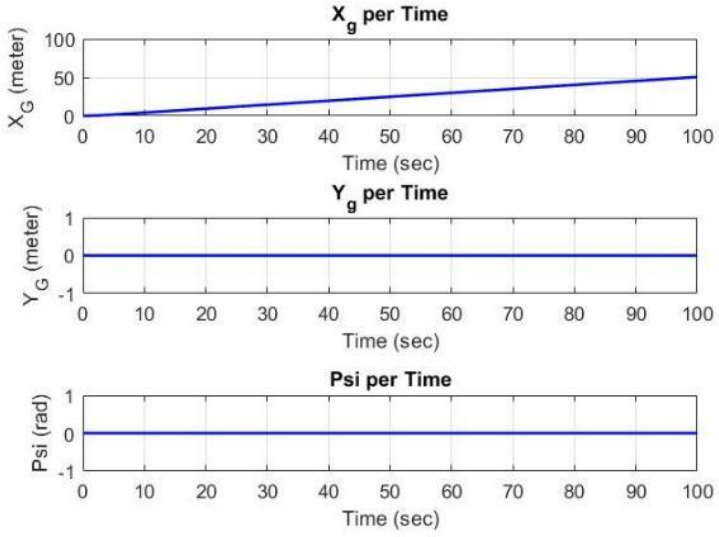
پس از طی مراحل بالا، تمامی متغیرهای حالت بروز رسانی شده و گام زمانی بعدی به همین ترتیب انجام می‌شود. در انتها پس از طی شدن تمامی گام‌های زمانی، نمودارهای موقعیت و سرعت خودرو رسم می‌شود. برای شبیه‌سازی و صحت سنجی سیستم دینامیکی، از سه دسته ورودی مختلف استفاده شده که در ادامه آورده شده است. هم‌چنین، در تمامی شبیه‌سازی‌ها مقادیر اولیه متغیرهای حالت برابر صفر در نظر گرفته شده است.

$$\delta = 0 \quad \alpha = 0.8 \quad Dande = First\ Gear \tag{۴-۶۶}$$

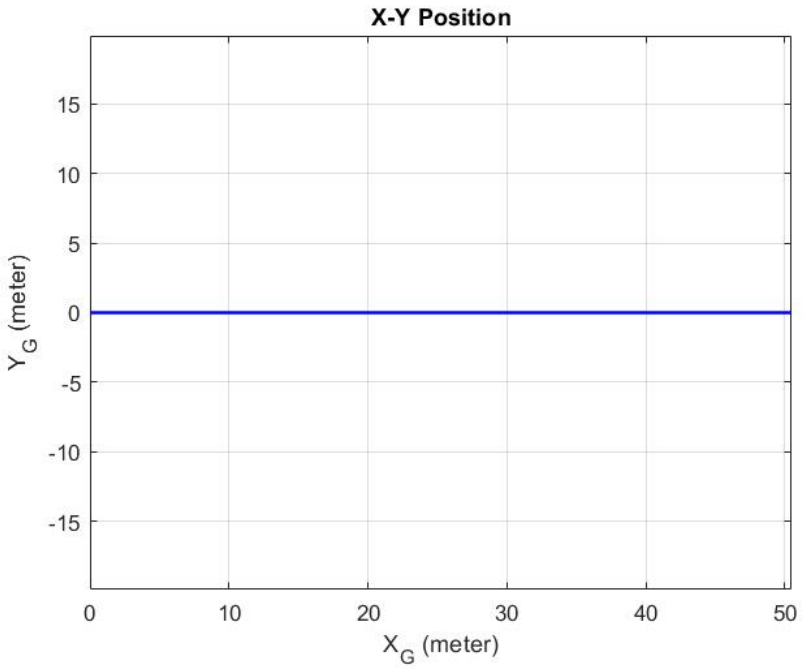
$$\delta = 20 \quad \alpha = 0.9 \quad Dande = First\ Gear \tag{۴-۶۷}$$

$$\delta = -2 \quad \alpha = 1 \quad Dande = Reverse\ Gear \tag{۴-۶۸}$$

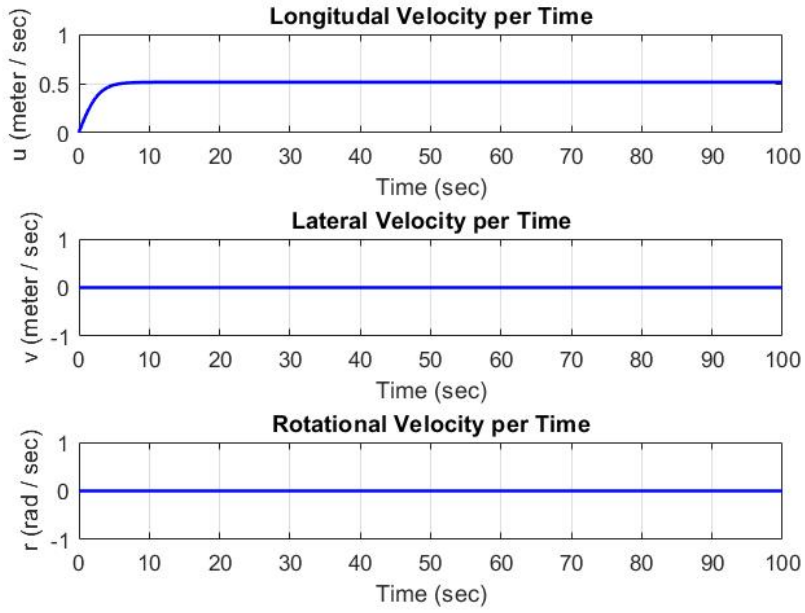
با استفاده از ورودی‌های بالا، نتایج شبیه‌سازی به ترتیب در ادامه آورده شده است.



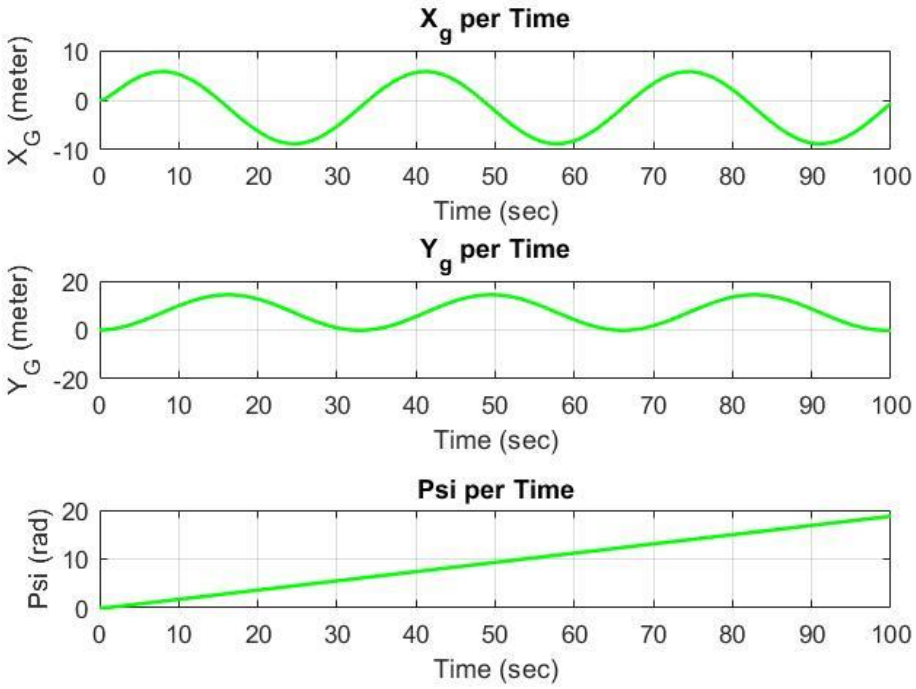
شکل۲۲-۴ مولفه‌های موقعیت بر حسب زمان برای ورودی‌های معادله ۶۶-۴



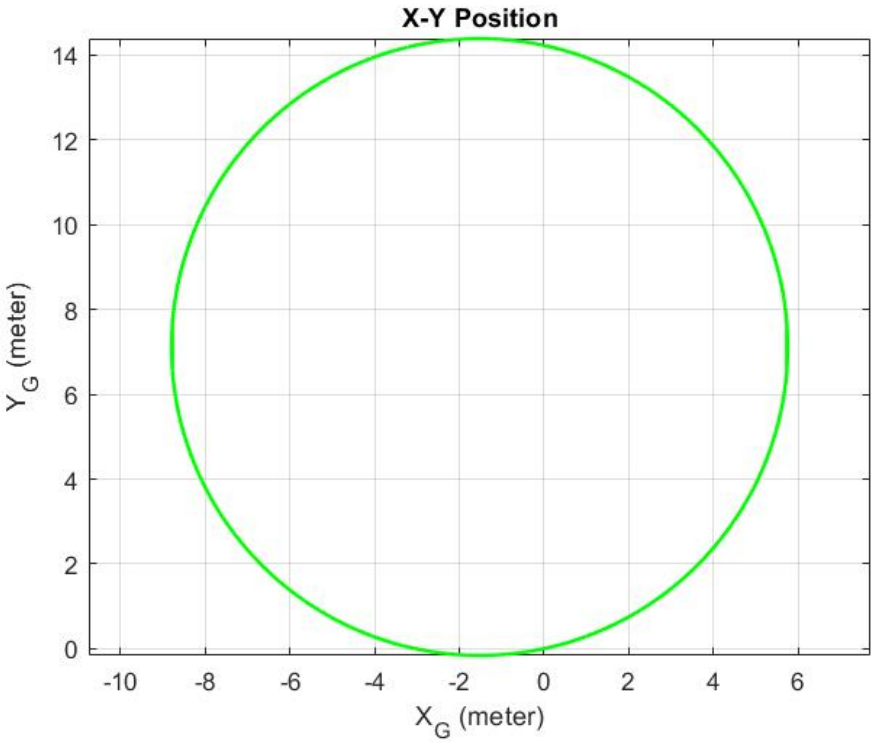
شکل ۲۳-۴ موقعیت خودرو در صفحه زمین برای ورودی‌های معادله ۶۶-۴



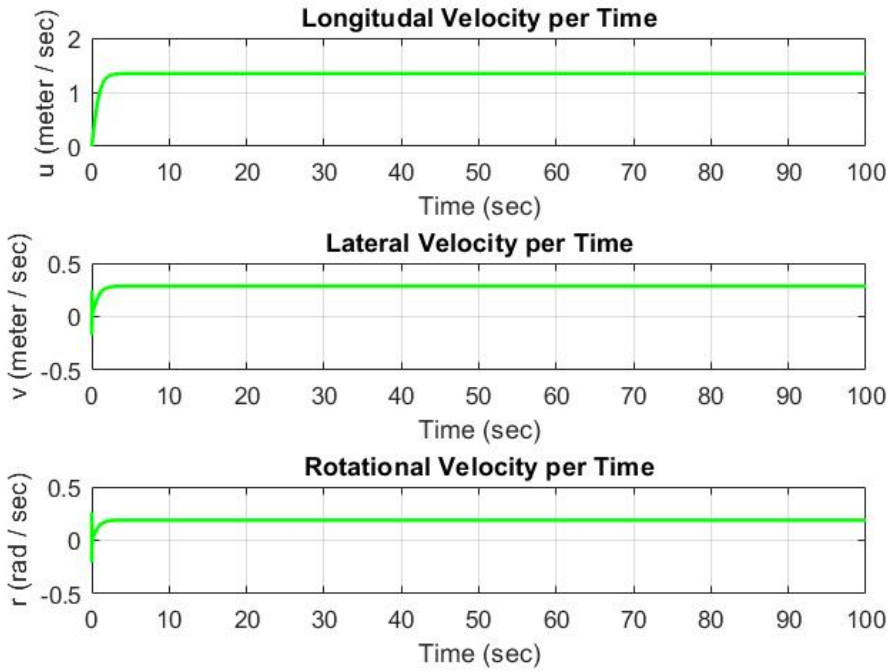
شکل ۲۴-۴ مولفه‌های سرعت خودرو بر حسب زمان برای ورودی‌های معادله ۶۶-۴



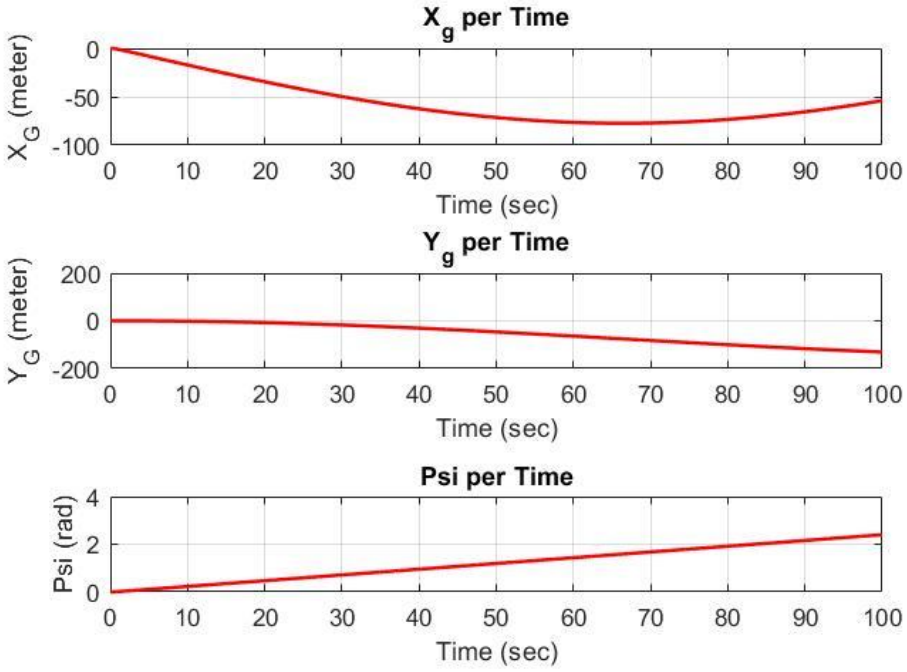
شکل ۲۵-۴ مولفه‌های موقعیت بر حسب زمان برای ورودی‌های معادله ۶۷-۴



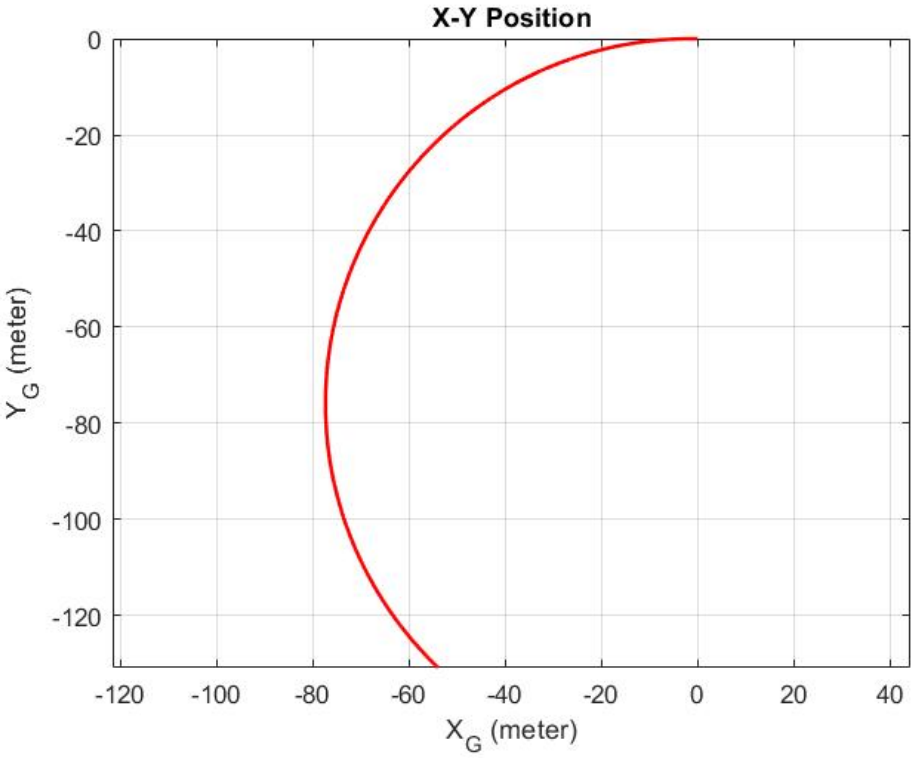
شکل ۲۶-۴ موقعیت خودرو در صفحه زمین برای ورودی‌های معادله ۶۷-۴



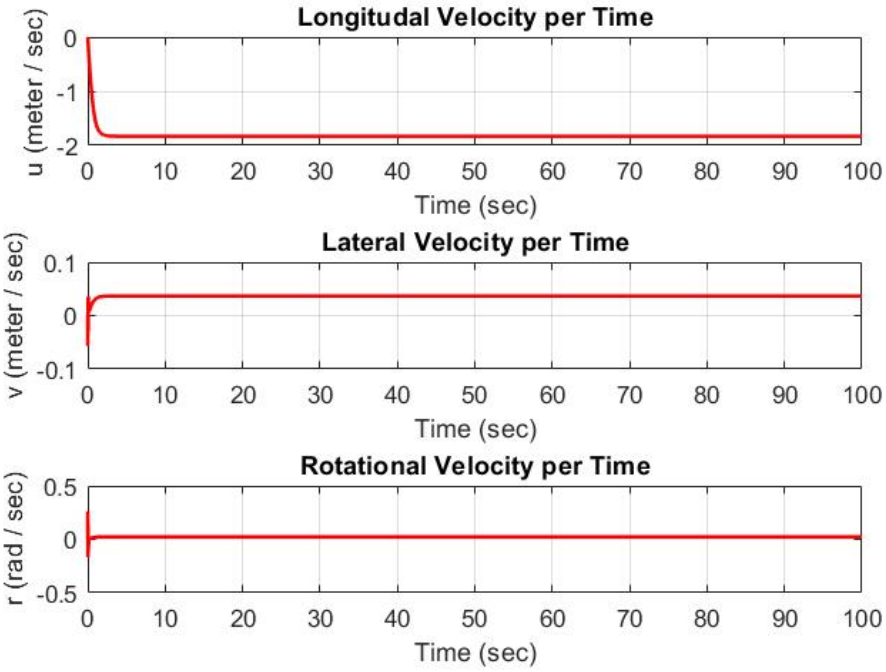
شکل ۲۷-۴ مولفه‌های سرعت خودرو بر حسب زمان برای ورودی‌های معادله ۶۷-۴



شکل ۲۸-۴ مولفه‌های موقعیت بر حسب زمان برای ورودی‌های معادله ۶۹-۴



شکل ۲۹-۴ موقعیت خودرو در صفحه زمین برای ورودی‌های معادله ۶۹-۴



شکل ۳۰-۴ مولفه‌های سرعت خودرو بر حسب زمان برای ورودی‌های معادله ۶۹-۴

با توجه به نتایج به دست آمده صحت و درستی مدل‌سازی تایید می‌شود. هم‌چنین در حالت واقعی، در صورت فشردن پدال گاز به میزان کمتر از ۰/۸ ، به دلیل بیشتر بودن گشتاور بار، خودرو توانایی حرکت را دارا نیست و خودرو با در نظر گرفتن کلاچ خاموش خواهد شد. در حالیکه در سیستم راننده خودکار، به دلیل عدم وجود کلاچ و اتوماتیک بودن گیربکس، فرض شده است که در این حالت، خودرو در حالت خلاصی قرار خواهد گرفت . شبیه سازی سیستم دینامیکی در داخل بستر پایتون نیز پیاده‌سازی شده اما تنها کد متلب آن آورده شده است.

۴-۲- توسعه قانون هدایت

برنامه ریزی مسیر برای اتوپارک به دو بخش تقسیم شده است. بخش اول شامل مسیر از نقطه شروع حرکت به نقطه شروع پارک و بخش دوم شامل مسیر از نقطه شروع پارک تا نقطه پارک تعیین شده می‌باشد.

۴-۲-۱- الگوریتم A^*

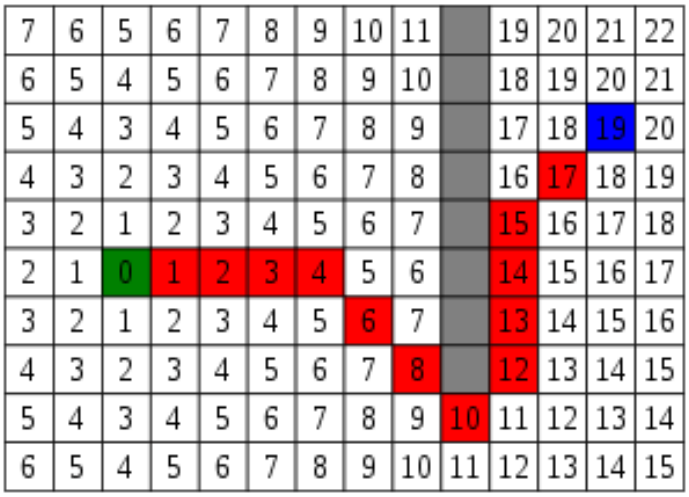
برای تولید مسیر بخش اول اتوپارک از الگوریتم A^* استفاده شده است. A^* یک الگوریتم جست‌وجوی آگاهانه است، به این معنی که بر اساس نمودارهای وزنی فرموله شده است. این الگوریتم از گره شروع خاص یک نمودار آغاز به کار می کند و هدف آن یافتن راهی به گره هدف داده شده، با کمترین هزینه می باشد. (حداقل مسافت طی شده ، کمترین زمان و غیره). A^* این کار را با حفظ یک درخت از مسیرهایی انجام می دهد که از گره شروع منشا می گیرد و مسیر را هر بار یک لبه گسترش می دهد تا زمانی که به گره هدف برسد.

در هر تکرار از حلقه اصلی خود ، A^* باید تعیین کند که کدام یک از مسیرهای خود را گسترش دهد. این کار را براساس هزینه مسیر و برآورد هزینه لازم برای گسترش مسیر تا رسیدن به هدف انجام می دهد. به طور خاص، A^* مسیری را انتخاب می کند که تابع زیر به حداقل برسد.

$$f(n) = g(n) + h(n) \qquad (۴-۶۹)$$

جایی که n گره بعدی مسیر است ، $g(n)$ هزینه مسیر از گره آغازین تا n است و $h(n)$ یک تابع ابتکاری است که هزینه ارزان‌ترین مسیر از گره n تا هدف را تخمین می زند. A^* هنگامی خاتمه می یابد که مسیری که برای گسترش انتخاب می کند مسیری از ابتدا به هدف باشد یا این که هیچ راهی واجد شرایط تمدید وجود نداشته باشد. تابع ابتکاری خاص مسئله است. اگر این تابع قابل قبول باشد، به این معنی که هرگز برای رسیدن به هدف، هزینه واقعی را بیش از حد تخمین نزند، A^* تضمین می کند که یک مسیر کم هزینه را از ابتدا به هدف بازگرداند[۲۹۱].

نمونه‌ی ساده‌ی الگوریتم A^* روی شکل ۳۱-۴ به صورت محیط ۱۰در۱۴ مشاهده می‌شود.



شکل ۳۱-۴ نمونه‌ی الگوریتم A^*

پیاده سازی های معمول A^* برای گسترش از صف اولویت برای انجام انتخاب مکرر گره های با حداقل هزینه‌ی برآورد شده، استفاده می کنند. این صف اولویت به عنوان مجموعه باز یا حاشیه شناخته می شود. در هر مرحله از الگوریتم، گره ای با کمترین مقدار $f(n)$ از صف حذف می شود، مقادیر f و g همسایگان آن به همین ترتیب به روز می شوند و این همسایگان به صف اضافه می شوند. الگوریتم تا جایی ادامه می یابد که گره حذف شده (گره ای که کمترین مقدار f را از همه گره های حاشیه ای دارد) گره هدف است. ارزش f آن هدف نیز هزینه کوتاهترین مسیر است ، زیرا در یک ابتکار قابل قبول h در هدف برابر صفر است.

اگر h اکتشافی شرط اضافی $h(x) \leq d(x,y) + h(y)$ را برای هر لبه (x, y) نمودار برآورده کند (جایی که d طول آن لبه را نشان می دهد) ، h یکنواخت، یا استوار نامیده می شود. با یک ابتکار استوار، تضمین می‌شود که A^* یک مسیر مطلوب را بدون پردازش بیش از یک بار گره ها پیدا می کند و A^* برابر می شود با اجرای الگوریتم Dijkstra با کاهش هزینه $d'(x,y) = d(x,y) + h(y) - h(x)$.

هنگامی که A^* جستجوی خود را خاتمه می دهد، مسیری را از ابتدا به هدف پیدا کرده است که هزینه واقعی آن کمتر از برآورد هزینه هر مسیر از ابتدا به هدف است. هنگامی که ابتکار پذیرفتنی باشد، این تخمین ها خوش بینانه هستند، بنابراین A^* می تواند با خیال راحت آن گره ها را نادیده بگیرد زیرا احتمالاً نمی توانند به راه حل ارزان تری نسبت به راه حل انتخاب شده منجر شوند. به عبارت دیگر، A^* هرگز از احتمال مسیر کم هزینه از ابتدا تا هدف غافل نخواهد شد.

زمانی که معیار قابل قبول یک راه حل بهینه را تضمین می کند، این بدان معنی است که A^* باید تمام مسیرهای به همان اندازه شایسته را بررسی کند تا مسیر بهینه پیدا کند. برای محاسبه کوتاهترین مسیرهای تقریبی، می توان با تخفیف در معیار پذیرش، سرعت جستجو را به قیمت بهینه سازی افزایش داد. اغلب اوقات ما می خواهیم این تخفیف را محدود کنیم، بنابراین می توانیم تضمین کنیم که مسیر راه حل، بدتر از $(1 + \varepsilon)$ برابر راه حل بهینه نخواهد بود. از این ضمانت نامه جدید به عنوان ε-قابل قبول یاد می شود.

الگوریتم های ε-قابل قبول متعددی وجود دارد:

- A^* وزنی توزین استاتیک. اگر $h_a(n)$ یک تابع اکتشافی قابل قبول است، در نسخه وزنی جستجو A^* از $h_w(n) = \varepsilon h_a(n)\,, \varepsilon > 1$ به عنوان تابع اکتشافی استفاده می شود و در ادامه جستجوی A^* را به طور معمول انجام می دهد (که از آنجا که گره های کمتری گسترش یافته اند سرانجام سریعتر از استفاده از h_a اتفاق می افتد). مسیری که از طریق الگوریتم جستجو پیدا می شود می تواند حداکثر هزینه ε برابر حداقل هزینه مسیر در نمودار داشته باشد.

- وزن دهی پویا از تابع هزینه $f(n) = g(n) + (1 + \varepsilon\omega(n))h(n)$ استفاده می‌کند. که جایی که $d\,(n)$ عمق جستجو است و N طول پیش بینی شده راه حل است. $w(n) = \begin{cases} 1 & d(n) \le N \\ \frac{d(n)}{N} & \text{otherwise} \end{cases}$

- توزین پویایی نمونه برداری، از نمونه برداری از گره ها برای تخمین و تخریب بهتر خطای ابتکاری استفاده می کند.

- A^*_ε از دو تابع اکتشافی استفاده می کند. تابع اول لیست FOCAL است که برای انتخاب گره های کاندید استفاده می شود و h_f دوم برای انتخاب امیدوار کننده ترین گره از لیست FOCAL استفاده می شود.

- A^*_ε گره هایی با عملکرد $Af(n) + Bh_f(n)$ را انتخاب می کند ، جایی که A و B ثابت هستند. اگر هیچ گره ای انتخاب نشود ، الگوریتم با عملکرد $Cf(n) + Dh_f(n)$ ، جایی که C و D ثابت هستند، محاسبه خواهد شد.

- $AlphA^*$ با ترجیح گره های اخیراً گسترش یافته سعی در بهره برداری از عمق اول دارد. $ALphA^*$ از تابع هزینه $f_\alpha(n) = (1 + w_\alpha(n))f(n)$ استفاده می کند.

$$w_\alpha(n) = \begin{cases} \lambda & g(\pi(n)) \leq g(\tilde{n}) \\ \Lambda & \text{otherwise} \end{cases} \qquad (۴-۷۰)$$

جایی که λ و Λ ثابت آن هستند و $\lambda \leq \Lambda$، $\pi(n)$ والد n است ، و $\tilde{n}$ آخرین گره ای است که گسترش یافته است.

پیچیدگی زمانی A^* به تابع ابتکار بستگی دارد. در بدترین حالت فضای جستجو نامحدود، تعداد گره های گسترش یافته در عمق راه حل نمایی است. (کوتاهترین مسیر) $O(b^d)$:d ، جایی که b فاکتور انشعاب است (میانگین تعداد جانشینان در هر بخش). فرض می شود که یک حالت هدف وجود دارد و از حالت شروع قابل دستیابی است. اگر اینگونه نباشد و فضای حالت بی نهایت باشد، الگوریتم خاتمه نمی یابد[۲۹۲].

عملکرد اکتشافی تأثیر عمده ای بر عملکرد عملی جستجوی A^* دارد، زیرا یک اکتشاف خوب به A^* اجازه می دهد تا بسیاری از گره های b^d را که جستجو ناآگاهانه گسترش می دهد، هرس کند. کیفیت آن را می توان با توجه به فاکتور انشعاب موثر b^* بیان کرد، که می تواند به طور تجربی برای یک مسئله با اندازه گیری تعداد گره های تولید شده توسط انشعاب ، N و عمق راه حل تعیین شود [۲۹۳].

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d.$$

(۴-۷۱)

اکتشافات خوب آنهایی هستند که ضریب انشعاب کمتری دارند (بهینه ۱ =b^*).

پیچیدگی زمانی چندجمله ای است زمانی که فضای جستجو یک درخت باشد ، یک حالت هدف واحد وجود داشته باشد ، و تابع اکتشافی h شرایط زیر را داشته باشد:

$$|h(x) - h^*(x)| = O(\log h^*(x)) \qquad (۴-۷۲)$$

که در آن h^* بهترین حالت ابتکاری و هزینه دقیق رسیدن از x به هدف است. به عبارت دیگر، خطای h سریعتر از لگاریتم "اکتشافی کامل" h^* که فاصله واقعی از x را به هدف بر می‌گرداند، رشد نخواهد کرد.

۲-۲-۴- درون یابی اسپلاین (spline)

مسیر به دست آمده توسط الگوریتم A^* باید توسط spline درون یابی شده و به منحنی تبدیل شود تا مسیر خودرو به صورت هموار تعیین شود. در ریاضیات ، spline یک تابع خاص است که به طور جزئی توسط چند جمله ای تعریف می شود. در حل مشکلات ، درون یابی spline اغلب به درون یابی چند جمله ای ترجیح داده می شود زیرا حتی در هنگام استفاده از چند جمله ای با درجه پایین نتایج مشابهی را به همراه دارد.

در زیرمجموعه های علوم کامپیوتر از طریق کمک رایانه و گرافیک رایانه ای، اصطلاح spline بیشتر به یک منحنی چند جمله ای (پارامتری) اشاره دارد. Spline ها به دلیل سادگی ساخت آنها ، سهولت و دقت ارزیابی آنها و ظرفیت تقریبی شکل های پیچیده از طریق اتصالات منحنی و طراحی منحنی تعاملی، منحنی های محبوب در این زیرشاخه ها هستند.

منحنی اسپیلاین با استفاده از مشتقات یک منحنی چندجمله‌ای درجه سوم میان هر دو نقطه ورودی ایجاد می‌شود. به عبارتی دیگر، منحنی درجه سوم تابع چندضابطه‌ای است که از چند تابع چند ضابطه‌ای که به یکدیگر چسبیده‌اند، تشکیل شده‌است. این توابع چند ضابطه‌ای آن چنان در محل اتصال با یکدیگر مطابقت دارند که تقریباً محل اتصال مشخص نیست. در حقیقت، اگر تمام تابع با یک تابع $p(t)$ توصیف شود، $p(t)$ آن قدر یکنواخت و هموار خواهد بود که دارای مشتق دوم در تمام نقاط و مشتق دوم پیوسته خواهد بود[۲۹۴].

روش مدل‌سازی ریاضی با $n+1$ نقطه $\{(x_i, y_i): i = 0,1,\cdots,n\}$ ، در حقیقت، درون‌یابی بین همه زوج نقطه‌های $(x_i - 1, y_i - 1)$ و (xi, yi) با استفاده از چندجمله‌ای $y = q_i(x)$ است که در آن $i = 1,2,\cdots,n$.

انحنای منحنی $y = f(x)$ به صورت زیر بیان می‌شود:

$$\kappa = \frac{y''}{(1 + y'^2)^{3/2}} \qquad (۴-۷۳)$$

از آنجایی که شکل اسپلاین به گونه‌ای است که خمش را (تحت قید عبور از همه نقاط) کمینه کند، y' و y''در همه جا و همه نقاط پیوسته خواهند بود. به عبارت دیگر، خواهیم داشت:

$$\begin{cases} q_i'(x_i) = q_{i+1}'(x_i) \\ q_i''(x_i) = q_{i+1}''(x_i) \end{cases} \qquad 1 \le i \le n-1 \qquad (۴-۷۴)$$

این امر زمانی امکان‌پذیر است که چندجمله‌ای‌ها از درجه ۵ یا بالاتر باشند. روش کلاسیک برای استفاده از چندجمله‌ای درجه ۳، اسپلاین مکعبی (Cubic Spline) نامیده می‌شود که در آن، مشتق اول پیوسته است، اما مشتق دوم خیر.

درون یابی اسپلاین مکعبی بین دو نقطه:

چندجمله‌ای مرتبه سوم $q(x)$ را در نظر بگیرید که زوج نقطه $(x1, y1)$ و $(x2, y2)$ را درون‌یابی می‌کند:

$$\begin{aligned} q(x_1) &= y_1 \\ q(x_2) &= y_2 \\ q'(x_1) &= k_1 \\ q'(x_2) &= k_2 \end{aligned} \qquad (۳-۷۵)$$

می‌توانیم فرم متقارن زیر را بنویسیم که تساوی‌های بالا در آن صدق می‌کنند:

$$q(x) = (1-t(x))\,y_1 + t(x)\,y_2 + t(x)(1-t(x))((1-t(x))\,a + t(x)\,b) \qquad (۴-۷۶)$$

که در آن:

$$t(x) = \frac{x-x_1}{\quad}, \qquad (۴-۷۷)$$

$$\begin{aligned} a &= k_1(x_2-x_1)-(y_2-y_1), \\ b &= -k_2(x_2-x_1) + (y_2-y_1). \end{aligned} \qquad (۴-۷۸)$$

با توجه به تساوی زیر:

این روابط را خواهیم داشت:

$$q' = \frac{y_2-y_1}{x_2-x_1} + (1-2t)\frac{a(1-t)+bt}{x_2-x_1} + t(1-t)\frac{b-a}{x_2-x_1}, \qquad (۴-۷۹)$$

$$q'' = 2\frac{b-2a+(a-b)3t}{(x_2-x_1)^2}. \quad (6)$$

با قرار دادن $x = x2$ و $x = x1$، به ترتیب در معادلات ۷۹-۴ با استفاده از ۷۷-۴، مشتق‌های اول $q'(x1) = k1$ و $q'(x2) = k2$و مشتق‌های دوم زیر را به دست خواهیم آورد:

$$q''(x_1) = 2\frac{b-2a}{(x_2-x_1)^2} \qquad (۴-۸۰)$$

$$q''(x_2) = 2\frac{a-2b}{(x_2-x_1)^2} \qquad (۴-۸۱)$$

اکنون اگر (x_i, y_i)را برای $i = 0,1, \dots, n$و به عنوان $n+1$ نقطه در نظر بگیریم، تعداد n چندجمله‌ای مرتبه سوم درون‌یاب yدر بازه $x_{i-1} \le x \le x_i$ برای $i = 1, \dots, n$ خواهیم داشت، به گونه‌ای که $q_i'(xi) = q_i' +$ $1(x_i)$برای $i = 1, \dots, n-1$ برقرار است:

$$q_i = (1-t)\,y_{i-1} + t\,y_i + t(1-t)\big((1-t)\,a_i + t\,b_i\big) \qquad (۴-۸۲)$$

که در آن، $i = 1,2, \dots, n$ و $t = \frac{x-x_{i-1}}{x_i-x_{i-1}}$ است و در نتیجه، nچندجمله‌ای خواهیم داشت که با یکدیگر تابعی مشتق‌پذیر در بازه $x0 \le x \le xn$ تعریف می‌کنند و برای $i = 1, \dots, n$ داریم:

$$a_i = k_{i-1}(x_i - x_{i-1}) - (y_i - y_{i-1}) \quad (۴-۸۳)$$

$$b_i = -k_i(x_i - x_{i-1}) + (y_i - y_{i-1}) \quad (۴-۸۴)$$

که در آن:

$$\begin{aligned} k_0 &= q_1'(x_0) \quad (12) \\ k_i &= q_i'(x_i) = q_{i+1}'(x_i) \qquad i = 1, \ldots, n-1 \\ k_n &= q_n'(x_n) \quad (14) \end{aligned} \quad (۴-۸۵)$$

اگر دنباله $k0$، $k1$، ...و kn به گونه‌ای باشد که برای $i = 1, \ldots, n-1$ تساوی $q_i''(x_i) = q_i'' + 1(x_i)$ برقرار باشد، آنگاه تابع حاصل یک مشتق دوم پیوسته نیز خواهد داشت.

از معادلات بالا رابطه زیر را برای $i = 1, \ldots, n-1$ نتیجه می‌گیریم:

$$\begin{aligned} &\frac{k_{i-1}}{x_i - x_{i-1}} + \left(\frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i}\right) 2k_i + \frac{k_{i+1}}{x_{i+1} - x_i} \\ &= 3\left(\frac{y_i - y_{i-1}}{(x_i - x_{i-1})^2} + \frac{y_{i+1} - y_i}{(x_{i+1} - x_i)^2}\right) \quad (15) \end{aligned} \quad (۴-۸۶)$$

در خط‌کش‌های کشسان که مدل درون یابی اسپلاین هستند، سمت چپ یا چپ‌ترین گره یا همان نقطه و سمت راست یا راست‌ترین گره خط‌کش می‌تواند آزادانه حرکت کند و به همین دلیل، با $q'' = 0$ فرم یک خط مستقیم

$$q_1''(x_0) = 2\frac{3(y_1 - y_0) - (k_1 + 2k_0)(x_1 - x_0)}{(x_1 - x_0)^2} = 0, \quad (۴-۸۷)$$

$$q_n''(x_n) = -2\frac{3(y_n - y_{n-1}) - (2k_n + k_{n-1})(x_n - x_{n-1})}{(x_n - x_{n-1})^2} = 0 \quad (۴-۸۸)$$

را به خود می‌گیرد. از آنجایی که q'' باید یک تابع پیوسته از x باشد، برای اسپلاین‌های طبیعی علاوه بر $n-1$ معادله خطی ۸۸-۴، باید داشته باشیم:

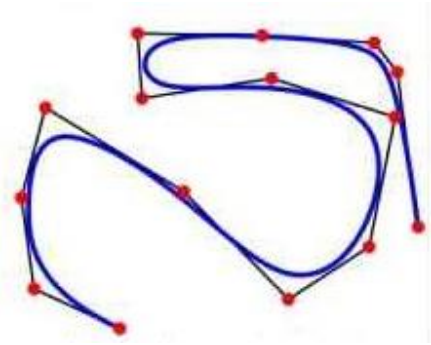
این یعنی:

$$\frac{2}{x_1-x_0}k_0+\frac{1}{x_1-x_0}k_1=3\frac{y_1-y_0}{(x_1-x_0)^2}, \quad (۴-۸۹)$$

$$\frac{1}{x_n-x_{n-1}}k_{n-1}+\frac{2}{x_n-x_{n-1}}k_n=3\frac{y_n-y_{n-1}}{(x_n-x_{n-1})^2}. \quad (۴-۹۰)$$

در نهایت، ۸۸-۴ همراه با ۸۹-۴ و ۹۰-۴، تعداد $n+1$ معادله خطی را تشکیل می‌دهند که به صورت یکتا پارامترهای $k0$، $k1$، ... و kn را تعریف می‌کنند[۲۹۵].

در شکل ۳۲-۴ یک نمونه از اعمال Spline را مشاهده می‌کنیم.



شکل ۳۲-۴ یک نمونه از اعمال spline

موارد دیگری نیز وجود دارند که می‌توان آن‌ها را نیز در نظر گرفت: اسپلاین مقید (Clamped Spline) که شیب را در انتهای اسپلاین مشخص می‌کند، و اسپلاین غیرگره‌ای(Not-a-knot Spline) ، که در آن، مشتق سوم نیز در نقاط x_1 وx_{N-1} باید پیوسته باشد. برای اسپلاین غیرگره‌ای، معادلات زیر را نیز داریم:

$$q_1'''(x_1) = q_2'''(x_1)$$
$$\Rightarrow \frac{1}{\Delta x_1^2}k_0 + \left(\frac{1}{\Delta x_1^2} - \frac{1}{\Delta x_2^2}\right)k_1 - \frac{1}{\Delta x_2^2}k_2 = 2\left(\frac{\Delta y_1}{\Delta x_1^3} - \frac{\Delta y_2}{\Delta x_2^3}\right) \quad (۴-۹۱)$$

$$q_{n-1}'''(x_{n-1}) = q_n'''(x_{n-1})$$
$$\Rightarrow \frac{1}{\Delta x_{n-1}^2}k_{n-2} + \left(\frac{1}{\Delta x_{n-1}^2} - \frac{1}{\Delta x_n^2}\right)k_{n-1} - \frac{1}{\Delta x_n^2}k_n = 2\left(\frac{\Delta y_{n-1}}{\Delta x_{n-1}^3} - \frac{\Delta y_n}{\Delta x_n^3}\right) \quad (۴-۹۲)$$

که در آن، $\Delta xi = x_i - x_{i-1}$ و $\Delta yi = y_i - y_{i-1}$

با استفاده از الگوریتم A^* و spline ، خودرو مسیر بهینه‌ی خود را با توجه به موانع تعریف شده، از وضعیت اولیه تعیین شده به نقطه شروع پارک پیدا می‌کند. نقطه شروع پارک بر اساس ابعاد خودرو از موقعیت پارک تعیین شده به دست می‌آید.

محیط تعریف شده در پروژه، یک محیط ۱۰۰ در ۱۰۰ می‌باشد. طول خودرو ۸ و عرض خودرو ۴ در نظر گرفته شده است. نقطه پارک تعیین شده مرکز خودروی پارک شده در نظر گرفته شده و با توجه به ابعاد خودرو، فاصله‌ی عرضی و طولی معینی برای نقطه شروع پارک تعیین شده است.

$S + L$ برابر است با فاصله طولی نقطه پارک تعیین شده با نقطه شروع پارک

$D + W$ برابر است با فاصله عرضی نقطه پارک تعیین شده با نقطه شروع پارک

$L = 8$

$W = 4$

$S = 8$

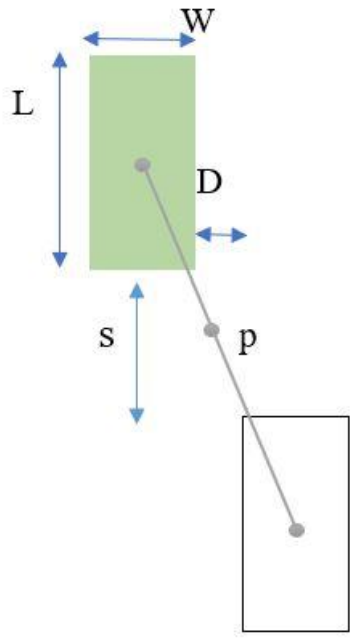
$D = 2$

۳-۲-۴- طراحی مسیر پارک دوبل

برای تولید مسیر از نقطه شروع پارک به نقطه پارک تعیین شده، طراحی مسیر به صورت گام به گام و با استفاده از دو کمان از دو دایره به شعاع های یکسان صورت گرفته است.

دو نقطه پارک و شروع پارک به هم وصل شده و وسط این خط به دست می‌آید. نقطه‌ی وسط p محل تغییر کمان می‌باشد.

مستطیل سبز نشان دهنده خودرو

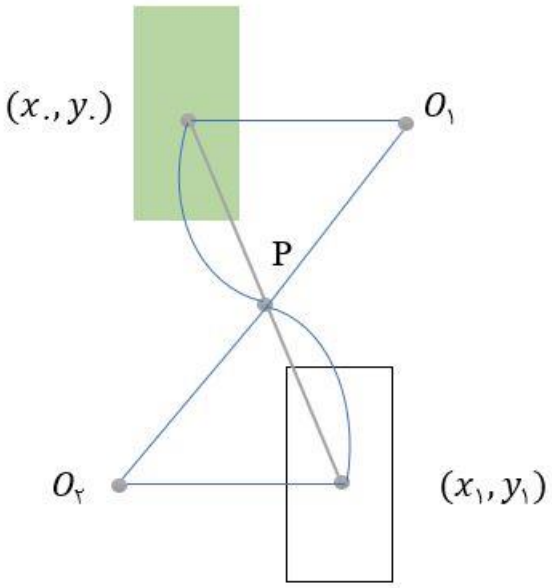


مستطیل سفید نشان دهنده جای پارک

شکل ۳۳-۴ طراحی مسیر پارک دوبل

با توجه به ابعاد ماشین طول خط واصل ۱۳.۴ به دست می‌آید. و نقطه وسط خط را به دو بخش به طول ۶.۷ تقسیم می‌کند.

با به دست آمدن نقطه تغییر کمان، نقاط شروع و پایان دو کمان به دست می‌آید . y نقطه‌ی مرکز کمان اول را برابر با y نقطه شروع پارک و y نقطه‌ی مرکز کمان دوم را برابر با y نقطه پارک تعیین شده در نظر می‌گیریم. زاویه‌ی بین شعاع کمان ها و خط واصل گفته شده به دست می‌آید.



شکل ۴-۳۴ کمان های به دست آمده مسیر طراحی شده
درجهت پارک دوبل

زاویه‌ی بین شعاع کمان ها و خط واصل به دست آمده، ۶۱ درجه به دست می‌آید و با توجه به برابر بودن شعاع یک کمان، مثلث رسم شده برای هر کمان متساوی الساقین بوده، یک زاویه‌ی دیگر ۶۱ درجه و زاویه مقابل کمان ۵۸ درجه به دست می‌آید.

با استفاده از طول ضلع مثلث و زاویه‌ی مجاور آن طول شعاع کمان ها به دست می‌آید.

$$R = \frac{6.7/2}{\cos 61} = 6.9 \qquad (۴-۹۳)$$

با به دست آمدن شعاع کمان ها، مراکز دو دایره مشخص می‌شوند.

$$y(O_1) = y_0 \qquad x(O_1) = x_0 + 6.9 \qquad (۴-۹۴)$$

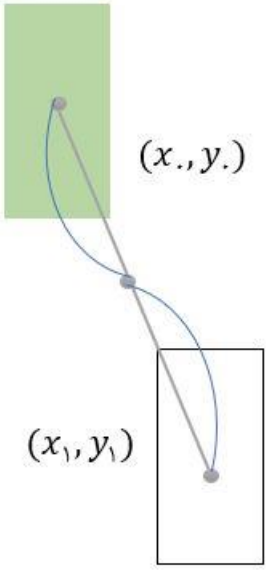
$$y(O_2) = y_1 \qquad x(O_2) = x_1 - 6.9 \qquad (۴-۹۵)$$

با توجه به مراحل طی شده، با داشتن نقطه پارک، در ابتدا نقطه شروع پارک به دست آمده، سپس مسیر پارک طراحی می‌شود. یک کمان از دایره اول به مرکز O_1 و شعاع ۷ به اندازه زاویه ۵۸ درجه طی می‌شود و خودرو به نقطه P می‌رسد. حال ادامه‌ی مسیر با کمان دایره دوم به مرکز O_2 و شعاع ۷ به اندازه زاویه ۵۸ درجه به دست می‌آید. با طی شدن این دو کمان، خودرو به محل پارک می‌رسد و به اصطلاح پارک دوبل می‌کند.

بسته به جهت خودرو و سمت راست یا چپ قرار گرفتن خودرو نسبت به محل پارک، معادلات دایره برای کمان‌های مورد نیاز متفاوت می‌باشند. چهار حالت برای شروع پارک وجود دارد.

۱. حالت بیان شده تا به این جا که محل شروع پارک بالا سمت چپ نقطه پارک قرار می‌گیرد.

$$x_0 = x_1 - D - w \qquad y_0 = y_1 + L + S \tag{۴-۹۶}$$



معادله دایره اول

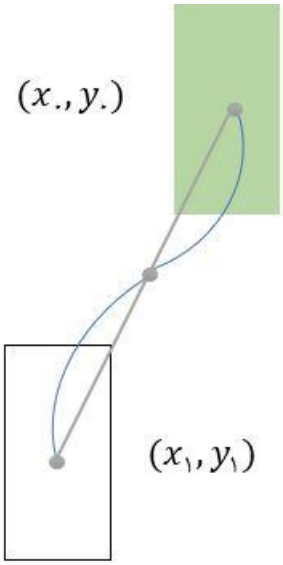
$$(x - (x_0 + 6.9))^2 + (y - y_0)^2 = 6.9^2 \tag{۴-۹۷}$$

معادله دایره دوم

$$(x - (x_1 - 6.9))^2 + (y - y_1)^2 = 6.9^2 \tag{۴-۹۸}$$

۲. محل شروع پارک بالا سمت راست نقطه پارک قرار می‌گیرد.

$$x_0 = x_1 + D + w \qquad y_0 = y_1 + L + S \tag{۴-۹۹}$$



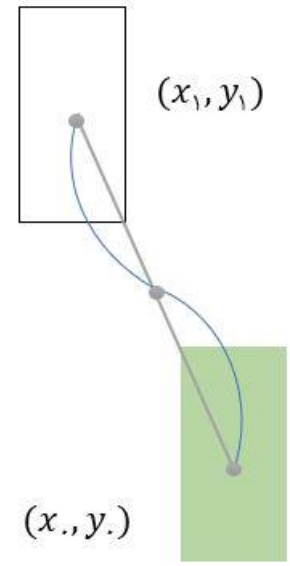
معادله دایره اول

$$(x - (x_0 - 6.9))^2 + (y - y_0)^2 = 6.9^2 \tag{۴-۱۰۰}$$

معادله دایره دوم

$$(x - (x_1 + 6.9))^2 + (y - y_1)^2 = 6.9^2 \tag{۴-۱۰۱}$$

۳. محل شروع پارک پایین سمت راست نقطه پارک قرار می‌گیرد.



$$x_0 = x_1 + D + w \qquad y_0 = y_1 - L - S \tag{۴-۱۰۲}$$

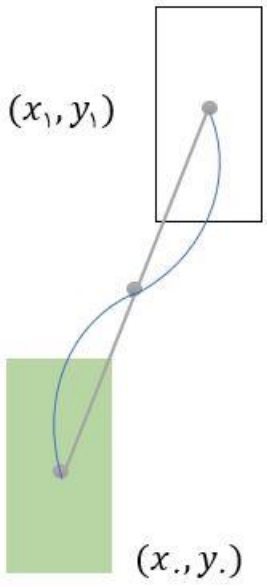
معادله دایره اول

$$(x - (x_0 - 6.9))^2 + (y - y_0)^2 = 6.9^2 \tag{۴-۱۰۳}$$

معادله دایره دوم

$$(x - (x_1 + 6.9))^2 + (y - y_1)^2 = 6.9^2 \tag{۴-۱۰۴}$$

۴. محل شروع پارک پایین سمت چپ نقطه پارک قرار می‌گیرد.



$$x_0 = x_1 + D + w \qquad y_0 = y_1 - L - S \tag{۴-۱۰۵}$$

معادله دایره اول

$$(x - (x_0 + 6.9))^2 + (y - y_0)^2 = 6.9^2 \tag{۴-۱۰۶}$$

معادله دایره دوم

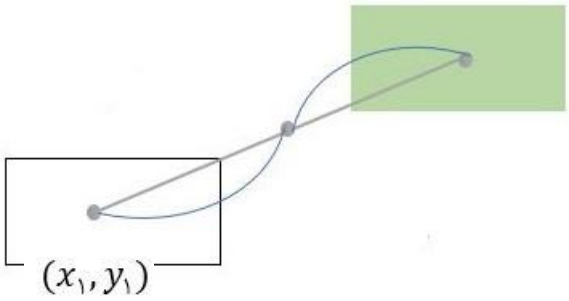
$$(x - (x_1 - 6.9))^2 + (y - y_1)^2 = 6.9^2 \tag{۴-۱۰۷}$$

این طراحی توسط خود تیم صورت گرفته و برداشت هایی از مقالات [۲۹۶-۲۹۷-۲۹۸] می‌باشد.

همچنین طراحی های پارک دوبل ماشین های افقی و طراحی پارک عمودی یا به اصطلاح گاراژ به همین شکل انجام می‌شود و پیاده سازی این پارک‌ها جزو چشم اندازهای پروژه می‌باشد. چهار حالت طراحی پارک دوبل افقی:

۱. محل شروع پارک بالا سمت راست نقطه پارک قرار می‌گیرد.

(x_0, y_0)

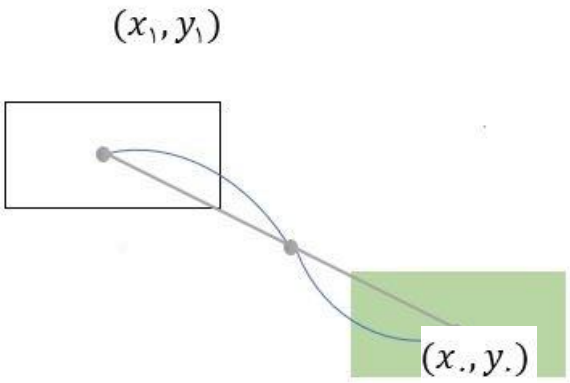


$$x_0 = x_1 + s + L \qquad y_0 = y_1 + D + W \tag{۴-۱۰۸}$$

$$(x - x_0)^2 + (y - (y_0 - 6.9))^2 = 6.9^2 \tag{۴-۱۰۹}$$

$$(x - x_1)^2 + (y - (y_1 + 6.9))^2 = 6.9^2 \tag{۴-۱۱۰}$$

۲. محل شروع پارک پایین سمت راست نقطه پارک قرار می‌گیرد.



$$x_0 = x_1 + S + L \qquad y_0 = y_1 - D - W \tag{۴-۱۱۱}$$

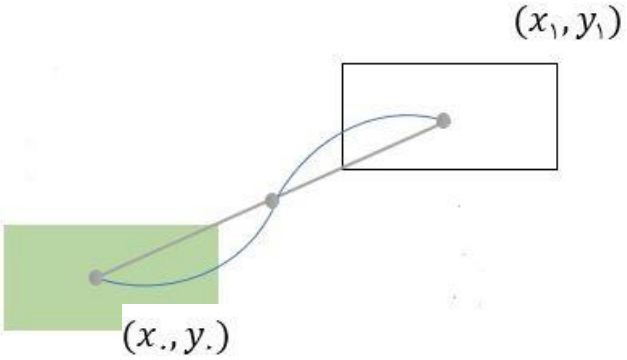
$$(x - x_0)^2 + (y - (y_0 + 6.9))^2 = 6.9^2 \tag{۴-۱۱۲}$$

$$(x - x_1)^2 + (y - (y_1 - 6.9))^2 = 6.9^2 \tag{۴-۱۱۳}$$

۳. محل شروع پارک پایین سمت چپ نقطه پارک قرار می‌گیرد.

$$x_0 = x_1 - S - L \quad y_0 = y_1 - D - W$$

(۴-۱۱۴)



$$(x - x_0)^2 + (y - (y_0 + 6.9))^2 = 6.9^2$$

(۴-۱۱۵)

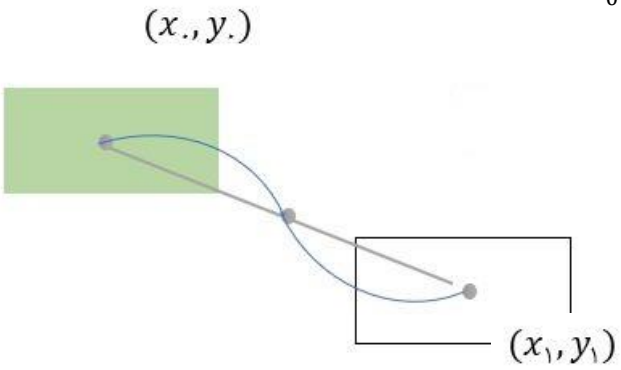
$$(x - x_1)^2 + (y - (y_1 - 6.9))^2 = 6.9^2$$

(۴-۱۱۶)

۴. محل شروع پارک بالا سمت چپ نقطه پارک قرار می‌گیرد.

$$x_0 = x_1 - S - L \quad y_0 = y_1 + D + W$$

(۴-۱۱۷)



$$(x - x_0)^2 + (y - (y_0 - 6.9))^2 = 6.9^2$$

(۴-۱۱۸)

$$(x - x_1)^2 + (y - (y_1 + 6.9))^2 = 6.9^2$$

(۴-۱۱۹)

۴-۳- توسعه و بیان قانون کنترل

به منظور کنترل مدل دینامیکی خودرو برای انجام مانور پارک خودکار از کنترل پیش‌بین مدل[۲۶۹] استفاده شده است. در ادامه به صورت مختصر این روش کنترلی به صورت مختصر توضیح داده شده است.

۴-۳-۱- مقدمه‌ای بر کنترل پیش‌بین مدل

در طول سالها ، سیستم های کنترل در حوزه روباتیک به شدت تغییر کرده است تا امکان انجام اقدامات پیچیده تر و پویایی غیر خطی فراهم شود. کنترل پیش بینی مدل یکی از استراتژی هایی است که اجازه رفتارهای پیچیده

[269] Model Predictive Control

به سیستم می دهد. این پیچیدگی ها شامل محیط های پویا یا محیط های غیرقابل دسترسی می باشند که اجازه دخالت انسان را نمی دهند. علاوه بر این، اکثر مدل های ربات بسیار غیر خطی هستند که این امر استراتژی های کنترل را دشوارتر می کند. استراتژی های معمول کنترل صنعتی مانند کنترل PD و کنترل PID در تضمین انواع مختلف ویژگی ها ناموفق می باشند. تحقیقات زیادی با استراتژی های مختلف کنترل مطلوب مانند کنترل تطبیقی و کنترل تقلید وجود دارد، اما کنترل پیش بین مدل (MPC) به وضوح در سطح پیشرفته برجسته تر است[۲۹۹].

کنترل پیشبین مدل در اواخر دهه هفتاد میلادی توسعه داد شد. این روش کنترلی به صورت ذاتی کنترلر حلقه باز است. کنترل پیشبین مدل ابتدا در کنترل فرآیندهای صنعتی شیمیایی مورد توجه قرار گرفت. اما امروزه در حوزه‌های دیگری همچون رباتیک[۳۰۰]، صنایع سیمان، برج‌های خنک کننده[۳۰۱] و مولدهای بخار[۳۰۲] مورد استفاده قرار گرفته است. دلیل موفقیت کنترل پیشبین مدل آن است که در صنعت برای صنعت توسعه داده شده است. استفاده از این روش کنترلی در حوزه‌های مختلف نشا ن دهنده پتانسیل بالای این روش در طراحی کنترلرهای با عملکرد مطلوب را نشان می‌دهد. هر چند این روش کنترلی یک روش خاص را ارائه نمی‌دهد، اما جز روشهایی است که از مدل فرآیند به صورت صریح برای تعیین سیگنال کنترلی استفاده می‌کند. همچنین این روش کنترلی جز خانواده کنترل پیشبین است و برای این منظور می‌توان به دلایل زیر اشاره کرد[۳۰۳].

- استفاده صریح از مدل برای تخمین زدن خروجی سیستم در آینده
- تعیین سیگنال کنترلی با استفاده از کمینه کردن تابع هزینه

از مزایای استفاده از کنترل پیشبین مدل می‌توان به قابلیت در نظر گرفتن قیود سیستم، قابل استفاده در فرآیندهای چند متغیره، استفاده از یک روش کنترل پیش خوراند[۲۷۰] برای جبران اغتشاشات سیستم، قابلیت استفاده برای سیستمهای غیرخطی و ... اشاره کرد. از طرف دیگر، این روش دارای معایبی نیز هست که می‌توان افزایش محاسبات کنترلر با افزایش پیچیدگی سیستم را نام برد که باعث ایجاد تاخیر در عملکرد سیستم می‌گردد. همچنین، در صورتی که دینامیک سیستم دارای مولفه‌های متغیر با زمان باشد، تمامی محاسبات مرتبط در هر گام زمانی تکرار می‌گردد که کاهش شدید سرعت عملکرد کنترلر را در بردارد. اصلی ترین عیب کنترل پیشبین مدل، نیاز آن به مدل مناسب از فرآیند است. در صورتی که مدل مناسبی از فرآیند در دسترس نباشد، عملکرد کنترلر را تحت تاثیر قرار خواهد داد. همچنین، کنترل پیشبین مدل به صورت پیشفرض پایداری و مقاوم بودن سیستم را تصمین نمی‌کند.

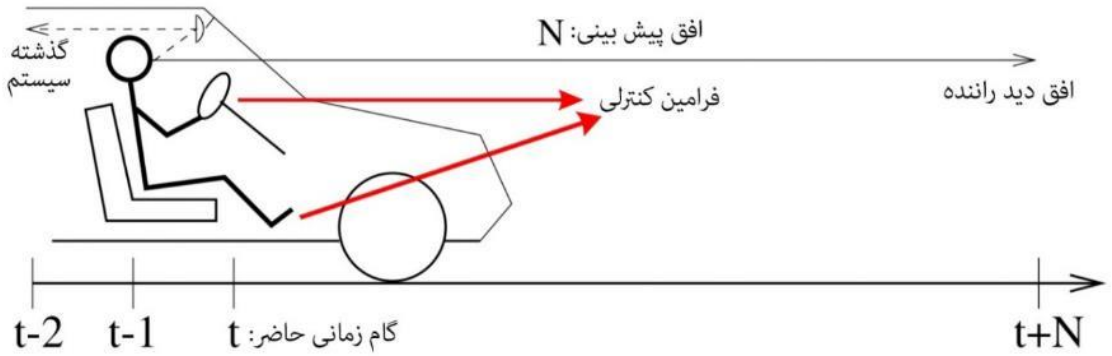
۱-۱-۳-۴- مقایسه کنترل تناسبی- انتگرالی- مشتقی (PID) و کنترل پیش بین مدل

[270] Feed Forward

در کنترل پیش بین مدل سیگنال کنترلی همواره از طریق حل یک مسئله بهینه‌سازی مقید حاصل می‌شود. تابع هزینه مسئله بهینه‌سازی، بسته به نوع سیستم می‌تواند انرژی مصرفی، سوخت مصرفی، انرژی خطای ردیابی و... باشد. به عنوان مقایسه‌ای بین کنترل‌کننده رایج تناسبی-انتگرالی-مشتقی PID و کنترل پیش‌بین مدل، کنترل سرعت یک موتور جریان مستقیم ۱۲ ولت را در نظر می‌گیریم. در هنگام طراحی پارامترهای کنترل‌کننده PI ما فقط مشخصه‌های حوزه فرکانس یا زمان (مانند درصد بالازدگی، زمان نشست، پهنای باند و...) را در نظر می‌گیریم و کاری به انرژی مصرفی سیستم کنترل نداریم. اما به دلیل این‌که موتور ۱۲ ولت است، نباید ولتاژی بیشتر از آن را به موتور اعمال کنیم، چون ممکن است موتور بسوزد. پس سیگنال کنترلی باید کوچکتر از ۱۲ ولت باشد که این قید را نمی‌توان در تنظیم پارامترهای کنترل‌کننده PI در نظر گرفت. از طرف دیگر، مسائل انرژی مصرفی و قیدهای مربوط به سیگنال‌های کنترلی و حتی خود حالت‌های سیستم را می‌توان در کنترل پیش‌بین مدل در نظر گرفت. برای مثال می‌توان کنترل‌کننده‌ای طراحی کرد که سیگنال کنترلی کمتر از ۱۲ ولت باشد، سرعت موتور هیچ وقت بزرگتر از ۱۰۰۰ دور بر دقیقه نشود و هم زمان با برآورده شدن این قیدها انرژی مصرفی موتور نیز حداقل شود[۳۰۴].

۲-۱-۳-۴- استراتژی کنترل پیش‌بین مدل

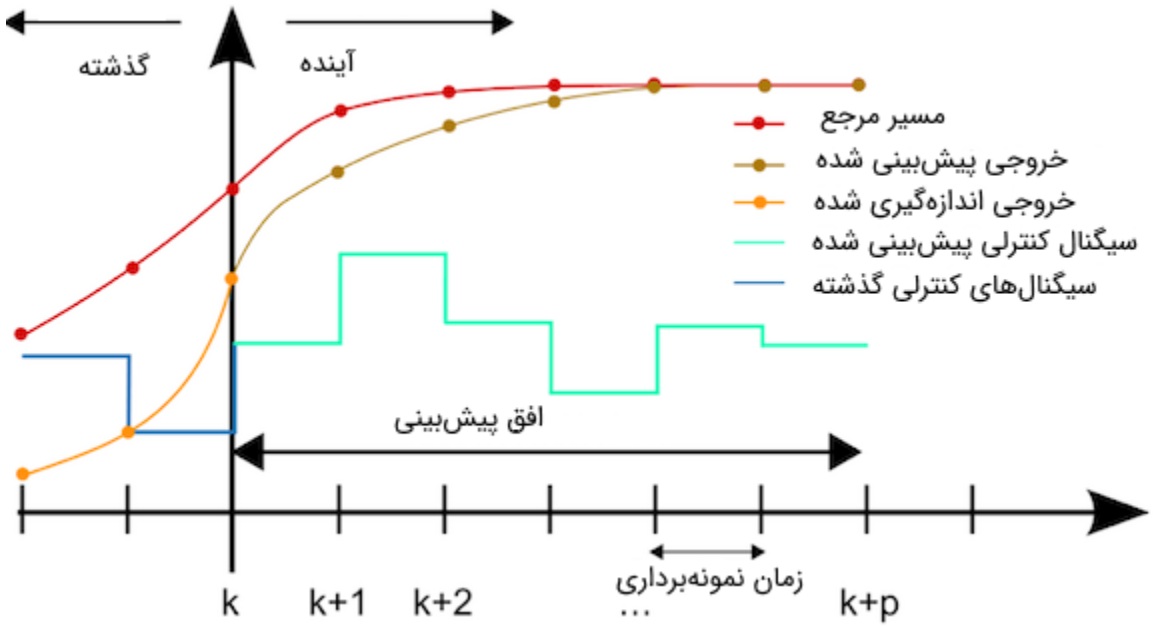
تصور کنید در یک اتاق تاریک قدم می زنید. شما سعی می کنید محیط اطراف را حس کنید و بهترین مسیر را در جهت یک هدف پیش بینی می کنید، اما درهر واحد زمان تنها یک قدم بر می دارید و چرخه را تکرار می کنید. به عنوان مثالی دیگر، راننده به جاده پیش رو نگاه می کند و مسیر مرجع مورد نظر را برای یک افق کنترل محدود می داند و با در نظر گرفتن ویژگی های خودرو (مدل ذهنی خودرو) تصمیم می گیرد که برای کنترل مسیر مورد نظر ، کدام اقدامات کنترلی (گاز ، ترمز ، فرمان) را انجام دهد که فقط اولین اقدام کنترلی در هر لحظه انجام می شود و این روش برای تصمیم گیری کنترل بعدی تکرار می شود[۳۰۵].



شکل ۳۵-۴ شباهت رانندگی اتومبیل با استراتژی کنترل پیش‌بین مدل[۳۰۵]

به طور مشابه ، فرایند MPC مانند قدم زدن در یک اتاق تاریک و رانندگی است. ماهیت MPC بهینه سازی ورودی های قابل دستکاری و پیش بینی رفتار فرآیند است.

MPC یک فرایند تکرار شونده برای بهینه سازی پیش بینی های وضعیت روبات ها در افق محدود آینده و در عین حال دستکاری ورودی ها برای یک افق معین است. پیش بینی با استفاده از مدل فرآیند انجام می شود. بنابراین ، داشتن یک مدل پویا هنگام اجرای MPC ضروری است. این مدل های فرآیندی عموماً غیر خطی هستند، اما برای دوره های کوتاه مدت روش هایی مناسب مانند بسط تیلور برای خطی سازی این مدل ها وجود دارد. این تقریب ها در خطی سازی یا تغییرات غیرقابل پیش بینی در مدل های پویا ممکن است باعث خطا در پیش بینی شود. بنابراین، MPC فقط بر اساس ورودی کنترل اولیه محاسبه شده عمل می کند و سپس پیش بینی های بهینه شده را بر اساس بازخورد و فیدبک دوباره محاسبه می کند. این بدان معناست که MPC یک استراتژی کنترل تکرار شونده ، مبتنی بر مدل ، پیش بینی ، بهینه سازی و فیدبک است.



شکل ۳۶-۴ اساس کار کنترلر MPC [۲۹۹]

شکل (۳۶-۳) اصول کار کنترل‌کننده MPC نشان می‌دهد. فرض کنید که وضعیت تمام سیگنال‌های موجود در سیستم تا نمونه k ام در دسترس باشد، یعنی حالت‌ها و خروجی‌های سیستم از نمونه ۰ تا نمونه k ام و همچنین

سیگنال کنترلی نیز از نمونه 0 تا $k-1$ام در دسترس باشند. اگر زمان نمونه‌بردای مشخص باشد، شماره نمونه به راحتی تبدیل به زمان واقعی خواهد شد .$(t = kT_s)$ به همین دلیل به جای نمونه از زمان استفاده می‌کنیم. حال هدف این است که سیگنال‌های کنترلی از زمان k تا زمان $k+p$ تا را طوری تعیین کنیم که خروجی سیستم از زمان k تا زمان $k+p$ تا حد ممکن به خروجی مرجع نزدیک باشد.

برای این کار باید خروجی سیستم از زمان k تا زمان $k+p$ تا پیش‌بینی شود، که با توجه به مشخص بودن مدل دینامیکی سیستم، به راحتی می‌توان پیش‌بینی را انجام داد. به پارامتر p افق پیش‌بینی گفته می‌شود و در واقع طول بازه پیش‌بینی را نشان می‌دهد.

MPC سه شرط اساسی برای کار دارد. اولین مورد یک تابع هزینه J است. دومین شرط مورد نیاز یک مدل پویا از روبات است که این مدل پویا MPC را قادر می سازد تا حالت های یک روبات را در یک افق معین با ورودی های مختلف شبیه سازی کند. مورد سوم الگوریتم بهینه سازی است که برای حل تابع بهینه سازی J استفاده می شود[۲۹۹].

$$\text{تابع هزینه} \quad : \quad J = \sum_{t=k}^{t=k+p} W_a(x_t - r_t) + W_b \Delta u_t{}^2 \qquad (۴-۱۲۰)$$

که متغیر ها عبارتند از :

J: تابع هزینه

x_t: وضعیت ربات در زمان t

r_t: وضعیت مرجع ربات در زمان t

u_t: ورودی پیش بینی شده روبات در زمان t

W_a و W_b: وزن ها با توجه به نیاز

تابع هزینه بالا خطای مسیر مرجع و همچنین تنش ناشی از انحراف شدید در ورودی های ربات را به حداقل می رساند.

به عنوان مثال فرض کنیم که مدل دینامیکی گسسته سیستم به صورت زیر باشد[۳۰۰]:

$$y(k+1) = y(k) + 0.5u(k) \qquad (۴-۱۲۱)$$

آنگاه با توجه به اینکه اطلاعات خروجی سیستم تا زمان k و اطلاعات سیگنال کنترلی تا زمان $k-1$ مشخص است، پس خروجی یک نمونه به جلو یا یک گام به جلو بصورت زیر خواهد بود:

$$\hat{y}(k+1 \mid k) = y(k) + 0.5u(k) \qquad (۴-۱۲۲)$$

از علامت $\hat{y}$ برای نشان دادن مقادیر پیش‌بینی استفاده می‌شود و $\hat{y}(k+1 \mid k)$ به معنی پیش‌بینی یک گام به جلو با داشتن اطلاعات تا زمان k است. حال با جایگذاری $k+1$ به جای k در رابطه بالا پیش‌بینی دو گام به جلو به دست می‌آید:

$$\hat{y}(k+2 \mid k) = y(k) + 0.5u(k) + 0.5u(k+1) \qquad (۴-۱۲۳)$$

به همین ترتیب، با جایگذاری‌های متوالی، پیش‌بینی تا p گام به جلو انجام می‌شود.

$$\hat{y}(k+p \mid k) = y(k) + 0.5u(k) + 0.5u(k+1) + \cdots 0.5u(k+p-1) \qquad (۴-۱۲۴)$$

واضح است که تمام پیش‌بینی‌ها، وابسته به سیگنال کنترلی در طول افق پیش‌بین (یعنی $u(k)$ تا $u(k+p-1)$) هستند. حال برای اینکه پیش‌بینی‌ها تا حد امکان به خروجی‌های مرجع نزدیک باشند، اغلب از تابع هزینه مجموع مربعات خطای ردیابی استفاده می‌شود. خطای ردیابی n گام به جلو به صورت زیر تعریف می‌شود:

$$e(k+n) = y_{ref}(k+n) - \hat{y}(k+n) \qquad (۴-۱۲۵)$$

که در آن n می‌تواند از یک تا p تغییر کند و $y_{ref}(k+n)$ مقدار سیگنال مرجع می‌باشد. حال مجموع مربعات خطا به صورت زیر تعریف می‌شود:

$$Cost = \sum_{n=1}^{n=p} (y_{ref}(k+n) - \hat{y}(k+n))^2 \qquad (۴-۱۲۶)$$

با بهینه‌سازی تابع هزینه بالا، سیگنال کنترلی در طول افق پیش‌بین به دست می‌آید. اما همان‌طور که قبلا اشاره شد، علاوه بر حداقل‌سازی تابع هزینه باید قیدهای فیزیکی نیز برآورده شوند. بهینه‌سازی تابع هزینه بالا ممکن است منجر به سیگنال کنترلی با دامنه بالا شود. برای رفع این مشکل، انرژی سیگنال کنترلی در طول افق پیش‌بین نیز به تابع هزینه اضافه می‌شود:

$$Cost = \sum_{n=1}^{n=p} (y_{ref}(k+n) - \hat{y}(k+n))^2 + \rho \sum_{n=1}^{n=p-1} u(k+n)^2 \qquad (۴-۱۲۷)$$

جمله دوم در تابع هزینه، باعث می‌شود که مصالحه‌ای بین دامنه سیگنال کنترلی و خطای ردیابی برقرار شود. پارامتر ρ به منظور تنظیم اهمیت نسبی خطای ردیابی نسبت به دامنه سیگنال کنترلی استفاده می‌شود. هرچه خطای ردیابی نسبت به دامنه سیگنال کنترلی اهمیت بیشتری داشته باشد، اندازه ρ کوچکتر انتخاب می‌شود.

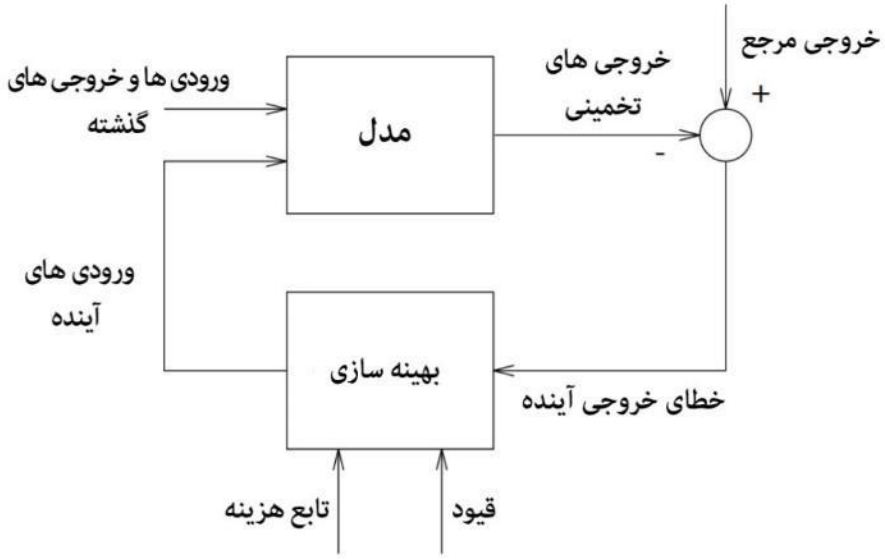
مسئله بهینه‌سازی بالا نامقید است و اگر بخواهیم قید سیگنال کنترلی صد در صد رعایت شود، باید مسئله بهینه‌سازی مقید زیر را محاسبه کنیم:

$$u(k) = \min \quad Cost = \sum_{n=1}^{n=p} (y_{ref}(k+n) - \hat{y}(k+n))^2 + \rho \sum_{n=1}^{n=p-1} u(k+n)^2 \qquad (۴-۱۲۸)$$
$$SubjectTo \quad u(t+k) \in [u_{min} \; u_{max}]$$

حال اگر خروجی‌های پیش‌بینی شده نیز مقید باشند (مثلا سرعت موتور هیچ وقت بزرگتر از ۱۰۰۰ دور بر دقیقه نشود)، مسئله بهینه‌سازی به صورت زیر تغییر پیدا می‌کند:

$$u(k) = \min Cost = \sum_{n=1}^{n=p} (y_{ref}(k+n) - \hat{y}(k+n))^2 + \rho \sum_{n=1}^{n=p-1} u(k+n)^2 \qquad (۴-۱۲۹)$$

$$SubjectTo \ \begin{cases} u(t+k) \in [u_{min} \ u_{max}] \\ y(t+k) \in [y_{min} \ y_{max}] \end{cases}$$



شکل ۳۷-۴ ساختار اساسی کنترل پیش بین مدل

۳-۱-۳-۴- اصل افق کاهنده[271]

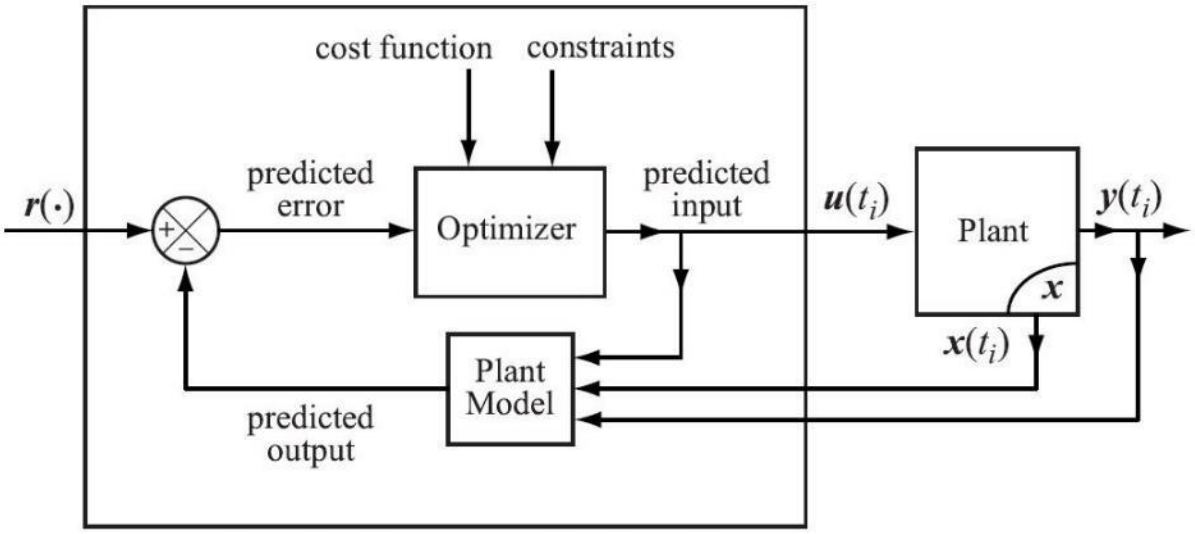
تابع هزینه تعریف شده دارای متغیرهای $u(k)$ تا $u(k+p-1)$ است و با حل مسئله بهینه‌سازی مقادیر بهینه آن‌ها محاسبه می‌شوند. اما طبق اصل افق کاهنده فقط اولین مقدار محاسبه شده مربوط به سیگنال کنترلی بهینه به پلانت اعمال می‌شود و در زمان بعدی یعنی $k+1$، تمام محاسبات قبل دوباره تکرار خواهد شد. پس فقط $u(k)$ به پلانت اعمال می‌شود و اثر آن یعنی $y(k+1)$ اندازه‌گیری می‌شود. حال ما اطلاعات خروجی

[271] Receding Horizon Principle

سیستم تا زمان $k+1$ را داریم و دوباره با طی کردن مسیر قبل، سیگنال کنترلی بهینه جدید $u(k+1)$ را به دست می‌آوریم و آن را به پلانت اعمال می‌کنیم تا $y(k+2)$ به دست آید و همین فرآیند را همواره تکرار می‌کنیم.

دلیل استفاده از اصل افق کاهنده، کاهش اثر عدم قطعیت‌ها، اغتشاشات و نویزهای موجود در سیستم است. زیرا این سیگنال‌ها اغلب دارای ماهیت تصادفی هستند و تا زمانی که رخ ندهند، نمی‌توانیم آن‌ها را اندازه‌گیری کنیم. حال اگر کل سیگنال کنترلی یعنی $u(k)$ تا $u(k+p-1)$ به پلانت اعمال شود، بدین معنی است که اثرات اغتشاشات در طول افق پیش‌بین لحاظ نخواهد شد. اما با p بار تکرار مسئله بهینه‌سازی و اندازه‌گیری اثرات اغتشاشات در هر نمونه‌برداری می‌توان اثرات آن‌ها را نیز جبران کرد[۳۰۳].

پس هدف تعیین سیگنال کنترلی در طول افق پیش‌بین برای حداقل‌سازی تابع هزینه و با در نظر گرفتن قیدهای حاکم بر سیستم است.



شکل ۳۸-۴ ساختار کامل کنترل پیش بین مدل [۳۰۳]

پس به طور خلاصه کنترل پیش بین ۳ مراحل زیر را پشت سر می‌گذارد:

۱. خروجیهای آینده سیستم در افق پیش بینی، در گامهای زمانی مشخص تخمین زده می‌شود. در واقع خروجیهای آینده سیستم به عنوان تابعی از ورودیها و خروجیهای گذشته و سیگنالهای کنترلی آینده سیستم که مدل نامیده می‌شود، پیش بینی می‌گردد.

۲. فرامین یا سیگنالهای کنترلی به وسیله بهینهسازی معیاری که تابع هزینه نامیده می شود ، استخراج می گردد. تابع هزینه در واقع معیاری برای همگرایی خروجی سیستم به خروجی مرجع است.
۳. از مجموعه سیگنالهای کنترلی محاسبه شده برای گامهای زمانی افق پیشبینی، تنها اولین سیگنال کنترلی که مربوط به گام زمانی حاضر است به سیستم اعمال میشود. در لحظه بعدی، تمامی مراحل تکرار میشود و سیگنالهای کنترلی بروزرسانی میشود.

اما استراتژی‌های مختلفی برای کنترل پیش‌بین مدل ذکر شده‌اند که در همه آنها از سه اصل زیر استفاده می‌شود:

- استفاده صریح از مدل دینامیکی فرآیند به منظور پیش‌بینی رفتار آینده سیستم
- محاسبه سیگنال کنترلی بهینه با بهینه‌سازی تابع هزینه و در نظر گرفتن قیود آن
- اصل افق کاهنده

تفاوت آنها در موارد زیر است:

- نوع مدل فرآیند برای پیش‌بینی: برخی از الگوریتم‌ها از پاسخ پله سیستم، برخی دیگر از مدل تابع تبدیل و برخی دیگر از مدل فضای حالت سیستم برای پیش‌بینی استفاده می‌کنند.
- شیوه در نظر گرفتن اغتشاشات وارد بر سیستم.
- نوع تابع هزینه: اغلب از مجموع مربعات خطای ردیابی و سیگنال کنترلی به عنوان تابع هزینه استفاده می‌شود. اما می‌توان مجموع قدرمطلق‌های خطای ردیابی و سیگنال کنترلی و یا هر تابع هزینه‌ای دیگری را بسته به نوع پلانت در نظر گرفت.

۴-۳-۱-۴- مزایای کنترل پیش‌بین مدل

- ذخیره‌سازی انرژی و هزینه : با درنظر گرفتن انرژی سیگنال کنترلی در تابع هزینه می‌توان مصرف انرژی سیستم را کاهش داد که به نوبه خود باعث کاهش هزینه‌های سیستم خواهد شد.
- جبران موثر اغتشاشات وارد بر سیستم : با استفاده از اصل افق کاهنده اثرات اغتشاشات (عدم قطعیت‌ها و نویزها) در کنترل پیش‌بین مدل نسبت به سایر کنترل‌کننده‌ها به صورت موثرتری حذف خواهد شد.

- کنترل سیستم‌های چند متغیر : تعمیم کنترل پیش‌بین مدل به سیستم‌های چند متغیره بسیار سرراست و مستقیم است و باعث پیچیدگی زیادی نمی‌شود. در حالی که طراحی کنترل‌کننده‌های کلاسیک مانندPID برای سیستم‌های چند متغیره بسیار سخت‌تر و چالش‌برانگیزتر از سیستم‌های تک‌متغیره است.
- پیاده‌سازی آسان در سیستم‌های دیجیتال :بر خلاف تئوری‌های پیچیده کنترل بهینه که نیازمند حل معادلات دیفرانسیل غیرخطی و پیچیده هستند، کنترل پیش‌بین مدل به راحتی در کامپیوترهای دیجیتال قابل پیاده‌سازی است.
- کاربرد در صنعت : کنترل‌کننده پیش‌بین مدل از صنعت نشات گرفته است و بسیاری از استراتژی‌های کنترل پیش‌بین مدل به خوبی بر روی پلانت‌های صنعتی مانند ربات‌ها، توربین‌های بادی و بخار، اتوپایلوت، نورد فلزات، تولید سیمان و صنایع نفت و پتروشیمی کارایی خودشان را نشان داده‌اند.

۴-۳-۱-۵- معایب کنترل پیش‌بین مدل

- بزرگترین عیب کنترل پیش بین مدل، نیاز آن به مدل دقیق فرآیند است، زیرا در این کنترل‌کننده در قدم اول باید رفتار آینده سیستم پیش‌بینی شود. بنابراین اگر مدل ریاضی سیستم دقیق نباشد، پیش‌بینی‌های خروجی سیستم نیز معتبر نخواهد بود و در نتیجه منجر به خطا خواهد شد.
- پیچیده شدن حل مسئله بهینه‌سازی برای سیستم‌های غیرخطی عیب دیگر کنترل‌پیش بین است. اگر دینامیک سیستم غیرخطی باشد آنگاه تابع هزینه کنترل پیش‌بین مدل یک تابع پیچیده از متغیرهای تصمیم (سیگنال کنترلی در طول افق کنترل) خواهد شد و بهینه‌سازی آن مشکلات زیادی را به همراه خواهد شد. البته برای رفع این مشکل می‌توان با تعریف یک تبدیل و نگاشت غیرخطی، سیستم غیرخطی را به یک سیستم خطی تبدیل کرد.

۲-۳-۴- مدل سیستم

معمول‌ترین مدل‌های مورد استفاده برای بیان فرآیند دینامیکی به عبارتند از[۳۰۵]:

- **مدل ضربه** : این مدل به صورت گسترده در صنعت مورد قبول واقع شده است.در این روش، مدل سیستم با استفاده از تحریک سیستم به وسیله ورودی ضربه به دست می‌آید. از مزیت‌های مدل ضربه می‌توان به کاربرد در فرآیندهای چند متغیره اشاره کرد. در طرف مقابل، نیاز به ضرایب بسیار برای توصیف رفتار دینامیکی سیستم از عیوب مدل ضربه است. در این مدل به طور کلی پاسخ سیستم با رابطه‌ی (۱۳۰-۳) بیان می‌شود:

$$y(t) = \sum_{i=1}^{N} h_i \, u(t-i) \qquad (۴-۱۳۰)$$

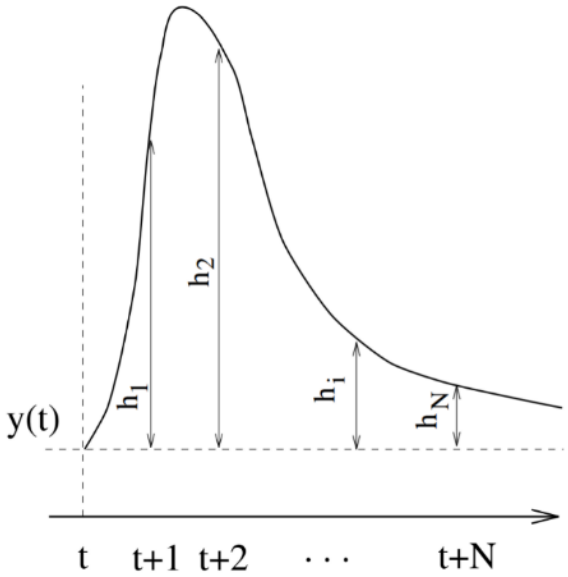
به طوری که معادله (۴-۱۳۰) شامل مولفه‌های زیر است:

h_i : مقدار پاسخ سیستم در گام زمانی نمونه برداری iام به ازای تحریک با ضربه واحد

$u(t-i)$: ورودی سیستم در گام زمانی $\mathrm{t}-\mathrm{i}$

N: افق پیش‌بینی

برای درک مولفه‌های بیان شده، شکل ۴-۳۹ آورده شده است.



شکل۴-۳۹ نمودار پاسخ سیستم فرضی به ازای ورودی ضربه واحد و مقادیر پاسخ سیستم در زمان[۳۰۵]

- **مدل پله:** این مدل بسیار شبیه به مدل ضربه می‌باشد. تنها تفاوت آن با مدل ضربه استفاده از تحریک پله واحد به جای ضربه واحد است. برای بیان مدل پله از معادله (۴-۱۳۱) استفاده می‌گردد.

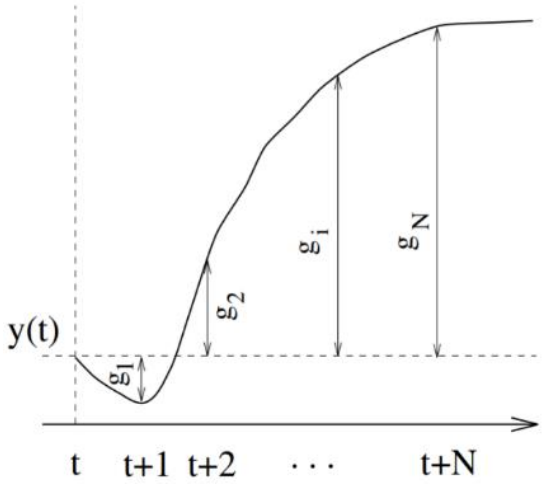
$$y(t) = \sum_{i=1}^{N} g_i \, \Delta u(t-i) \qquad (۴-۱۳۱)$$

به صورتی که دارای مولفه‌های زیر است:

$$\Delta u(t-i) = u(t) - u(t-1) \qquad (۴-۱۳۲)$$

g_i: مقدار پاسخ سیستم در گام زمانی نمونه برداری iام به ازای تحریک باظ پله واحد

برای درک مولفه‌های بیان شده، شکل۴۰-۴ آورده شده است.



شکل ۴۰-۴ نمودار پاسخ سیستم فرضی به ازای ورودی پله واحد و مقادیر پاسخ سیستم در زمان[۳۰۵]

- **مدل تابع تبدیل:** در این مدل از تابع تبدیل سیستم در فضای گسسته استفاده می‌شود. به دلیل در نظر گرفتن آسان تر زمان مرده[۲۷۲] و نیاز به ضرایب کمتر، از این مدل به صورت گسترده در فرآیندهای صنعتی استفاده می‌شود. با توجه به صورت و مخرج تابع تبدیل، می‌توان مدل را با مجموعه معادله (۴-۱۳۳) بیان کرد.

$$G = \frac{B}{A}$$
$$A(z^{-1})\, y(t) = B(z^{-1})\, u(t) \qquad (۴-۱۳۳)$$

به طوری که صورت و مخرج تابع تبدیل در فضای گسسته با معادلات (۳-۱۳۴) بیان می‌گردد.

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{na} z^{-na}$$
$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{nb} z^{-nb} \qquad (۴-۱۳۴)$$

از مدل تابع تبدیل برای بیان رفتار فرآیندهای ناپایدار استفاده می‌شود. بدین ترتیب، مقدار پاسخ سیستم در گام زمانی t+k که در گام زمانی t تخمین زده شده است، به صورت معادله (۳-۱۳۵) بیان می‌شود.

$$\hat{y}(t+k|t) = \frac{B(z^{-1})}{A(z^{-1})}\, u(t+k|t) \qquad (۴-۱۳۵)$$

[272] Dead time

- **مدل فضای حالت:** این مدل مشهورترین مدل برای بیان رفتار دینامیکی سیستم است که با معادلات (۴-۱۳۶) بیان می‌شود.

$$x(t) = Ax(t-1) + Bu(t-1)$$
$$y(t) = Cx(t) \qquad (۴-۱۳۶)$$

به طوری که بردار x بردار حالت سیستم و A و B و C ماتریس‌های سیستم هستند. از مدل فضای حالت، می‌توان برای بیان رفتار دینامیکی سیستم‌های چند متغیره استفاده نمود. به این ترتیب، مقدار پاسخ سیستم در گام زمانی t+k که در گام زمانی t تخمین زده شده است، به صورت معادله (۴-۱۳۷) بیان می‌شود.

$$\hat{y}(t+k|t) = C\hat{x}(t+k|t)$$
$$= C[A^k x(t) + \sum_{i=1}^{k} A^{i-1}Bu(t+k-i|t) \qquad (۴-۱۳۷)$$

۳-۳-۴- مدل اغتشاشات

برای مدل‌سازی اغتشاشات، از مدل‌های مختلفی استفاده می‌شود. یکی از مدل‌هایی که به صورت گسترده مورد استفاده قرار می‌گیرد، مدل میانگین متحرک خودهمبسته یکپارچه کنترل شده[۲۷۳] یا به اختصار کاریما[۲۷۴] است. در این مدل اغتشاش با اختلاف بین خروجی اندازه‌گیری شده سیستم و خروجی مدل سیستم بیان می‌شود که در معادلات (۴-۱۳۸) در فضای گسسته آورده شده است[۳۰۵].

$$n(t) = \frac{C(z^{-1})}{D(z^{-1})}\, e(t)$$
$$e(t) = y(t) - \hat{y}(t) \qquad (۴-۱۳۸)$$

به طوری که شامل مولفه‌های زیر است:

$C(z^{-1})$: عبارت چند جمله‌ای که عموما برابر با یک در نظر گرفته می‌شود

$D(z^{-1})$: عبارت چند جمله‌ای که عموما شامل انتگرال‌گیر است

[273] Controlled Auto Regressive and Integrated Moving Average
[274] CARIMA

خروجی اندازه‌گیری شده سیستم: $y(t)$

خروجی مدل سیستم: $\hat{y}(t)$

با توجه به این تفاوت‌ها، مهمترین استراتژی‌های کنترل پیش‌بین مدل عبارتند از[۳۰۰]:

- الگوریتم MAC: استفاده از پاسخ ضربه تجربی سیستم برای پیش‌بینی سیستم
- الگوریتم DMC: استفاده از پاسخ پله تجربی سیستم برای پیش‌بینی سیستم
- الگوریتم PFC: استفاده از مدل‌سازی فضای حالت سیستم برای پیش‌بینی سیستم
- الگوریتم GPC: استفاده از مدل‌سازی تابع تبدیل گسسته سیستم برای پیش‌بینی سیستم
- الگوریتم کنترل پیش‌بین مدل غیرخطی NMPC: کنترل پیش‌بین مدل برای سیستم‌های غیرخطی

۴-۳-۴- پیاده سازی در بستر نرم افزاری

پیاده‌سازی کنترلر ابتدا در نرم افزار متلب و سپس در محیط پایتون کدنویسی شده است. هم‌چنین، برای پیاده‌سازی کنترلر پیش‌بین مدل، از مدل دینامیکی فضای حالت استفاده شده است. مقادیر اولیه متغیرهای حالت برابر صفر در نظر گرفته شده است. مدت زمان شبیه‌سازی ۵۰ ثانیه، گام نمونه برداری برابر ۰/۱ ثانیه، افق پیش‌بینی و افق کنترل برابر ۵ گام جلوتر معادل ۰/۵ثانیه در نظر گرفته شده است. توضیحات مذکور در قالب کد متلب در ادامه آورده شده است.

```
clc;clear all;close all;
global Parameters
run('Specification')


%% Initializing

% Time
T0 = 0;      % [sec] Initial Time
Ts = 0.001;  % [sec] Time Step
Tf =100;     % [sec] Final Time
t  = T0:Ts:Tf;
Nt = numel(t);

% Initial Value of States
x0   = 0;
y0   = 0;
psi0 = 0;
u0   = 0;
v0   = 0;
r0   = 0;

```

```
% System Inputs
Theta0 = 0.8; % [%] [0.1 1] Accelerator
Delta0 = 0;   % [Deg] [-60 60]
Dande  = 1;   % 1 = First Gear , -1 = Reverse

Theta = Theta0 * ones(Nt,1);
Delta = Delta0 * (pi/180) * ones(Nt,1);

% Others
Gama = 0 * pi/180;     % [Deg] Slope Angle of Earth

%% Matrices Construction
% States
States = zeros(6 , Nt); % [ x_g, y_g, psi, u, v, r ]

States(1,1) = x0;
States(2,1) = y0;
States(3,1) = psi0;
States(4,1) = u0;
States(5,1) = v0;
States(6,1) = r0;
```

با توجه به توضیحات گفته شده در قسمت دینامیک، برای ورودی‌های سیستم محدودیت در نظر گرفته شده است. بدین ترتیب، فشردگی پدال گاز در بازه [۰/۵ ۱] و زاویه فرمان در بازه [۶۰- ۶۰] در نظر گرفته شده است. به جز این قیود، قیود دینامیکی دیگری برای سیستم در نظر گرفته نشده است. مسیر مرجع کنترلر در قالب ماتریس ref که شامل موقعیت‌های طولی و عرضی مرجع مرکز جرم در طی بازه زمانی مد نظر می‌باشد. توضیحات مذکور در قالب کد زیر آورده شده است.

```
%% NLC Conditions

In = zeros(2,Nt);
lb(1,1) = 0.5;  lb(2,1) = -60 * pi / 180 ;
ub(1,1) = 1;    ub(2,1) = 60 * pi / 180 ;

LB = repmat(lb , Tp/Ts,1) ;  % lower bond of input signal
UB = repmat(ub , Tp/Ts,1) ;  % upper bond of input signal

Uopt = zeros(Tp/Ts , 2) ;

Ref(1,:) = t;                % X_g  Reference Position
Ref(2,:) = 2*sin(2*pi*t/Tf); % Y_g  Reference Position

%% Constraints

A = [] ;
B = [] ;
Aeq = [] ;
Beq = [] ;
Fval  = zeros(size(t)) ;
```

با توجه به توضیحات گفته شده، عملکرد کنترلر شروع گردیده، مقادیر بهینه ورودی‌ها در افق پیش‌بینی موجود برای میل به مسیر مرجع در گام زمانی حاضر محاسبه گردیده و بنابر قاعده افق عقب‌نشینی[۲۷۵] تنها ورودی اول که مربوط به گام زمانی بعدی است به سیستم دینامیکی اعمال می‌گردد. خروجی سیستم دینامیکی، متغیرهای حالت بروز شده برای گام بعدی خواهد بود. این روند تا طی شدن مدت زمان شبیه‌سازی و برای تمامی گام‌های زمانی تکرار خواهد شد. توضیحات مذکور در قالب کد زیر آمده است:

```
1.  for i = 2:Nt
2.
3.      States0  = States(: , i-1);
4.
5.      [Uopt , Fval(i)] = fmincon(@(U)CostFunction(Parameters, States0 , repmat(transpose(Ref
    (:,i)), (Tp/Ts)+1 ,1), U , Ts , Tp , Gama) , Uopt , A , B , Aeq , Beq , LB , UB);
6.
7.      In(1,i-1) = Uopt(1,1) ;
8.      In(2,i-1) = Uopt(1,2) ;
9.
10.     States(:,i) = CarDynamic(Parameters, States(:,i-1), Uopt(1,:), Gama, Ts);
11.
12. end
```

تابع هزینه در نظرگرفته شده برای کنترلر تنها شامل مجموع مربعات اختلاف مقدار متغیرهای حالت مکانی و موقعیت‌های مرجع در افق پیش‌بینی در نظر گرفته شده است. به طور خلاصه تابع هزینه در رابطه ۱۳۹-۴ آورده شده است:

$$J(N_p) = \sum_{j=1}^{N_p} [x_r(t+j) - \hat{x}(t+j|t)]^2 + [y_r(t+j) - \hat{y}(t+j|t)]^2 \qquad (۴-۱۳۹)$$

معادله بالا شامل پارامترهای زیر است:

$x_r(t+j)$: موقعیت طولی مرجع در گام زمانی t+j

$\hat{x}(t+j|t)$: موقعیت طولی پیش‌بینی شده در گام زمانی t+j

$y_r(t+j)$: موقعیت عرضی مرجع در گام زمانی t+j

$\hat{y}(t+j|t)$: موقعیت عرضی پیش‌بینی شده در گام زمانی t+j

N_p: افق پیش‌بینی

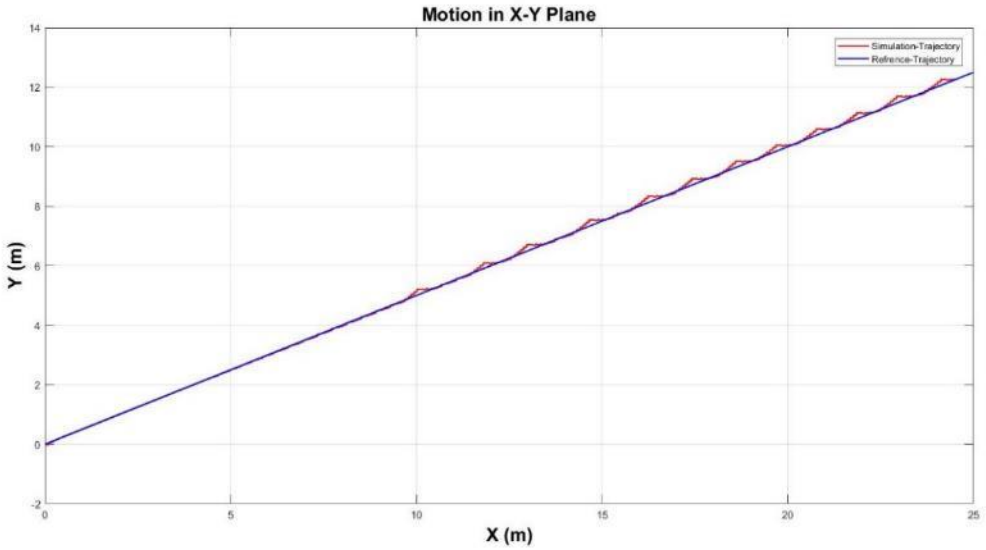
با توجه به توضیحات داده شده، برای صحت‌سنجی عملکرد کنترلر، دو مسیر مرجع زیر در نظر گرفته شده است:

[275] Receding Horizon

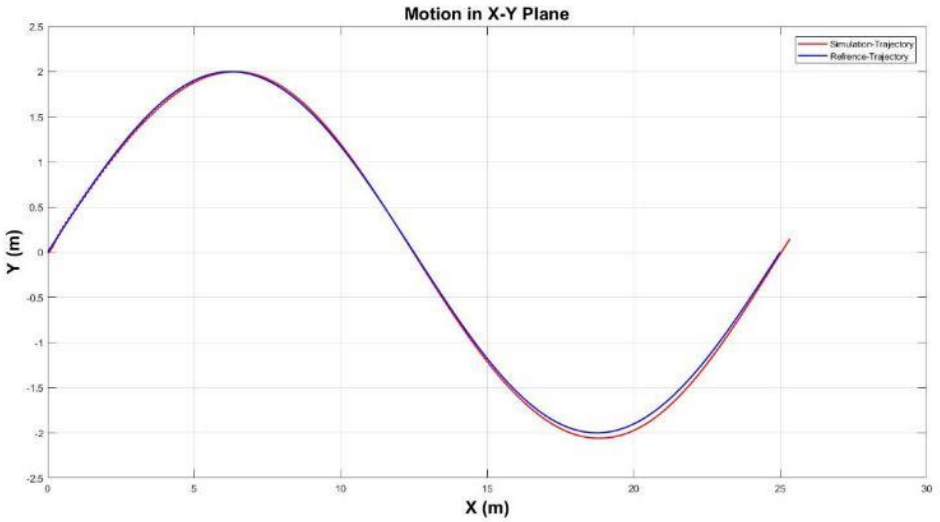
$$x = \frac{t}{2} \quad y = \frac{t}{4} \tag{۴-۱۴۰}$$

$$y(t) = 2 * \sin\left(\frac{2\pi x}{50}\right) \tag{۴-۱۴۱}$$

در ادامه، نتایج‌شبیه‌سازی کنترلر برای مسیرهای تعیین شده آورده شده است.



شکل ۴-۴۱ نتیجه شبیه‌سازی برای مسیر مرجع معادله ۴-۱۴۰



شکل ۴-۴۲ نتیجه شبیه‌سازی برای مسیر مرجع معادله ۴-۱۴۱

کنترلر طراحی شده، دارای عملکرد مطلوب برای تمامی مانورها نیست. علت این امر، محدودیت سرعت خودرو در دنده یک و عدم توانایی در رسیدن به سرعت مطلوب برای رهگیری مسیر مرجع است. به همین دلیل، دنده‌های بالاتر که سرعت بیشتری را برای سیستم فراهم می‌کنند، باید به مدل دینامیکی سیستم اضافه گردد. با توجه به کمبود وقت در هنگام انجام پروژه، برای انجام شبیه‌سازی نهایی، از مدل ساده شده سینماتیکی استفاده شده در مراجع استفاده گردیده[۳۰۶] و اضافه شدن مدل کامل دینامیکی سیستم به همراه تمامی دنده‌ها به آینده موکول شده است.

$$\dot{x} = u\cos(\psi) \quad (۴-۱۴۲)$$

$$\dot{y} = u\sin(\psi) \quad (۴-۱۴۳)$$

$$\dot{u} = a \quad (۴-۱۴۴)$$

$$\dot{\psi} = \frac{u\,tan(\delta)}{L} \quad (۴-۱۴۵)$$

این مجموعه معادلات شامل مولفه‌های زیر است:

x: موقعیت طولی مرکز جرم خودرو

y: موقعیت عرضی مرکز جرم خودرو

ψ: موقعیت زاویه‌ای خودرو

u: سرعت طولی مرکز جرم خودرو

a: شتاب طولی مرکز جرم خودرو

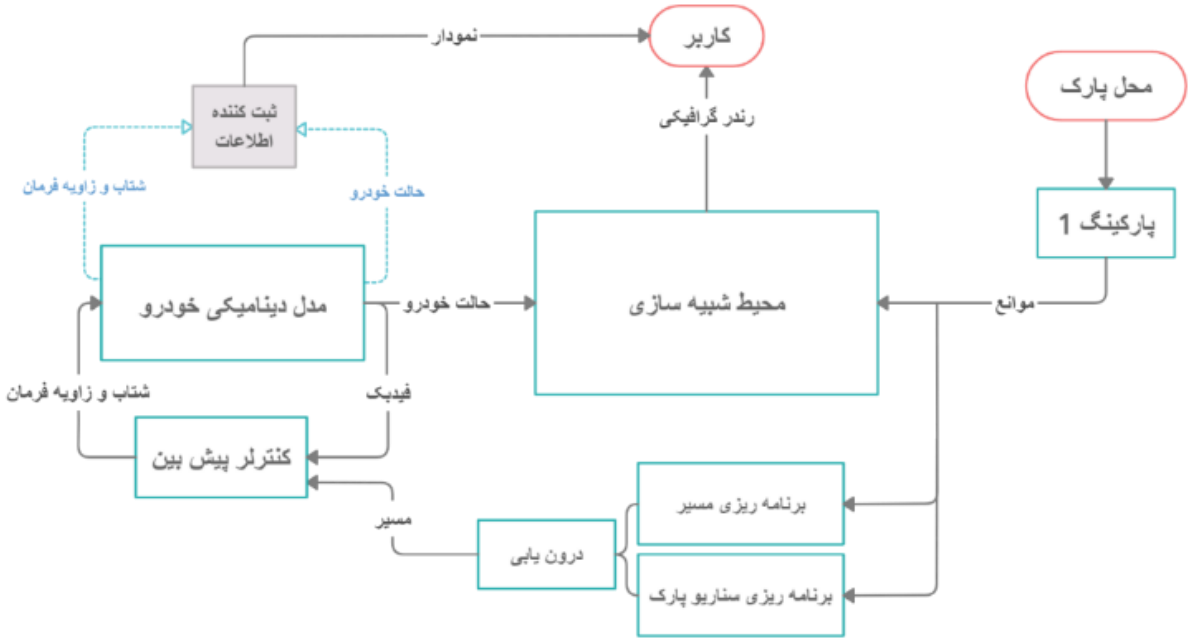
δ: زاویه فرمان چرخ‌ها

L: فاصله محورهای جلو و عقب خودرو

باقی مولفه‌های کنترلر در قسمت تشریح نرم‌افزار در حلقه مورد بررسی قرار گرفته‌اند. ورودی های سیستم سینماتیکی شتاب طولی و زاویه فرمان چرخ است و خروجی آن موقعیت مکانی مرکز جرم خودرو و زاویه موقعیت آن به صورت بیان شده در مختصات زمین است.لازم به ذکر است که تمامی قسمت های تشریح شده در بستر پایتون به صورت یکپارچه درآمده است که کد آن به پیوست تقدیم و در بخش بعد تشریح گردیده است.

۵- شبیه سازی پارک خودکار

به طور کلی، نرم‌افزار توسعه داده شده در بستر پایتون، به صورت نمودار بلوکی زیر خلاصه می‌گردد. اطلاعات اولیه خودرو شامل محل پارک و نقطه شروع حرکت خودرو توسط کاربر در برنامه وارد می‌گردد. این اطلاعات به بلوک پارکینگ فرستاده شده و سایر خودروها و دیوارها تحت عنوان موانع تولید می‌شود. در ادامه موانع تولید شده، به بلوک‌های محیط شبیه سازی، برنامه‌ریزی مسیر و برنامه‌ریزی سناریو پارک فرستاده شده تا عمل مسیریابی به درستی صورت بگیرد. بلوک درون‌یابی در ابتدا از مسیرهای طراحی شده، نمونه گیری می‌کند و سپس، به‌وسیله اسپلاین پایه[۲۷۶]، درون یابی می‌شود. مسیر نهایی، به قسمت هدایت خودرو ارسال شده و رهگیری مسیر در یک حلقه کنترلی شامل مدل دینامیکی خودرو و کنترلر پیش‌بین انجام می‌شود. در هرلحظه حالت فعلی خودرو به محیط شبیه‌سازی ارسال شده و خروجی گرافیکی برای نمایش به کاربر رندر می‌گردد.



شکل۱-۵ دیاگرام بلوکی نرم‌افزار درحلقه پارک خودکار

[276] Basis spline (B-spline)

۱-۵- زبان برنامه نویسی

پایتون[۲۷۷] یک زبان برنامه‌نویسی شیءگرا[۲۷۸]، تفسیری[۲۷۹]، سطح بالا[۲۸۰] و همه منظوره است که اولین بار در سال ۱۹۹۱ منتشر شده‌است. فلسفه اصلی طراحی پایتون خوانایی بالای کد است و فاصله‌های خالی در آن معنادار هستند و مکرر استفاده می‌شوند. ساختار زبانی و دیدگاه شیءگرا در پایتون به گونه‌ای طراحی شده‌است که به برنامه‌نویس امکان نوشتن کد منطقی و واضح (بدون ابهام) را برای پروژه‌های کوچک و بزرگ می‌دهد.

زبان برنامه نویسی پایتون از ماژول‌ها[۲۸۱] و بسته‌ها[۲۸۲] استفاده می‌کند، بدین معنا که برنامه‌های این زبان قابل طراحی به سبک ماژولار[۲۸۳] هستند و کدهای نوشته شده در یک پروژه در پروژه‌های گوناگون دیگر نیز قابل استفاده مجدد است. هنگامی که کاربری ماژول یا بسته مورد نیاز خود را توسعه دهد، می‌تواند آن را برای استفاده در دیگر پروژه‌ها گسترش به‌کار گیرد.

۱-۱-۵- مفهوم شیءگرایی

برنامه نویسی شیء گرا (OOP)، یک نوع شیوه‌ی برنامه نویسی کامپیوتری است که در آن طراحی نرم افزار به جای توابع، حول محور اشیاء و داده‌ها می‌چرخد. برنامه نویسی شیء گرا، کدنویسی را در پروژه‌های بزرگ بسیار آسان‌تر و خواناتر کرده و کدها منظم‌تر خواهند بود. پایتون نیز یکی از زبان‌های شیء گرا است که از اساس به صورت شیء نوشته شده است. در این زبان هر متغیر و داده‌ای که تعریف می‌کنیم، یک شیء است.

درنتیجه برای پیاده‌سازی الگوریتم‌های پارک خودکار از زبان پایتون استفاده شده است. هر بخش به صورت یک شیء در نامه تعریف شده است و این اشیاء با تبادل اطلاعات، فرآیند پارک را به سرانجام می‌رسانند.

۲-۱-۵- معرفی کتاب‌خانه ها

در پیاده‌سازی پارک خودکار با استفاده از زبان پایتون، از کتاب‌خانه‌های زیر (ماژول) جهت محاسبات عددی و پردازش گرافیکی اطلاعات استفاده شده است.

---

[277] Python
[278] Object-oriented programming (OOP)
[279] Interpreted
[280] High-Level language
[281] Modules
[282] Packages
[283] Modular

- **Numpy** : ماژول نامپای برای کار با اعداد و به صورت ماتریسی و آرایه‌های چندبعدی استفاده می‌شود.
- **OpenCV**[284] : این ماژول مجموعه ای از کتابخانه‌های برنامه‌نویسی پردازش تصویر و یادگیری ماشین است. این ماژول بیشتر بر پردازش تصویر بی درنگ[۲۸۵] تمرکز دارد.
- **Math** : این ماژول امکان دسترسی به توابع ریاضیاتی را در محیط پایتون فراهم می‌سازد.
- **Scipy** : ماژول سای‌پای، یک پکیج علمی و اوپن سورس مبتنی بر زبان پایتون است و برای انجام محاسبات علمی و مهندسی مورد استفاده قرار می‌گیرد. کتابخانه‌ی SciPy بر مبنای کتابخانه‌ی NumPy است و امکان کار با آرایه‌های n بُعدی را فراهم می‌کند. این کتابخانه برای کار با آرایه‌های Numpy ایجاد شده است و بسیاری از عملیات محاسباتی و بهینه‌سازی را به طور کارا ممکن می‌کند.
- **Time** : ماژول time برای کار با زمان در پایتون استفاده می‌شود.
- **Matplotlib** : ماژول Matplotlib برای مصورسازی داده‌ها و رسم نمودار به کار می‌رود.
- **OS** : این ماژول امکان استفاده از برخی قابلیت‌های وابسته به سیستم‌عامل را فراهم می‌آورد.

۲-۵- ورود اطلاعات

در توسعه کد، تاحد امکان سعی شده است که تجربه کاربری[۲۸۶] مناسبی ارائه داده شود و از ارتباط مستقیم کاربر با کد پرهیز شود. به همین منظور بلوک پارکینگ۱ طراحی شده تا به عنوان واسط، ارتباط بین کاربر و نرم‌افزار را شکل دهد. اجرای برنامه با استفاده دستور زیر در محیط خط فرمان[۲۸۷]، صورت می‌گیرد. مختصات نقطه شروع خودرو و شماره محل پارک خودرو در پارکینگ توسط کاربر انتخاب شده و هدایت خودرو مطابق با اطلاعات کاربر برنامه‌ریزی می‌شود.

```
1.  python main_autopark.py --x_start 0 --y_start 90 --parking 12
```

برای نمونه در دستور بالا، شروع پارک از مختصات (0,90) آغاز شده و پارک موازی در نقطه محل شماره ۱۲ برنامه‌ریزی می‌شود.

[284] Open Computer Vision Library
[285] Real Time
[286] User Experience
[287] Command Line

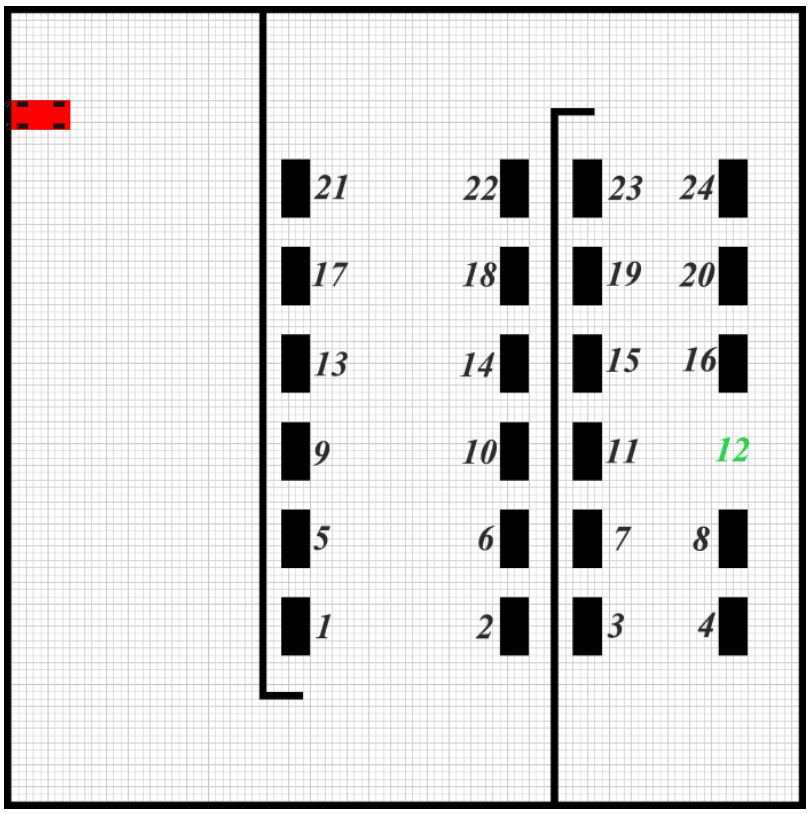
۱-۲-۵- پارکینگ ۱

برای تسریع فرآیند قرارگیری موانع شامل دیوارها و خودروها و هم‌چنین ساده‌سازی تعامل برنامه با کاربر، پارکینگ شماره ۱ طراحی شده است. این پارکینگ دارای ۲۴ محل پارک بوده و محل پارک خودرو توسط کاربر انتخاب می‌شود. سایر محل‌های پارک با موانع پر می‌شود تا شرایط پارک دوبل به وجود آید.

```
class Parking1:
    def __init__(self, car_pos):
        self.car_obstacle = self.make_car()
        self.walls = [[70,i] for i in range(-5,90) ]+\
                     [[30,i] for i in range(10,105)]+\
                     [[i,10] for i in range(30,36) ]+\
                     [[i,90] for i in range(70,76) ] #+ [[i,20] for i in range(-5,50)]
        # self.walls = [0,100]
        self.obs = np.array(self.walls)
        self.cars = {1 : [[35,20]], 2 : [[65,20]], 3 : [[75,20]], 4 : [[95,20]],
                     5 : [[35,32]], 6 : [[65,32]], 7 : [[75,32]], 8 : [[95,32]],
                     9 : [[35,44]], 10: [[65,44]], 11: [[75,44]], 12: [[95,44]],
                     13: [[35,56]], 14: [[65,56]], 15: [[75,56]], 16: [[95,56]],
                     17: [[35,68]], 18: [[65,68]], 19: [[75,68]], 20: [[95,68]],
                     21: [[35,80]], 22: [[65,80]], 23: [[75,80]], 24: [[95,80]]}
        self.end = self.cars[car_pos][0]
        self.cars.pop(car_pos)

    def generate_obstacles(self):
        for i in self.cars.keys():
            for j in range(len(self.cars[i])):
                obstacle = self.car_obstacle + self.cars[i]
                self.obs = np.append(self.obs, obstacle)
        return self.end, np.array(self.obs).reshape(-1,2)

    def make_car(self):
        car_obstacle_x, car_obstacle_y = np.meshgrid(np.arange(-2,2), np.arange(-4,4))
        car_obstacle = np.dstack([car_obstacle_x, car_obstacle_y]).reshape(-1,2)
        return car_obstacle
```

خروجی تابع generate_obstacles() مختصات نقطه مقصد به همراه مجموعه‌ای از موانع شامل خودروها و دیوار است که برای قرارگیری و پردازش گرافیکی به شیء محیط شبیه‌سازی داده می‌شود. خودروها به وسیله تابع make_car() در مبدا مختصات ساخته شده و سپس به مختصات تعیین شده انتقال می‌یابد. هدف از طراحی ماژولار موانع و ساخت پارکینگ ۱، گسترش انواع مختلف پارکینگ و راحتی توسعه آن است. اتصال محیط شبیه‌سازی و پارکینگ به شکل زیر صورت می‌گیرد.

```
start = np.array([args.x_start, args.y_start])

parking1 = Parking1(args.parking)
end, obs = parking1.generate_obstacles()

env = Environment(obs)
```



شکل ۲-۵ محل‌های قابل پارک در پارکینگ

۲-۲-۵- ورود اطلاعات به صورت دستی

علی‌رغم طراحی پارکینگ۱ برای ساده سازی تعامل با کد، امکان ورود مختصات اولیه و نهایی پارک به همراه مختصات موانع دلخواه نیز در برنامه وجود دارد. ورود مختصات اولیه و نهایی خودرو در کد main_autopark.py بخش default variables صورت می‌گیرد. هم‌چنین مختصات موانع دلخواه در بخش defining obstacles در توسط تابع make_square() تعریف می‌شود.

```
1.      ########################## defining obstacles ##############################
2.
3.      # add squares
4.      # square1 = make_square(10,65,20)
5.      # square2 = make_square(15,30,20)
6.      # square3 = make_square(50,50,10)
7.      # obs = np.vstack([obs,square1,square2,square3])
8.
9.      # Rahneshan logo
10.     # start = np.array([50,5])
11.     # end = np.array([35,67])
12.     # rah = np.flip(cv2.imread('READ_ME/rahneshan_obstacle.png',0), axis=0)
13.     # obs = np.vstack([np.where(rah<100)[1],np.where(rah<100)[0]]).T
14.
15.     # new_obs = np.array([[78,78],[79,79],[78,79]])
16.     # obs = np.vstack([obs,new_obs])
```

۳-۵- نمایش اطلاعات

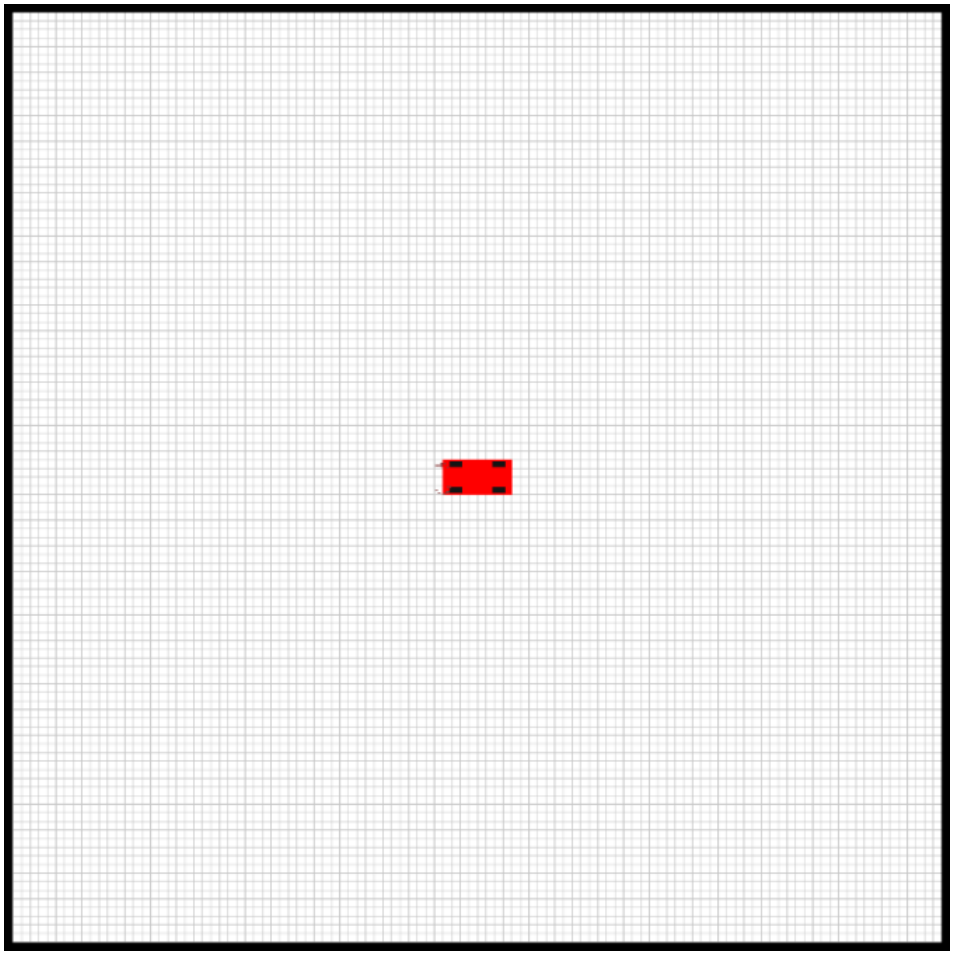
نمایش اطلاعات به صورت ساده، از مهم‌ترین بخش‌های نرم‌افزار است. خلاصه‌سازی فرآیندی پیچیده مانند پارک خودکار به صورت گرافیکی، روشی ساده و قابل فهم برای ارائه به کاربر می‌باشد. مطابق با دیاگرام بلوکی محیط شبیه سازی در مرکز برنامه، تمامی اطلاعات را از بخش‌های مختلف دریافت کرده و با جمع آوری و رسم آن‌ها در یک محیط دوبعدی در هر لحظه به کاربر نمایش می‌دهد.

۱-۳-۵- محیط شبیه سازی

اولین نیاز برای تست هر الگوریتم، وجود یک عامل[۲۸۸] در یک محیط[۲۸۹] شبیه‌سازی شده است. محیط‌های آماده طراحی شده زیادی مانند gym در پایتون و سایر زبان‌های برنامه‌نویسی وجود دارد اما محدودیت‌های آن‌ها و عدم قابلیت ارتقا، تیم را به سمت طراحی یک محیط شبیه‌سازی از پایه سوق داد. با استفاده از کتاب‌خانه numpy یک آرایه ۱۰۰۰ در ۱۰۰۰ پیکسل به عنوان پایه شبیه سازی درنظر گرفته می‌شود. دلیل استفاده از آرایه ۱۰۰۰ در ۱۰۰۰، افزایش رزولوشن نمایش است. این آرایه به یک شبکه ۱۰۰ در ۱۰۰ تقسیم‌بندی شده و سایر الگوریتم‌ها در فضای ۱۰۰ در ۱۰۰ اجرا می‌شود. در اطراف شبکه نیز موانعی به عنوان دیوار قرار داده شده تا همه الگوریتم‌ها به این فضا محدود شوند.

شیء محیط با استفاده از دستور env=Environment(obs) ساخته می‌شود. دستور res=env.render() برای قرارگیری عامل در محیط و رندرگرافیکی آن طراحی شده است. عامل در این محیط، یک خودرو شامل یک بدنه و ۴ چرخ می‌باشد. همه رندرهای گرافیکی در مختصات $(0,0)$ انجام شده، سپس به وسیله تابع rotate_car() به میزان لازم دوران یافته و سپس با توجه به مختصات آن انتقال می‌یابد. نکته قابل ذکر، توجه به جزئیات طراحی، حتی در تولید گِل‌های چرخ‌های عقب به صورت رندم است که حرکت خودرو را طبیعی جلوه می‌دهد.

[288] Agent
[289] Environment



شکل ۳-۵ محیط شبیه‌سازی توسعه داده شده

تابع render در هر لحظه خودرو و ۴ چرخ آن را در مبدا مختصات در متغیر بارگذاری می‌کند. سپس همه این اجزا به اندازه زاویه خودرو (دو چرخ جلو، به اندازه زاویه خودرو به اضافه زاویه چرخ) دوران می‌یابد. سپس با استفاده از مختصات خودرو تمامی اجزا انتقال یافته و روی صفحه محیط، رسم می‌شود. برای حرکت طبیعی خودرو، گِل‌های چرخ عقب به صورت رندم تولید می‌شود و با همان فرآیند دوران-انتقال به محل مورد نظر انتقال می‌یابد.

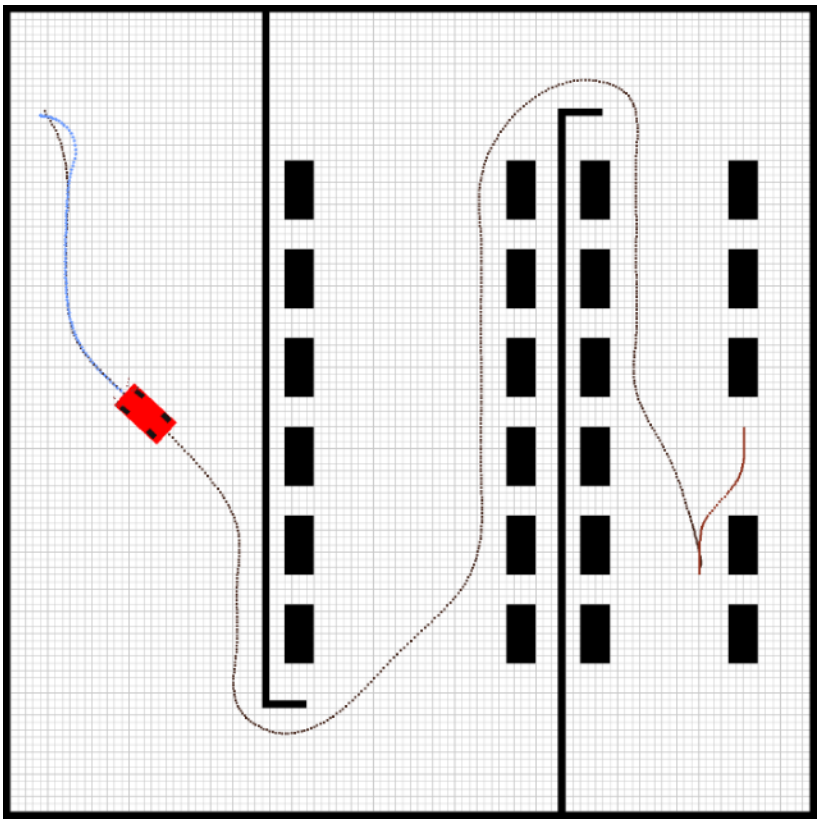
```
    def render(self, x, y, psi, delta):
        # x,y in 100 coordinates
        x = int(10*x)
        y = int(10*y)
        # x,y in 1000 coordinates
        # adding car body
        rotated_struct = self.rotate_car(self.car_struct, angle=psi)
        rotated_struct += np.array([x,y]) + np.array([10*self.margin,10*self.marg
in])
        rendered = cv2.fillPoly(self.background.copy(), [rotated_struct], self.co
lor)

        # adding wheel
        rotated_wheel_center = self.rotate_car(self.wheel_positions, angle=psi)
        for i,wheel in enumerate(rotated_wheel_center):

            if i <2:
                rotated_wheel = self.rotate_car(self.wheel_struct, angle=delta+ps
i)
            else:
                rotated_wheel = self.rotate_car(self.wheel_struct, angle=psi)
            rotated_wheel += np.array([x,y]) + wheel + np.array([10*self.margin,1
0*self.margin])
            rendered = cv2.fillPoly(rendered, [rotated_wheel], self.wheel_color)

        # gel
        gel = np.vstack([np.random.randint(-50,-
30,16),np.hstack([np.random.randint(-20,-10,8),np.random.randint(10,20,8)])]).T
        gel = self.rotate_car(gel, angle=psi)
        gel += np.array([x,y]) + np.array([10*self.margin,10*self.margin])
        gel = np.vstack([gel,gel+[1,0],gel+[0,1],gel+[1,1]])
        rendered[gel[:,1],gel[:,0]] = np.array([60,60,135])/255

        new_center = np.array([x,y]) + np.array([10*self.margin,10*self.margin])
        self.background = cv2.circle(self.background, (new_center[0],new_center[1
]), 2, [255/255, 150/255, 100/255], -1)

        rendered = cv2.resize(np.flip(rendered, axis=0), (700,700))
        return rendered
```

خروجی بلوک درون‌یابی شامل مسیر نهایی، برای رسم به محیط شبیه‌سازی داده شده و خط موجع در محیط شبیه‌سازی به وسیله دستور draw_path() رسم می‌شود. رنگ مسیر نیز برای تنوع به صورت رندم انتخاب می‌گردد. علاوه بر خط مرجع، موقعیت خودرو در هر لحظه نیز برای مقایسه، در محیط شبیه‌سازی نشان داده می‌شود.

```
    def draw_path(self, path):
        path = np.array(path)*10
        color = np.random.randint(0,150,3)/255
        path = path.astype(int)
        for p in path:
            self.background[p[1]+10*self.margin:p[1]+10*self.margin+3,p[0]+10*sel
f.margin:p[0]+10*self.margin+3]=color
```

رسم مسیر در کد اصلی main_autopark.py صورت می‌گیرد.

```
    env.draw_path(interpolated_path)
    env.draw_path(interpolated_park_path)
```



شکل ۴-۵ رندر اطلاعات گرافیکی در محیط شبیه سازی توسعه داده شده

۲-۳-۵- ثبت اطلاعات

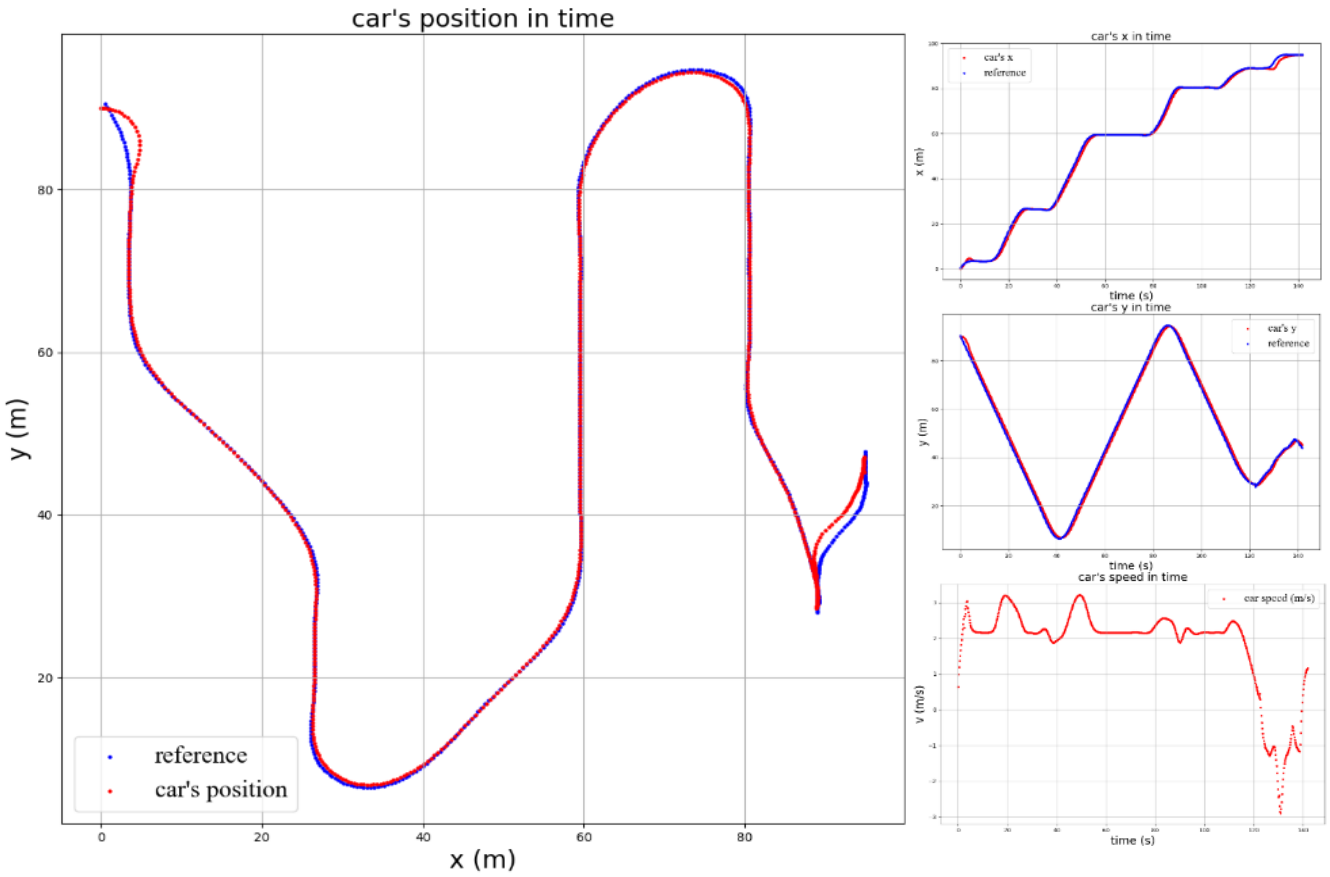
برای بررسی عملکرد کنترل‌کننده، لازم است تمامی اطلاعات در طول زمان ثبت شده و به صورت نمودار به کاربر نمایش داده شود. بدین منظور کلاس DataLogger در ماژول utils.py پیاده سازی شده تا همزمان با نمایش گرافیکی اطلاعات، ثبت آن‌ها نیز صورت بگیرد. هر شیء از این کلاس، سه لیست در خود دارد که اطلاعات مسیر مرجع، متغیرهای حالت خودرو و دستور تعیین شده توسط کنترلر، با دستور log() در آن‌ها نگه داشته می‌شود. در انتهای فرآیند پارک، با فراخوانی دستور save_data() تمامی اطلاعات به صورت نمودار در پوشه log results ذخیره می‌شود.

```
logger = DataLogger()

logger.log(point, my_car, acc, delta)
logger.save_data()

class DataLogger:
    def __init__(self):
        self.path = []
        self.car_state = []
        self.u = []

    def log(self, point, my_car, acc, delta):
        self.path.append(point)
        self.car_state.append([my_car.x, my_car.y, my_car.v, my_car.psi])
        self.u.append([acc, delta])

    def save_data(self):
        os.makedirs('log results', exist_ok=True)
        t = np.arange(0,len(self.path)/5,0.2)
        self.path = np.array(self.path)
        self.car_state = np.array(self.car_state)
        self.u = np.array(self.u)
        font = font_manager.FontProperties(family='Times New Roman', weight='bold',style='
normal', size=20)

        # plot x
        plt.figure(figsize=(12,8))
        plt.plot(t, self.path[:,0], color='b', linewidth=5)
        plt.plot(t, self.car_state[:,0], color='r', linewidth=4)
        plt.title('car\'s x in time',fontsize=20)
        plt.xlabel('time (s)',fontsize=20)
        plt.ylabel('x (m)',fontsize=20)
        plt.grid()
        plt.legend(['reference', 'car\'s x'], prop=font) # using a named size
        plt.savefig('log results/x.png')

        # plot y
        plt.figure(figsize=(12,8))
        plt.plot(t, self.path[:,1], color='b', linewidth=5)
        plt.plot(t, self.car_state[:,1], color='r', linewidth=4)
        plt.title('car\'s y in time',fontsize=20)
        plt.xlabel('time (s)',fontsize=20)
        plt.ylabel('y (m)',fontsize=20)
        plt.grid()
```

```
        plt.legend(['reference', 'car\'s y'], prop=font) # using a named size
        plt.savefig('log results/y.png')

        # plot v
        plt.figure(figsize=(12,8))
        plt.plot(t, self.car_state[:,2], color='r', linewidth=4)
        plt.title('car\'s speed in time',fontsize=20)
        plt.xlabel('time (s)',fontsize=20)
        plt.ylabel('v (m/s)',fontsize=20)
        plt.grid()
        plt.legend(['car speed (m/s)'], prop=font) # using a named size
        plt.savefig('log results/v.png')

        # plot psi
        plt.figure(figsize=(12,8))
        plt.plot(t, np.rad2deg(self.car_state[:,3]), color='r', linewidth=4)
        plt.title('car\'s angle in time',fontsize=20)
        plt.xlabel('time (s)',fontsize=20)
        plt.ylabel('psi (degree)',fontsize=20)
        plt.grid()
        plt.legend(['car angle (degree)'], prop=font) # using a named size
        plt.savefig('log results/psi.png')

        # plot position
        plt.figure(figsize=(12,12))
        plt.plot(self.path[:,0], self.path[:,1], color='b', linewidth=5)
        plt.plot(self.car_state[:,0], self.car_state[:,1], color='r', linewidth=4)
        plt.title('car\'s position in time',fontsize=20)
        plt.xlabel('x (m)',fontsize=20)
        plt.ylabel('y (m)',fontsize=20)
        plt.grid()
        plt.legend(['reference','car\'s position'], prop=font) # using a named size
        plt.savefig('log results/position.png')

        # plot accelerate
        plt.figure(figsize=(12,8))
        plt.plot(t, self.u[:,0], color='r', linewidth=4)
        plt.title('car\'s accelerate in time',fontsize=20)
        plt.xlabel('time (s)',fontsize=20)
        plt.ylabel('accelerate (m^2/s)',fontsize=20)
        plt.grid()
        plt.legend(['car accelerate (m^2/s)'], prop=font) # using a named size
        plt.savefig('log results/accelerate.png')

        # plot delta
        plt.figure(figsize=(12,8))
        plt.plot(t, np.rad2deg(self.u[:,1]), color='r', linewidth=4)
        plt.title('car\'s steer in time',fontsize=20)
        plt.xlabel('time (s)',fontsize=20)
        plt.ylabel('steer (degree)',fontsize=20)
        plt.grid()
        plt.legend(['car steer (degree)'], prop=font) # using a named size
        plt.savefig('log results/steer.png')

        print('all data saved on log results ...')
```

نمودارهای ذخیره شده شامل موارد زیر است.

- موقعیت x بر حسب زمان
- موقعیت y بر حسب زمان
- سرعت v بر حسب زمان
- زاویه خودرو psi بر حسب زمان
- موقعیت x بر حسب موقعیت y
- شتاب acc بر حسب زمان
- زاویه چرخ delta بر حسب زمان



شکل ۵-۵ نمودارهای ذخیره شده توسط ثبت کننده اطلاعات

۵-۴- برنامه ریزی مسیر

برنامه ریزی مسیر در ۲ مرحله انجام شده و ۲ مسیر به یک دیگر وصل می شوند تا مسیر نهایی را تشکیل دهند.

۵-۴-۱- برنامه ریزی مسیر بخش اول

در ماژول pathplanning.py، کلاس AStarPlanner الگوریتم A* را در بر دارد. در این کلاس هر گره خود یک شیء از کلاس Node بوده و گره قبل خود را ذخیره می‌کند. تابع calc_heuristic() نیز وظیفه محاسبه هزینه را برعهده دارد.

برای استفاده از این کلاس، کلاس دیگری به نام PathPlanning() نوشته شده که موانع را از پارکینگ دریافت کرده و تغییر مقیاس آن‌ها را انجام می‌دهد. سپس مسیر مناسب با استفاده از شیء الگوریتم A* پیدا شده و خروجی داده می‌شود.

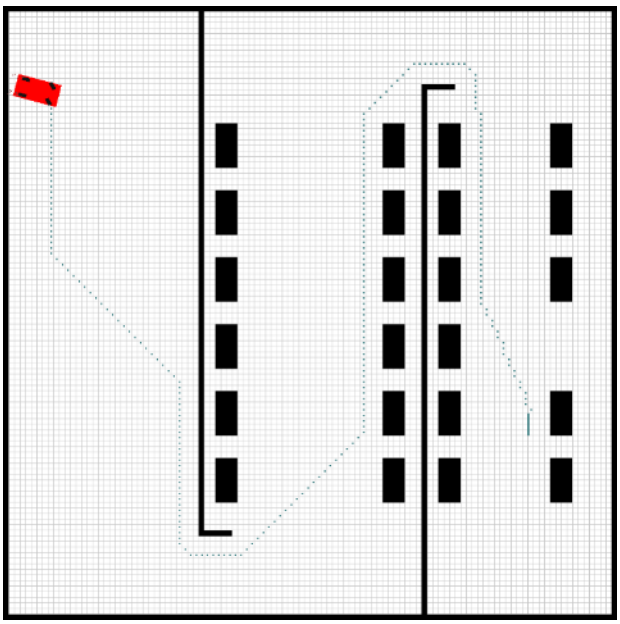
```
1. class PathPlanning:
2.     def __init__(self,obstacles):
3.         self.margin = 5
4.         #sacale obstacles from env margin to pathplanning margin
5.         obstacles = obstacles + np.array([self.margin,self.margin])
6.         obstacles = obstacles[(obstacles[:,0]>=0) & (obstacles[:,1]>=0)]
7.
8.         self.obs = np.concatenate([np.array([[0,i] for i in range(100+self.margin
   )]),
9.                                    np.array([[100+2*self.margin,i] for i in range(
   100+2*self.margin)]),
10.                                   np.array([[i,0] for i in range(100+self.margin)
   ]),
11.                                   np.array([[i,100+2*self.margin] for i in range(
   100+2*self.margin)]),
12.                                   obstacles])
13.
14.         self.ox = [int(item) for item in self.obs[:,0]]
15.         self.oy = [int(item) for item in self.obs[:,1]]
16.         self.grid_size = 1
17.         self.robot_radius = 4
18.         self.a_star = AStarPlanner(self.ox, self.oy, self.grid_size, self.robot_r
   adius)
19.
20.     def plan_path(self,sx, sy, gx, gy):
21.         rx, ry = self.a_star.planning(sx+self.margin, sy+self.margin, gx+self.mar
   gin, gy+self.margin)
22.         rx = np.array(rx)-self.margin+0.5
23.         ry = np.array(ry)-self.margin+0.5
24.         path = np.vstack([rx,ry]).T
25.         return path[::-1]
```

بلوک درون‌یابی نیز در این کلاس پیاده‌سازی شده و توسط تابع interpolate_path() برای هر برنامه‌ریز مسیر پیاده‌سازی شده است. درون‌یابی مسیر به کمک اسپلاین پایه صورت می‌گیرد.

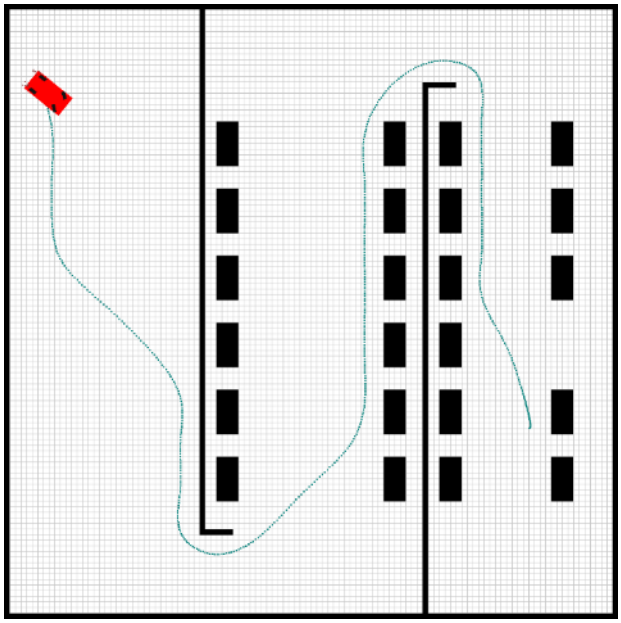
مسیر پیدا شده با استفاده از تابع interpolate_path(self, path) برای افزایش آزادی ابتدا نمونه‌گیری می‌شود. سپس با استفاده از تابع make_interp_spline() از ماژول scipy اسپلاین با درجه ۳ را روی مسیر درون‌یابی می‌کند.

```
1.      def interpolate_b_spline_path(self, x, y, n_path_points, degree=3):
2.          ipl_t = np.linspace(0.0, len(x) - 1, len(x))
3.          spl_i_x = scipy_interpolate.make_interp_spline(ipl_t, x, k=degree)
4.          spl_i_y = scipy_interpolate.make_interp_spline(ipl_t, y, k=degree)
5.          travel = np.linspace(0.0, len(x) - 1, n_path_points)
6.          return spl_i_x(travel), spl_i_y(travel)
7.
8.      def interpolate_path(self, path):
9.          choices = np.arange(0,len(path),int(len(path)/32))
10.         way_point_x = path[choices,0]
11.         way_point_y = path[choices,1]
12.         n_course_point = int(len(choices)*18)
13.         rix, riy = self.interpolate_b_spline_path(way_point_x, way_point_y, n_cou
    rse_point)
14.         new_path = np.vstack([rix,riy]).T
15.         # new_path[new_path<0] = 0
16.         return new_path
```

درنتیجه مسیر خروجی A* با شکستی‌های زیاد که در شکل ۶-۴ مشاهده می‌کنیم، به صورت منحنی های نشان داده شده در شکل۷-۴ تغییر پیدا می‌کند.



شکل۶-۵ مسیر با شکستگی‌های زیاد



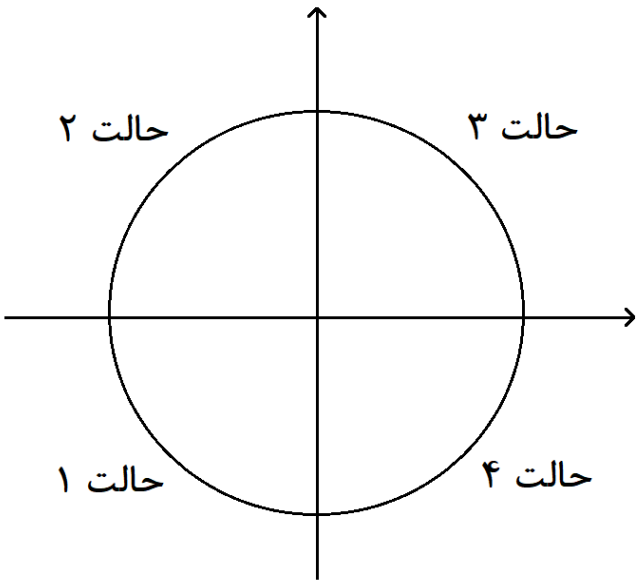
شکل ۷-۵ مسیر بدون شکستگی و به صورت منحنی

۲-۴-۵- برنامه ریزی مسیر بخش دوم

اطلاعات دریافتی از پارکینگ شامل موانع و مختصات نقاط اولیه و نهایی، به بلوک برنامه‌ریزی سناریو پارک وارد می‌شود. وظیفه این بخش، انتخاب حالت پارک مناسب از ۴ حالت ذکر شده و برنامه‌ریزی مسیر پارک براساس دو کمان است. به‌وسیله الگوریتم A*، مسیر مناسب از نقطه اولیه تا نقطه نهایی برای کسب اطلاعات لازم جهت نحوه ورود خودرو به محل پارک پیدا می‌شود. سپس زاویه نهایی ورود خودرو براساس معکوس تانژانت ۱۰ نقطه پایانی مسیر پیدا شده و بر اساس این زاویه یکی از حالات ذکر شده انتخاب می‌گردد.

```
    ############################## path planning ##############################
    park_path_planner = ParkPathPlanning(obs)
    path_planner = PathPlanning(obs)

    print('planning park scenario ...')
    new_end, park_path, ensure_path1, ensure_path2 = park_path_planner.generate_park_scenario(int(start[0]),int(start[1]),int(end[0]),int(end[1]))

    print('routing to destination ...')
    path = path_planner.plan_path(int(start[0]),int(start[1]),int(new_end[0]),int(new_end[1]))
    path = np.vstack([path, ensure_path1])

    print('interpolating ...')
    interpolated_path = path_planner.interpolate_path(path)
    interpolated_park_path = park_path_planner.interpolate_park_path(park_path)
    interpolated_park_path = np.vstack([ensure_path1[::-1], interpolated_park_path, ensure_path2[::-1]])

    env.draw_path(interpolated_path)
    env.draw_path(interpolated_park_path)
    ###########################################################################
```

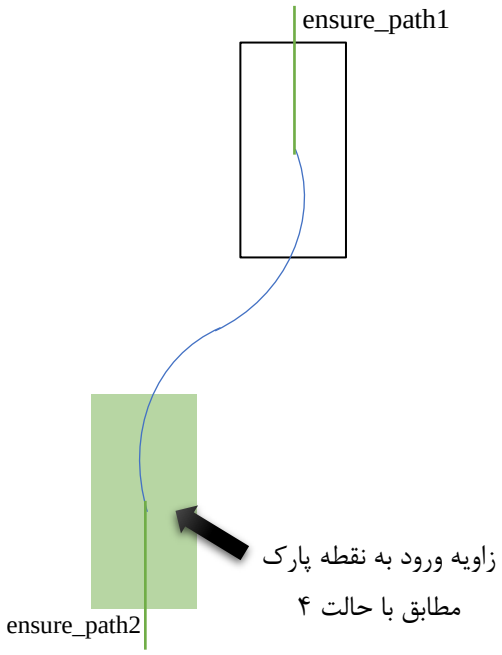
با انتخاب معادلات دایره مناسب، مختصات (x, y) مسیر پارک تولید می‌شود. ۴ تابع برای ۴ حالت ذکر شده نوشته شده که براساس زاویه محاسبه شده، یکی از آن‌ها انتخاب می‌شود.



شکل ۸-۵ چهار حالت ممکن برای ورود به نقطه پارک

- **حالت ۱:** مسیر پارک با استفاده از تابع plan_park_down_right() طراحی می‌شود.
- **حالت ۲:** مسیر پارک با استفاده از تابع plan_park_down_left() طراحی می‌شود.
- **حالت ۳:** مسیر پارک با استفاده از تابع plan_park_up_left () طراحی می‌شود.
- **حالت ۴:** مسیر پارک با استفاده از تابع plan_park_up_right() طراحی می‌شود.

مسیر طراحی شده تابع interpolate_park_path() درون‌یابی شده و خروجی داده می‌شود. دو مسیر کوتاه ensure_path1 و ensure_path2 نیز برای اطمینان از مناسب بودن زاویه خودرو در زمان پارک، به مسیر پارک اضافه می‌شود. برای نمونه در حالت ۴ طراحی مسیر با استفاده از تابع plan_park_up_right() صورت گرفته است.



شکل ۹-۵ طراحی مسیر پارک براساس حالت چهارم

```
        elif math.atan2(1,0) < computed_angle <= math.atan2(0,-1):
            x_ensure2 = gx
            y_ensure2 = gy
            x_ensure1 = x_ensure2 + d + w
            y_ensure1 = y_ensure2 + l + s
            ensure_path1 = np.vstack([np.repeat(x_ensure1,3/0.25), np.arange(y_en
sure1,y_ensure1+3,0.25)]).T
            ensure_path2 = np.vstack([np.repeat(x_ensure2,3/0.25), np.arange(y_en
sure2-3,y_ensure2,0.25)]).T
            park_path = self.plan_park_up_right(x_ensure2, y_ensure2)
```

تابع plan_park_up_right() نیز با دریافت محل پارک به صورت زیر مسیر پارک را بر مبنای معادلات ذکر شده، برنامه‌ریزی می‌کند.

```
1.  def plan_park_up_right(self, x1, y1):
2.              s = 4
3.              l = 8
4.              d = 2
5.              w = 4
6.
7.              x0 = x1 + d + w
8.              y0 = y1 + l + s
9.
10.             curve_x = np.array([])
11.             curve_y = np.array([])
12.             y = np.arange(y1,y0+1)
13.             circle_fun = (6.9**2 - (y-y0)**2)
14.             x = (np.sqrt(circle_fun[circle_fun>=0]) + x0-6.9)
15.             y = y[circle_fun>=0]
16.             choices = x>x0-6.9/2
17.             x=x[choices]
18.             y=y[choices]
19.             curve_x = np.append(curve_x, x[::-1])
20.             curve_y = np.append(curve_y, y[::-1])
21.
22.             y = np.arange(y1,y0+1)
23.             circle_fun = (6.9**2 - (y-y1)**2)
24.             x = (np.sqrt(circle_fun[circle_fun>=0]) + x1+6.9)
25.             y = y[circle_fun>=0]
26.             x = (x - 2*(x-(x1+6.9)))
27.             choices = x<x1+6.9/2
28.             x=x[choices]
29.             y=y[choices]
30.             curve_x = np.append(curve_x, x[::-1])
31.             curve_y = np.append(curve_y, y[::-1])
32.
33.             park_path = np.vstack([curve_x, curve_y]).T
34.             return park_path
```

۵-۵- مدل سینماتیکی خودرو

مدل سینماتیکی خودرو در کلاس Car_Dynamics پیاده شده است.

$$\dot{x} = vcos(\psi) \quad (۵-۱)$$

$$\dot{y} = vsin(\psi) \quad (۵-۲)$$

$$\dot{v} = a \quad (۵-۳)$$

$$\dot{\psi} = \frac{vtan(\delta)}{L} \quad (۵-۴)$$

هر شیء از این کلاس، دارای متغییرهای حالت x، y، v و psi می باشد. که متناسب با ورودی خودرو توسط تابع update_state تغییر می کنند.

```
1. class Car_Dynamics:
2.     def __init__(self, x_0, y_0, v_0, psi_0, length, dt):
3.         self.dt = dt             # sampling time
4.         self.L = length          # vehicle length
5.         self.x = x_0
6.         self.y = y_0
7.         self.v = v_0
8.         self.psi = psi_0
9.         self.state = np.array([[self.x, self.y, self.v, self.psi]]).T
10.
11.    def move(self, accelerate, delta):
12.        x_dot = self.v*np.cos(self.psi)
13.        y_dot = self.v*np.sin(self.psi)
14.        v_dot = accelerate
15.        psi_dot = self.v*np.tan(delta)/self.L
16.        return np.array([[x_dot, y_dot, v_dot, psi_dot]]).T
17.
18.    def update_state(self, state_dot):
19.        # self.u_k = command
20.        # self.z_k = state
21.        self.state = self.state + self.dt*state_dot
22.        self.x = self.state[0,0]
23.        self.y = self.state[1,0]
24.        self.v = self.state[2,0]
25.        self.psi = self.state[3,0]
```

از تابع move برای اعمال ورودی شتاب و زاویه فرمان استفاده می شود. با اعمال ورودی، تغییرات متغیرهای حالت در واحد زمان محاسبه شده و در برداری قرار می گیرد. این بردار تغییرات، با متغییرهای حالت فعلی خودرو جمع شده و وضعیت خودرو در حالت بعد به دست می آید.

۶-۵- کنترل کننده پیش بین

در بخش کنترل کننده پیش بین ۲ رویکرد برای پیاده سازی انتخاب شده است. در رویکرد اول، وضعیت خودرو به کنترل کننده داده می شود. سپس کنترل کننده ۱۰ حالت بعدی خودرو (به اندازه افق پیش بین) را در نظر گرفته و ورودی آن را بهینه می کند تا اختلاف بین مکان خودرو و رفرنس حداقل شود. سپس شتاب و زاویه فرمان بهینه حالت اول را به دست می آورد.

```
1. class MPC_Controller:
2.     def __init__(self):
3.         self.horiz = None
4.         self.R = np.diag([0.01, 0.01])                # input cost matrix
5.         self.Rd = np.diag([0.01, 1.0])                # input difference cost ma
   trix
6.         self.Q = np.diag([1.0, 1.0])                  # state cost matrix
7.         self.Qf = self.Q                              # state final matrix
8.
9.     def mpc_cost(self, u_k, my_car, points):
10.        mpc_car = copy.copy(my_car)
11.        u_k = u_k.reshape(self.horiz, 2).T
12.        z_k = np.zeros((2, self.horiz+1))
13.
14.        desired_state = points.T
15.        cost = 0.0
16.
17.        for i in range(self.horiz):
18.            state_dot = mpc_car.move(u_k[0,i], u_k[1,i])
19.            mpc_car.update_state(state_dot)
20.
21.            z_k[:,i] = [mpc_car.x, mpc_car.y]
22.            cost += np.sum(self.R@(u_k[:,i]**2))
23.            cost += np.sum(self.Q@((desired_state[:,i]-z_k[:,i])**2))
24.            if i < (self.horiz-1):
25.                cost += np.sum(self.Rd@((u_k[:,i+1] - u_k[:,i])**2))
26.        return cost
27.
28.    def optimize(self, my_car, points):
29.        self.horiz = points.shape[0]
30.        bnd = [(-5, 5),(np.deg2rad(-60), np.deg2rad(60))]*self.horiz
31.        result = minimize(self.mpc_cost, args=(my_car, points), x0 = np.zeros((2*
   self.horiz)), method='SLSQP', bounds = bnd)
32.        return result.x[0],  result.x[1]
```

در روش دوم، کنترل کننده ابتدا مدل غیرخطی خودرو را در نقطه کار خطی می کند. به این ترتیب ماتریس های A، B و C به دست می آید سپس مراحل بهینه سازی روش اول بر روی مدل خطی سازی شده انجام می شود.

$$z_{k+1} = z_k + f(z_k, u_k)dt \rightarrow z_{k+1} = Az_k + Bu_k + C \qquad (۵-۵)$$

```python
class Linear_MPC_Controller:
    def __init__(self):
        self.horiz = None
        self.R = np.diag([0.01, 0.01])                # input cost matrix
        self.Rd = np.diag([0.01, 1.0])                # input difference cost matrix
        self.Q = np.diag([1.0, 1.0])                  # state cost matrix
        self.Qf = self.Q                              # state final matrix
        self.dt=0.2
        self.L=4

    def make_model(self, v, psi, delta):
        # matrices
        # 4*4
        A = np.array([[1, 0, self.dt*np.cos(psi)       , -self.dt*v*np.sin(psi)],
                      [0, 1, self.dt*np.sin(psi)       , self.dt*v*np.cos(psi) ],
                      [0, 0, 1                         , 0                     ],
                      [0, 0, self.dt*np.tan(delta)/self.L, 1                     ]])
        # 4*2
        B = np.array([[0    , 0                                  ],
                      [0      , 0                                ],
                      [self.dt, 0                                ],
                      [0      , self.dt*v/(self.L*np.cos(delta)**2)]])

        # 4*1
        C = np.array([[self.dt*v* np.sin(psi)*psi             ],
                      [-self.dt*v*np.cos(psi)*psi             ],
                      [0                                      ],
                      [-self.dt*v*delta/(self.L*np.cos(delta)**2)]])

        return A, B, C

    def mpc_cost(self, u_k, my_car, points):

        u_k = u_k.reshape(self.horiz, 2).T
        z_k = np.zeros((2, self.horiz+1))
        desired_state = points.T
        cost = 0.0
        old_state = np.array([my_car.x, my_car.y, my_car.v, my_car.psi]).reshape(4,1)

        for i in range(self.horiz):
            delta = u_k[1,i]
            A,B,C = self.make_model(my_car.v, my_car.psi, delta)
            new_state = A@old_state + B@u_k + C

            z_k[:,i] = [new_state[0,0], new_state[1,0]]
            cost += np.sum(self.R@(u_k[:,i]**2))
            cost += np.sum(self.Q@((desired_state[:,i]-z_k[:,i])**2))
            if i < (self.horiz-1):
                cost += np.sum(self.Rd@((u_k[:,i+1] - u_k[:,i])**2))

            old_state = new_state
        return cost

    def optimize(self, my_car, points):
        self.horiz = points.shape[0]
        bnd = [(-5, 5),(np.deg2rad(-60), np.deg2rad(60))]*self.horiz
        result = minimize(self.mpc_cost, args=(my_car, points), x0 = np.zeros((2*self.hori
z)), method='SLSQP', bounds = bnd)
        return result.x[0],  result.x[1]
```

۷-۵- ساختار اصلی کد

کد main_autopark.py دربرگیرنده ساختار اصلی کد و محل استفاده و پیوند اجزای توسعه‌داده شده می‌باشد. با توجه به اینکه تمامی اجزا درقالب اسکریپت جدا توسعه داده شده اند، استفاده از آن‌ها به راحتی با اضافه کردن کلاس‌ها صورت می‌گیرد.

```
1. import cv2
2. import numpy as np
3. from time import sleep
4. import argparse
5.
6. from environment import Environment, Parking1
7. from pathplanning import PathPlanning, ParkPathPlanning
8. from control import Car_Dynamics, MPC_Controller, Linear_MPC_Controller
9. from utils import angle_of_line, make_square, DataLogger
```

ماژول argparser وظیفه گرفتن متغیر در محیط خط فرمان را برعهده دارد. بعد از دریافت متغیرها، اطلاعات پردازش شده و سپس شیء ثبت‌کننده اطلاعات ساخته می‌شود.

```
1.      parser = argparse.ArgumentParser()
2.      parser.add_argument('--x_start', type=int, default=0, help='X of start')
3.      parser.add_argument('--y_start', type=int, default=90, help='Y of start')
4.      parser.add_argument('--phi_start', type=int, default=0, help='phi of start')
5.      parser.add_argument('--x_end', type=int, default=90, help='X of end')
6.      parser.add_argument('--y_end', type=int, default=80, help='Y of end')
7.      parser.add_argument('--
    parking', type=int, default=1, help='park position in parking1 out of 24')
8.
9.      args = parser.parse_args()
10.     logger = DataLogger()
```

پس دریافت متغیرها، پارکینگ ۱ متناسب با محل انتخابی کاربر ساخته شده و موانع را تولید می‌کند.

```
1.      parking1 = Parking1(args.parking)
2.      end, obs = parking1.generate_obstacles()
```

سپس به ترتیب اشیاء محیط شبیه‌سازی، خودرو و کنترلر با افق پیش‌بین ۱۰ ساخته شده و محیط برای اولین بار برای کاربر رندر می‌شود.

```
    ########################## initialization ##############################
    env = Environment(obs)
    my_car = Car_Dynamics(start[0], start[1], 0, np.deg2rad(args.phi_start), length=4, dt=0.2)
    MPC_HORIZON = 10
    controller = MPC_Controller()
    # controller = Linear_MPC_Controller()

    res = env.render(my_car.x, my_car.y, my_car.psi, 0)
    cv2.imshow('environment', res)
    key = cv2.waitKey(1)
    ########################################################################
```

در بخش مسیریابی، دو شیء مسیریابی اصلی و مسیریابی پارک با توجه به موانع تولید شده ساخته می‌شوند و سپس سناریو پارک برنامه‌ریزی شده و مسیریابی با توجه به موقعیت شروع پارک توسط الگوریتم A* انجام می‌گردد. بخش درون‌یابی ابتدا از مسیرها نمونه گیری کرده و سپس با استفاده از اسپلاین، مسیرها را درون‌یابی می‌کند. دو مسیر ensure_path1 و ensure_path2 در ادامه تولید شده و به مسیرها متاسب با شرایط گفته شده اضافه می‌گردد.

```
    ############################## path planning ###################################
    park_path_planner = ParkPathPlanning(obs)
    path_planner = PathPlanning(obs)

    print('planning park scenario ...')
    new_end, park_path, ensure_path1, ensure_path2 = park_path_planner.generate_park_scenario(int(start[0]),int(start[1]),int(end[0]),int(end[1]))

    print('routing to destination ...')
    path = path_planner.plan_path(int(start[0]),int(start[1]),int(new_end[0]),int(new_end[1]))
    path = np.vstack([path, ensure_path1])

    print('interpolating ...')
    interpolated_path = path_planner.interpolate_path(path)
    interpolated_park_path = park_path_planner.interpolate_park_path(park_path)
    interpolated_park_path = np.vstack([ensure_path1[::-1], interpolated_park_path, ensure_path2[::-1]])

    env.draw_path(interpolated_path)
    env.draw_path(interpolated_park_path)
    ################################################################################
```

دربخش کنترلی، کنترلر پیش‌بین، دستور کنترلی را مطابق با نقاط مسیر بهنیه کرده و به خودرو اعمال می‌کند. ورودی ها شامل شتاب و زاویه ی فرمان می باشند که شتاب بین ۵- تا ۵ و زاویه ی فرمان بین ۶۰- تا ۶۰ درجه محدود شده اند. متغیرهای حالت خودرو، متناسب با ورودی تغییر می‌کند و خودرو جابجا می‌شود. رندرگرافیکی برای کاربر پردازش شده و اطلاعات هر لحظه خودرو توسط ثبت‌کننده اطلاعات ذخیره می‌گردد.

```
1.      ################################### control ###################################
2.      print('driving to destination ...')
3.      for i,point in enumerate(interpolated_path):
4.
5.              acc, delta = controller.optimize(my_car, interpolated_path[i:i+MPC_HORIZON])
6.              my_car.update_state(my_car.move(acc,  delta))
7.              res = env.render(my_car.x, my_car.y, my_car.psi, delta)
8.              logger.log(point, my_car, acc, delta)
9.              cv2.imshow('environment', res)
10.             key = cv2.waitKey(1)
11.             if key == ord('s'):
12.                 cv2.imwrite('res.png', res*255)
```

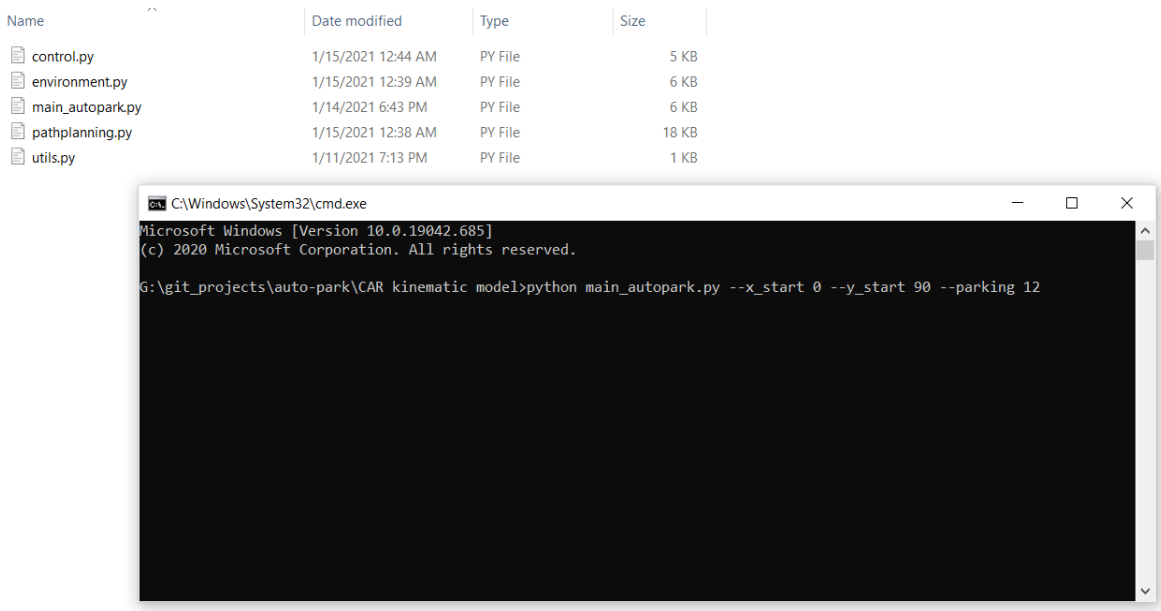
فرآیند کنترلی تا رسیدن خودرو به مقصد ادامه می‌یابد و بعد از اتمام فرآیند پارک، ثبت‌کننده اطلاعات، نمودارهای تولید شده را در پوشه log results ذخیره می‌کند.

```
1. logger.save_data()
```

۵-۸- ارائه یک مثال از شبیه سازی

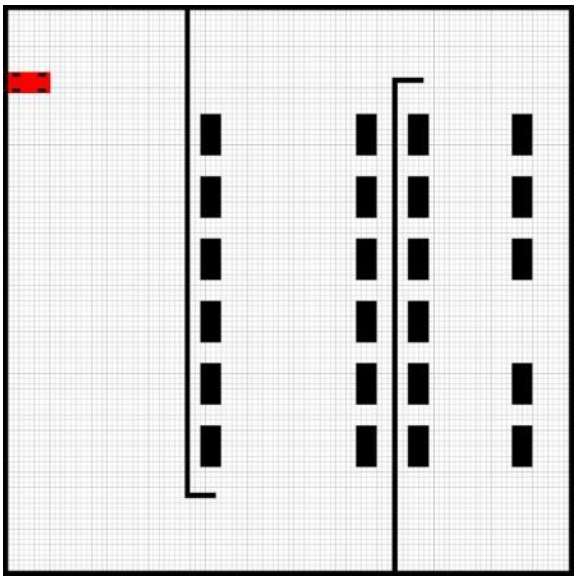
هدف از این بخش بررسی یک نمونه پارک خودرو، در محل پارک شماره ۱۲ می‌باشد. برای این‌کار، با توجه به دستور توضیح داده در بخش قبل، مختصات اولیه خودرو را $(0, 90)$ انتخاب کرده و نقطه نهایی را محل پارک ۱۲ وارد می‌کنیم.

```
python main_autopark.py --x_start 0 --y_start 90 --parking 12
```



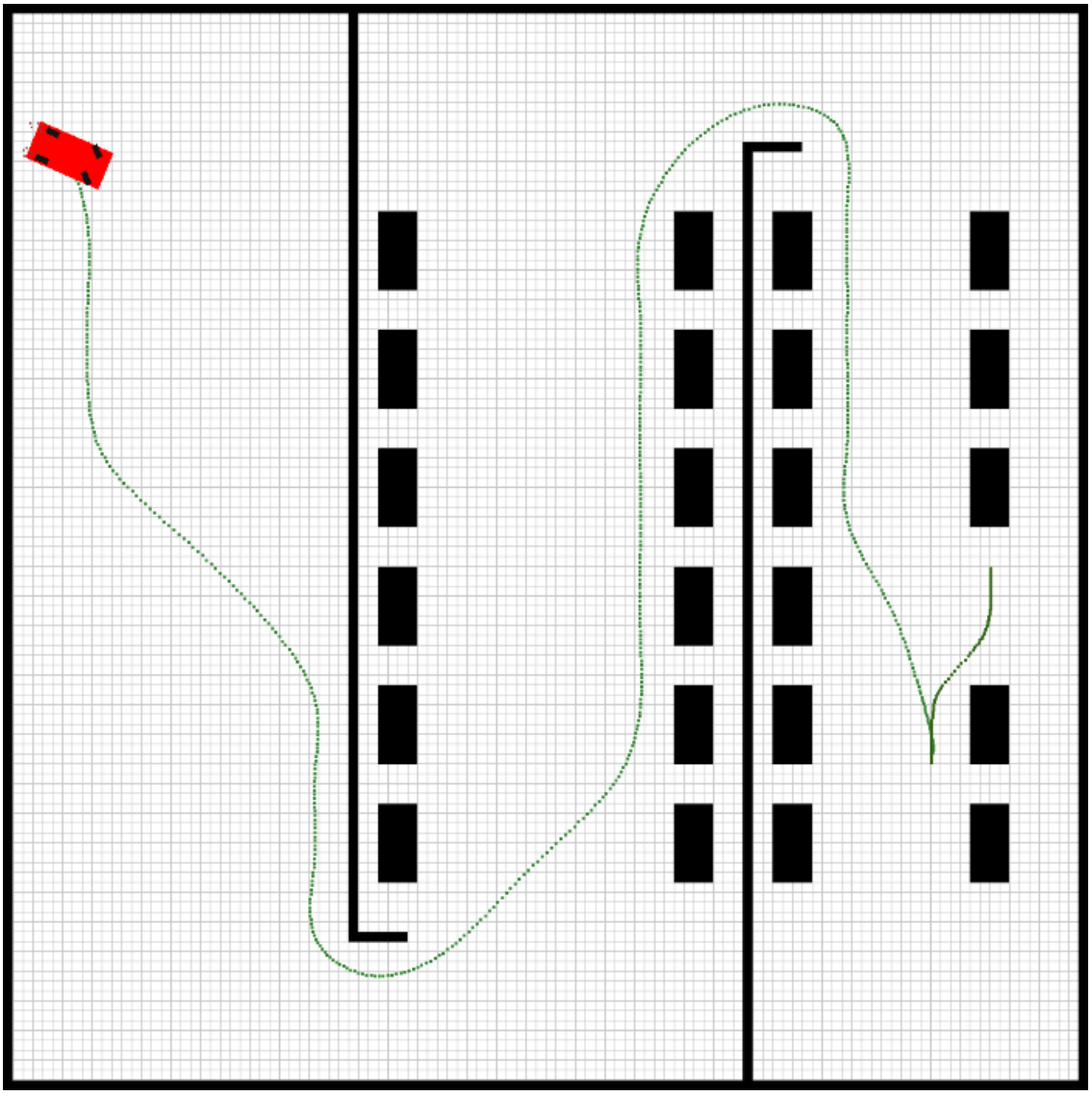
شکل ۱۰-۵ ورود اطلاعات در شبیه ساز پارک خودکار

موانع و دیوارها توسط پارکینگ۱ تولید شده و محیط شبیه سازی، رندر اولیه را انجام می‌دهد.



شکل ۱۱-۵ رندر اولیه شبیه ساز مطابق با اطلاعات وارد شده

درادامه برنامه‌ریز مسیر پارک سناریو محل پارک را تولید کرده و مسیر پارک را برنامه‌ریزی می‌کند. با برنامه‌ریزی شدن مسیر پارک، نوبت به برنامه‌ریزی مسیر می‌رسد. برنامه‌ریز مسیر نیز به کمک الگوریتم A* مسیری از ابتدا تا محل مناسب تولید می‌کند. هر دو مسیر برنامه‌ریزی شده، به تابع درون‌یابی مربوط به خود داده شده تا مسیر نهایی بدست آید. در انتها محیط شبیه‌سازی، مسیرهای برنامه ریزی شده را برای دنبال کردن، رسم می‌کند.



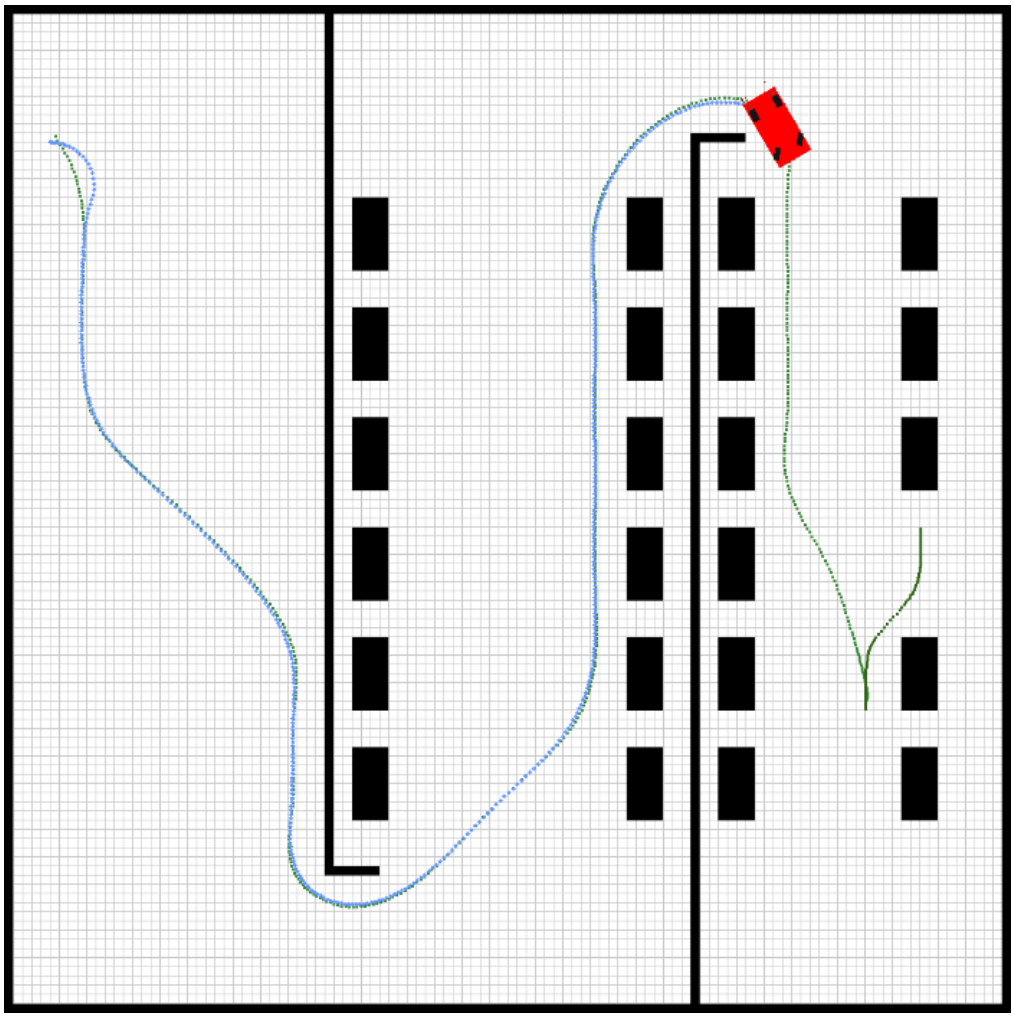
شکل ۱۲-۵ برنامه ریزی مسیر

بعد از برنامه ریزی مسیر نوبت به رهگیری آن می‌رسد. در این قسمت کنترل‌کننده پیش‌بین براساس مختصات مسیر تولید شده، به هدایت خودرو می‌پردازد. کنترلر با دریافت مختصات، زاویه و سرعت مطلوب تابع هزینه تعریف

شده را در افق پیش‌بین ۵ کمینه کرده و شتاب و زاویه چرخ مناسب را به خودرو اعمال می‌کند. متغیرهای حالت خودرو نیز براساس ورودی به‌روز می‌شود.

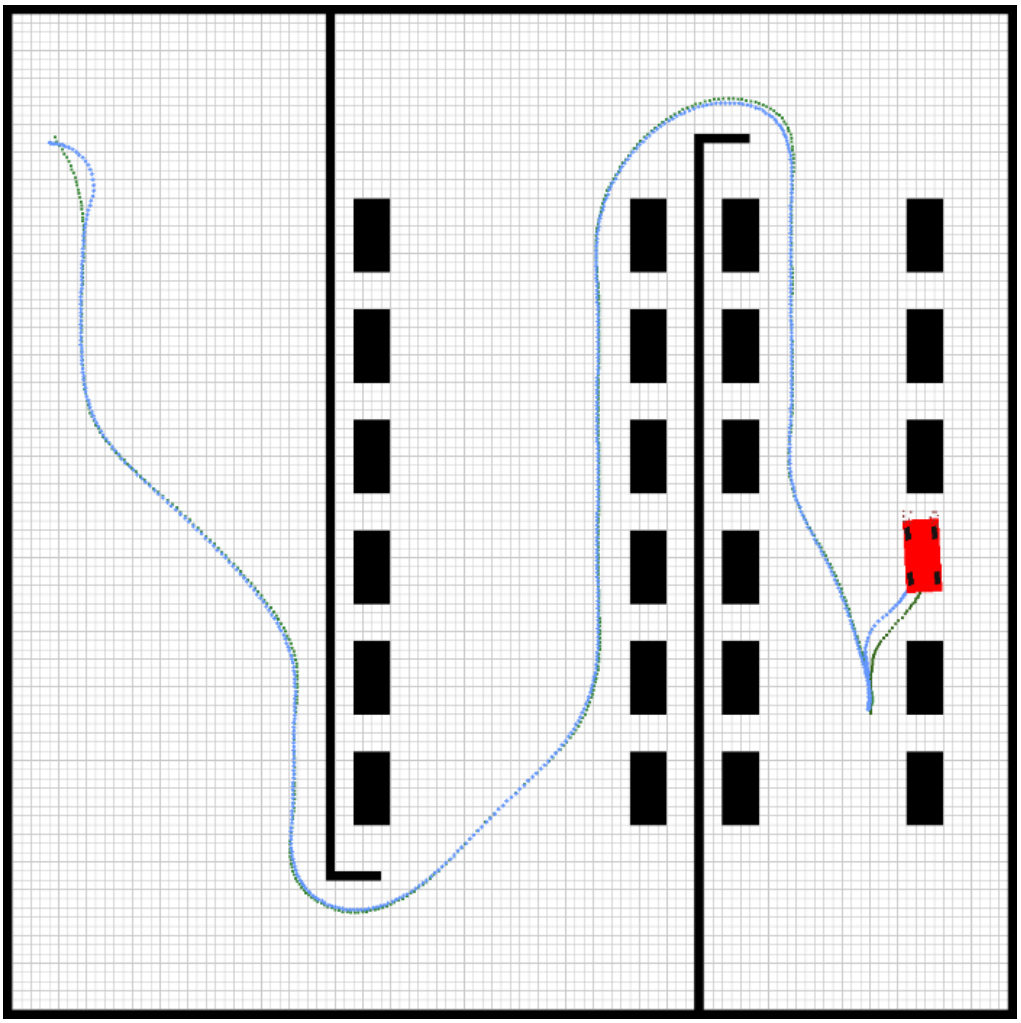
```
planning park scenario ...
Find goal
routing to destination ...
Find goal
interpolating ...
driving to destination ...
```

در هر لحظه موقعیت خودرو نیز برروی محیط برای مقایسه بهتر ثبت می‌گردد.



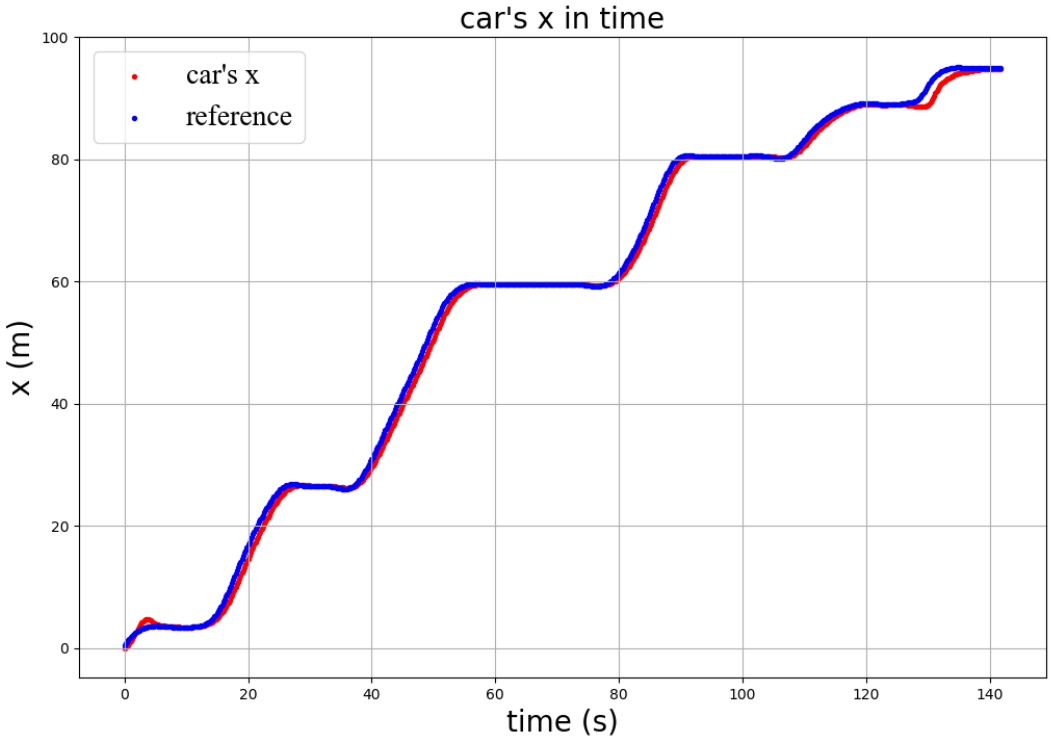
شکل ۱۳-۵ کنترل و هدایت خودرو در مسیر برنامه ریزی شده

در نهایت با اتمام فرآیند رهگیری خودرو به صورت دوبل در محل تعیین شده پارک می‌کند.

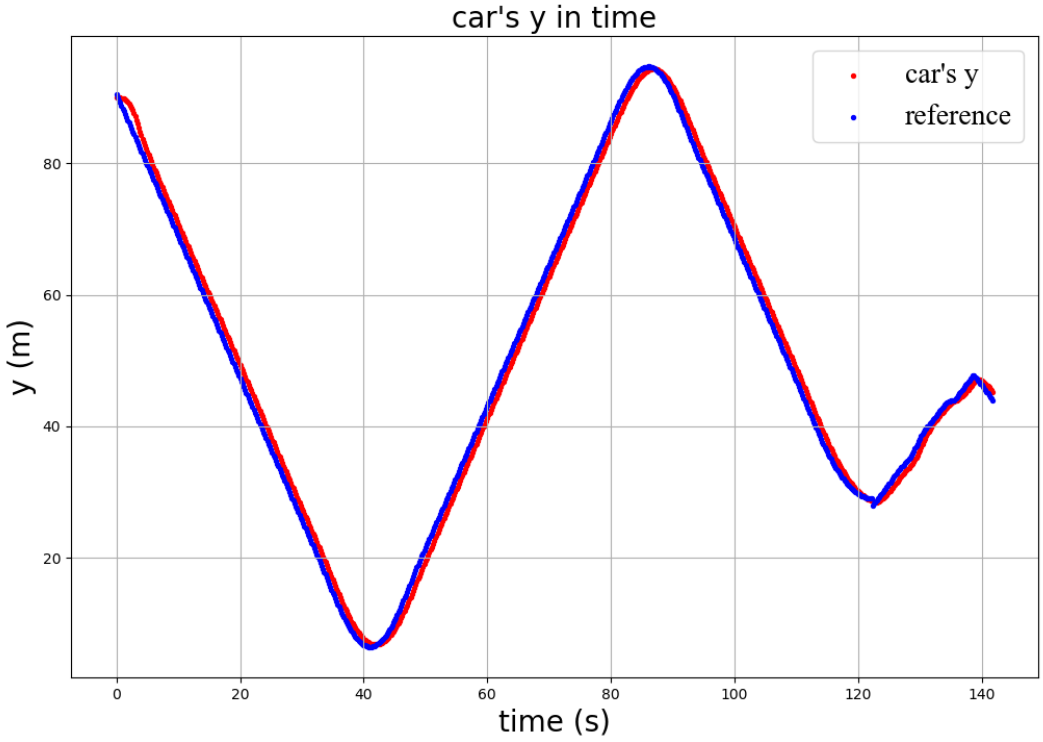


شکل ۱۴-۵ پارک موازی در محل تعیین شده

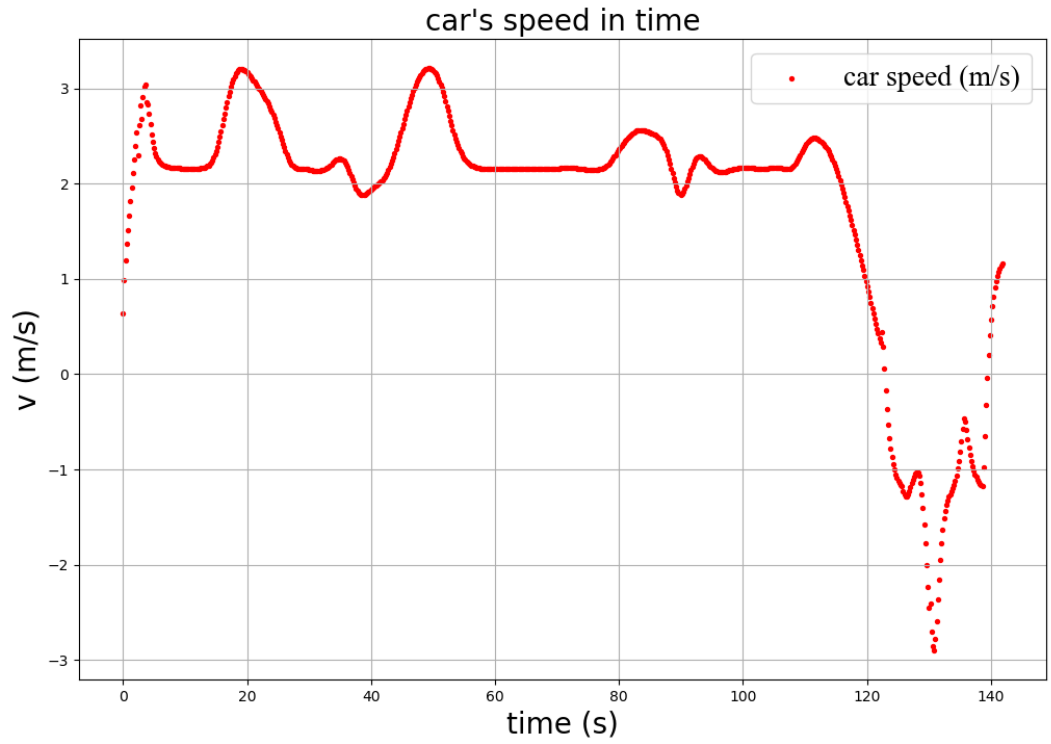
در تمام مدت حرکت خودرو، شیء logger ، اطلاعات مربوط به خودرو را ثبت نموده و در انتها به صورت نمودار در پوشه log results ذخیره می‌کند. نمودارهای متغییرهای حالت خودرو در ادامه آورده شده است.



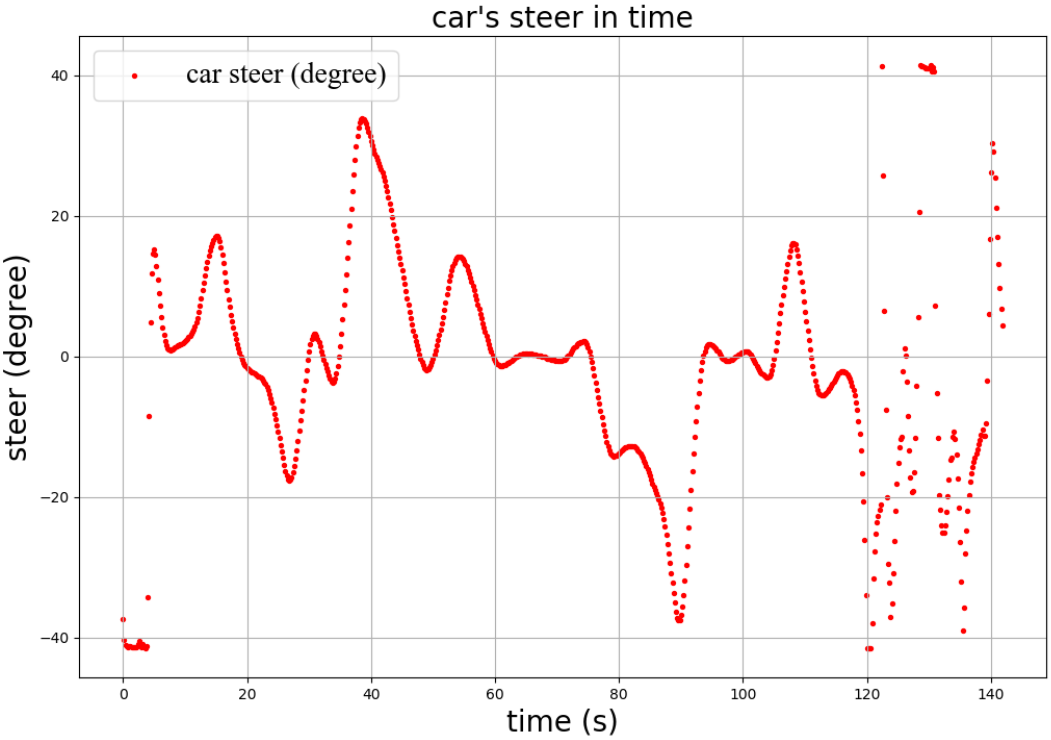
شکل ۱۵-۵ نمودار موقعیت افقی برحسب زمان



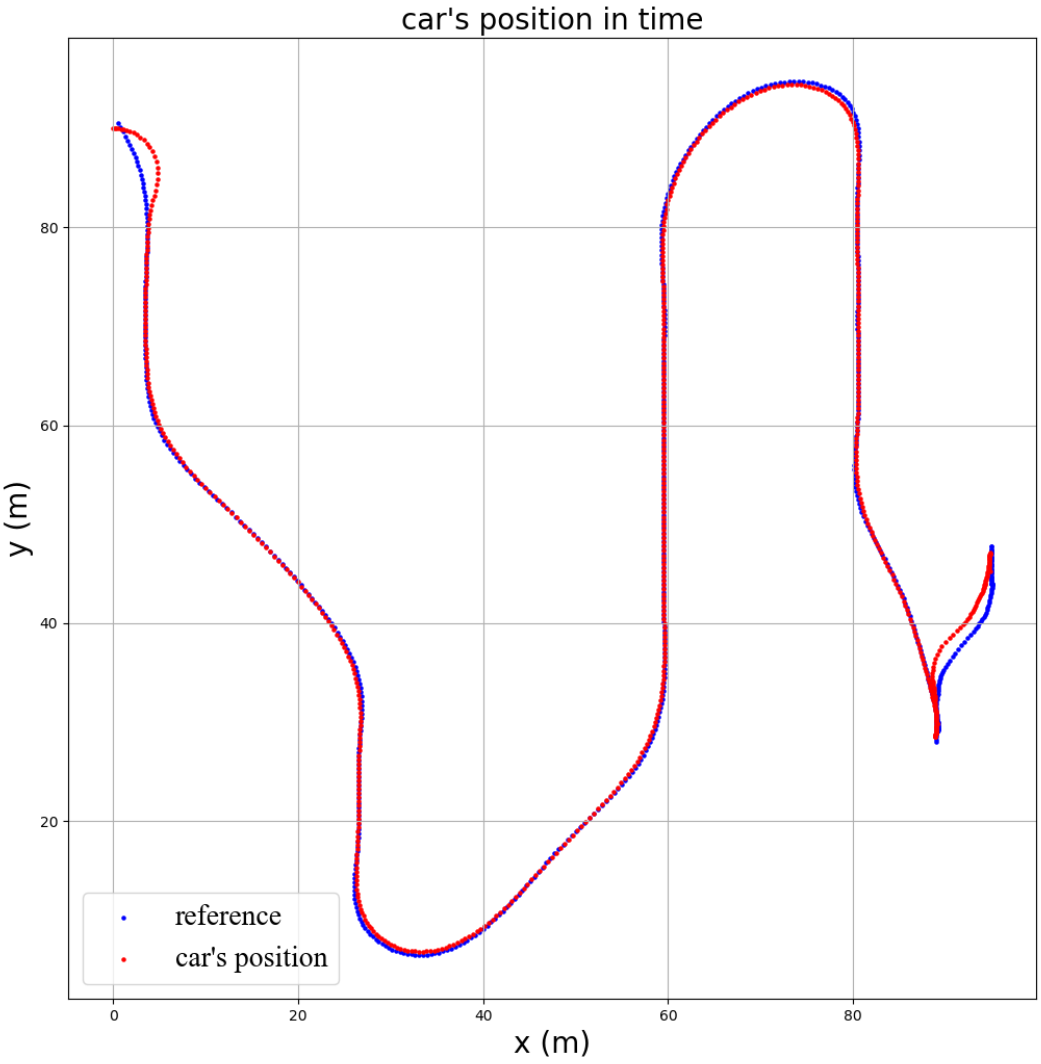
شکل ۱۶-۵ نمودار موقعیت عمودی برحسب زمان



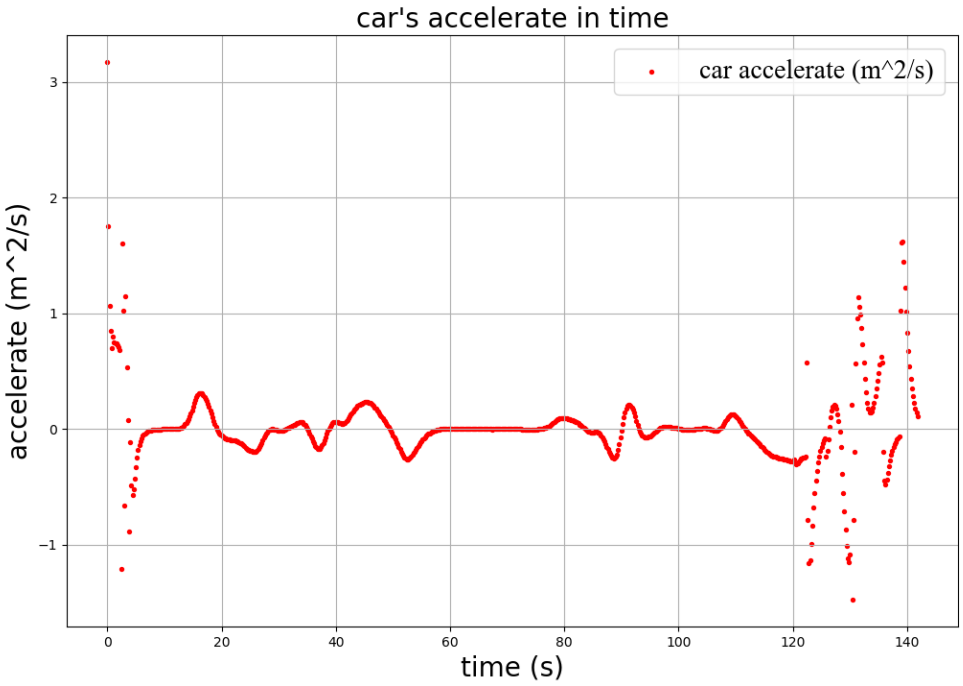
شکل ۱۷-۵ نمودار سرعت خودرو برحسب زمان



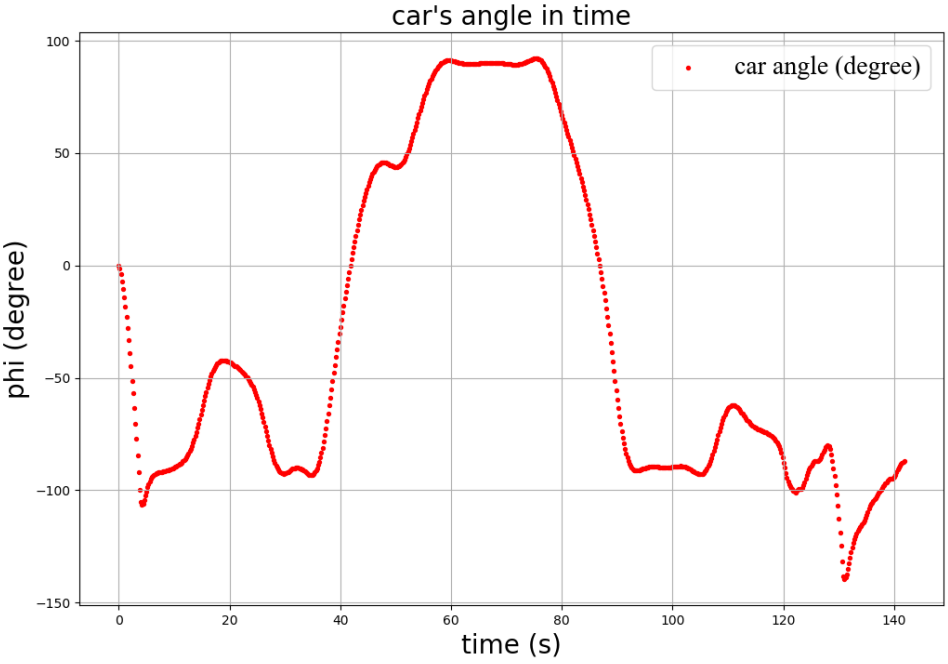
شکل ۱۸-۵ نمودار زاویه فرمان خودرو برحسب زمان



شکل ۱۹-۵ نمودار موقعیت عمودی خودرو برحسب موقعیت افقی آن



شکل ۲۰-۵ نمودار شتاب خودرو برحسب زمان



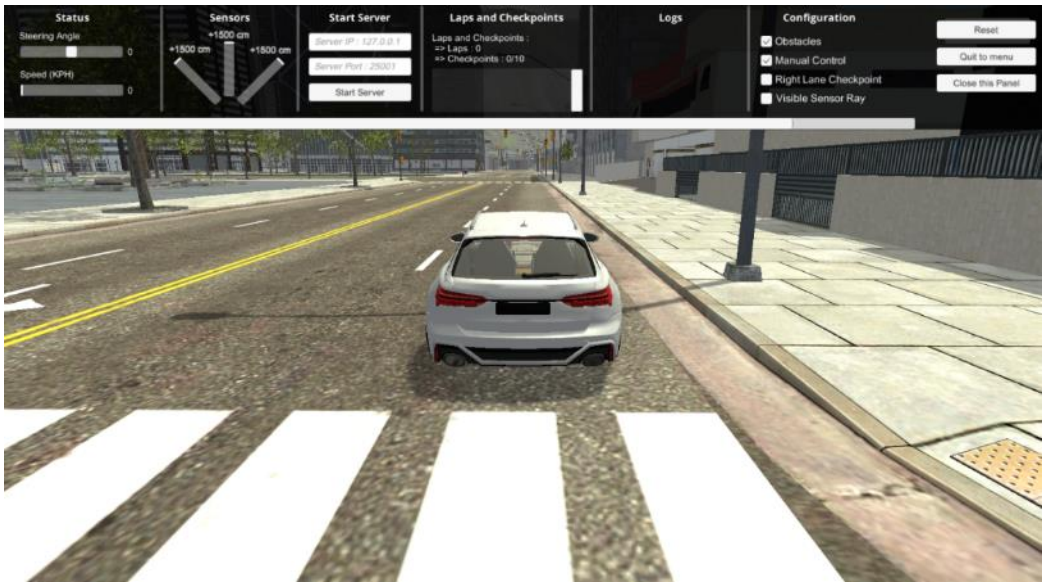
شکل ۲۱-۵ نمودار زاویه خودرو (psi) برحسب زمان

۶- شبیه سازی شهری و بین شهری

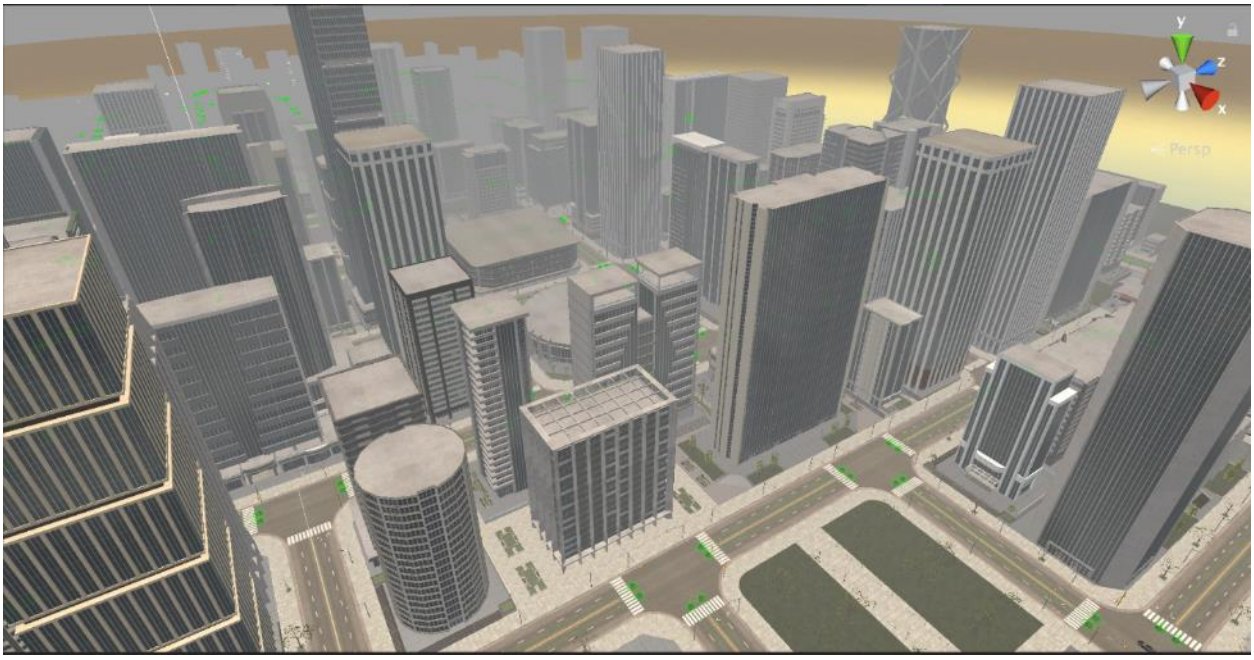
۱-۶- محیط شبیه سازی

در تست نرم افزاری این پروژه، محیط تستی بر مبنای شبیه ساز یونیتی [۳۰۷] توسعه داده شده است که در ادامه توضیحاتی در مورد این شبیه ساز ارائه می گردد. لازم به ذکر است که محیط تست مذکور، نخستین بار توسط شرکت ایویس انجین [۳۰۸] معرفی گردیده است و لازم می دانیم که از حامی پژوهشی این پروژه، دانشگاه امیرکبیر و همچنین جناب دکتر سروش صادقی نژاد و آقایان محمد حسین طیب زاده، امیرمحمد و امیرمهدی ظریف جهت توسعه ی اولیه ی این شبیه ساز، کمال قدردانی را به عمل آوریم.

ایویس انجین (به انگلیسی : AVIS Engine)، مختصر شده ی عبارت Autonomous Vehicles Intelligent simulation می باشد که در واقع یک محیط شبیه سازی برای پیاده سازی نرم افزارهای توسعه داده شده در حوزه ی خودروهای خودران می باشد. در واقع بدنه ی اولیه ی محیط شبیه سازی مورد استفاده در این پروژه نیز روی این شبیه ساز بنا شده، اما توسعه ی جزئیات مورد نیاز محیط شبیه سازی به فراخور عملگر های مورد تست پروژه مانند پیاده سازی دینامیک مورد نظر، زاویه ی قرارگیری سنسور ها، محدوده ی سرعتی خودرو، زاویه ی چرخش چرخ ها، کلیه ی فرآیند کنترل خودرو شامل سیستم جلوگیری از برخورد و ترمز اضطراری، پیاده سازی انکودر دیجیتال و همچنین کلیه ی فرآیند بینایی ماشین و پردازش تصویر پیاده سازی شده، توسط خود ما انجام شده است.



تصویر ۱-۶ تصویری از طراحی گرافیکی اولیه از محیط شبیه سازی



شکل ۲-۶ تصویری از نقشه ی استراتژیک اولیه از محیط شبیه سازی

شکل ۳-۶ تصویری از طراحی گرافیکی پیاده سازی سایه و آفتاب از محیط شبیه سازی

شکل ۴-۶ تصویری از یکی از نقشه های شهری مورد استفاده برای شبیه سازی

شکل ۵-۶ تصویری از یکی از نقشه های شهری مورد استفاده برای شبیه سازی

شکل ۶-۶ تصویری از یکی از نقشه های بین شهری مورد استفاده برای شبیه سازی

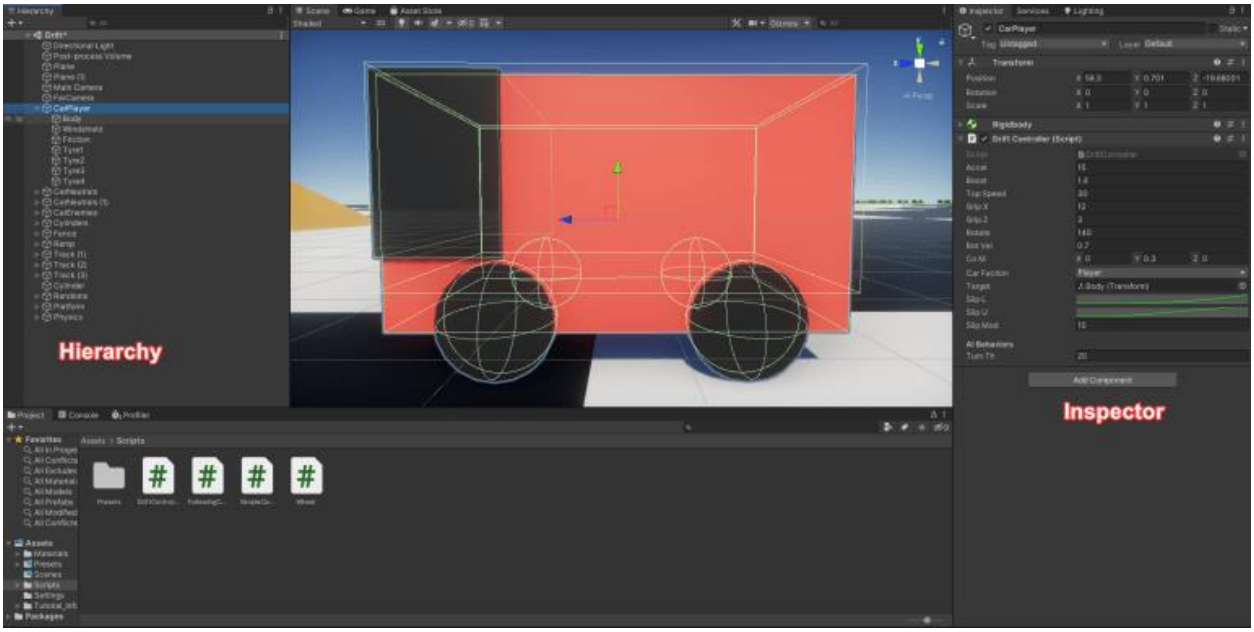
۶-۲- دینامیک شبیه سازی خودرو و محیط

به طور کلی یک دینامیک پیش فرض توسط خود شرکت یونیتی برای خودروی مورد استفاده در این شبیه سازی و محیط تست آن ارائه شده است. توضیحات جامع و کامل در مورد دینامیک خودرو شامل معادلات، نیروهای وارد بر تایر و خودرو، متغیر های حالت و دیگر مباحث در بخش های قبلی به صورت کامل مرقوم گردیده است و شبیه ساز یونیتی نیز به صورت پیش فرض از همان معادلات برای طراحی دینامیک خودرو استفاده کرده است، اما در این قسمت سعی شده تا برخی از موارد قابل تغییر در دینامیک پیاده سازی شده توسط یونیتی، به فراخور نیاز های پروژه تغییر یا اصلاح گردد.

۶-۲-۱- فیزیک خودرو

در این شبیه ساز کلیه ی مشخصات یک خودرو در دنیای واقعی، شامل یک بدنه، چهار پنجره ی جانبی و دو پنجره در جلو و عقب خودرو، چهار چرخ و طراحی های معمول محیط بیرونی یک خودرو مانند چراغ های ترمز پیاده سازی شده که مجموعا جرمی معادل یک و دو دهم تن برای آن تعریف شده است.

$$M = 1200\,kg \qquad (۶-۱)$$



شکل ۷-۶ مراحل تنظیم مشخصات فیزیکی و جرم خودرو

شکل ۸-۶ طراحی نهایی خودروی مورد استفاده شده در شبیه سازی

۲-۲-۶- تنظیمات برخورد

به طور کلی، در این قسمت مشخص می گردد که چه رویدادی را می توان به عنوان یک برخورد در نظر گرفت و چه رویداد هایی ناشی از ایجاد یک برخورد نیست و ناشی از فیزیک خود خودرو می باشد، چرا که در صورت بروز برخورد، باید عواقب آن مانند کاهش ناگهانی و سریع سرعت و شتاب خوردو در محیط شبیه ساز به دینامیک خودرو اعمال گردد. به طور کلی در دنیای واقعی، نیروهای فراوانی به یک خودرو وارد می گردد. یک مثال از این نیروها، لرزش ناشی از کار کردن موتور در خودرو است که معمولا زیر پای سرنشینان جلویی خودرو احساس می گردد. مثال دیگر این نیرو ها، نیروی ناشی از مقاومت هوا در سرعت های بالاست که میزان آن از حالتی که خودرو با یک سرعت پایین در محیط شهری در حال تردد است بیشتر می باشد و به همین دلیل، باید آستانه ی نیرویی در نظر گرفته می شد تا نیروهای اعمالی بالاتر از آستانه ی مشخص شده به خودرو به عنوان برخورد در نظر گرفته شود و عواقب ناشی از برخورد روی خودرو شبیه سازی گردد. همچنین، این آستانه نباید شامل نیروهای معمول وارد به خودرو مانند لرزش موتور و مقاومت هوا گردد. همچنین مدت زمانی که در آن بازه، خودرو دچار این نیرو شده نیز مورد توجه است و می توان مثال زد که نیروی وارد به یک خودرو در یک تصادف با سرعت ۲۰ کیلومتر بر ساعت، به مراتب کمتر از نیروی ناشی از مقاومت هوا و اصطکاک در سرعت ۱۲۰ کیلومتر بر ساعت است، اما نیروی ناشی از تصادف در مدت زمانی کمتر از نیم ثانیه رقم میخورد و تغییرات نیرو در بازه ی زمانی بسیار بالاست، اما نیروی ناشی از مقاومت هوا، تغییرات اندکی در بازه ی زمانی مشابه دارد، در نتیجه معیار تغییرات نیرو بر حسب زمان به عنوان معیار نهایی تشخیص برخورد در تنظیمات شبیه ساز استفاده شده است. آستانه ی مورد استفاده در این قسمت از تنظیمات دینامیک خودروی شبیه سازی شده حدود ده هزار نیوتن بر ثانیه می باشد. برای تشریح عملیات انجام شده در این قسمت، به ذکر یک مثال عملی می پردازیم:

فرض کنیم خودروی مورد استفاده در شبیه ساز با جرم ۱۲۰۰ kg به یکی از موانع برخورد کرده است. اگر سرعت برخورد و سرعت برگشت خودرو به ترتیب ۵۴ km/h و ۹ km/h باشد و تصادف ۰.۱۵ sطول بکشد، محاسبات مورد استفاده برای قسمت تنظیمات برخورد به شکل زیر می باشد:

$$v_1 = 54\ \frac{km}{h} = 15\ \frac{m}{s} \quad (۶-۲)$$

$$v_2 = -9\ \frac{km}{h} = -2.5\ \frac{m}{s} \quad (۶-۳)$$

$$P_1 = m \times v_1 = 1200 \times 15 = 18 \times 10^3\ \frac{kg.m}{s} \quad (۶-۴)$$

$$P_2 = m \times v_2 = 1200 \times -2.5 = -3 \times 10^3\ \frac{kg.m}{s} \quad (۶-۵)$$

$$\Delta P = -3 \times 10^3 \frac{kg.m}{s} - 18 \times 10^3 \frac{kg.m}{s} = -21 \times 10^3 \frac{kg.m}{s} \quad (۶-۶)$$

$$F = \frac{\Delta P}{\Delta t} = \frac{-21 \times 10^3 \frac{kg.m}{s}}{0.15\ s} = -14 \times 10^4\ N \quad (۶-۷)$$

$$\text{معیار تشخیص برخورد} = \frac{F}{\Delta t} = \frac{-14 \times 10^4\ N}{0.15\ s} \approx 10^6 \frac{N}{s} > 10^4 \frac{N}{s} \quad (۶-۸)$$

می بینیم که متریک مورد استفاده برای تشخیص برخورد در این تصادف بیشتر از آستانه ی تعریف شده در تنظیمات دینامیک می باشد، بنابراین، اعمال این نیرو در این بازه ی زمانی به خودرو، نیروی ناشی از برخورد در نظر گرفته می شود و کلیه ی عواقب برخورد مانند کاهش ناگهانی سرعت و شتاب روی دینامیک خوردو شبیه سازی می گردد.

این مورد در شبیه ساز به این صورت قابل مشاهده است که در صورت اعمال ترمز روی خودرو، حدود ۳ ثانیه طول می کشد تا خودرو از سرعت ۵۴ کیلومتر بر ساعت به توقف کامل برسد. نیروی ترمز اعمال شده به خودرو در شبیه ساز به صورت زیر قابل محاسبه است:

$$P_1 = m \times v_1 = 1200 \times 15 = 18 \times 10^3 \frac{kg.m}{s} \quad (۶-۹)$$

$$P_2 = m \times v_2 = 1200 \times 0 = 0 \frac{kg.m}{s} \quad (۶-۱۰)$$

$$\Delta P = 0 \frac{kg.m}{s} - 18 \times 10^3 \frac{kg.m}{s} = -18 \times 10^3 \frac{kg.m}{s} \quad (۶-۱۱)$$

$$F = \frac{\Delta P}{\Delta t} = \frac{-18 \times 10^3 \frac{kg.m}{s}}{3\ s} = -6 \times 10^3\ N \quad (۶-۱۲)$$

$$\text{معیار تشخیص برخورد} = \frac{F}{\Delta t} = \frac{-6 \times 10^3\ N}{3\ s} \approx 2 \times 10^3 \frac{N}{s} < 10^4 \frac{N}{s} \quad (۶-۱۳)$$

می بینیم که نیروی ترمز اعمال شده به خودرو کمتر از آستانه ی تشخیص برخورد می باشد، بنابراین شبیه سازی های مرتبط با شرایط پسا برخورد روی خودرو اعمال نمی گردد و مدت زمان ۳ ثانیه تا توقف کامل خودرو لازم می باشد و به محض فشردن دکمه ی ترمز، دقیقا ۳ ثانیه طول می کشد تا سرعت مشاهده شده در پنل بالایی شبیه ساز به ۰ برسد.

اما در صورت برخورد با مانع، این زمان به ۰.۱۵ ثانیه کاهش می یابد که ناشی از تشخیص برخورد و شبیه سازی نیروهای ناشی از برخورد به دینامیک خودرو است و در محیط شبیه ساز نیز مشاهده می گردد که در صورت برخورد مانع با سرعت ۵۴ کیلومتر بر ساعت با یک مانع، دقیقا مطابق با محاسباتی که بالاتر ارائه شد، سرعت مشاهده شده در قسمت بالایی شبیه ساز در مدت زمان ۰.۱۵ ثانیه به ۰ می رسد و نمودار کاهش سرعت و به عبارتی شیب منفی، شیب قابل توجهی پیدا می کند.

به این صورت دینامیک ناشی از برخورد با مانع در محیط شبیه سازی تنظیم و به خودرو اعمال می گردد.

شکل ۹-۶ تصویری از برخورد خودرو با مانع در شبیه ساز و صفر شدن آنی سرعت

پس در این قسمت، تنظیمات برخورد مشخص شد و در صورتی که تغییرات نیروهای وارد بر خودرو در بازه ی زمان مشخص، از معیار از پیش تعیین شده بیشتر باشد، برخورد تشخیص داده می شود و به نسبت نیروی اعمال شده به خودرو، شتاب منفی مناسب روی معادلات دینامیکی حرکت خودرو شبیه سازی می گردد.

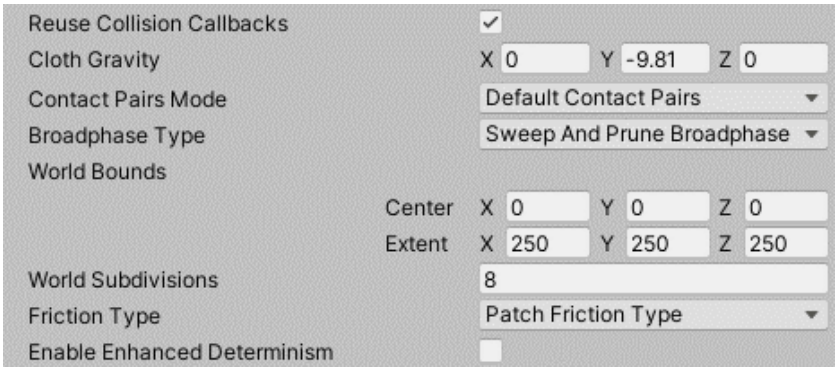
۳-۲-۶- تنظیمات اصطکاک و شتاب گرانشی

در این قسمت از طراحی دینامیک خودرو و محیط، تنظیمات مرتبط با اصطکاک و شتاب گرانشی پیاده سازی می گردد.

شتاب گرانشی شتابی است که توسط کره ی زمین به خودرو وارد می گردد. در نقاط مختلف زمین، خودرو ها با شتابی بین ۹٫۷۸ تا ۹٫۸۲ $\frac{m}{s^2}$ به سمت زمین جذب می‌شوند که به عرض جغرافیایی آن نقطه بستگی دارد.

$$g = G\frac{M}{(R + h)^2} \qquad (۶-۱۳)$$

جاذبه ی پیاده سازی شده در این محیط برابر ۹٫۸۱ $\frac{m}{s^2}$ که معادل شتاب گرانشی شهر تهران با توجه به عرض جغرافیایی آن و ارتفاع شهر از سطح دریاست، قرار داده شده است.

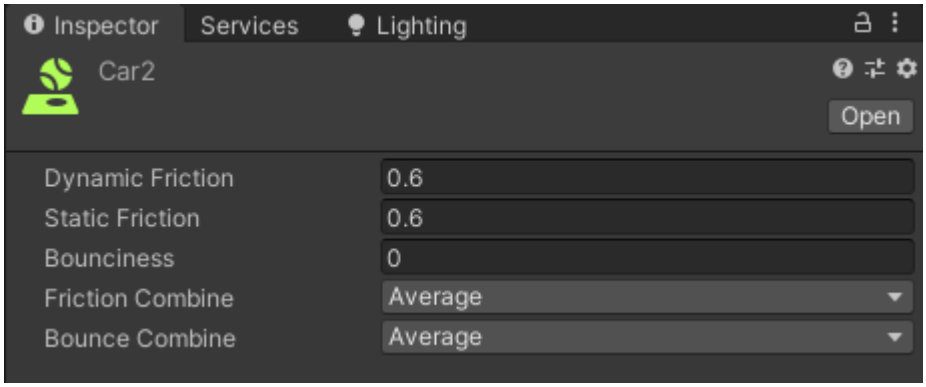


شکل ۶-۱۰ تنظیمات شتاب گرانشی در شبیه‌ساز

در مورد اصطکاک محیط، ضریبی بین ۰ و ۱ تعریف می گردد که مقدار ۰ نشان دهنده ی محیطی مانند یخ است که خودرو کاملا روی آن سر می خورد و مقدار ۱، حداکثر اصطکاک ممکن است. در این شبیه ساز، مقدار ۰.۶ برای هر دو مقدار اصطکاک استاتیک و دینامیک قرار داده شده است.

$$F_s = \mu_s \times N \quad , \quad \mu_s = 0.6 \qquad (۶-۱۴)$$

$$F_k = \mu_k \times N \quad , \quad \mu_k = 0.6 \qquad (۶-۱۵)$$



شکل ۱۱-۶ تنظیمات ضرایب اصطکاک در محیط شبیه-ساز

۳-۶- قابلیت‌ها

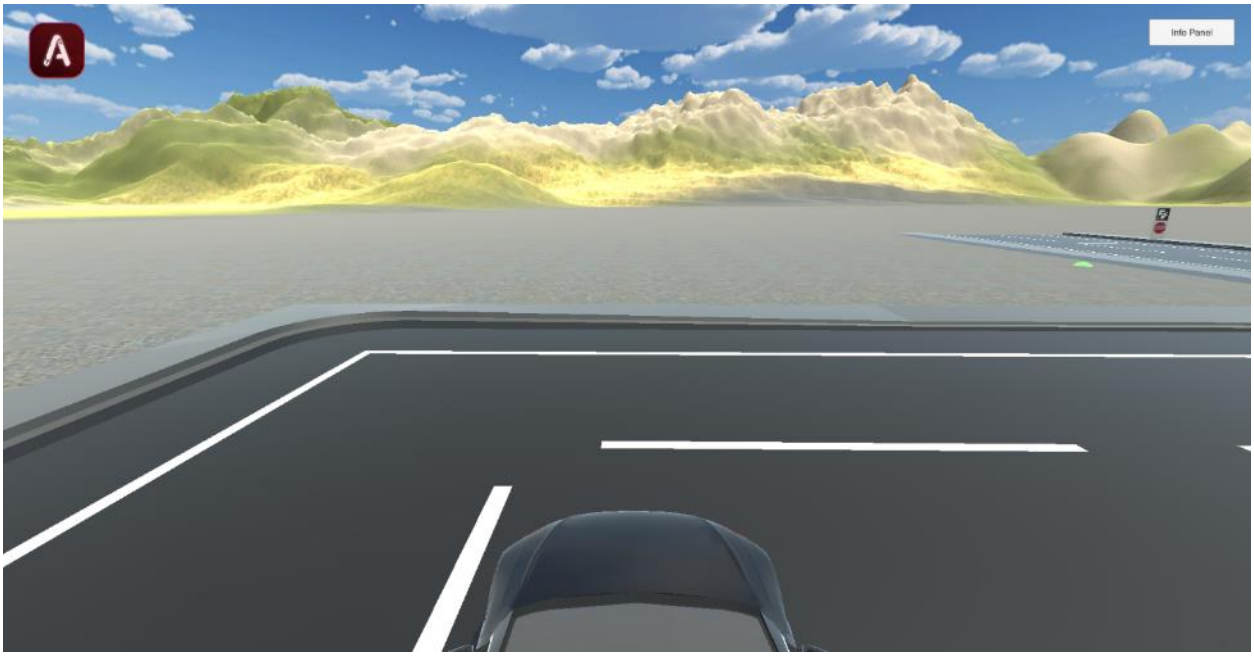
در این قسمت به معرفی قابلیت ها و ویژگی های گوناگون سیستم شبیه سازی و امکانات تعبیه شده درون محیط شبیه ساز پرداخته می شود.

۱-۳-۶- محیط‌ها

دو محیط شهری و بین شهری در این شبیه ساز طراحی شده است که در هر کدام، متناسب به قوانین راهنمایی رانندگی مختص خودشان طراحی گردیده اند.

محیط های شهری به علت عدم وجود گاردریل و امکان سبقت در این محیط ها دارای خط کشی های مقطع سفید رنگ هستتند ، و همچنین دارای تابلوها و چراغ های راهنمایی و رانندگی، چهار راه و بن بست می باشند.

شکل ۱۲-۶ تصویری از یک محیط شهری در شبیه ساز



شکل ۱۳-۶ تصویری از یک بن بست در محیط شهری در شبیه ساز

شکل ۱۴-۶ تصویری از یک چهار راه در محیط شهری در شبیه ساز

محیط های بین شهری دارای به علت وجود گاردریل و عدم امکان سبقت در این گونه محیط ها خط کشی های ممتد زرد رنگ هستند و همچنین شامل پیچ های تندی از ۴۵ درجه تا ۱۱۰ درجه می باشند.

شکل ۱۵-۶ تصویری از یک محیط بین شهری در شبیه ساز

پیچ های تعبیه شده در جاده :

شکل ۱۶-۶ تصویری از یک پیچ جاده ای تعبیه شده در محیط بین شهری در شبیه ساز

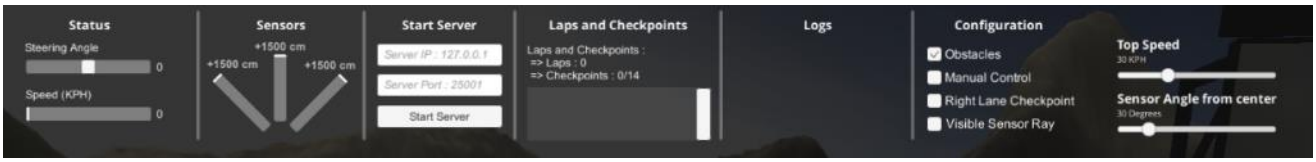
شکل ۱۷-۶ تصویری از یک پیچ جاده ای تعبیه شده در محیط بین شهری در شبیه ساز

۲-۳-۶- داشبورد و پنل نمایش

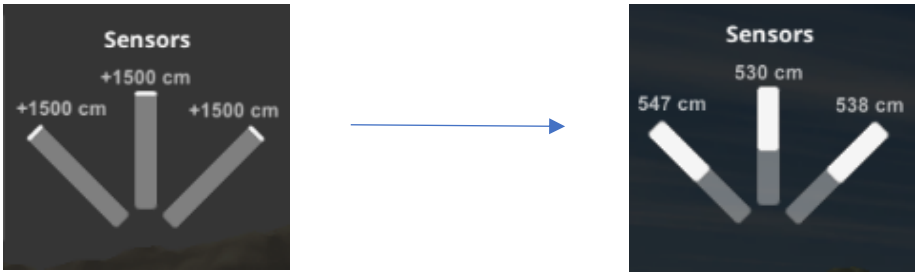
در شبیه ساز استفاده شده برای این پروژه، یک پنل نمایشی در قسمت بالایی شبیه ساز قرار داده شده است که از طریق آن می توان سرعت خودرو، زاویه ی قرار گیری چرخ ها را بدون وقفه [۲۹۰]مشاهده نمود و این دیتا برای اعمال فرمان های کنترلی به خودرو بسیار کاربردی می باشد.

همچنین امکان تغییر زاویه ی سنسور های اولترا سونیک تعبیه شده درون خودرو و همچنین امکان محدود کردن حداکثر سرعت خودرو نیز از طریق این پنل وجود دارد.

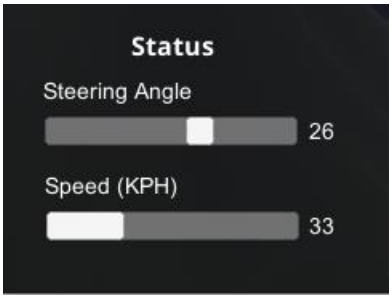
دیتای ارسالی توسط سنسورهای اولتراسونیک نیز در این پنل قابل مشاهده است.



شکل ۱۸-۶ تصویر داشبورد شبیه ساز

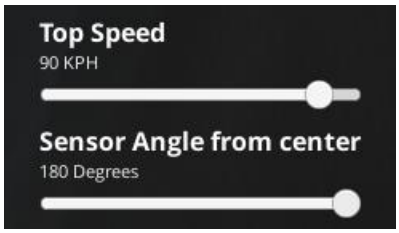


شکل ۱۹-۶ تصویر تغییر وضعیت دیتای ارسالی از سنسور ها هنگام تشخیص مانع



شکل ۲۰-۶ تصویری از پنل نمایش سرعت و زاویه ی چرخ ها به صورت بدون وقفه

[290] Real Time



شکل ۲۱-۶ تصویری از پنل تنظیم حداکثر سرعت و زاویه ی سنسور ها

۳-۳-۶- موانع

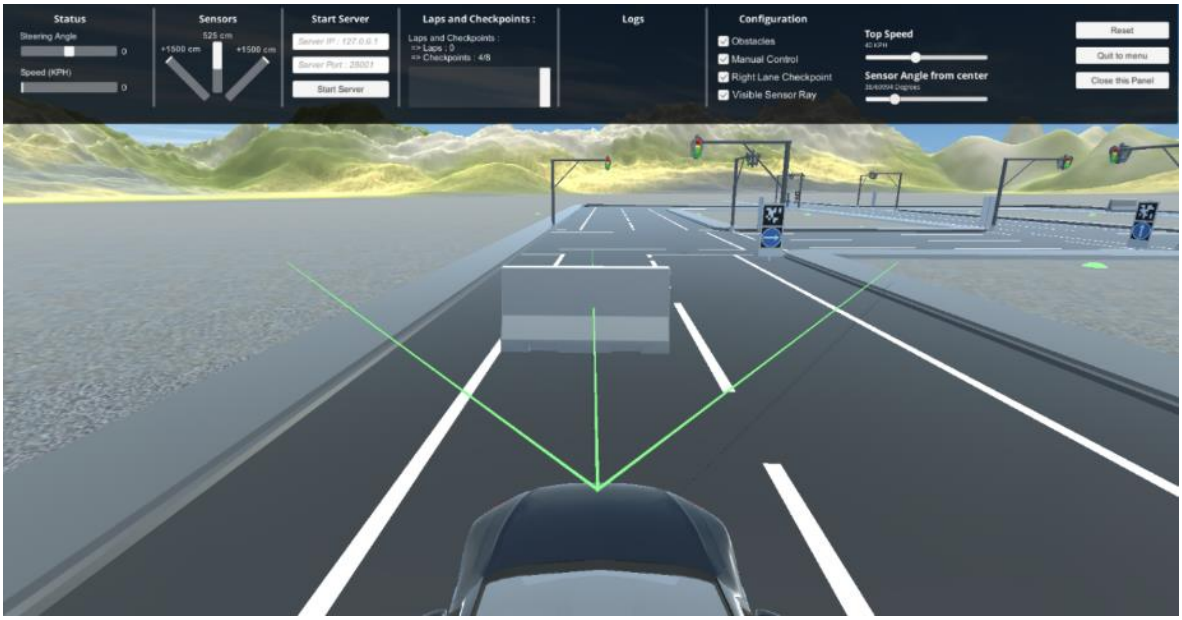
محیط شبیه سازی به گونه ای توسعه داده شده است که امکان قرار دادن مانع در بخش های مختلف خیابان، هم در قسمت شهری و هم در قسمت بین شهری وجود داشته باشد و در بخش کنترل خودرو برای تشخیص این موانع، هم از تکنیک های پردازش تصویر و بینایی ماشین و هم از سنسور های اولتراسونیک استفاده شده است.

شکل ۲۲-۶ تصویری از قرار گیری موانع در محیط بین شهری

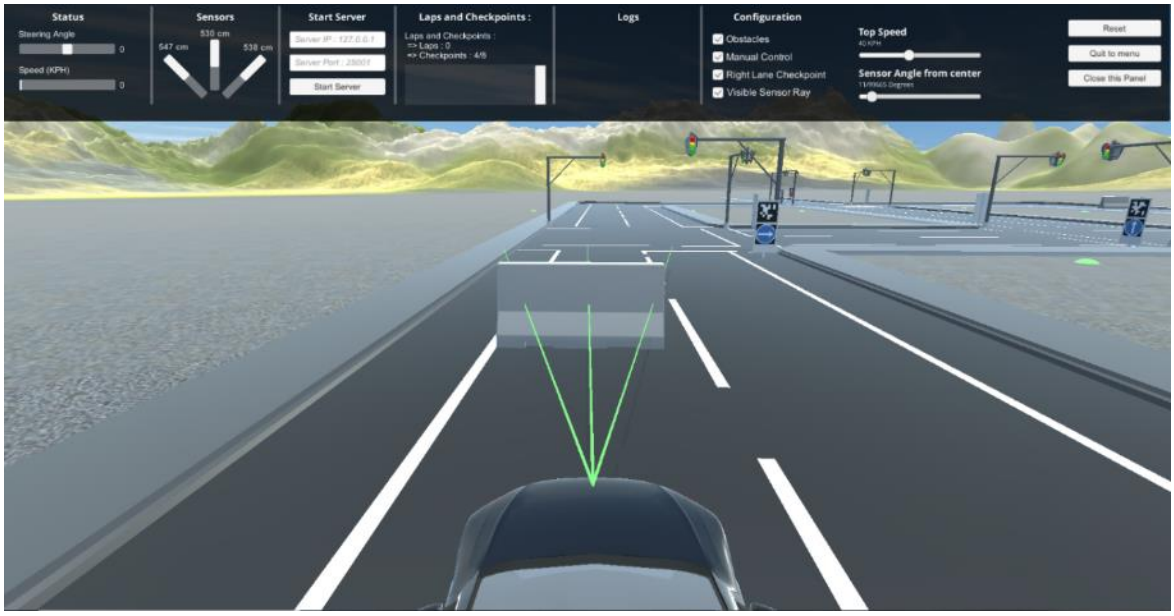
شکل ۲۳-۶ تصویری از قرار گیری موانع در محیط شهری

۴-۳-۶- سنسور های اولتراسونیک

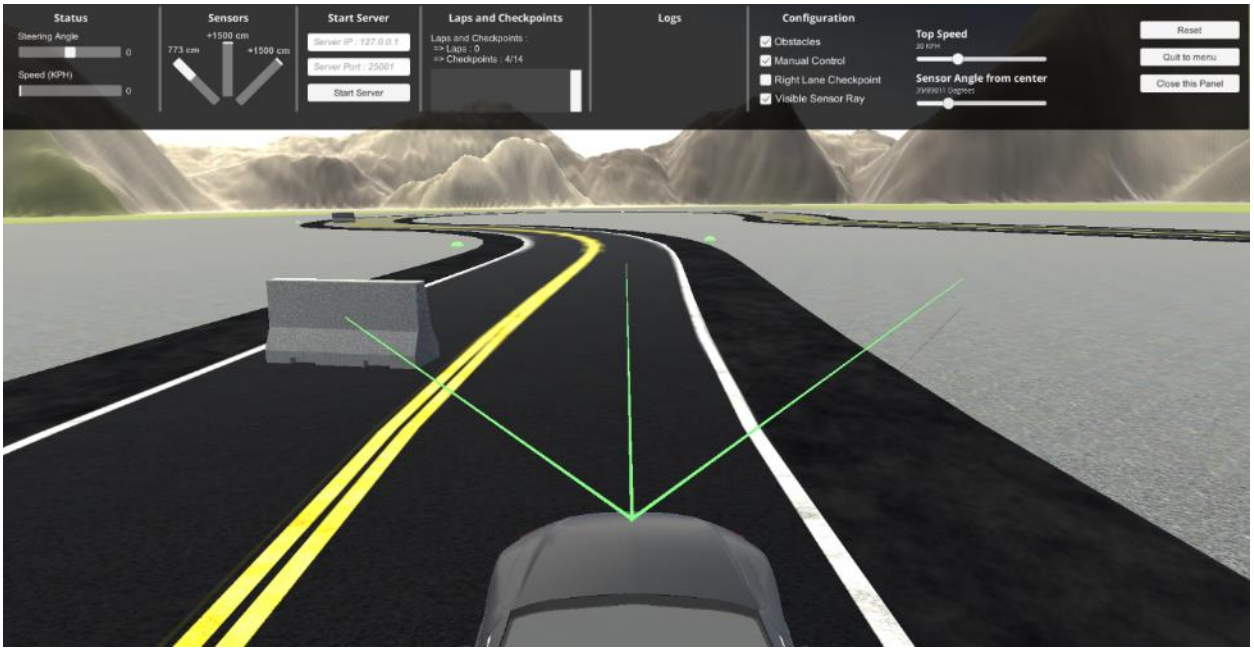
در این محیط تست نرم افزاری پروژه، سه سنسور اولتراسونیک شبیه سازی شده روی خودرو تعبیه شده اند که برد آن ها ۱۵ متر می باشد و قابلیت تشخیص هر گونه مانع یا عابر پیاده را در فاصله های کمتر از ۱۵ متری دارا می باشند. همچنین سنسور های تعبیه شده، دارای قابلیت تنظیم زاویه نیز می باشند و زاویه ی تابش آن ها می تواند از ۰ تا ۱۸۰ درجه تغییر کند.



شکل ۲۴-۶ تصویری از عملکرد سنسور های اولتراسونیک با زاویه ی ۳۹ درجه



شکل ۲۵-۶ تصویری از عملکرد سنسور های اولتراسونیک با زاویه ی ۱۲ درجه



شکل ۲۶-۶ تصویری از عملکرد سنسور های اولتراسونیک با زاویه ی ۴۰ درجه

۵-۳-۶- تابلو ها و علائم راهنمایی و رانندگی

در محیط شبیه سازی شهری، انواع تابلوهای راهنمایی و رانندگی قرار داده شده که در ادامه با الگوریتم های بینایی ماشین تشخیص داده می گردند و مطابق با دستورات آن ها، فرمان کنترلی به خودرو اعمال می گردد.

شکل ۲۷-۶ نمونه ای از تابلوی عبور مستقیم در محیط شبیه ساز

شکل ۲۸-۶ نمونه ای از تابلوی گردش به راست در محیط شبیه ساز

شکل ۲۹-۶ نمونه ای از تابلوی ایست در محیط شبیه ساز

در قسمت پایین، لیست کاملی از علائم طراحی شده و استفاده شده در محیط شبیه ساز تست پروژه آورده شده است :

جدول ۱-۶ تابلوهای راهنمایی و رانندگی موجود در شبیه‌ساز

نام علامت	تصویر علامت	حرکت مجاز
ورود ممنوع		خودرو نباید به خیابان هایی که ابتدای آن ها این علامت نصب شده وارد گردد.
بن بست		خودرو نباید به خیابان هایی که ابتدای آن ها این علامت نصب شده وارد گردد.
گردش به راست		خودرو در صورت مشاهده ی این تابلو، تنها مجاز به ادامه ی حرکت به سمت راست می باشد.
گردش به چپ		خودرو در صورت مشاهده ی این تابلو، تنها مجاز به ادامه ی حرکت به سمت چپ می باشد.
عبور مستقیم		خودرو در صورت مشاهده ی این تابلو، تنها مجاز به ادامه ی حرکت به صورت مستقیم می باشد.
ایست		خودرو به محض تشخیص این علامت باید توقف کامل انجام بدهد.

۷- شبیه سازی شهری

۷-۱- تشخیص خطوط

شکل ۷-۱ نمایی از جاده

یکی از مراحل حیاتی و اساسی در کنترل خودرو تشخیص خطوط جاده می‌باشد که بر مبنای آن تصمیم گرفته می‌شود خودرو هم اکنون در کجای جاده قرار دارد و بر اساس آن عملیات کنترل بر روی خودرو اعمال می‌شود و مقادیر سرعت و زاویه فرمان تنظیم می‌گردد. با توجه به شبیه ساز AvisEngine و ثابت بودن شرایط محیطی و رنگ‌های موجود در این شبیه‌ساز بهترین رویکرد برای تشخیص خطوط استفاده از پردازش تصویر کلاسیک می‌باشد که به واسطه رنگ‌های مشخص هر شی می‌توان با دقت خوبی خطوط جاده را تشخیص داد. همانطور که در تصویر ۷-۱ که نمایی از جاده قابل مشاهده توسط خودرو می‌باشد مشخص است، خطوط به رنگ سفید می‌باشند.

بخش تشخیص خطوط به دو بخش تشخیص خطوط افقی و عمودی تقسیم می‌شود. از خطوط عمودی تشخیص داده شده به منظور کنترل خودرو در مسیر جاده و از خطوط افقی به منظور تشخیص دور زدن خودرو و ایستادن پشت چهار راه استفاده می‌شود.

۷-۱-۱- تشخیص خطوط عمودی

برای تشخیص خطوط عمودی از الگوریتم تشخیص خط هاف استفاده شده که تئوری و نحوه عملکرد این الگوریتم در بخش ۲ بیان شده است. در این قست به بررسی کد آن پرداخته شده است که چگونه این الگوریتم توانایی تشخیص خطوط موجود در تصویر با دقت بالایی دارد.

برای تشخیص یک خط در تصویر باید گام‌های زیر طی شود.

۱-۱-۱-۷ گام اول: تشخیص خطوط سفید

اولین گام برای تشخیص خطوط، مشخص کردن رنگ آن است. خروجی این بخش به صورت تصویر ماسک[۲۹۱] است و از آن به عنوان ورودی به الگوریتم هاف استفاده می‌شود. با توجه به مشخص بودن رنگ سفید در شبیه ساز، به راحتی می‌توان با پیدا کردن بازه تغییرات پیکسلی رنگ سفید در تصویر، ماسک مناسبی از هر فریم ایجاد کرد. کد آن به صورت مقابل آورده شده است:

```
1.  white_mask = cv2.inRange(Frame, Min_Value, Max_Value) * (1-car_mask)
```

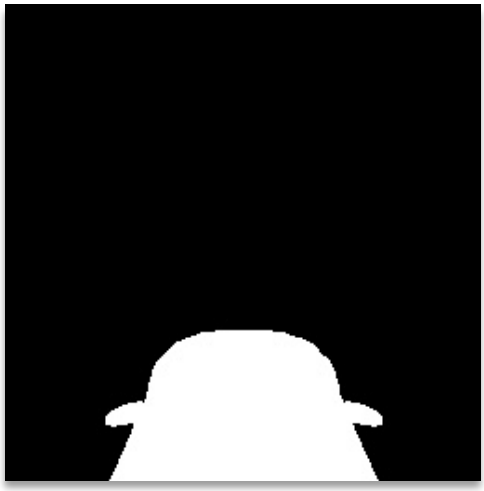
- **Frame**: تصویر ورودی
- **Min**: کمترین مقادیر آبی، سبز و قرمز (BGR) در تصویر

np.array([240,240,240]

- **Max**: بیشترین مقادیر آبی، سبز و قرمز (BGR) در تصویر

np.array([255,255,255])

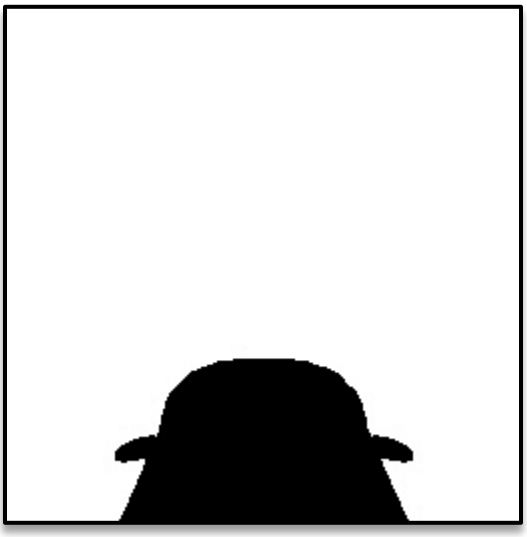
- **Car_mask**: این متغیر، ماسک خودرو می‌باشد که از قبل تعریف شده است. بدلیل نزدیک بودن رنگ خودرو به رنگ خطوط سفید، ممکن است در ماسک ایجاد شده تصویر خودرو نیز باشد. ماسک خودرو به صورت تصویر ۲-۷ می‌باشد.



شکل ۲-۷ ماسک خودرو

[291] Mask

به همین دلیل با بدست آوردن ماسک خودرو و تفاضل آن با یک، باعث می‌شود رنگ جاده به صورت سفید با مقدار پیکسلی یک و رنگ خودرو به رنگ سیاه با مقدار پیکسلی صفر تبدیل شود. خروجی آن به صورت تصویر ۳-۷ می‌باشد.



شکل ۳-۷ ماسک محیط اطراف خودرو

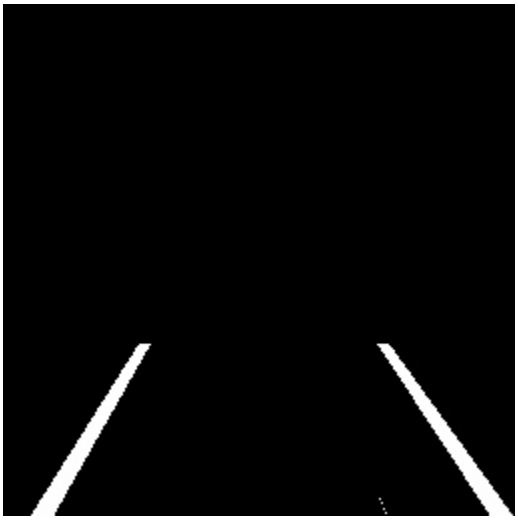
حال با ضرب کردن *(1-car_mask)* در ماسک تصویر، خروجی ایجاد می‌شود.

- **خروجی:** ماسک ایجاد شده که در آن خطوط با رنگ سفید مشخص شده‌اند. نمونه ای از خروجی در تصویر ۴-۷ آورده شده است:

شکل ٤-۷ ماسک خطوط جاده

۲-۱-۱-۷ گام دوم: اعمال منطقه مورد نظر[۲۹۲] به خروجی گام اول

بعد از بدست آوردن ماسک ایجاد شده بر اساس رنگ سفید، به منظور کنترل خودرو، فقط به رنگ خطوط جاده که در مقابل خودرو قرار دارد نیاز می‌باشد. به همین منظور با اعمال منطقه مورد نظر به صورت ذوزنقه‌ای شکل، فقط خطوط سفید جاده که در مقابل خودرو است مشخص می‌گردد که نمونه ای از خروجی آن بعد از اعمال گام اول در تصویر ۵-۷ آورده شده است:



شکل ۵-۷ ماسک خطوط جاده بعد از اعمال منطقه مورد نظر

این کار توسط تابع *region_of_interest* انجام می‌شود:

```
1.  def region_of_interest(image):
2.      (height, width) = image.shape
3.      mask = np.zeros_like(image)
4.      polygon = np.array([[
5.          (0, height),
6.          (0, 180),
7.          (80, 130),
8.          (256-80,130),
9.          (width, 180),
10.         (width, height)]], np.int32)
11.
12.     cv2.fillPoly(mask, polygon, 255)
13.     masked_image = image * (mask)
14.     masked_image[:170, :]=0
15.     return masked_image
```

[292] ROI

- **ورودی:** خروجی گام اول
- **خروجی:** خطوط سفید جاده در مقابل خودرو

۳-۱-۱-۷ گام سوم: تشخیص خطوط عمودی به وسیله الگوریتم هاف

در کتابخانه OpenCV تشخیص خط با استفاده از تبدیل هاف را می‌توان با دو تابع HoughLines یا HoughLinesP پیاده‌سازی کرد. HoughLinesP مربوط به نوع دیگری از تبدیل هاف است که «تبدیل هاف احتمالاتی» (Probabilistic Hough Transform) نام دارد. این تابع آرگومان‌های زیر را به عنوان ورودی دریافت می‌کند.

- **edges :** خروجی بعد از اعمال منطقه مورد نظر بر روی ماسک خطوط تصویر
- **Lines :** یک بردار برای ذخیره‌سازی مختصات شروع و پایان خطوط.
- **Min_threshold :** تعداد کمینه نقاط تقاطع برا تشخیص یک خط.

کدهای مربوط به پیاده‌سازی تبدیل هاف در زبان پایتون در ادامه آورده شده است:

```
1. rho = 1
2. angle = np.pi / 180
3. min_threshold = 10
4. lines = cv2.HoughLinesP(edges, rho, angle, Min_threshold, np.array([]), minLineLength=8, ma
   xLineGap=4)
```

با توجه به رابطه []، ρ برابر با فاصله عمودی خط از مبدا بر حسب پیکسل و θ زاویه خط با مبدا است که بر حسب رادیان اندازه گرفته می‌شود.

در نهایت تمامی عملیات تعریف شده در تابع *detect_lines* انجام می‌شود:

```
1. def detect_lines(image):
2.     rho = 1  # precision in pixel, i.e. 1 pixel
3.     angle = np.pi / 180  # degree in radian, i.e. 1 degree
4.     min_threshold = 10  # minimal of votes
5.     lines = cv2.HoughLinesP(image, rho, angle, min_threshold, np.array([]), minLineLength=8,
6.                             maxLineGap=4)
7.
8.     return lines
```

۲-۱-۷- تشخیص خطوط افقی

برای تشخیص خطوط افقی نیز همانند تشخیص خطوط عمودی از الگوریتم تشخیص خط می‌شود. در این قست به بررسی کد آن پرداخته شده است که چگونه این الگوریتم توانایی تشخیص خطوط افقی موجود در تصویر با دقت بالایی دارد.

برای تشخیص خطوط افقی در تصویر باید گام‌های زیر طی شود.

۱-۲-۱-۷ گام اول: تشخیص خطوط سفید

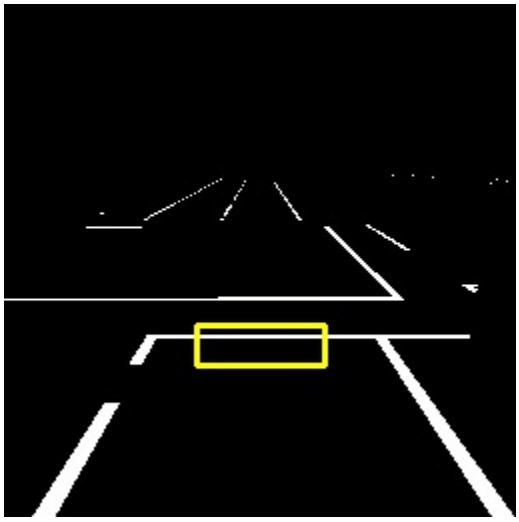
از خروجی گام تشخیص خطوط سفید که در بخش ۱-۱-۱-۷ توضیح داده شد در این قسمت نیز استفاده می‌شود.

۲-۲-۱-۷ گام دوم: اعمال منطقه مورد نظر به خروجی گام اول

منطقه مورد نظر به صورت مقابل تعریف می‌شود:

```
1.  roi = mask[160:180, 96:160]
```

با اعمال کد مقابل منطقه ای که در تصویر ۶-۷ مشخص شده است با رنگ زرد جدا می‌شود و با توجه به آن مشخص می‌شود که آیا در این منطقه خط افقی قرار دارد یا خیر.



شکل ٦-۷ مشخص کردن منطقه مورد نظر برای تشخیص خطوط افقی

۳-۲-۱-۷ گام سوم: تشخیص خطوط افقی به وسیله الگوریتم هاف

کد و مشخصات پارامتری آن همانند تشخیص خطوط عمودی می‌باشد که بر روی منطقه مورد نظر به منظور تشخیص خطوط افقی اعمال می‌شود.

۴-۲-۱-۷ گام چهارم: آیا در تصویر خط افقی هست یا خیر؟

بعد از انجام مراحل فوق می‌بایست توجه داشت که آیا خطی که تشخیص داده شده افقی هست یا خیر. بدین منظور از تابع *horiz_lines* استفاده می‌شود.

```
1.  def horiz_lines(mask):
2.      roi = mask[160:180, 96:160]
3.      try:
4.          lines = detect_lines(roi)
5.          lines = lines.reshape(-1,2,2)
6.          slope = (lines[:,1,1]-lines[:,0,1]) / (lines[:,1,0]-lines[:,0,0])
7.
8.          if (lines[np.where(abs(slope)<0.2)]).shape[0] != 0:
9.              detected = True
10.         else:
11.             detected = False
12.     except:
13.         detected = False
14.     return detected
```

همانطور که گفته شده است، خروجی الگوریتم هاف به صورت نقاط موجود در خط به صورت $[(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)]$ می‌باشد، از این نقاط برای بدست آوردن شیب خط استفاده می‌گردد و سپس با در نظر گرفتن آستانه ای مشخص می‌گردد که آیا این خط افقی هست یا خیر.

مقدار آستانه شیب در نظر گرفته برابر ۰.۲ می‌باشد، بدین منظور که اگر شیب خط حاصل کمتر از آن باشد، خط افقی تشخیص داده شده و تابع عبارت True باز می‌گرداند و اگر شیب خط بیشتر از مقدار مشخص شده باشد، تابع عبارت False خروجی می‌دهد.

- **ورودی:** خروجی گام دوم
- **خروجی:** آیا در تصویر منطقه مورد نظر خط هست یا خیر

۲-۷- سنسورهای مورد استفاده

۱-۲-۷- سنسور انکودر

انکودر یک نوع سنسور حرکت است. اگر حرکتی که انکودر می‌سنجد، از نوع دورانی باشد به آن انکودر زاویه یا دوار می‌گوییم و اگر از نوع خطی باشد به آن انکودر خطی می‌گوییم. علت اصلی به کاربردن انکودر در صنایع مختلف گرفتن فیدبک (feedback) از محیط است و با استفاده از آن می توان یک سیستم حلقه بسته ایجاد کرد [۳۰۹][۳۱۰] (close loop)

ترجمه فارسی انکودر، رمزگذار است. علت این نامگذاری این است که انکودر حرکت را رمزگذاری می‌کند و آن را به صورت کدهای دیجیتال در می آورد. بعد از آن با اتصال انکودر به یک دستگاه رمزخوان مانند PLC ، مقدار حرکت ثبت شده توسط انکودر قابل خواندن می‌شود.

شکل ۷-۷ ساختمان یک نمونه انکودر

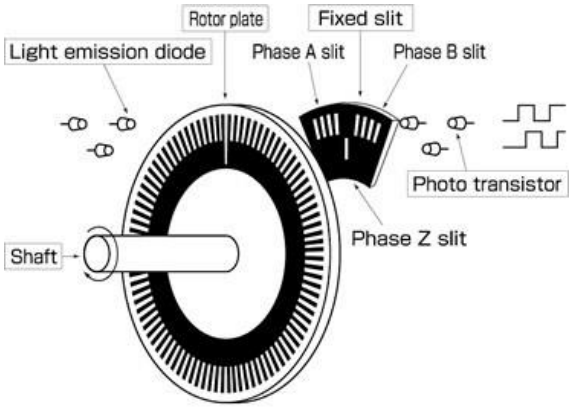
انکودرهای دوار به دو نوع انکودر مطلق یا absolute و انکودر افزایشی یا incremental تقسیم می شوند.

۱-۱-۲-۷- سنسور افزایشی (Incremental)

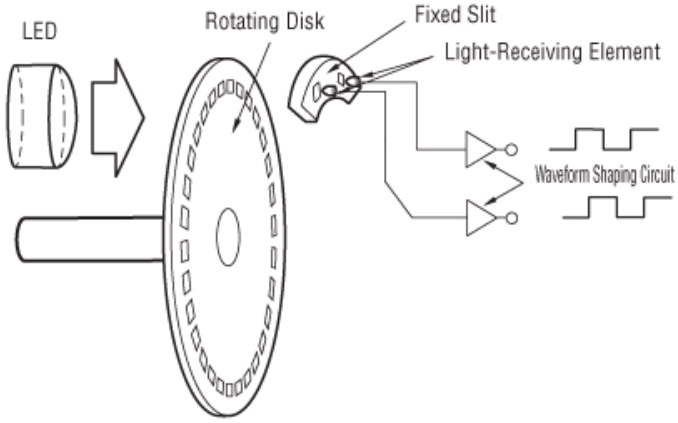
انکودر افزایشی دورانی با نام انکودر پالس مربعی (Quadrature Encoder) نیز شناخته می‌شود. عملکرد این نوع انکودر به وسیله نور انجام می‌شود. به این صورت که درون انکودر افزایشی یک صفحه دوار قرار دارد که بر روی آن صفحه پر از سوراخ است. هر بار که صفحه حرکت می‌کند و سوراخ ها از مقابل منبع نور عبور می‌کنند، نور از سوراخ عبور کرده و به سنسور نور برخورد می‌کند. در نتیجه انکودر متوجه می‌شود که حرکت انجام شده است. با هر بار عبور نور از سوراخ اصطلاحا گفته می‌شود که انکودر یک پالس می اندازد[۳۱۰].

یکی از معایب انکودر افزایشی این است که تنها مقدار تغییرات حرکت را مشخص می‌کند و قادر به تعیین موقعیت نیست. همچنین دقت انکودر افزایشی به تعداد سوراخ های آن وابسته است به این صورت که هر چه تعداد این سوراخ ها بیشتر باشند، دقت انکودر نیز بیشتر است. به عنوان مثال اگر در یک انکودر تعداد ۳۶۰ سوراخ داشته باشیم، از آنجا که کل صفحه ۳۶۰ درجه است، در نتیجه انکودر مورد نظر از دقتی برابر با ۱ درجه برخوردار است. در واقع دقت انکودر افزایشی از رابطه زیر محاسبه میشود:

$$\text{دقت} = \frac{360}{\text{تعداد پالس}} \qquad (۷-۱)$$

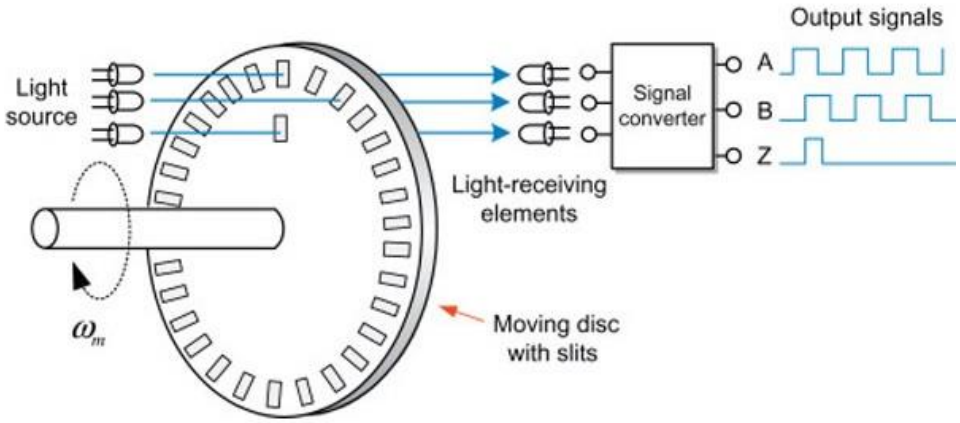


شکل ۸-۷ ساختمان و نحوه کار انکودر دوار افزایشی



شکل ۹-۷ ساختمان و نحوه کار انکودر دوار افزایشی

همچنین اگر از این انکودر در رباتی استفاده شده باشد و لحظه ای برق قطع شود، پس از اتصال دوباره برق ربات باید دوباره از نقطه صفر شروع به کار کند.



شکل ۱۰-۷ ساختمان و نحوه کار انکودر دوار افزایشی و سیگنال های خروجی آن

انکودرافزایشی مقدار مشخصی از پالس ها را در یک چرخش دیسک انکودر فراهم می کند و دارای دو سیگنال خروجی A و B است که هنگام جابجایی دستگاه پالس صادر می کنند. خروجی می تواند یک خط پالس (یک کانال A) یا دو خط پالس (دو کانال A وB) باشد که برای تعیین جهت چرخش جابجا می شوند. سیگنال های A و B وقوع و جهت حرکت را نشان می دهند. با چرخش چرخ ما دو پالس خروجی داریم که با یکدیگر اختلاف فاز دارند و براساس آن می توان جهت چرخش را نیز مشخص نمود که به این فرایند مرحله بندی بین دو سیگنال (quadrature) می‌گویند. بسیاری از انکودر های افزایشی دارای یک سیگنال خروجی اضافی هستند که به طور معمول با شاخص Z مشخص شده اند، که نشان می دهد رمزگذار در یک موقعیت مرجع خاص قرار دارد. همچنین، برخی از خروجی انکودر ها وضعیتی را ارائه می دهند که نشان دهنده شرایط خطای داخلی مانند خرابی بلبرینگ یا اشتباه در عملکرد سنسور است. درواقع علاوه بر سوراخ هایی که روی لبه صفحه قرار گرفته، یک تک سوراخ هم در میانه های صفحه ایجاد شده است. این سوراخ تعیین کننده زاویه صفر و نقطه شروع حرکت است و حرکت انکودر را کالیبره میکند و تعداد دورهای کامل را می‌شمارد. از طریق این سوراخ می‌توان در حرکت های طولانی خطای انکودر را اصلاح کرد. به این صورت که ممکن است وقتی که یک دور کامل زد، به علت خطا ۳۶۰ درجه را ۳۵۰ درجه اعلام کند که وقتی به نقطه Z می‌رسد، با صفر شدن زاویه، خطا نیز اصلاح می‌شود[۳۱۱].

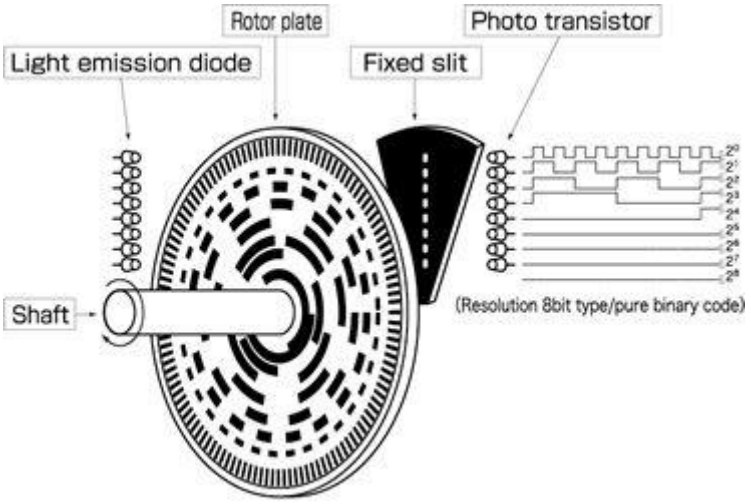
مزایای انکودر های incremental:

۱. مناسب برای شمارش پالس های ساده یا مانیتورینگ فرکانس هایی همچون سرعت ، جهت و موقعیت
۲. مرقون به صرفه و مکانیزم ساده نسبت به انکودر های مطلق
۳. اندازه گیری مغناطیسی
٤. انکودر های INCREMENTAL (افزایشی) دارای رزولوشن ۵۰۰۰ PPR[293] هستند.

[293] PULSE PER REVOLUTION

۲-۱-۲-۷- سنسور مطلق (Absolute)

انکودر مطلق علاوه بر حرکت، تعیین موقعیت نیز انجام می‌دهد. نحوه کارکرد این نوع انکودر به این صورت است که درون انکودر مطلق یک صفحه دوار قرار دارد که این صفحه متشکل از چندین شبکه است و هر کدام از این شبکه ها به چندین قطاع تقسیم شده اند. به هر یک از قطاع های شبکه ها یک بیت گفته می‌شود و به هر کدام از شبکه ها یک الگوی رنگی منحصر بفرد اختصاص داده می‌شود. با توجه به اینکه شاخص روی چه الگوی رنگی قرار گرفته باشد، انکودر مطلق تشخیص می‌دهد که موقعیت فعلی در چه زاویه ای قرار دارد.[۳۱۰]



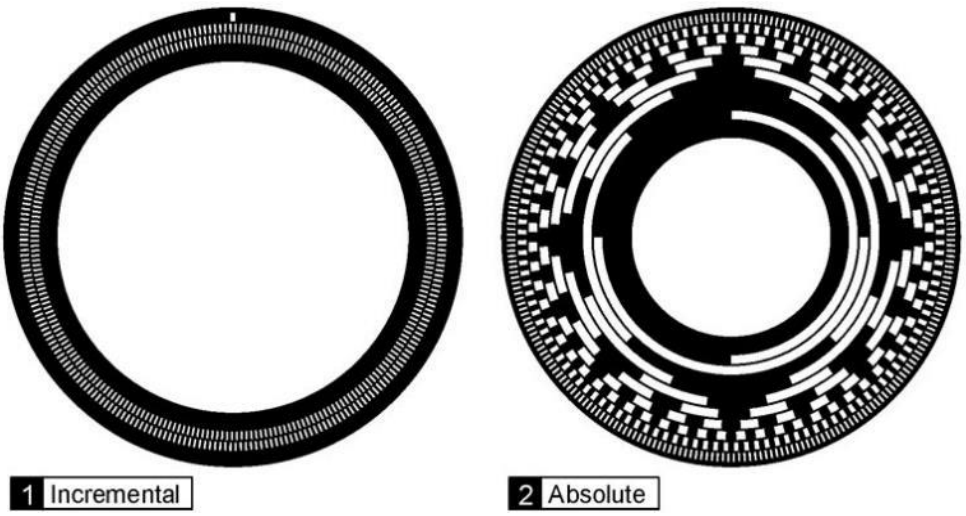
شکل ۱۱-۷ ساختمان و نحوه کار انکودر دوار مطلق

یک انکودر چرخشی absolute موقعیت خود را با استفاده از نقطه رفرنس تعیین می نماید. این روش بسته به اینکه آیا انکودر چرخشی absolute نوری و یا مغناطیسی است، کمی متفاوت میباشد. اما اصل در هر دو حالت یکسان است. دو دیسک وجود دارد ، هر دو با حلقه های متحدالمرکز . یک دیسک به محور مرکزی متصل است و دیگری آزادانه حرکت می کند. همانطور که دیسک می چرخد نشانگر ها در امتداد مسیر انکودر absolute ، موقعیت روی دیسک ثابت را تغییر میدهند. هر پیکربندی در امتداد دیسک انکودر، کد باینری منحصر به فردی است. نگاهی به کد باینری درون انکودر، موقعیت مطلق شئ را تعیین میکند. برای انکودر های مطلق نوری، نشانگر از طریق نور عمل میکند. برای انکودر های مطلق مغناطیسی (magnetic) نشانگر ها یک ردیف از سنسورهای مغناطیسی هستند که از یک آهنربا عبور می‌کنند و موقعیت قطب های مغناطیسی را تشخیص می دهند.

انکودر مطلق در مقایسه با انکودر افزایشی از دقت بیشتری برخوردار است. دقت انکودر مطلق با بیت بیان میشود و از فرمول زیر محاسبه میشود:

$$\text{دقت} = \frac{360}{2^{\text{تعداد بیت}}} \qquad (۷-۲)$$

همچنین انکودر چرخشی مطلق Absolute می تواند موقعیت را به صورت مطلق ارسال نماید. یعنی با خاموش شدن دستگاه و قطع برق و روشن شدن مجدد آن، انکودر می تواند موقعیت خود را به صورت دقیق اعلام نماید و مشکلی پیش نمی آید.



شکل ۱۲-۷ مقایسه صفحه دوار انکودر دوار افزایشی و مطلق

مزایای انکودر های absolute:

۱. حفظ و به یادآوری موقعیت بعد از خاموشی دستگاه و تداوم مانیتورینگ موقعیت
۲. امکان تعیین موقعیت ماشین و امکان ذخیره سازی دیتا الکترونیک
۳. امکان استفاده از گزینه های چند گانه مانند analog , Ethernet , fieldbus, parallel , serial
٤. استفاده از اندازه گیری نوری و مغناطیسی
٥. انکودر های مطلق دارای رزولوشن ۱۶ بیت و یا ۶۵.۵۳۶ پالس در هر چرخش دارند(PPR)

۳-۷- سیستم دور زدن در محیط شبیه ساز شهری

ما می‌خواهیم که خودرو همواره در لاین راست حرکت کند. لذا پس از تشخیص مانع، باید مانع را دور بزند. یعنی ابتدا برای عدم برخورد به مانع به لاین سمت چپ رفته و پس از گذر از مانع به لاین راست برگردد.

تابع turn_the_car به صورت زیر تعریف شده است.

```
def turn_the_car(car,s,t):
    time1 = time.time()
    while((time.time()-time1)<t):
        car.getData()
        car.setSteering(s)
        car.setSpeed(15)
```

ورودی های این تابع، خودرو (car) ، زاویه ی چرخش (s) و مدت زمان چرخش (t) است. و بدین صورت عمل می کند که خودرو به مدت t ثانیه، زاویه ی s را به فرمان داده و با این زاویه حرکت می کند.

این تابع به نوعی شبیه سازی انکودر می‌باشد. در واقع با حرکت چرخشی خودرو با زاویه‌ی تعیین شده، هر لحظه توسط تابع car.getData() از خودرو فیدبک گرفته می‌شود که همان نقش انکودر می‌باشد و میزان جابه‌جایی خودرو مشخص می‌شود.

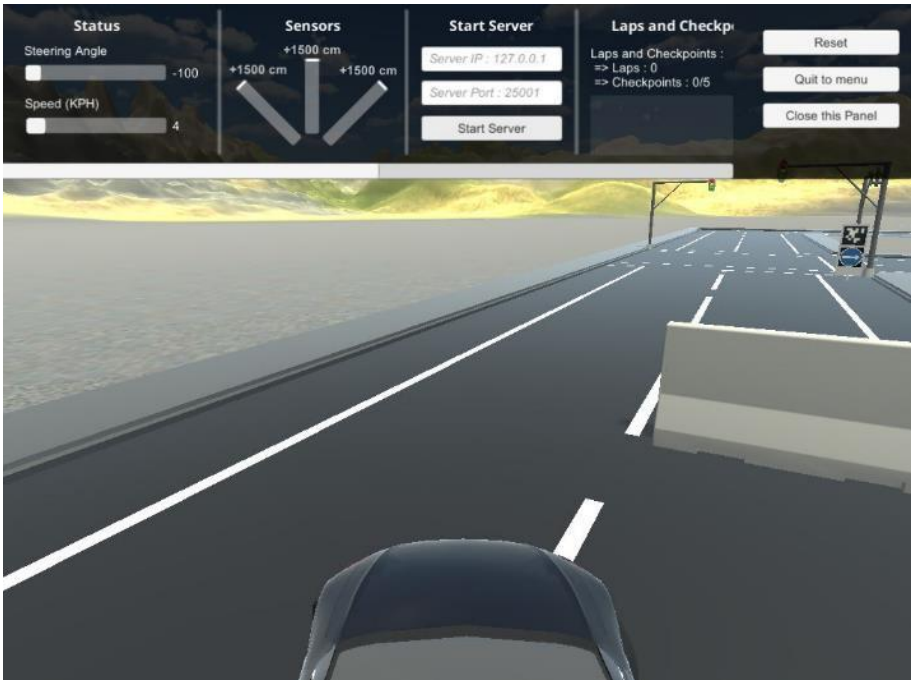
برای سیستم دور زدن، کد زیر نوشته شده است.

```
if sensors[1] < 700:
        ret = utils.stop_the_car(car)
        print('side_pix :', side_pix)
        time.sleep(3)
        if side_pix > 128:
            utils.turn_the_car(car,-100,5.5)
            utils.turn_the_car(car,100,6.5)
            utils.turn_the_car(car,-100,2.5)
        else:
            utils.turn_the_car(car,100,4)
```

سیستم بدین صورت است که اگر سنسور وسط، فاصله ی کمتر از ۷۰۰ سانتی متر تشخیص دهد که نشان دهنده‌ی تشخیص مانع است، در ابتدا با استفاده از تابع utils.stop_the_car خودرو می ایستد و به مدت ۳ ثانیه صبر می‌کند.

شکل ۱۳-۷ مشاهده‌ی مانع توسط خودرو

سپس در صورتی که متغیر side_pix بزرگتر از ۱۲۸ باشد، که نشان از سمت راست بودن خودرو است، در سه مرحله از تابع utils.turn_the_car استفاده می شود تا خودرو گردش کند. در ابتدا برای هر مرحله عددهای تصادفی برای زاویه و زمان چرخش داده شد و با توجه به فیدک داده شده توسط انکودر شبیه سازی شده، عدد های نهایی به دست آمدند. در ابتدا خودرو به مدت ۵.۵ ثانیه، زاویه ی منفی ۱۰۰ را به فرمان می دهد. در این حالت خودرو به لاین چپ می رود.



شکل ۱۴-۷ دور زدن مانع توسط خودرو

در گام بعدی باید به لاین راست برگردد پس به مدت ۶.۵ ثانیه زاویه‌ی مثبت ۱۰۰ را به فرمان می دهد. حال خودرو به لاین راست بازگشته است ولی جهت خودرو مستقیم نیست.

شکل ۱۵-۷ بازگشت خودرو به لاین خود

در نتیجه خودرو باید دوباره گردش به چپ داشته باشد ولی مدت زمان این گردش کم است چرا که فقط به منظور صاف شدن جهت خودرو می‌باشد. در این مرحله خودرو به مدت ۲.۵ ثانیه، زاویه‌ی منفی ۱۰۰ را به فرمان داده و خودرو صاف می‌شود و بدین صورت خودرو مانع را به طور کامل دور می‌زند. لازم به ذکر است که این مقادیر زمانی و زاویه ها با توجه به سرعت تعیین شده برای خودرو و فیدبک گرفته شده با این سرعت، تعیین شده اند و در صورتی که سرعت خودرو را تغییر دهیم، این پارامتر ها نیز باید تغییر کنند.

شکل ۱۶-۷ زاویه‌ی چرخش منفی برای بازگشت به خط مربوطه

۴-۷- سنسور آلتراسونیک

به طور کلی امواج صوتی با فرکانس‌هایی بالاتر از محدوده شنوایی انسان، را اولتراسوند، التراسونیک و یا فراصوت می‌نامند. محدوده شنوایی انسان بین ۲۰ هرتز تا ۲۰ کیلو هرتز است پس امواج صوتی بیش از ۲۰ کیلو هرتز را امواج التراسونیک می‌نامند. این امواج توسط بسیاری از حیوانات قابل شناسایی است. خفاش، گربه، سگ و عمده‌ی حشرات و ماهی‌ها از این دسته هستند. خفاش از جمله حیواناتی هستند که از این امواج برای پرواز (ناوبری) خود استفاده می‌کند.

۱-۴-۷- معرفی سنسور آلتراسونیک

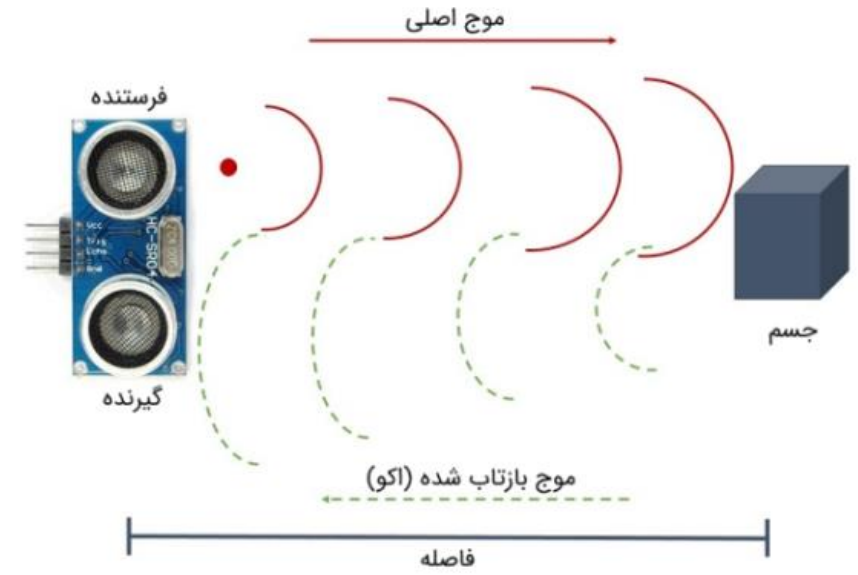
سنسور فاصله سنج التراسونیک یا سنسور مافوق صوت یک سنسور مجاورتی است که با ارسال پالس صوتی کوتاه، در فرکانس بالاتر از محدوده شنوایی انسان و دریافت آن پس از انعکاس از سطح جسم مورد نظر، فاصله را از طریق محاسبه زمان رفت و برگشت امواج مافوق صوت در هوا، با وضوح بالا اندازه گیری می نماید. سنسور التراسونیک هیچ حساسیتی به رنگ، نور یا بو نداشته و بدون تماس با جسم، فاصله تا هدف را پیدا می کند. سطح سنج، ارتفاع سنج و یا سنسور التراسونیک بدلیل دقت سرعت بالا ، عدم تماس با جسم و نداشتن خورندگی و سایش، هزینه تعمیر و نگهداری پایین و همچنین قیمت مناسب، یکی از بهترین راهکارهای اندازه گیری فاصله و آشکار سازی اجسام در صنعت شناخته شده است[۳۱۲].

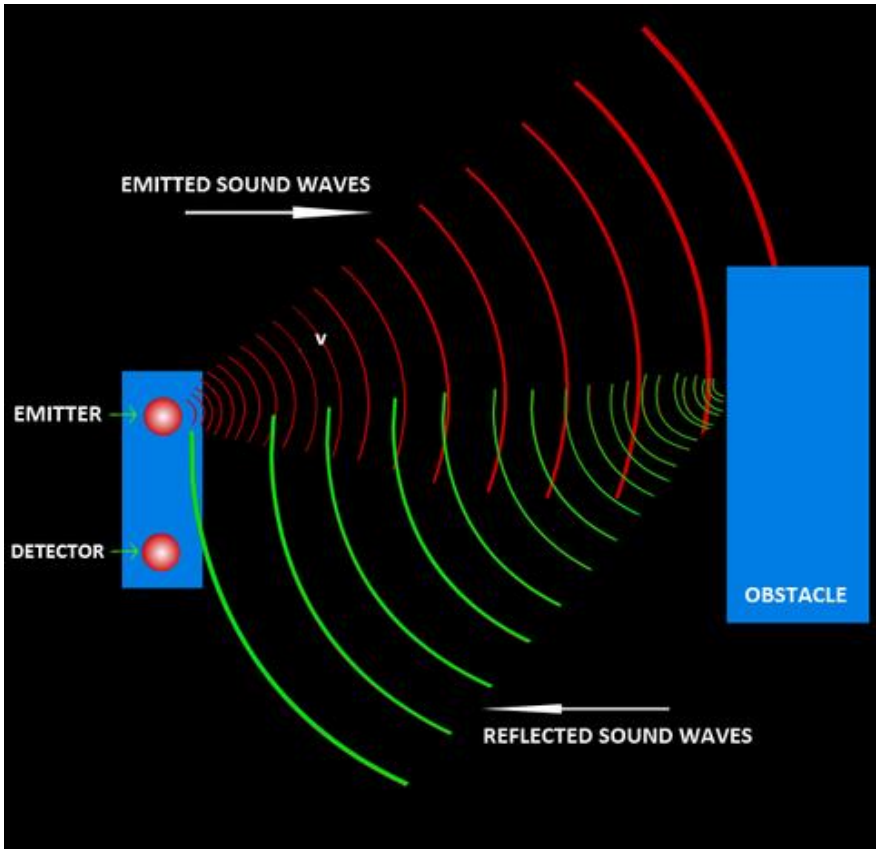
شکل ۱۷-۷ سنسور التراسونیک

به صورت کلی سنسور های التراسونیک در سه دستهDiffuse ، Reflex و Thru-Beam قرار می‌گیرند. مدل Diffuse حالت استاندارد سنسور است. در مدل Reflex قرار دادن یک رفلکتور در رنج عملکرد سنسور موجب ایجاد موج برگشتی یا اکو می‌شود. به این ترتیب اگر جسمی در برابر سیگنال برگشتی قرار گیرد آنگاه سیگنال به سنسور نخواهد رسید. در این حالت سنسور وجود جسم را تشخیص می‌دهد. نوع سوم این سنسور یا Thru-Beam شامل فرستنده و گیرنده بوده می‌باشد. در این نوع سنسور قرار گرفتن جسم بین فرستده و گیرنده موجب تغییر حالت در گیرنده می‌شود.

عملکرد سنسور فاصله سنج التراسونیک را می‌توان در چهار مرحله زیر خلاصه کرد[۳۱۳].

۱. سنسور التراسونیک امواج صوتی با فرکانس بالا را به سمت جسم مورد نظر ساطع می‌کند.
۲. جسم هدف امواج صوتی را دریافت می‌کند.
۳. سپس امواج صوتی منعکس شده و به سمت سنسور التراسونیک بازتاب می‌شوند.
٤. مدت زمان بازگشت موج صوتی برای اندازه‌گیری فاصله بین استفاده می‌شود.





شکل ۱۸-۷ فرستادن امواج صوتی و دریافت امواج اکو توسط سنسور التراسونیک

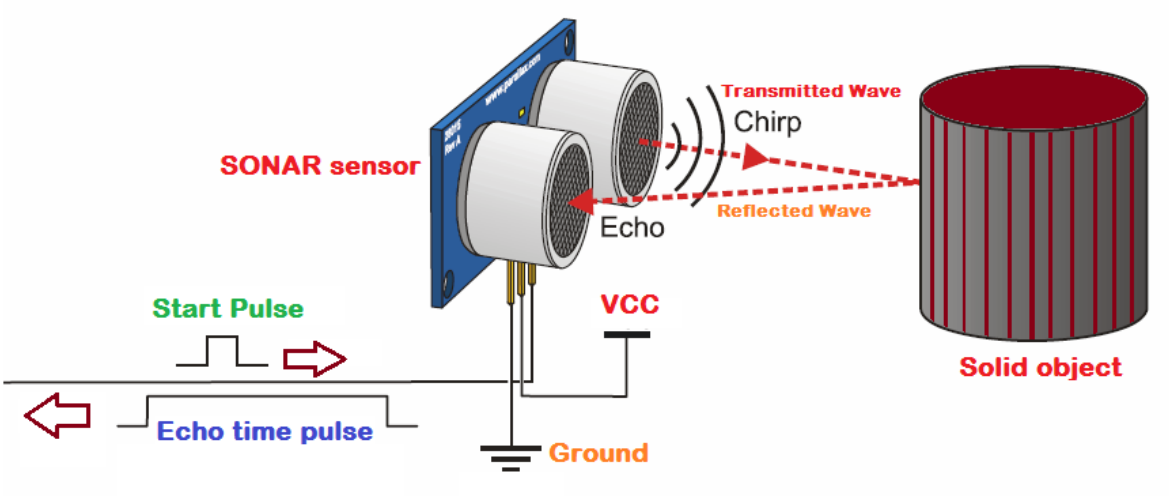
فرمول این محاسبه به صورت فرمول ۳-۷ می‌باشد.

$$D = \frac{1}{2} T \times C \qquad (۷-۳)$$

که در این فرمول D فاصله است، T زمان است و C سرعت صدا ۳۴۳ متر در ثانیه است). به عنوان مثال، اگر دانشمندی یک سنسور اولتراسونیک را با هدف تشخیص فاصله با یک جعبه تنظیم کند و ۰/۲۵۵ ثانیه طول بکشد تا صدا برگردد ، فاصله بین سنسور اولتراسونیک و جعبه به صورت فرمول ۴-۷ محاسبه خواهد شد :

$$D = 0.5 \times 0.025 \times 343 \approx 4.2875\ meter \qquad (۷-۴)$$

سنسورهای اولتراسونیک با یک دستگاه ارتعاشی معروف به مبدل، پالس های فراصوتی را منتشر می کنند که در یک پرتو مخروطی شکل حرکت می کنند و باعث تولید موج فراصوت می شوند. فرکانس ارتعاش تولیدی، محدوده سنسور اولتراسونیک را تعیین می کند. اولتراسونیک برد کوتاه در فرکانس های بالاتر بهترین عملکرد را دارند ، در حالی که سنسورهای اولتراسونیک دوربرد در فرکانس های پایین بهتر کار می کنند.



شکل ۱۹-۷ نحوه عملکرد سنسور التراسونیک

فرستنده ها و گیرنده ها در این سنسورها بسیار اهمیت دارند و لازم به ذکر است که در این سنسورها به طور معمول از فرستنده و گیرنده هایی استفاده می‌شود که با تبدیل انرژی الکتریکی به صوتی امواج صوتی بالاتر از ۲۰ کیلو هرتز ایجاد می کنند و با دریافت بازتاب امواج صوت را بار دیگر به انرژی الکتریکی تبدیل می‌کنند تا قابل اندازه‌گیری و نمایش باشند.

سنسور التراسونیک چهار پایه دارد که وظایف مختلفی دارند.

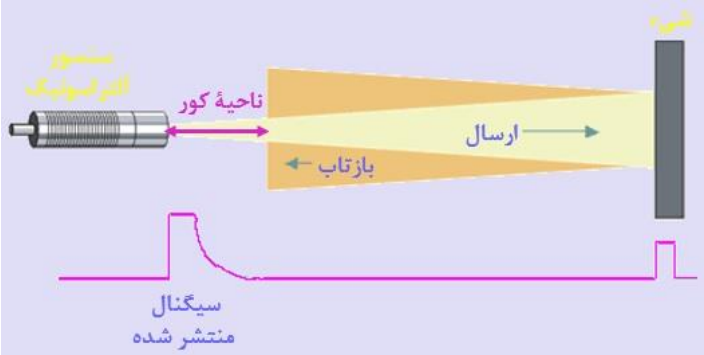
شکل ۲۰-۷ اجزا و پین‌های سنسور آلتراسونیک

Ultrasonic Sensor Pins Description	
Pin Name	**Pin Description**
Vcc	Voltage supply (5V)
Trig	Trigger pulse input pin
Echo	Echo pulse output pin
GND	Ground (0V)

شکل ۲۱-۷ پین های سنسور التراسونیک و توضیحات هر یک

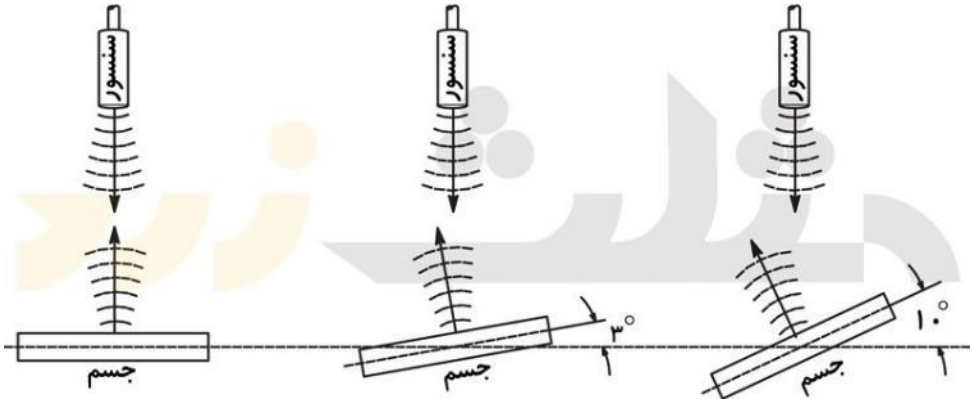
نکته بسیار مهمی که وجود دارد این است که هر ماده ای که در جهان ما وجود دارد به یک کیفیت خاص امواج التراسونیک را از خود عبور و مقداری از آن را باز تابش می دهد. بنابراین برای دست یابی به موارد ذکر شده نوع ماده نیز از اهمیت بسیاری برخوردار می باشد. به صورت کلی در اجسامی که انعکاس پذیری سطح آن ها پایین بوده یا ارتفاع پستی بلندی های سطح شان بزرگ تر از طول موج امواج صوتی التراسونیک باشد دقت اندازه گیری تا حدودی کاهش می‌یابد.

همچنین سنسور التراسونیک دارای یک ناحیه Blind zone می‌باشد. ناحیه کور ناحیه ای غیرقابل استفاده می باشد. اگر چرخه تشخیص قبل از اینکه سنسور انتقال خود را کامل کند تمام شود، سنسور نمی تواند اکو را به طور دقیق دریافت کند[۳۱۴].



شکل ۲۲-۷ ناحیه عملکرد و طرز عملکرد سنسور التراسونیک

هر سنسور صوتی دارای خصوصیات منحصر به فرد بوده و باید متناسب با شرایط ماشین‌آلات، نوع کاربری، محل نصب و ... انتخاب شود. برای انتخاب سنسور صوتی مناسب باید به کاتالوگ شرکت سازنده نیز مراجعه شود. در کاتالوگ‌ها اطلاعات مهمی در مورد محل نصب، زاویه‌ی نصب، رطوبت، وجود قطرات آب، رسوب‌های سخت، تنظیم و سیم‌بندی آن‌ها درج شده است. رعایت نکردن این موارد می‌تواند باعث عدم عملکرد صحیح سنسور شود. با توجه به تصویر زیر مشاهده می‌کنید که سنسور مافوق صوت باید عمود بر جسم مورد کنترل باشد چرا که اختلاف زاویه باعث عدم عملکرد سنسور خواهد شد[۳۱۵].



شکل ۲۳-۷ زاویه های متفاوت قرارگیری سنسور التراسونیک در برابر جسم و تفاوت عملکرد

از کاربردهای این سنسورها می‌توان به موارد زیر اشاره کرد:

۱. اندازه‌گیری فاصله
۲. سنسورهای رباتیک
۳. خودروهای هوشمند
٤. هواپیماهای بدون سرنشین (پهپاد)

۵. کنترل سطح مایع
٦. تشخیص افراد برای شمارش
۷. تشخیص حضور

مزایای سنسور التراسونیک عبارتند از:

۱. تحت تأثیر رنگ و شفافیت جسم قرار نمی‌گیرد، زیرا فاصله را از طریق امواج صوتی تشخیص می‌دهد.
۲. در مکان‌هایی که کم‌نور هستند، به خوبی کار می‌کند.
۳. مصرف جریان و برق کمتری دارد.
٤. چندین گزینه رابط برای اتصال با میکروکنترلر و غیره دارد.

چند مورد از معایب سنسور التراسونیک نیز به شرح زیر است:

۱. محدوده تشخیص محدود
۲. وضوح پایین و سرعت تازه‌سازی (Refresh) پایین آن که موجب می‌شود برای تشخیص اهدافی که سریع حرکت می‌کنند مناسب نباشد.
۳. عدم توانایی در اندازه‌گیری فاصله اشیایی که بافت و سطح پیچیده دارند.

۲-۴-۷- سنسورهای التراسونیک در شبیه سازی:

در شبیه ساز استفاده شده، سنسور التراسونیک نقش بسیار مهمی را ایفا می کند. در واقع سه سنسور التراسونیک جلوی ماشین قرار داده شده است. یک سنسور وسط و دو سنسور دیگر در طرفین و با زاویه ی ۳۰ درجه قرار گرفته اند که زاویه ی سنسورهای طرفین قابل تغییر می‌باشد. رنج تشخیص این سنسور ها ۱۵۰۰ سانتی متر می باشد و این بدان معناست که تا زمانی که جسمی فاصله ی بیشتر از ۱۵۰۰ سانتی متر از هر یک از التراسونیک ها دارد، تشخیص داده نمی شوند.

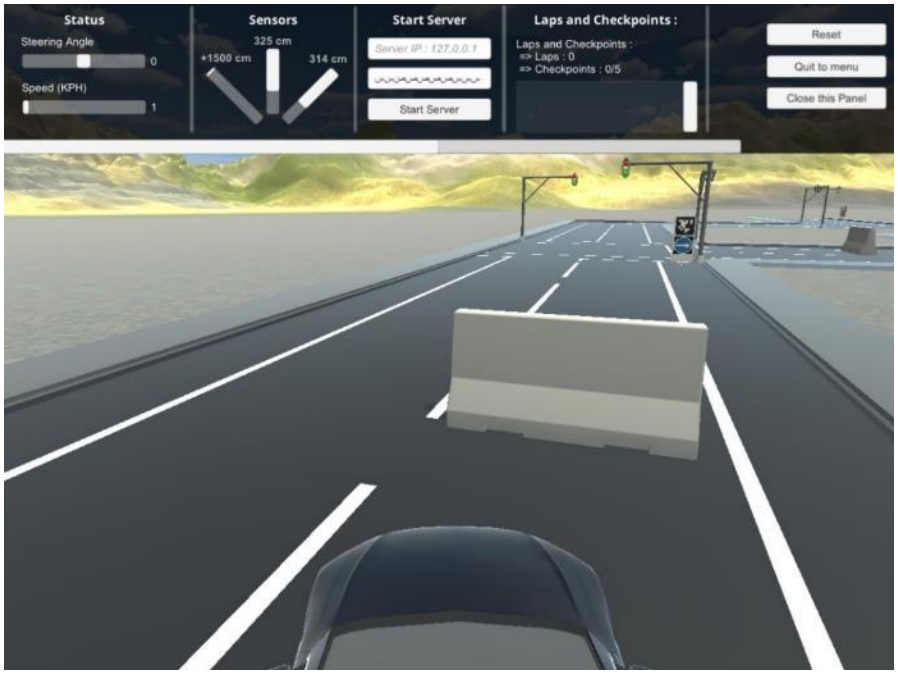
شکل ۲۴-۷ محیط شبیه سازی و بخش سنسورهای التراسونیک

در شکل بالا خروجی سنسورهای شبیه سازی شده را مشاهده می‌کنیم و در این فریم، مانع بیش از ۱۵۰۰ سانتی متر از سنسورها فاصله دارد و در نتیجه هنوز تشخیص داده نشده است.

شکل ۲۵-۷ عملکرد سنسور التراسونیک وسط در محیط شبیه سازی

حال در تصویر بالا، ماشین به جلو حرکت کرده و به مانع نزدیک شده است. در این حالت، مشاهده می کنیم که سینسور التراسونیک وسط، به ما فاصله ی ۵۷۴ سانتی متر را نشان می دهد. در واقع فاصله کمتر از ۱۵۰۰ سانتی متر است و امواج به مانع برخورد می‌کنند و با توجه به محاسبه زمانی برگشت امواج، فاصله محاسبه می‌شود.

شکل ۲۶-۷ عملکرد سنسور التراسونیک وسط و چپ در محیط شبیه سازی



شکل ۲۷-۷ عملکرد سنسور التراسونیک وسط و راست در محیط شبیه سازی

همچنین در تصاویر بالا مشاهده می‌کنیم که با زاویه گرفتن خودرو، هم سنسور وسط و هم سنسورهای طرفین وجود مانع را تشخیص داده و فاصله‌ی آن را اندازه می‌گیرند.

در کد شبیه ساز، سنسور در تابع getData موجود در کلاس car جست و جو شده و در متغیر sensorTagCheck ریخته می‌شود.

```
def getData(self):
    self.image_mode = 1
    self.sensor_status = 1
    self.updateData()
    self.sock.sendall(self.data_str.encode("utf-8"))
    recive = self.sock.recv(131072).decode("utf-8")
    imageTagCheck = re.search('<image>(.*?)<\/image>', recive)
    sensorTagCheck = re.search('<sensor>(.*?)<\/sensor>', recive)
    speedTagCheck = re.search('<speed>(.*?)<\/speed>', recive)
```

در ادامه مقدار تشخیص داده شده‌ی سنسورها در متغیر sensors ریخته می‌شود. همچنین تعیین شده است که اگر سنسورها مانعی تشخیص نداده باشند، مقادیر نشان داده شده‌ی هر سه سنسور ۱۵۰۰ سانتی متر باشند. که در شکل ۲۴-۷ این مورد را در عمل مشاهده می‌کنیم.

```
if(sensorTagCheck):
   sensorData = sensorTagCheck.group(1)
   sensor_arr = re.findall("\d+", sensorData)
   sensor_int_arr = list(map(int, sensor_arr))
   self.sensors = sensor_int_arr
else:
   self.sensors = [1500,1500,1500]
```

در انتهای کد شبیه ساز، تابع getSensors تعریف شده است که مقدار سنسورها را بر می‌گردانیم و خروجی می‌دهیم.

```
def getSensors(self):
     return self.sensors
```

در کدهای بخش‌های شهری و بزرگراه، مقدار سنسور ها به صورت کد زیر خوانده می‌شود.

```
sensors = car.getSensors()
```

و بسته به هدف های کنترلی از جمله نخوردن به موانع، دور زدن موانع و غیره، از مقادیر این سنسورها در طول حرکت خودرو استفاده می‌شود.

۷-۵- سیستم عدم برخورد با مانع و ترمز اضطراری

یکی از موارد مهم در خودرو خودران تشخیص موانع می‌باشد که از برخورد خودرو نسبت به آن جلوگیری می‌کند. با توجه به شبیه ساز AvisEngine و ثابت بودن شرایط محیطی و رنگ‌های موجود در این شبیه‌ساز بهترین رویکرد برای تشخیص موانع استفاده از پردازش تصویر کلاسیک می‌باشد که به واسطه رنگ مشخص شی می‌توان با دقت خوبی خطوط آن را تشخیص داد. همانطور که در شکل ۷-۲۷ آورده شده است، نمایی از جاده به همراه مانع قابل مشاهده توسط خودرو آورده شده است:

شکل ۷-۲۷ نمایی از جاده

برای تشخیص یک مانع در تصویر جاده شهری کد مقابل اجرا می‌شود:

```
1.  if sensors[1] < 700:
2.    ret = utils.stop_the_car(car)
3.    time.sleep(3)
4.    if side_pix > 128:
5.        utils.turn_the_car(car,-100,5.5)
6.        utils.turn_the_car(car,100,6.5)
7.        utils.turn_the_car(car,-100,2.5)
8.    else:
9.        utils.turn_the_car(car,100,4)
```

برای ایجاد سیستم تشخیص و دور زدن مانع با توجه به کد بالا چند گام می‌بایست طی شود.

۱-۵-۷- گام اول: مقادیر سنسور آلتراسونیک

اگر مقدار سنسور آلتراسونیک که در وسط قرار دارد کمتر از ۷۰۰ باشد، آنگاه مانع تشخیص داده شده و عملیات دور زدن مانع شروع می‌شود.

۲-۵-۷- گام دوم: ایست پشت مانع

بعد از تشخیص مانع در کمتر از مقدار ۷۰۰ نسبت به خود، خودرو توسط تابع stop_the_car سرعت آن صفر شده و سپس به مدت سه ثانیه پشت مانع می‌ایستد.

تابع stop_the_car به صورت مقابل می‌باشد که در آن به سیستم سرعت خودرو تا زمان ایستادن، سرعت معکوس ۱۰۰ وارد می‌شود:

```
1.  def stop_the_car(car):
2.      car.setSteering(0)
3.      while car.getSpeed():
4.          car.setSpeed(-100)
5.          car.getData()
6.      car.setSpeed(0)
7.      return True
```

۳-۵-۷- گام سوم: شرط side_pix را بررسی کن

اگر مقدار side_pix که بیانگر .... می‌باشد بیشتر از مقدار رفرنس ۱۲۸ باشد، آنگاه گام‌های بعد را اجرا کن.

در غیر این صورت خودرو را با زاویه فرمان ۱۰۰ درجه به مدت ۴ ثانیه بچرخان.

۴-۵-۷- گام چهارم: سیستم دور زدن مانع

ابتدا خودرو به مدت زمان ۵.۵ ثانیه زاویه فرمان را به اندازه ۱۰۰- درجه می‌چرخاند. به عبارتی دیگر فرمان با زاویه ۱۰۰ درجه به سمت چپ چرخانده می‌شود.
سپس فرمان به مدت زمان ۶.۵ ثانیه با زاویه فرمان ۱۰۰ درجه به سمت راست چرخانده می‌شود.
در نهایت به منظور صاف شدن خودرو در مسیر، فرمان به مدت ۲.۵ ثانیه با زاویه ۱۰۰- درجه چرخانده می‌شود.
با طی کردن مراحل فوق خودرو به درستی و با دقت مانع را دور زده و به مسیر خود باز میگردد.

۶-۷- شناسایی تابلوهای راهنمایی و رانندگی در محیط شبیه ساز

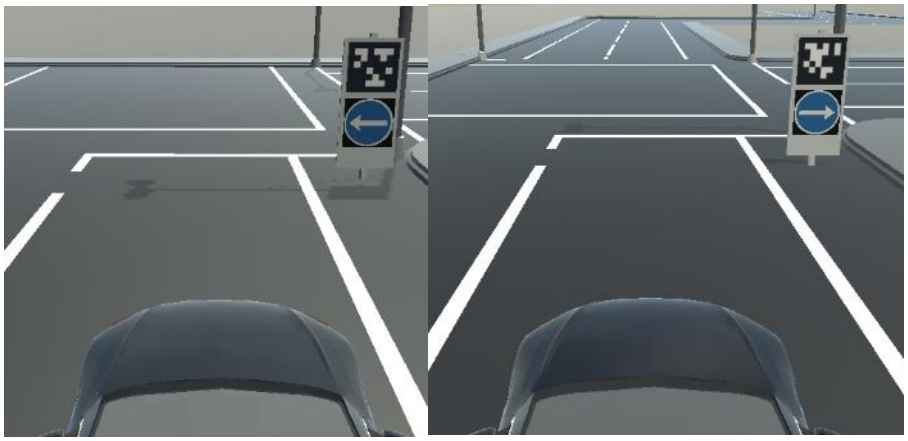
یکی از مراحل اصلی در رانندگی شهری، تشخیص تابلوهای راهنمایی و رانندگی در سطح شهر در محیط شبیه‌ساز می‌باشد. با استفاده از این تابلوها، خودرو متوجه می‌شود که برای دریافت حداکثر امتیاز نیاز است چه مسیری را طی کند. در این بخش، به بررسی تابلوهای راهنمایی و رانندگی موجود در شبیه‌ساز پرداخته می‌شود، همچنین نحوه‌ی شناسایی آن‌ها نیز به طور کامل شرح داده خواهد شد.

۱-۶-۷- تابلوهای راهنمایی و رانندگی موجود در شبیه ساز

در سطح شهر تابلوهای مختلفی اعم از محدودیت‌های سرعت، نشانگرهای محیط شهری و غیرشهری و ... وجود دارد. با توجه به اهمیت مسیریابی در شبیه‌ساز و موجود نبودن محدودیت سرعت، تابلوهای راهنمایی و رانندگی در دو کلاس "جهت دهی" و "توقف" می‌باشد.

در کلاس جهت‌دهی، ۳ نوع تابلوی مختلف وجود دارد: ۱. گردش به راست، ۲. گردش به چپ و ۳. ادامه‌ی مسیر مستقیم.

شایان ذکر است که برای هرکدام از تابلوها، علاوه بر شکل اصلی تابلو، یک اپریل‌تگ[۲۹۴] نیز وجود دارد که برای راحت تر بودن شناسایی تابلو است. لکن، در این بخش از تصویر اصلی تابلو برای شناسایی استفاده می‌شود و اپریل‌تگ مورد بررسی قرار نمی‌گیرد. نمونه‌ی تصاویر از تابلوها در ادامه مشاهده می‌شود.

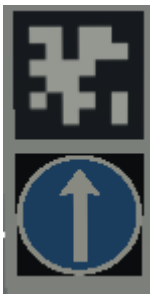


شکل ۲۸-۷ دیده شدن تابلوی گردش به راست و چپ توسط خودرو در محیط شبیه ساز

[294] ArilTag

شکل ۲۹-۷ تابلوی گردش به راست

شکل ۳۰-۷ تابلوی گردش به چپ



شکل ۳۱-۷ تابلوی ادامه به صورت مستقیم

شکل ۳۲-۷ تابلوی توقف

۲-۶-۷- نحوه‌ی انجام شناسایی

برای شناسایی تابلوها ابتدا با استفاده از رنگ تابلوها، اصل تابلو شناسایی می‌شود و سپس وارد یک مدل کلسیفایر می‌شود و در نهایت نوع تابلو مشخص می‌شود. برای تابلوهای توقف، با توجه به این که تنها یک نوع تابلو و آن هم با رنگ قرمز موجود است، در صورت شناسایی شدن دیگر نیازی به کلسیفایر نمی‌باشد. برای انجام شناسایی بر اساس رنگ، تصویر دریافت شده از شبیه‌ساز به فضای HSV برده می‌شود.

۱-۲-۶-۷- شناسایی تابلوهای جهت‌دهی

در مرحله‌ی اول ابتدا فریم دریافتی که به فضای HSV برده شده است وارد تابع `detect_sign` می‌شود، و نوع تابلو خروجی گرفته می‌شود:

```
sign = utils.detect_sign(frame, hsv_frame)
```

با توجه به این که رنگ محیط اصلی تابلو به ضورت آبی می‌باشد، یک بازه‌برای این رنگ فیلتر می‌شود و خروجی در داخل یک متغیر به اسم ماسک قرار می‌گیرد. این بازه برای تابلوهای آبی به صورت زیر می‌باشد.

```
def detect_sign(frame, hsv_frame):
    types = ['left', 'straight', 'right']
mask = cv2.inRange(hsv_frame, np.array([100,160,90]), np.array([160,220,220]))
```

در ماسک به دست آمده، ابتدا کانتورهای موجود در ماسک پیدا شده و سپس بر اساس اندازه مرتب می‌شوند.

```
points, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
sorted_points = sorted(points, key=len)
```

و سپس با توجه به این که در هر لحظه حداکثر یک تابلوی آبی‌رنگ وجود دارد، بزرگترین کانتور به عنوان کانتور تابلو در نظر گرفته می‌شود. شایان ذکر است که که این عملیات در داخل یک بلوک try-except قرار می‌گیرد زیرا در همه حال تابلو وجود ندارد.

خروجی این ماسک برای شکل ۲۸-۷ به صورت شکل ۳۳-۷ می‌باشد.

شکل ۳۳-۷ تشخیص محیط اصلی تابلو و ماسک به دست آمده

در نهایت با استفاده کانتور به دست‌آمده به یک باندیگ باکس محیط می‌شود و به صورت یک تصویر ۲۵ پیکسل در ۲۵ پیکسل تغییر اندازه داده می‌شود. در انتها نیز این تصویر وارد یک کلسیفایر از پیش آموزش‌داده‌شده می-شود و کلاس مرتبط با تابلوی مذکور از بین ۳ کلاس موجود تعیین می‌شود و به برنامه برمی‌گردد.

```
x,y,w,h = cv2.boundingRect(sorted_points[-1])
if (x>5) and (x+w<251) and (y>5) and (y+h<251):
    sign = frame[y:y+h,x:x+w]
    sign = cv2.resize(sign, (25,25))/255
    frame = cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
    return types[np.argmax(sign_model.predict(sign.reshape(1,25,25,3)))]
```

۲-۲-۶-۷- شناسایی تابلوی توقف

همانند شناسایی تابلوهای آبی‌رنگ جهت‌دهی، ابتدا یک فیلتر مناسب برای تابلوی قرمزرنگ پیدا می‌شود و ماسک مرتبط با آن فیلتر به دست می‌آید.

```
red_mask = cv2.inRange(hsv_frame, np.array([140,70,0]),
np.array([255,255,255])) * (1-car_mask)
```

سپس با استفاده از یک تابع، تشخیص داده می‌شود که ماسک به دست آمده حاوی تابلوی توقف می‌باید یا خیر. برای این کار، ابتدا کانتورهای موجود در ماسک به دست آمده و به ترتیب اندازه مرتب می‌شوند.

```
def red_sign_state(red_mask):
    points, _ = cv2.findContours(red_mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    sorted_points = sorted(points, key=len)
```

سپس در صورتی که ابعاد بزرگترین کانتور (در صورتی که کانتوری وجود داشته باشد) بزرگتر از ۵۰ پیکسل باشد به این معنی است که یک تابلوی توقف در تصویر موجود می‌باشد و این مورد با چاپ شدن عبارت

“red sign detected”

به کاربر اعلام می‌شود، همچنین مقدار True به برنامه بازگردانده می‌شود و در نهایت سیاست توقف خودرو اعمال می‌شود.

```
    try:
        red_area = cv2.contourArea(sorted_points[-1])
        if red_area > 50:
            print('red sign detected!')
            return True
        else:
            return False
    except:
        return False
```

۳-۲-۶-۷- کلسیفایر استفاده شده برای تشخیص تابلوهای جهت دهی

پس از آن‌که اصل تابلوی آبی‌رنگ پیدا شد، می‌بایست وارد یک کلسیفایر شود تا کلاس آن تشخیص داده شود. این عملیات به این صورت انجام‌گردد که ابتدا مدل کلسیفایر که به صورت یک شبکه‌ی عصبی عمیق می‌باشد در برنامه فراخوانی می‌شود و سپس تابلوهای یافت شده وارد کلسیفایر شده و طبقه‌بندی انجام‌می‌شود.

```
from tensorflow.keras.models import load_model
sign_model = load_model('best_model.h5')
```

نام مدل از پیش‌آموزه داده شده best_model.h5 می‌باشد که با آموزش یک شبکه‌ی عصبی عمیق به دست آمده است. ساختار این مدل نیز به صورت شکل ۳۴-۷ می‌باشد.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_52 (Conv2D)           (None, 25, 25, 32)        896
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 12, 12, 64)        18496
_________________________________________________________________
flatten_26 (Flatten)         (None, 9216)              0
_________________________________________________________________
dropout_26 (Dropout)         (None, 9216)              0
_________________________________________________________________
dense_37 (Dense)             (None, 3)                 27651
=================================================================
Total params: 47,043
Trainable params: 47,043
Non-trainable params: 0
_________________________________________________________________
```

شکل ۳۴-۷ ساختار شبکه‌ی عصبی استفاده شده برای یافتن کلاس هر تابلو

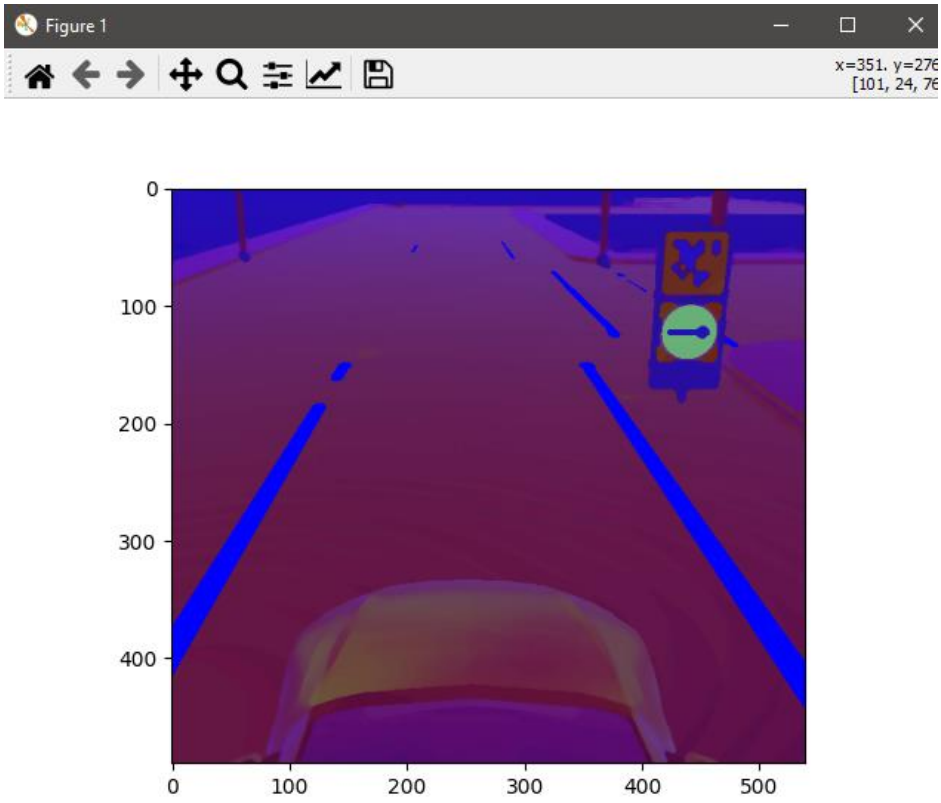
همان طور که در شکل مشاهده می‌شود، شبکه‌ی مذکور دارای دولایه‌ی کانولوشنی و دو لایه‌ی خطی با فعالسازهای ReLU و sigmoid می‌باشد. در مجموع شبکه-ی مورد استفاده دارای ۴۷۰۴۳ پارامتر می‌باشد.

برای آموزش شبکه نیز از تصاویر جمع‌آوری شده‌ی تابلوها استفاده شد.

۳-۶-۷- نحوه‌ی پیدا کردن ماسک

برای پیدا کردن ماسک رنگی هر کدام از قسمت‌های مورد نیاز، لازم است که تصویر به فضای HSV برده شود و با استفاده از اسکریپت‌های color_detection.py و hsvshow.py ماسک ها به‌دست می‌آیند.

اسکریپت hsvshow.py تصویر را به فضای HSV برده و سپس با استفاده از کتابخانه‌ی matplotlib نمایش میدهد. با حرکت دادن نشانگر ماوس بر روی قسمتی که میخواهیم ماسک آن را به دست آوریم، بازه‌های مورد نیاز در فضای HSV را به‌دست می‌آوریم.



شکل ۳۵-۷ خروجی اسکریپت hsvshow.py

پس از یافتن تقریبی مقادیر مورد نیاز، برای تصویر مورد نظر کد color_detection.py اجرا می‌شود که در آن با استفاده از دکمه‌های کشویی مقادیر کمنیه و بیشینه برای کانال‌های HSV تنظیم می‌شود. پس از آن نیز به صورت دستی تنظیم می‌شود.

ابتدا کتابخانه‌های مورد نیاز وارد برنامه می‌شود:

```
import cv2
import numpy as np
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('--image', type = str, default = 'sshot-1.png' , help =
'Choose the name of the image you want.')
args = parser.parse_args()
```

سپس با استفاده از کتابخانه‌ی OpenCV یک منو با دکمه‌های کشویی ایجاد می‌شود.

```
def nothing(x):
    pass

term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)
trackwindow = (0,0,240,320)

cap = cv2.VideoCapture(0)

cv2.namedWindow('trackbars')
cv2.createTrackbar('high H : ', 'trackbars', 0, 255, nothing)
cv2.createTrackbar('high S : ', 'trackbars', 0, 255, nothing)
cv2.createTrackbar('high V : ', 'trackbars', 0, 255, nothing)
cv2.createTrackbar('low H : ', 'trackbars', 0, 255, nothing)
cv2.createTrackbar('low S : ', 'trackbars', 0, 255, nothing)
cv2.createTrackbar('low V : ', 'trackbars', 0, 255, nothing)
```

سپس در یک حلقه، هر بار مقادیر مورد نیاز برای پارامترهای بهینه و بیشینه‌ی هر سه کانال فضای HSV تنظیم می‌شود.

```
while True:
    high_h = cv2.getTrackbarPos('high H : ', 'trackbars')
    high_s = cv2.getTrackbarPos('high S : ', 'trackbars')
    high_v = cv2.getTrackbarPos('high V : ', 'trackbars')
    low_h = cv2.getTrackbarPos('low H : ', 'trackbars')
    low_s = cv2.getTrackbarPos('low S : ', 'trackbars')
    low_v = cv2.getTrackbarPos('low V : ', 'trackbars')

    upper = np.array([high_h, high_s, high_v])
    lower = np.array([low_h, low_s, low_v])
```

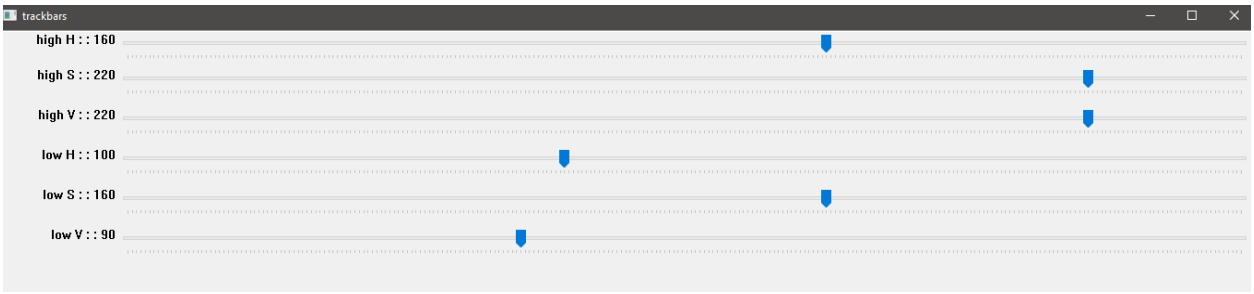
پس از این مرحله، تصویر خوانده شده و به فضای HSV برده می‌شود، سپس قسمت‌های با مقدار کوچکتر از مقادیر کمینه و همچنین مقادیر بیشتر از مقادیر بیشینه صفر می‌شوند و تنها مقادیری که در این بازه قرار دارد باقی می‌ماند.

```
    if 'npy' in args.image:
        frame = np.load(args.image)
    else:
        frame = cv2.imread(args.image)
    hsv_frame = frame
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    hsv_frame = cv2.medianBlur(hsv_frame, 7)
    # hsv_frame = cv2.medianBlur(hsv_frame, 5)
    # hsv_frame = cv2.blur(hsv_frame,(6,6))
    mask = cv2.inRange(hsv_frame, lower, upper)
    # mask = cv2.GaussianBlur(mask , (7,7), 0)
    mask = cv2.medianBlur(mask , 5)
    result = cv2.bitwise_and(frame, frame, mask = mask)
```

پس از آن، کانتورهای موجود در داخل ماسک یافت شده و نتیجه نشان داده می‌شود.

```
    points,_ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(frame, points, -1, (0,0,255), 2)

    # plotting ROI
    result = cv2.rectangle(result,(0,140),(512,190),(0,255,255),1)
    what_to_show = np.vstack((frame, result))
    cv2.imshow('mask', mask)
    cv2.imshow('frame + result + mask', what_to_show)
    key = cv2.waitKey(1)
    if key == ord('q'):
        break
    if key == ord('w'):
        cv2.imwrite('./frame2.jpg', mask)
```

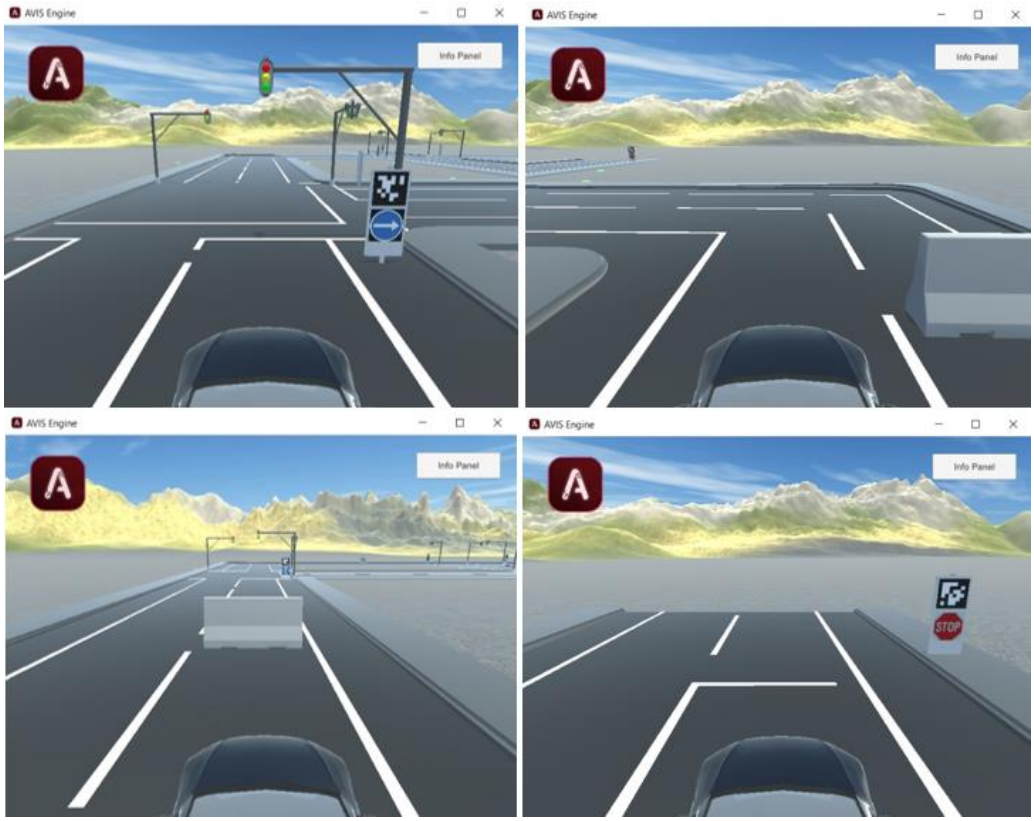
شکل دکمه‌های کشویی در زیر مشاهده می‌شود.



شکل ۳۶-۷ دکمه‌های کشویی موجود در برنامه color_detection.py

۴-۶-۷- سیستم تصمیم‌گیری در محیط شهری

یکی از مهم ترین زیرسیستم های یک خودروی خودران بخش تصمیم گیری آن است. یک خودروی خودران زمانی که در ی شرایطی قرار می گیرد، ابتدا لازم درک کاملی از آن محیط به دست بیاورد. محل جاده، تابلوها، عابران پیاده و سایر خودروها را مشخص کند. پس درک محیط و به دست آوردن اطلاعات در مورد اشیاء موجود در آن، لازم است تصمیم درست انتخاب شود تا متناسب با آن عمل مناسب اجرا شود. اهمیت زیرسیستم تصمیم گیری در آن است که تمامی اقداماتی که در ادامه توسط خودرو انجام می شود بر اساس تصمیم اتخاذ شد، می-باشد. محیط شبیه سازی Avis نیز به گونه ای طراحی شده است تا بتواند پیچیدگی های یک محیط شهری واقعی را به خوبی مدل کند و تصمیم گیری خودرو را به چالش بکشد.



شکل ۳۷-۷ محیط شهری شبیه ساز Avis شامل موانع، چهارراه، پیچ و انواع تابلو های گردش به چپ، گردش به راست و مستقیم

اطلاعات دریافتی از بخش ادراک خودرو شامل محل خطوط سفید جاده، محل خطوط افقی، اطلاعات مربوط به حاشیه جاده، وجود یا عدم وجود موانع و درنهایت وضعیت تابلوهای موجود در چهارراه می باشد. خطوط سفید جاده با استفاده از ماسک رنگ سفید و استخراج خطوط عمودی به دست آمده و همین مراحل برای استخراج

خطوط افقی جاده نیز تکرار می شود. استخراج حاشیه جاده و تشخیص محل خودرو نیز بر اساس رنگ آن صورت می گیرد. سه سنسور آلتراسونیک موجود در جلوی خودرو برای تشخیص وجود موانع به کار رفته و تابلوها نیز برمبنای شبکه عصبی کانولوشنی که توضیح داده شده است، شناسایی می‌شوند. در نهایت تمامی این اطلاعات به زیرسیستم تصمیم گیری برای انتخاب اقدام مناسب داده می شود. دستورات زیر درجهت استخراج این اطلاعات استفاده می‌شوند.

```
# getting data
car.getData()
sensors = car.getSensors()
frame = car.getImage()
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
white_mask = cv2.inRange(frame, np.array([240,240,240]), np.array([255,255,255])) *
(1-car_mask)
side_mask = cv2.inRange(frame, np.array([130,0,108]), np.array([160,160,200])) * (1-
car_mask)
red_mask = cv2.inRange(hsv_frame, np.array([140,70,0]), np.array([255,255,255])) *
(1-car_mask)

# vertical lines
lines = utils.detect_lines(utils.region_of_interest(white_mask))
two_line_mask, CURRENT_PXL = utils.mean_lines(white_mask, lines)

# white horiz line
horiz_detected = utils.horiz_lines(white_mask)

# only on turns and obstacle
mean_pix = utils.turn_where(white_mask)
side_pix = utils.detect_side(side_mask)

# detecting sign type
red_sign = utils.red_sign_state(red_mask)
sign = utils.detect_sign(frame, hsv_frame)
if sign == 'left':
    sign_state = 'left'
elif sign == 'straight':
    sign_state = 'straight'
elif sign == 'right':
  sign_state = 'right'
```

قبل از شرح زیرسیستم تصمیم گیری، به توضیح ۳ تابع که عمل های خودرو را مشخص می کنند، می‌پردازیم.

هدف تابع stop_the_car متوقف سازی خودرو قبل از چهارراه و پیچ می‌باشد. این تابع به این صورت عمل می کند که سرعت خودرو را در حلقه while قرار داده و تا زمانی که مثبت است، به خودرو شتاب منفی وارد می کند.

```
def stop_the_car(car):
    car.setSteering(0)
```

```
    while car.getSpeed():
        car.setSpeed(-100)
        car.getData()
    car.setSpeed(0)
    return True
```

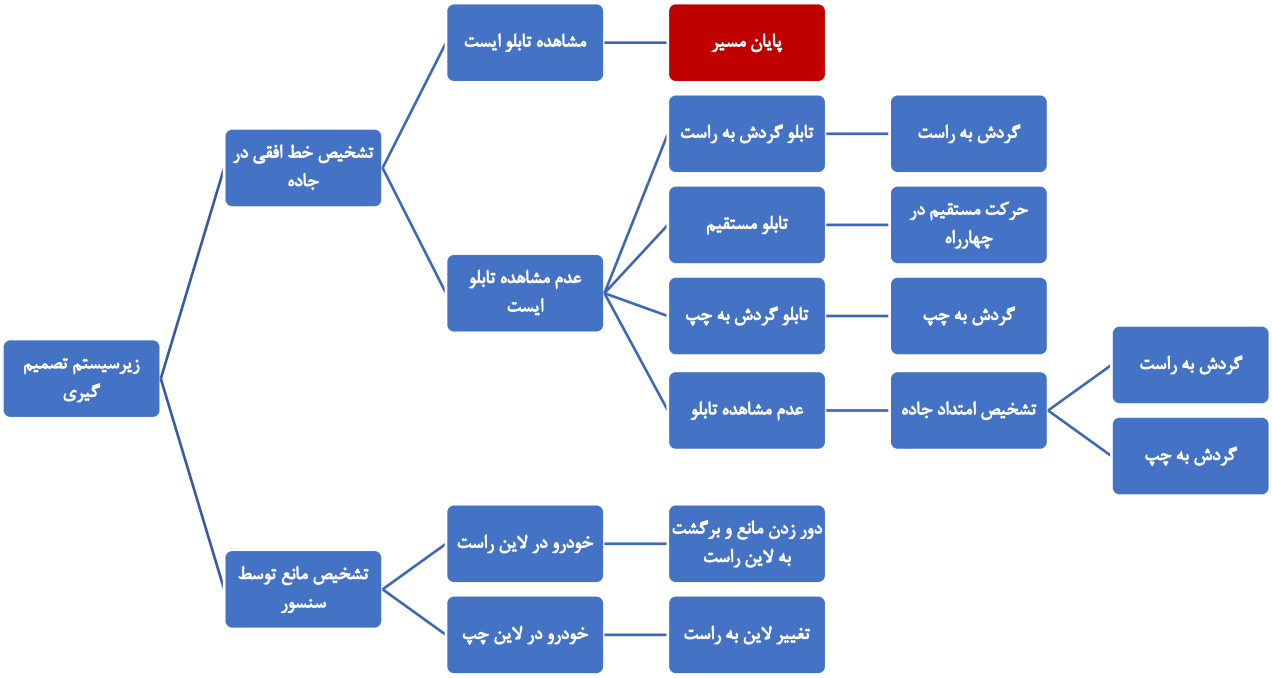
هدف تابع turn_the_car گردش خودرو در چهارراه و پیچ می باشد و عملکرد آن متناسب با زاویه فرمان مورد نظر و زمان تعیین شده است.

```
def turn_the_car(car,s,t):
    time1 = time.time()
    while((time.time()-time1)<t):
        car.getData()
        car.setSteering(s)
        car.setSpeed(15)
```

یکی از مشکلات گردش خودرو در پیچ کمبود فضای کافی برای دور زدن می باشد. برای حل این مشکل تابع go_back جهت برگشت خودرو استفاده می شود. این تابع به این صورت عمل می کند که زمان لازم برای حرکت به عقب به خودرو داده شده و خودرو جهت تعمین فضای کافی به عقب برمی‌گردد.

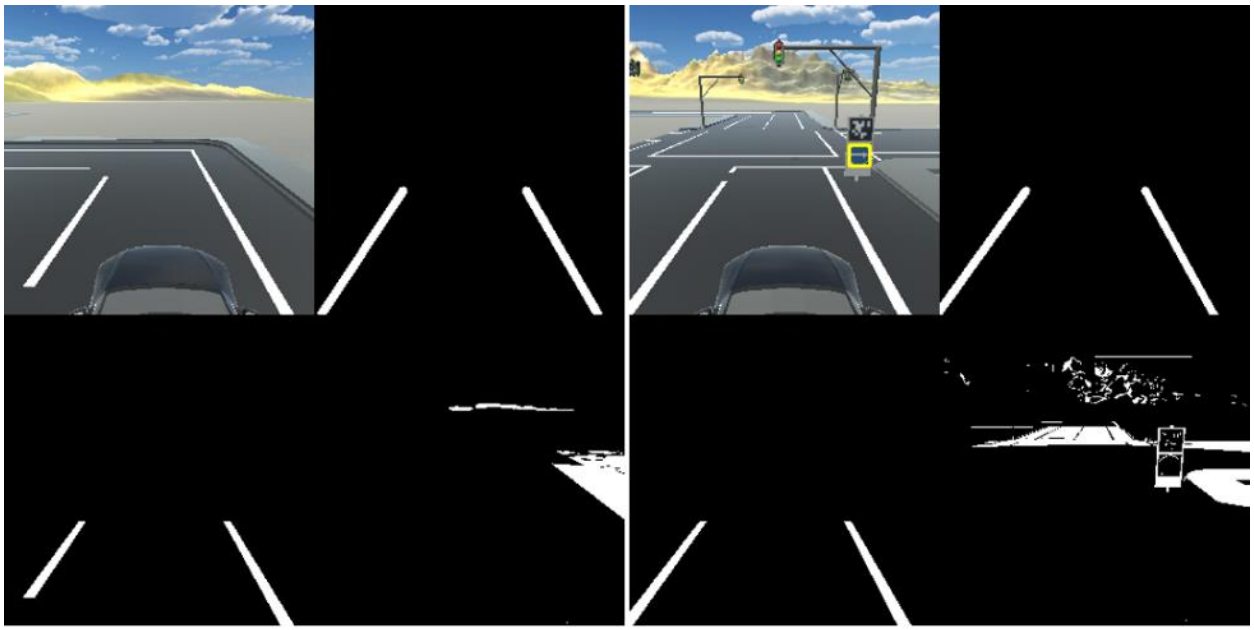
```
def go_back(car, t):
    time1 = time.time()
    while((time.time()-time1)<t):
        car.getData()
        car.setSpeed(-15)
  car.setSpeed(۰)
```

در زیرسیستم تصمیم گیری خودرو، ۲ سناریو بر مبنای شبیه ساز در نظر گرفته شده است. سناریو اول زمانی است که خودرو خطی افقی در جاده مشاهده کند که معنی این است که خودرو در چهارراه یا پیچ قرار گرفته است. سناریو دوم نیز در زمان مشاهده مانع توسط سنسورهای خودرو اتفاق می‌افتد.



شکل ۳۸-۷ سناریو

سناریوی اول خود شامل چندین حالت مختلف می باشد که به صورت درختی به حالت‌های جزئی تر تقسیم می‌شود. پس از تشخیص خط افقی لازم است ابتدا بررسی شود که خودرو به در انتهای مسیر قرار دارد یا خیر. انتهای مسیر با تابلوی stop در کنار خط افقی سفید مشخص شده است و تنها لازم است که خودرو وجود تابلو stop را بررسی کند. در صورت عدم مشاهده تابلو stop خودرو باید تعیین کند که در چهارراه یا پیچ قرار دارد. نکته تمایز چهارراه و پیچ جاده در وجود یا عدم وجود تابلو می باشد. در صورتی که تابلویی در حافظه خودرو (sign_state) ثبت شده باشد خودرو تصمیم خود را برمبنای چهارراه می‌گیرد و درغیر این صورت تصمیم بر اساس پیچ جاده گرفته می شود.



شکل ۳۹-۷ شکل بالا بینایی خودرو را در محیط شهری نشان می‌دهد. دید اصلی خودرو در بالاسمت راست و ماسک حاشیه جاده در ماسک جاده در پایین چپ قرار دارد. شکل سمت راست خودرو را در چهار راه نشان می-دهد. تابلو با باندینگ باکس زرد مشخص شده است همچنین ماسک حاشیه جاده نشان می‌دهد خودرو در سمت راست قرار دارد. شکل سمت چپ نیز بینایی خودرو در پیچ جاده نشان می دهد. همان‌طور که مشاهده می شود در پیچ جاده تابلویی وجود ندارد و تصمیم گیری بر اساس ادامه خطوط سفید جاده صورت می گیرد.

در چهارراه، متاسب با تابلوی مشاهده شده (چپ، راست و یا مستقیم) و همچنین لاینی که خودرو در آن قرار گرفته است (side_pix)، خودرو به آن جهت می پیچد و مسیر را دنبال می‌کند. فرآیند گردش در پیچ جاده نیز به همین صورت است، با این تفاوت که خودرو لازم است برای تعمین فضای مورد نیاز جهت گردش مقدار به عقب برگردد، همچنین جهت چرخش خودرو بر اساس ادامه لاین جاده (mean_pix) انتخاب می‌شود. به این صورت که اگر ادامه خطوط سفید جاده در سمت راست مجتمع شده باشد، خودرو به راست پیچیده و در غیر این صورت مسیر چپ را انتخاب می کند.

```
    if horiz_detected:
        car.setSteering(0)
        print(sign_state)
        ret = utils.stop_the_car(car)
        time.sleep(3)

        if not red_sign:
```

```
            if sign_state == 'nothing':
                # turn based on mean_pix
                mean_pix = utils.turn_where(white_mask)
                print('mean_pix :', mean_pix)

                if mean_pix < 128:
                    if side_pix > 128:
                        utils.go_back(car,4.5)
                        utils.turn_the_car(car,-100,10)
                    else:
                        utils.turn_the_car(car,-80,8)
                else:
                    if side_pix < 128:
                        utils.go_back(car,8)
                    else:
                        utils.go_back(car,4.5)
                    utils.turn_the_car(car,100,10)

            elif sign_state == 'left':
                if side_pix > 128:
                    utils.turn_the_car(car,-45,13)
                else:
                    utils.turn_the_car(car,-50,12)
                sign_state == 'nothing'

            elif sign_state == 'straight':
                utils.turn_the_car(car,0,11)
                sign_state == 'nothing'

            elif sign_state == 'right':
                if side_pix > 128:
                    utils.turn_the_car(car,65,9.5)
                else:
                    utils.turn_the_car(car,70,11)

                sign_state == 'nothing'

            sign_state = 'nothing'

        else:
            print('red sign detected. stopping the car ...')
    break
```

در سناریوی دوم با مشاهده مانعی در کمتر از ۷ متری خودرو، خودرو ابتدا متوقف می شود. سپس براساس ماسک حاشیه جاده و میانگین گیری از آن (side_pix) تعیین می کند که در سمت راست یا چپ جاده قرار دارد. با مشخص شدن محل دقیق خودرو تصمیم مناسب برای گذر از مانع اتخاذ می شود. در صورتی که مانع در سمت

راست قرار داشته باشد، خودرو ابتدا به لاین چپ رفته و سپس به لاین راست بازمی‌گردد اما در صورتی که مانع در سمت چپ قرار داشته باشد، خودرو تنها لاین خود را به راست تغییر می دهد.

```
    if sensors[1] < 700:
       ret = utils.stop_the_car(car)
       print('side_pix :', side_pix)
       time.sleep(3)
       if side_pix > 128:
           utils.turn_the_car(car,-100,5.5)
           utils.turn_the_car(car,+100,6.5)
           utils.turn_the_car(car,-100,2.5)
       else:
    utils.turn_the_car(car,+100,4)
```

۵-۶-۷- سیستم کنترل موقعیت خودرو در حالت شهری

در این حالت از حرکت خودرو، برای تعیین موقعیت و همچنین ادامه‌ی مسیر خودرو در مسیر مورد نظر که از پیش با توجه به سیسات حرکت خودرو تعیین شده است، نیاز به طراحی یک کنترل‌کننده می‌باشد. کنترل‌کننده‌ی مورد استفاده در این بخش از نوع تناسبی در حوزه‌ی گسسته بوده و به صورت فرمول‌های ۵-۷ و ۶-۷ می‌باشد.

$$e(t) = REF - Position \quad (۷-۵)$$

$$Control\ Signal = K_p \times e(k) \quad (۷-۶)$$

برای پیاده‌سازی این مورد نیز، ابتدا میزان خطای سیستم در پله‌ی مورد بررسی ابتدا محاسبه می‌شود. سپس خطا در مقدار ضریب تناسبی کنترل‌کننده ضرب شده و به عملگر داده می‌شود.

```
error = REFRENCE - CURRENT_PXL
steer = -(kp * error)
car.setSteering(int(steer))
```

در این حالت خودرو با استفاده از این کنترل‌کننده می‌تواند در مسیر مورد نظر خود به‌درستی حرکت نماید و به‌سوی مقصد پیشروی کند.

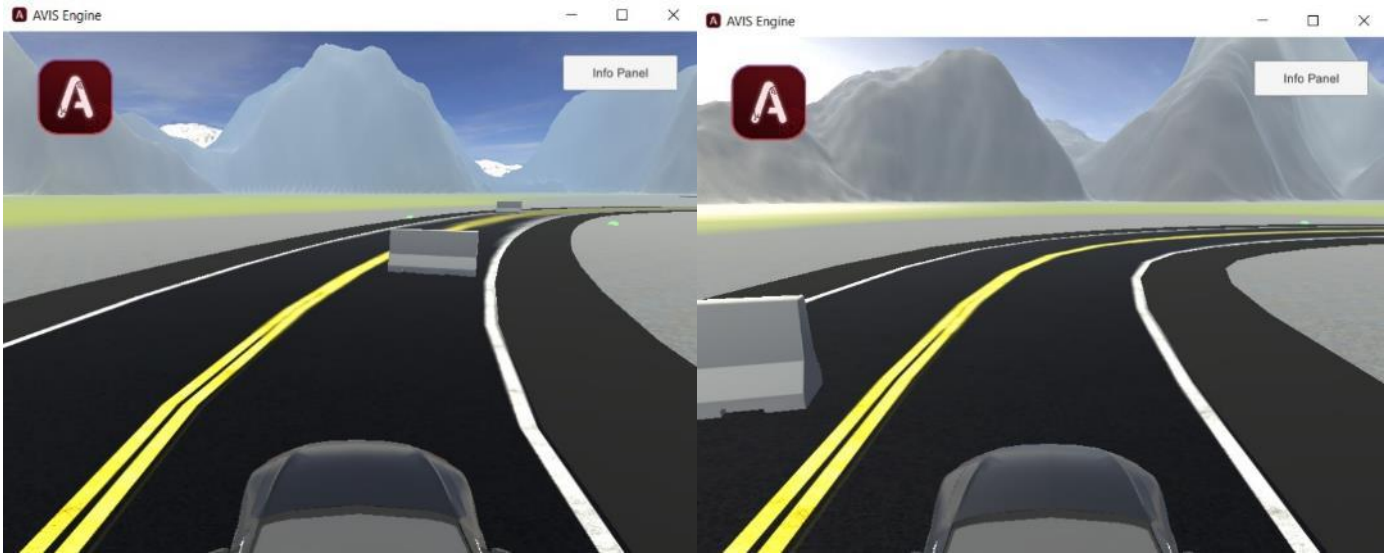
شکل ۴۰-۷ عملکرد کنترل‌کننده در حالت شهری

۸- شبیه سازی بین شهری

در این فصل، به بررسی نحوه‌ی شبیه‌سازی خودرو خودران در یک محیط بین‌شهری و یا بزرگراه در محیط شبیه‌ساز یونیتی پرداخته می‌شود.

۸-۱- تشخیص لاین در بخش بزرگراه

یکی از مهم‌ترین بخش های رانندگی خودران در بزرگراه، تشخیص وضعیت و مکان خودرو در جاده است. این مسئله از این رو اهمیت دارد که مرکز جاده، رفرنس مناسب برای کنترل زاویه فرمان را مشخص می کند. شبیه ساز استفاده شده دارای انواع موانع در دو لاین جاده بوده و شامل پیچ های تند و پشت سرم هم می باشد. همچنین مهم است که در بزرگراه خودرو همواره در لاین راست حرکت کند و تنها جهت دور زدن مانع، تغییر لاین دهد. در شکل زیر محیط شبیه ساز نشان داده شده است.



شکل ۸-۱ محیط بزرگراه شبیه ساز، شامل موانع در جاده، پیچ های تند و اهمیت حرکت در لاین راست نشان داده شده است

روش های متعددی از جمله استفاده از خطوط، مرکز و حتی حاشیه‌های جاده برای تعیین وضعیت خودرو وجود دارد. در بخش بزرگراه روش متفاوتی نسبت به روش اشاره شده در بخش شهری استفاده شده است. مبنای این روش، استفاده از تفاوت رنگ جاده با سایر محیط بوده و طبق مراحل زیر، دو لاین جاده به صورت مجزا برای تعیین رفرنس مورد نظر استفاده می شود.

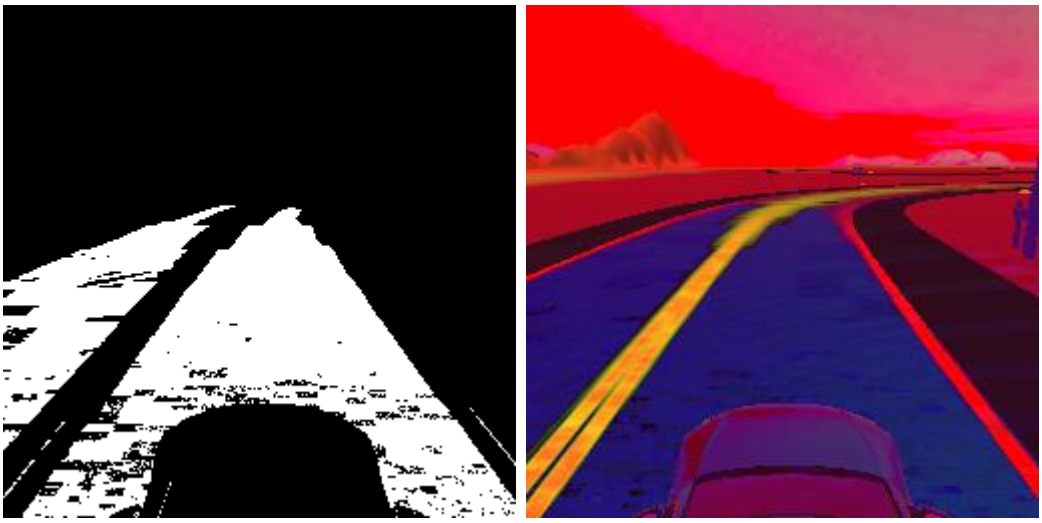
در اولین گام، با استفاده از دوربین تعبیه شده بر روی خودرو، یک فریم از شبیه ساز دریافت می شود. فریم دریافت شده در فرمت رنگی RGB بوده و به همین دلیل تفکیک رنگ ها در آن دشوار است. برای حل این مشکل ابتدا فریم دریافتی به فرمت رنگی HSV تبدیل می شود تا تفاوت رنگ ها به خوبی مشخص شود.

```
car.getData()
frame = car.getImage()

hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

شکل ۲-۸ تصویر خام دریافتی از دوربین با فرمت رنگی RGB در سمت راست و تصویر تبدیل شده به فرمت HSV در سمت چپ نشان داده شده است

با اعمال محدوده بندی مناسب بر روی تصویر، ماسک کلی جاده استخراج می شود. دستور inrange با بررسی مقادیر تصویر، مقدار ۱ را برای پیکسل هایی که در محدوده انتخابی قرار می‌گیرند، برمی‌گرداند. بدین ترتیب ماسک باینری بخش مورد نظر به دست می آید.

```
mask = cv2.inRange(hsv_frame, np.array([100,10,25]), np.array([120,50,60]))
```



شکل ۳-۸ تصویر محیط با فرمت رنگی HSV در سمت راست و ماسک باینری جاده در سمت چپ نشان داده شده است.

به دلیل در نظر گرفتن شرایط نوری مختلف در شبیه ساز، محدوده رنگ مورد نظر، در زوایای مختلف دچار تغییر می شود. لذا ممکن است ماسک به دست آمده، نویزی باشد و یا تعدادی از داده ها از بین برود. برای حل این مشکل، ۲ مرحله Erosion و Dilation به ترتیب بر روی تصویر اعمال می شود.

Erosion، با استفاده از فیلتر تعییر شده مقادیر را به مقدار مینیمم درون فیلتر جایگزین می کند، به این ترتیب اگر در محدوده فیلتر نویزی وجود داشته باشد اثر آن از بین می رود. این مرحله ممکن است باعث از بین رفتن داده های اصلی ماسک نیز شود. لذا در گام بعد Dilation بر روی تصویر اعمال می شود. Dilation عملکردی معکوس با Erosion دارد و مقدار ماکزیمم درون فیتر را نگه داشته و آن را جایگزین می کند. با این روش نویزهای ناخواسته ماسک حذف شده و سپس مقادیر داده های اصلی ماسک بزرگتر می شوند. نتایج مرحله به مرحله استخراج ماسک جاده مناسب، در تصویر زیر نشان داده شده است.

(ج) (ب) (الف)

شکل ۴-۸ تصویر (الف) ماسک خام جاده را نشان می دهد. همان طور که مشاهده می شود نویز کوچکی در حاشیه جاده قرار دارد. (ب) خروجی ماسک را پس از عمل Erosion نشان می‌دهد. (ج) خروجی نهایی ماسک را پس از Dilation نشان می دهد.

دستور پایتون این بخش نیز به شرح زیر است.

```
kernel = np.ones((2,2),np.uint8)
mask = cv2.erode(mask, kernel, iterations=2)
kernel = np.ones((3,3),np.uint8)
mask = cv2.dilate(mask, kernel, iterations=2)
```

برای تمرکز بر روی جاده، قسمتی از ماسک به دست آمده به عنوان محدوده مطلوب (Region of Interest) انتخاب می شود تا از تاثیر قسمت های اضافی ماسک در محاسبه رفرنس جلوگیری شود.

در آخرین گام، با بدست آوردن قسمت های سفید با بزرگترین مساحت، می توان ۲ لاین جاده را بدست آورد. می‌دانیم که از دید دوربین، بزرگترین مساحت مربوط به لاینی است که خودرو در آن قرار داشته و مساحت دوم لاین دیگر را نشان می دهد. این فرآیند با استفاده از الگوریتم تشخیص کانتورها پیاده سازی شده است. کانتور را می توان به عنوان یک محنی بسته در نظر گرفتن که نقاط با شدت و رنگ یکسان را به یکدیگر متصل می کند. با اعمال این الگوریتم بر روی ماسک جاده، ۲ کانتور شناسایی می شود. با مرتب سازی این کانتورها بر اساس

(ب) (الف)

شکل ۵-۸ در سمت راست ماسک لاین اصلی با بزرگترین مساحت و در سمت چپ ماسک لاین دیگر قرار گرفته است. مرکز ناحیه سفید در هر کدام از ماسک ها، رفرنس آن را مشخص می کند.

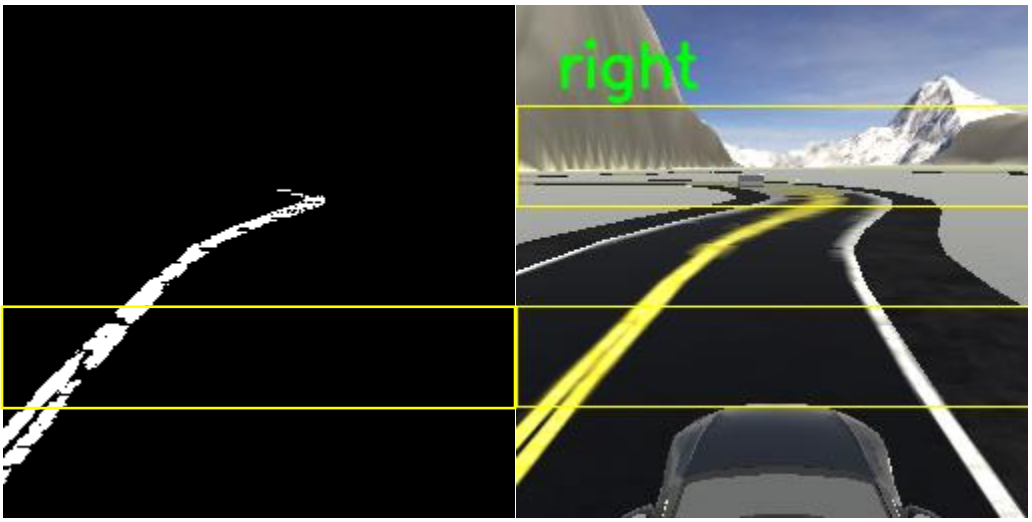
مساحت، ۲ لاین جاده استخراج می شود. مرکز هر کانتور نیز رفرنس نهایی جاده را مشخص می کند که با CURRENT_PXL به عنوان رفرنس لاین اصلی و SECOND_PXL به عنوان لاین ثانویه مشخص شده است.

دستور پایتون این بخش نیز به شرح زیر است.

```
lane_contours, _ = cv2.findContours(mask[130:200,:], cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
sorted_lanes = sorted(lane_contours, key=cv2.contourArea, reverse=True)

if len(sorted_lanes) > 1:
        right_lane_mask = cv2.drawContours(np.zeros((70,256)), sorted_lanes, 0, 255,
-1)
        left_lane_mask = cv2.drawContours(np.zeros((70,256)), sorted_lanes, 1, 255,
-1)

elif len(sorted_lanes) == 1:
        right_lane_mask = cv2.drawContours(np.zeros((70,256)), sorted_lanes, 0, 255,
-1)

CURRENT_PXL = np.mean(np.where(right_lane_mask>0), axis=1)[1]
SECOND_PXL = np.mean(np.where(left_lane_mask>0), axis=1)[1]
CURRENT_PXL = np.nan_to_num(CURRENT_PXL, nan = 128)
SECOND_PXL = np.nan_to_num(SECOND_PXL, nan = 128)
```

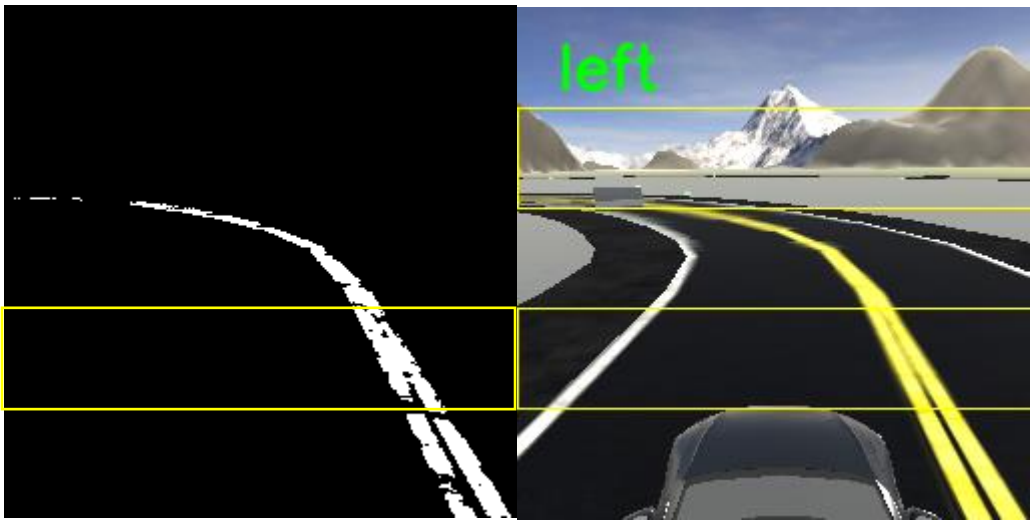
۸-۲- تشخیص خط زرد و تعیین محل خودرو

تعیین لاین فعلی خودرو براساس خط زرد جاده صورت می‌گیرد. به دلیل ادامه دار بودن خط زرد در پیچ جاده، محدوده دید محدودی در ماسک خط زرد انتخاب شده است که تصمیم گیری بر اساس آن انجام شود.



شکل ۸-۶ خط زرد وسط خیابان در سمت چپ

در تصویر زیر محل خودرو در لاین چپ جاده تشخیص داده شده است، لذا تمام تصمیم گیری های بعدی بر مبنای این موقیت یابی صورت می‌گیرد.



شکل ۸-۷ خط زرد وسط خیابان در سمت راست

در ادامه دستور این بخش در پایتون قرار داده شده است.

```
yellow_mask = cv2.inRange(hsv_frame, np.array([28,115,154]), np.array([31,180,255]))
YELLOW_PXL = np.mean(np.where(yellow_mask[140:190,:]>0), axis=1)[1]
YELLOW_PXL = np.nan_to_num(YELLOW_PXL, nan = 128)

if YELLOW_PXL>128:
      position = 'left'
else:
   position = 'right'
```

۳-۸- سنسورهای مورد استفاده

نحوه ی کار و ماهیت سنسور های استفاده شده در بخش بین شهری دقیقا مطابق با بخش شهری می باشد، اما در الگوریتم مورد استفاده برای استفاده از این سنسور ها در بخش بین شهری تفاوت هایی با بخش شهری وجود دارد.

در تردد های شهری، معمولا سرعت کم می باشد و تمرکز روی روی عدم وقوع تصادفات و رعایت قوانین راهنمایی و رانندگی و عمل به دستورات تابلو هاست، اما در تردد های بین شهری، تابلوهای دستوری وجود ندارد و سرعت رانندگی بسیار بالاتر از بخش شهری است، بنابراین کنترل خودرو در سرعت بالا و دریافت پیوسته ی دیتا از سنسور های تعبیه شده اهمیت بالایی پیدا می کند.

پس از استفاده از الگوریتم استفاده شده در بخش شهری برای قسمت بین شهری، معضلاتی از جمله لرزش خودرو، انحراف خودرو به طرفین و بهینه نبودن عملکرد کنترلر پیاده شده روی خودرو به وجود آمد که پس از بررسی های تخصصی، مشخص شد مشکل اصلی این است که در دریافت داده از سنسور های اولتراسونیک، پرش وجود دارد که می تواند به دلایل متعددی از جمله وجود نویز، خطای سنسور و بحث عدم تطابق زمانی یا ریل تایم نبودن سیستم انتقال دیتا به وجود بیاید.

با توجه به متکی بودن ضرایب کنترلر PID طراحی شده به دیتای دریافتی از سنسور ها، این پرش لحظه ای دیتا در سرعت بالای خودرو در قسمت بین شهری باعث لرزش خودرو و انحراف آن به طرفین می گشت.

روش پیاده سازی شده برای جلوگیری از این مشکل، پیاده سازی دیجیتالی سنسور حافظه دار بود، به این معنی که آرایه ای از ده دیتای آخر دریافت شده توسط سنسور به وجود آید و از میانگین این ده دیتا در کنترلر استفاده شود.

```
1.  sensors_array = np.round(where_avg * (np.array(sensors)) + (1-where_avg) * sensors_array, 1)
```

به این ترتیب، خلا وجود دیتا در بخش بین شهری با تکنیک سنسور حافظه دار یا همان میانگین گیری از چندین دیتای آخر دریافتی از سنسور، مرتفع گشت و مشکل لرزش خودرو در سرعت بالای بخش بین شهری حل گردید.

۸-۴- سیستم عدم برخورد با مانع در بزرگراه

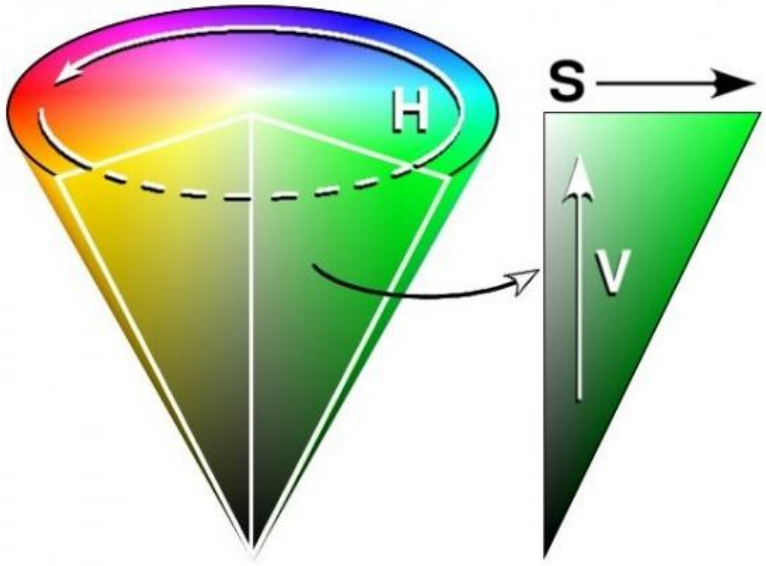
یکی از موارد مهم خودروهای خودران توانایی سبقت در بزرگراه از موانع می‌باشد. خودرو می‌بایست ابتدا مانع را به درستی تشخیص دهد و سپس بر اساس آن تصمیم بگیرد که چگونه نسبت به آن مانع عمل کند. با توجه به شبیه ساز مورد استفاده شده AvisEngine و ثابت بودن شرایط محیطی و رنگ‌های موجود در این شبیه‌ساز، بهترین رویکرد برای تشخیص موانع استفاده از پردازش تصویر کلاسیک می‌باشد که به واسطه رنگ مشخص شی می‌توان با دقت خوبی خطوط آن را تشخیص داد. نمایی از جاده بزرگراه به همراه مانع در مقابل خودرو در تصویر ۸-۸ آورده شده است.

شکل ۸-۸ جاده بزرگراه به همراه مانع در مقابل آن

برای تشخیص یک مانع و تصمیم‌گیری به منظور عکس‌العمل در مقابل آن در چهار گام به صورت مقابل مشخص شده است.

۱-۴-۸- گام اول: تبدیل تصویر از BGR به HSV

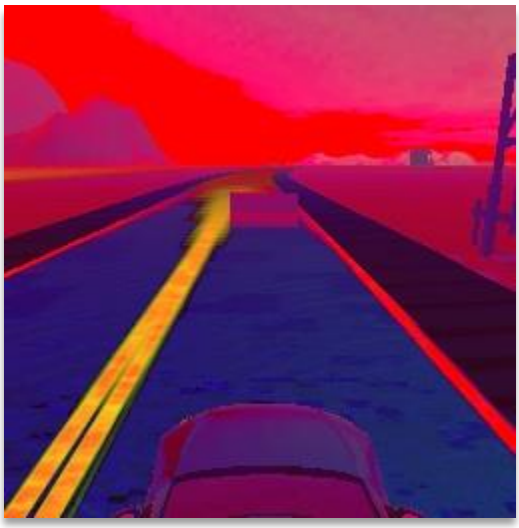
ابتدا تصویر به دست آمده می‌بایست به فضای HSV برده شود. در HSV از فضای رنگی مختلف استفاده می‌شود و این امکان را می‌دهد تا فضای رنگی دلخواه را به راحتی از طیف دیگر رنگ ها جدا کرد. فرض کنید که شما قصد دارید رنگ سبز را در تصویر فیلتر نمایید این بازه شامل طیفی می باشد که یک سمت آن سبز تیره و در سمت دیگر آن سبز روشن می باشد برای جدا کردن آن در فضای رنگی RGB این امکان وجود ندارد که شما بتوان به صورت خطی یعنی هر کانال با یک شرط بازه رنگ دلخواه را انتخاب نمائید. پس به خاطر چنین مشکلاتی تصویر را به فضای رنگی HSV انتقال داده می‌شود که این فضا از اجزای رنگدانه[۲۹۵]، اشباع[۲۹۶] و روشنایی[۲۹۷] تشکیل شده است. برای تفکیک رنگ سبز در این فضای رنگی کافیست محدوده Hue خود که مربوط به رنگ مورد نظر را انتخاب کرده و سپس کل محدوده اشباع و در نهایت انتخاب محدوده دلخواه برای روشنایی، رنگ دلخواه را مشخص کرد.



شکل ۹-۸ جداسازی رنگ سبز دلخواه با استفاده از فضای رنگی HSV

[295] Hue
[296] Saturation
[297] Value

از تابع cvtColor از کتابخانه OpenCv برای تبدیل تصویر RGB به HSV استفاده می‌شود. تصویری از تبدیل یافته تصویر RGB به HSV در شکل ۱۰-۸ آورده شده است.



شکل ۱۰-۸ نمایی از جاده در فضای HSV

۸-۴-۲ گام دوم: بدست آوردن ماسک مانع

اولین گام برای تشخیص مانع، تعیین بازه تغییرات رنگ RGB شی مورد نظر می‌باشد که خروجی آن تصویر ماسک[۲۹۸] است. با توجه به مشخص بودن رنگ خاکستری در شبیه ساز، به راحتی می‌توان با پیدا کردن بازه تغییرات پیکسلی، ماسک مناسبی از هر فریم ایجاد کرد. کد آن به صورت مقابل آورده شده است:

```
2.  obstacle_mask = cv2.inRange(Frame, Min, Max)
```

- **Frame**: تصویر ورودی به فرمت HSV
- **Min**: کمترین مقادیر آبی، سبز و قرمز (BGR) در تصویر

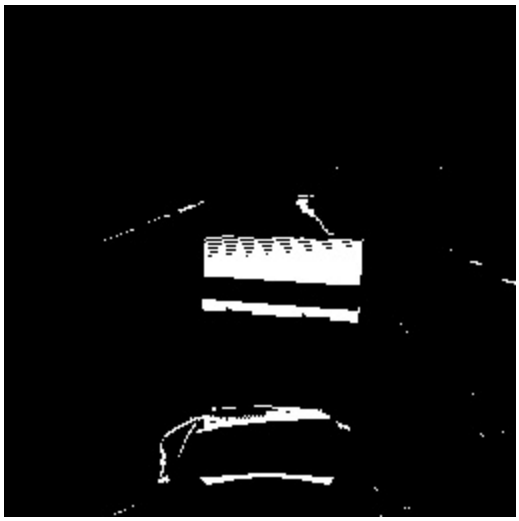
```
np.array([95,0,95]
```

- **Max**: بیشترین مقادیر آبی، سبز و قرمز (BGR) در تصویر

```
np.array([180,20,160])
```

- **خروجی:** ماسک ایجاد شده که در آن مانع با رنگ سفید مشخص شده‌اند. نمونه ای از خروجی در تصویر ۱۱-۸ آورده شده است:

[298] Mask



شکل ۱۱-۸ ماسک خطوط جاده

همانطور که در تصویر ۱۱-۸ مشاهده می‌شود علاوه بر ماسک مانع، به دلیل تشابه رنگی نویزهایی نیز در ماسک خروجی ایجاد شده است. به منظور تصمیم‌گیری دقیقتر برای رفتار در مقابل مانع، می‌بایست تا جای امکان تاثیرات نویزهای ایجاد شده را کم کرد.

۳-۴-۸ گام سوم: حذف نویز با عملیات مورفولوژی

عملیات مورفولوژیک به دو دست سایش (Erosion) و گسترش (Dialation) تقسیم می‌شود.

- **عملیات مورفولوژی سایش(Erosion)**

عملیات مورفولوژی سایش (Erosion) یکی از پایه ای ترین عملگرها در پردازش تصویر است. عملگر سایش، نواحی سفید رنگ تصویر باینری را تحلیل می‌کند، در واقع تصویر سفید را کوچک و قسمت های اضافی را حذف می‌کند. به عبارتی دیگر این عملیات شبیه به انقباض نواحی سفید عمل می‌کند.

سایش (Erosion) یکی از دو عملگر پایه در زمینه مورفولوژی ریاضی است. عملگر دوم گسترش (Dilation)است که معمولاً برای تصاویر باینری اعمال می شود. اما نسخه هایی هم وجود دارد که روی تصاویر مقیاس خاکستری (Grayscale) کار می کنند. تأثیر اصلی این عملگر بر روی یک تصویر باینری، از بین بردن مرزهای منطقه‌ها (به طور معمول پیکسل های سفید) از پیکسل های پیش زمینه (رنگ سیاه) است. بنابراین نواحی پیکسل پیش زمینه (سفید رنگ) از لحاظ اندازه، کوچک می‌شوند و سوراخ های مشکی درون مناطق سفید، بزرگتر می‌شوند.

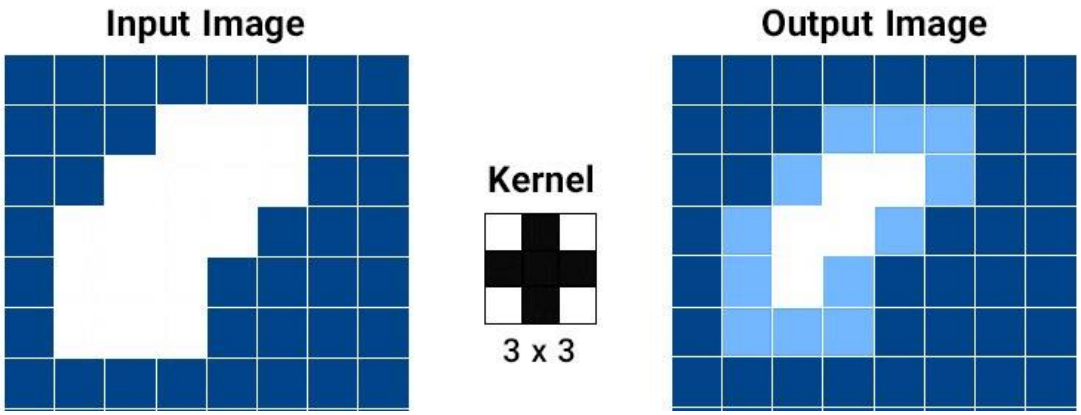
کد آن در کتابخانه Opencv به صورت مقابل می‌باشد.

```
1. kernel = np.ones((2,2),np.uint8)
2. mask = cv2.erode(mask, kernel, iterations=2)
```

اپراتور سایش سه نوع داده به عنوان ورودی می گیرد. اولی تصویری است که باید عملیات سایش روی آن انجام شود. مورد دوم مجموعه ای (معمولاً کوچک) از نقاط است که به عنوان یک عنصر ساختاری (Kernel) شناخته می شود. اثر دقیق سایش توسط عنصر ساختاری (Kernel) بر روی تصویر ورودی تعیین می‌شود. مورد سوم نیز نشان دهنده تعداد بارهای اعمال این عملگر بر روی تصویر باینری ورودی می‌باشد.

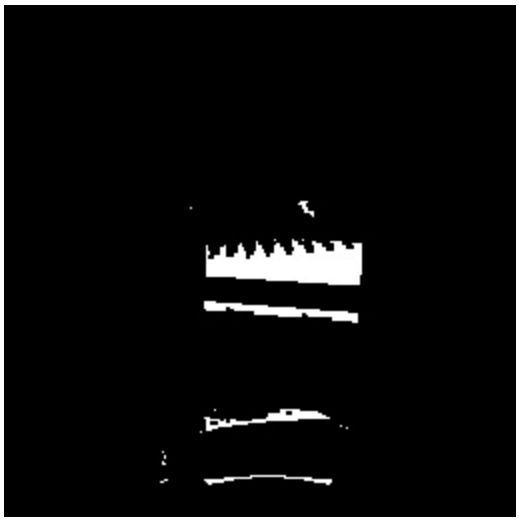
برای اعمال سایش، ماتریس کرنل (Kernel) را در مختصات ۰ و ۰ عکس قرار می دهیم. کرنل یک ماتریس است که تعداد سطر و ستون آن فرد است. اگر مرکز کرنل روی یک نقطه سفید رنگ (نقطه هدف) قرار بگیرد، پیکسل های اطراف آن ناحیه مورد بررسی قرار می‌گیرد. اگر پیکسلی در همسایگی نقطه هدف وجود داشته باشد که رنگ آن مشکی باشد، نقطه‌ی هدف به رنگ مشکی در می‌آید. این عملیات روی تمام نقاط تصویر اعمال می‌شود. درنهایت نقاط سفید رنگی که در مرز قرار دارند، دچار سایش می‌شوند و به رنگ سیاه تبدیل می‌شوند.

به عنوان مثال برای یک عنصر ساختاری ۳ × ۳، اثر سایش به این گونه است که هر پیکسل پیش زمینه که کاملاً توسط پیکسل های سفید دیگر احاطه نشده باشد (با فرض اتصال ۸ تایی) به رنگ سیاه تبدیل می‌شود. چنین پیکسل هایی باید در لبه های مناطق سفید قرار داشته باشند ، بنابراین نتیجه اصلی این است که مناطق پیش زمینه (مناطق سفید) کوچک می شوند و سوراخ های (نقاط مشکی) داخل یک منطقه سفید، رشد می کنند.



شکل ۱۲-۸ عملیات مورفولوژی سایش Erosion با کرنل ۳×۳

بعد از اعمال عملگر مورفولوژی سایش برای ماسک اولیه مانع، خروجی به صورت تصویر ۱۳-۸ می‌شود.



شکل ۱۳-۷ ماسک مانع بعد از اعمال عملگر سایش

همانطور که در تصویر ۱۳-۸ مشخص است، عملگر سایش توانسته به خوبی اکثر نویزهای که در ماسک تصویر وجود داشت حذف کند.

- **عملیات مورفولوژی گسترش (Dialation)**

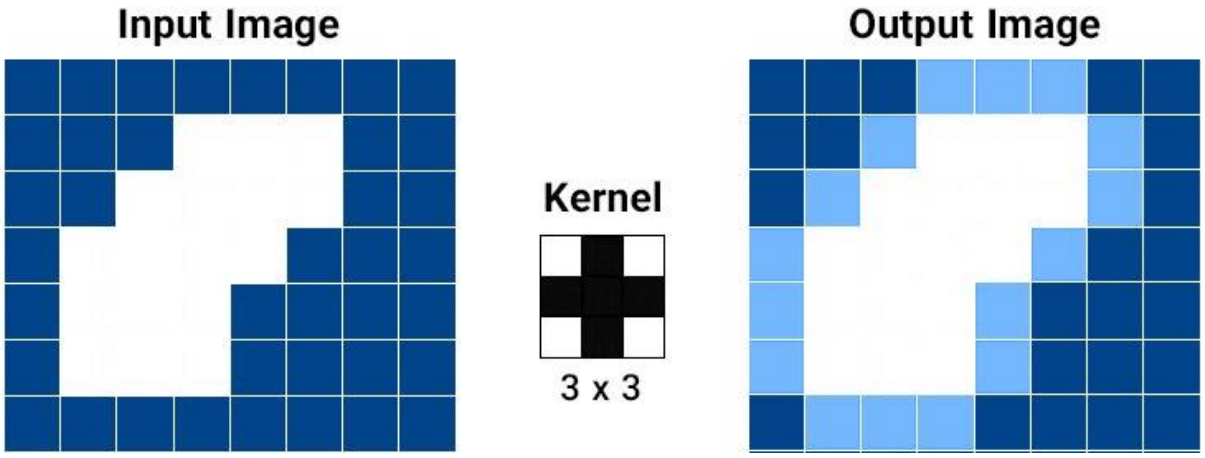
حال به منظر دقت و وضوح بهتر نسبت به ماسک مانع از عملگر گسترش استفاده می‌شود. این عملگر باعث بزرگ شدن یا گسترش مناطق سفید می‌شود. تأثیر اصلی این اپراتور بر روی یک تصویر باینری، بزرگ کردن مرزهای مناطق پیکسل پیش زمینه (به طور معمول پیکسل‌های سفید) است. بنابراین مناطقی از پیکسل های پیش زمینه (سفید رنگ) در اندازه رشد می کنند در حالی که سوراخ های داخل آن مناطق (سوراخ‌های مشکی) کوچکتر می شوند.

کد آن در کتابخانه Opencv به صورت مقابل می‌باشد.

```
1. kernel = np.ones((3,3),np.uint8)
2. mask = cv2.dilate(mask, kernel, iterations=2)
```

عملگر گسترش در پردازش تصویر سه نوع داده را به عنوان ورودی دریافت می‌کند. اولی همان تصویری است که باید فرایند گسترش روی آن انجام شود. مورد دوم مجموعه‌ای (معمولاً کوچک) است از نقاط، که به عنوان یک عنصر ساختاری (Kernel) شناخته می‌شود. اثر دقیق گسترش مناطق سفید، توسط عنصر ساختار (Kernel)بر روی تصویر ورودی تعیین می‌شود. مورد سوم نیز نشان دهنده تعداد بارهای اعمال این عملگر بر روی تصویر باینری ورودی می‌باشد.

برای اعمال گسرش، عنصر ساختاری (Kernel) را در موقعیت مکانی ۰ و ۰ عکس قرار می‌دهیم. کرنل یک ماتریس است که تعداد سطر و ستون آن باید فرد باشد. اگر مرکز کرنل روی یک نقطه سفید رنگ (نقطه هدف) قرار بگیرد پیکسل‌های اطراف آن ناحیه مورد بررسی قرار می‌گیرد. اگر پیکسل A در همسایگی نقطه هدف وجود داشته باشد و رنگ آن مشکی باشد، رنگ نقطه A به سفید تغییر رنگ می دهد. این عملیات روی تمام نقاط تصویر اعمال می‌شود. درنهایت نقاط سیاه رنگی که در مرز مناطق قرار دارند، دچار افزایش می شوند و به رنگ سفید تبدیل می‌شوند.



شکل ۱۴-۸ ماسک مانع بعد از اعمال عملگر سایش [۳۱۶]

عملیات delation باعث اتصال نواحی گسسته می‌شود و همچنین خوردگی‌های تصویر را پر می‌کند. این عملیات اندازه‌ی شیء را بزرگتر می‌کند

بعد از اعمال عملگر مورفولوژی گسترش بر روی خروجی حاصل شده از عملگر سایش، خروجی به صورت تصویر ۱۵-۸ می‌شود.

شکل ۱۵-۸ ماسک مانع بعد از اعمال عملگر گسترش

همانطور که در تصویر ۱۵-۸ مشخص است، عملگر گسترش توانسته به خوبی مانع را با وضوح و مقدار پیکسلی بیشتری نشان دهد.

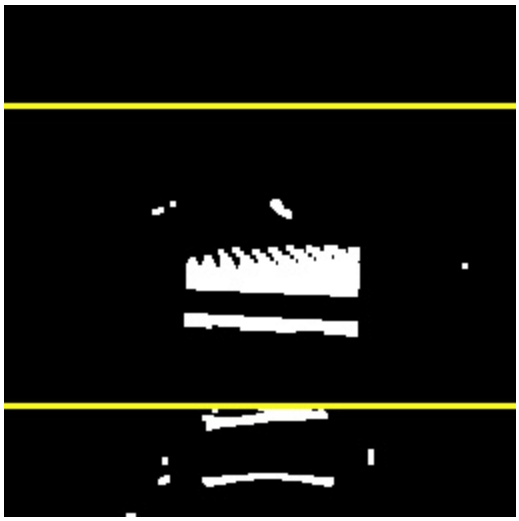
۸-۴-۴ گام چهارم: اعمال منطقه مورد نظر[۲۹۹] به خروجی گام سوم

به منظور تشخیص دقیقتر مانع و حذف نویز ایجاد شده به دلیل تشابه رنگی رنگ خودرو و مانع، می‌بایست با تعیین کردن محدوده‌ای عملیات تشخیص مانع را فقط در همان محدوده تعیین شده انجام داد. این کار به صورت مقابل انجام می‌شود:

```
1.  mask = mask[130: 200, :]
```

که منطقه مورد نظر با رنگ زرد در تصویر ۱۶-۸ آورده شده است.

[299] ROI



شکل ۱۶-۸ ماسک مانع و منطقه مورد نظر

۵-۴-۸ گام پنجم: استخراج کانتور (خطوط مرزی)

حال می‌بایست در منطقه محدود کننده مانع را تشخیص داد. خطوط مرزی، منحنی‌هایی از تصویر هستند که به یکدیگر متصل شده‌اند. منحنی‌ها نقاط پیوسته متصل در یک تصویر است و هدف کانتورها تشخیص اشیا است.

```
1.  points, _ = cv2.findContours(obstacle_mask[50:200,:],cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

همانطور که در تصویر ۱۶-۸ مشخص است، بعد از اعمال عملگرهای مورفولوژی باز هم اندکی نویز مشاهده می‌شود. خروجی کد بالا نقاط مرزی هر شی تشخیص داده شده می‌باشد. آنچه واضح است، شی ای که دارای بیشترین مساحت بر اساس نقطه مرزی باشد می‌توان به عنوان مانع استفاده کرد، به همین دلیل در مرحله بعد نقاط مرزی هر شی تشخیص داده شده بر اساس مساحت ترتیب بندی می‌شوند.

```
2.  sorted_area = sorted(points, key=cv2.contourArea, reverse=True)
```

با تعیین آستانه‌ای به اندازه ۱۰، اگر مساحت شی تشخیص داده شده بیشتر از مقدار تعریف شده بود، به دور شی مورد نظر مستطیلی توسط cv2.boundingRect رسم می‌شود که خروجی آن مختصات (X, Y) نقطه بالا سمت چپ مستطیل و اندازه عرض و طول آن می‌باشد. سپس نقطه میانی مانع توسط mean_obstacle تعریف می‌شود.

```
1.  if sorted_area > 10:
2.      x,y,w,h = cv2.boundingRect(sorted_points[-1])
3.      mean_obstacle = x + w//2
```

با توجه به بخش ۲-۸ که در آن مسیر حرکتی خودرو که در کدام لاین در حال رانندگی است تحت عنوان position مشخص می‌شود، می‌توان وضعیت خودرو نسبت به مانع را مشخص کرد.

اگر خودرو در سمت چپ خیابان قرار داشته باشد دو شرط بررسی خواهد شد، در غیر این صورت شرطی که به صورت پیش فرض در قسمت else آورده شده اجرا می‌شود. این مراحل با شرط‌های متفاوت برای سمت راست خیابان نیز برقرار است. کد آن به صورت مقابل است:

```
1.  if (position=='left'):
2.          if ((sensors_array[0]<1450 or sensors_array[1]<1450) and (mean_obstacle<obs_yellow)):
3.              error = (REFRENCE - SECOND_PXL)
4.              time1 = time.time()
5.              steer = -(kp * error)
6.              steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
7.              car.setSteering(int(steer_array))
8.
9.          elif((time.time()-time1)>0.8 and (min(sensors_array) > 1450)):
10.             error = (REFRENCE - SECOND_PXL)
11.             steer = -(kp * error)
12.             steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
13.             car.setSteering(int(steer_array))
14.
15.         else:
16.             error = REFRENCE - CURRENT_PXL
17.             steer = -(kp * error)
18.             steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
19.             car.setSteering(int(steer_array))
```

شرط اول: اگر سنسور آلتراسونیک سمت چپ یا وسط مقدار کمتر از ۱۴۵۰ نشان داد و همچنین مقدار میانگین خط زرد مرکزی جاده بیشتر از مقدار نقطه میانی مانع بود، آنگاه اجرا کن:

ابتدا مقدار خطا به صورت error = (REFRENCE - SECOND_PXL) مشخص می‌شود. در آن مقدار نقطه رفرنس که برابر ۱۲۸ می‌باشد از مقدار میانگین ماسک جاده سمت چپ کم خواهد شد و به عنوان خطا ثبت می‌شود. سپس با توجه به خطا بدست آمده مقدار فرمان با توجه به ضریب کنترل‌کننده تعیین می‌گردد. به منظور دقت بهتر و چرخش نرمتر فرمان این مقدار در steer_array به روز شده و مقدار بدست آمده جهت چرخش فرمان تنظیم می‌شود. در این شرط همچنین مقدار time1 که بیانگر زمان اجرای برنامه است و در ابتدای برنامه تعریف شد، به روز می‌شود.

شرط دوم: اگر تفاضل زمان حال اجرای برنامه و مقدار time1 ،که در ابتدای برنامه تعریف شده است و یا در شرط اول به روز شده، بیشتر از آستانه ۰.۸ ثانیه باشد و همچنین تمام سنسورهای آلتراسونیک مقداری بیشتر از ۱۴۵۰ نشان دهند، آنگاه اجرا کن:

همانند شرط اول ابتدا مقدار خطا به صورت error = (REFRENCE - SECOND_PXL) مشخص می‌شود. در آن مقدار نقطه رفرنس که برابر ۱۲۸ می‌باشد از مقدار میانگین ماسک جاده سمت چپ کم خواهد شد و به عنوان خطا ثبت می‌شود. سپس با توجه به خطا بدست آمده مقدار فرمان با توجه به ضریب کنترل‌کننده تعیین می‌گردد. به منظور دقت بهتر و چرخش نرمتر فرمان این مقدار در steer_array به روز شده و مقدار بدست آمده جهت چرخش فرمان تنظیم می‌شود.

در غیر این صورت: اگر دو شرط بالا بر قرار نبود، آنگاه اجرا کن:

ابتدا مقدار خطا به صورت error = (REFRENCE - CURRENT_PXL) مشخص می‌شود. در آن مقدار نقطه رفرنس که برابر ۱۲۸ می‌باشد از مقدار میانگین ماسک جاده سمت راست کم خواهد شد و به عنوان خطا ثبت می‌شود. سپس با توجه به خطا بدست آمده مقدار فرمان با توجه به ضریب کنترل‌کننده تعیین می‌گردد. به منظور دقت بهتر و چرخش نرمتر فرمان این مقدار در steer_array به روز شده و مقدار بدست آمده جهت چرخش فرمان تنظیم می‌شود.

اگر خودرو در سمت راست جاده باشد، شروط آن به صورت مقابل است:

```
else:
    if ((sensors_array[2]<1450 or sensors_array[1]<1450) and (mean_obstacle>obs_yellow)):
        error = (REFRENCE - SECOND_PXL)
        time1 = time.time()
        steer = -(kp * error)
        steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
        car.setSteering(int(steer_array))

    else:
        error = REFRENCE - CURRENT_PXL
        steer = -(kp * error)
        steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
        car.setSteering(int(steer_array))
```

شرط اول: اگر سنسور آلتراسونیک سمت راست یا وسط مقدار کمتر از ۱۴۵۰ نشان داد و همچنین مقدار میانگین خط زرد مرکزی جاده کمتر از مقدار نقطه میانی مانع بود، آنگاه اجرا کن:

ابتدا مقدار خطا به صورت (REFRENCE - SECOND_PXL) = error مشخص می‌شود. در آن مقدار نقطه رفرنس که برابر ۱۲۸ می‌باشد از مقدار میانگین ماسک جاده سمت چپ کم خواهد شد و به عنوان خطا ثبت می‌شود. سپس با توجه به خطا بدست آمده مقدار فرمان با توجه به ضریب کنترل‌کننده تعیین می‌گردد. به منظور دقت بهتر و چرخش نرمتر فرمان این مقدار در steer_array به روز شده و مقدار بدست آمده جهت چرخش فرمان تنظیم می‌شود. در این شرط همچنین مقدار time1 که بیانگر زمان اجرای برنامه است و در ابتدای برنامه تعریف شد، به روز می‌شود. در این قسمت خودرو سعی می‌کند مانع شناسایی شده را دور بزند.

در غیر این صورت: اگر شرط بالا بر قرار نبود، آنگاه اجرا کن:

ابتدا مقدار خطا به صورت (REFRENCE - CURRENT_PXL) = error مشخص می‌شود. در آن مقدار نقطه رفرنس که برابر ۱۲۸ می‌باشد از مقدار میانگین ماسک جاده سمت راست کم خواهد شد و به عنوان خطا ثبت می‌شود. سپس با توجه به خطا بدست آمده مقدار فرمان با توجه به ضریب کنترل‌کننده تعیین می‌گردد. به منظور دقت بهتر و چرخش نرمتر فرمان این مقدار در steer_array به روز شده و مقدار بدست آمده جهت چرخش فرمان تنظیم می‌شود. در این قسمت خودرو در مسیر خود مانع مشاهده نمی‌کند و مسیر مستقیم خود را در سمت راست جاده طی می‌کند.

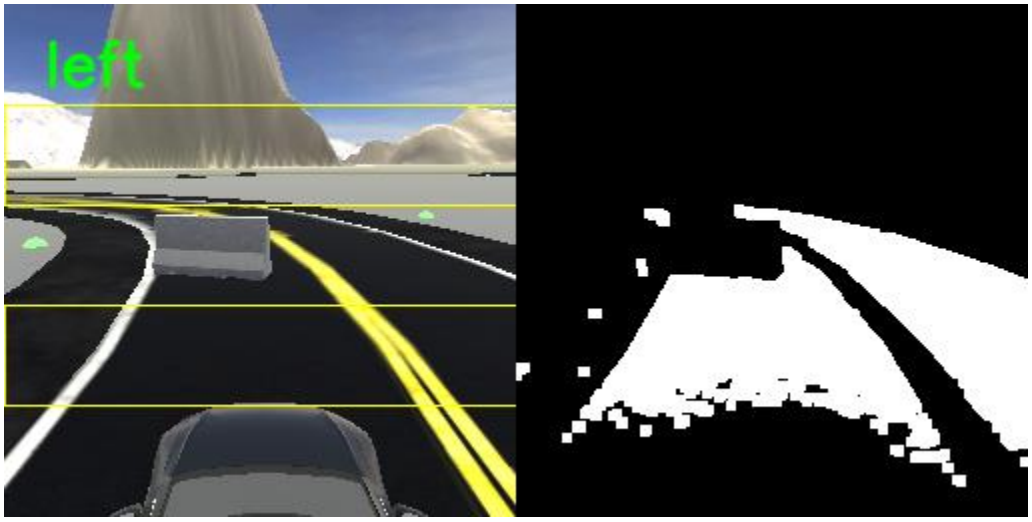
با طی کردن شرایط فوق و عملیات کنترل، خودرو به درستی ابتدا موانع را تشخیص می‌دهد و سپس سعی می‌کند آن را دور زده و به مسیر قبلی خود بازگردد.

۵-۸- سیستم تصمیم گیری در محیط بین شهری

تصمیم گیری یکی از اساسی ترین زیرسیستم‌های خودروی خودران به شمار می‌رود. سرعت بالای خودرو در محیط بین شهری اهمیت درک محیط و تصمیم گیری را نیز چندین برابر می کند. زیرا تمامی اقداماتی که در

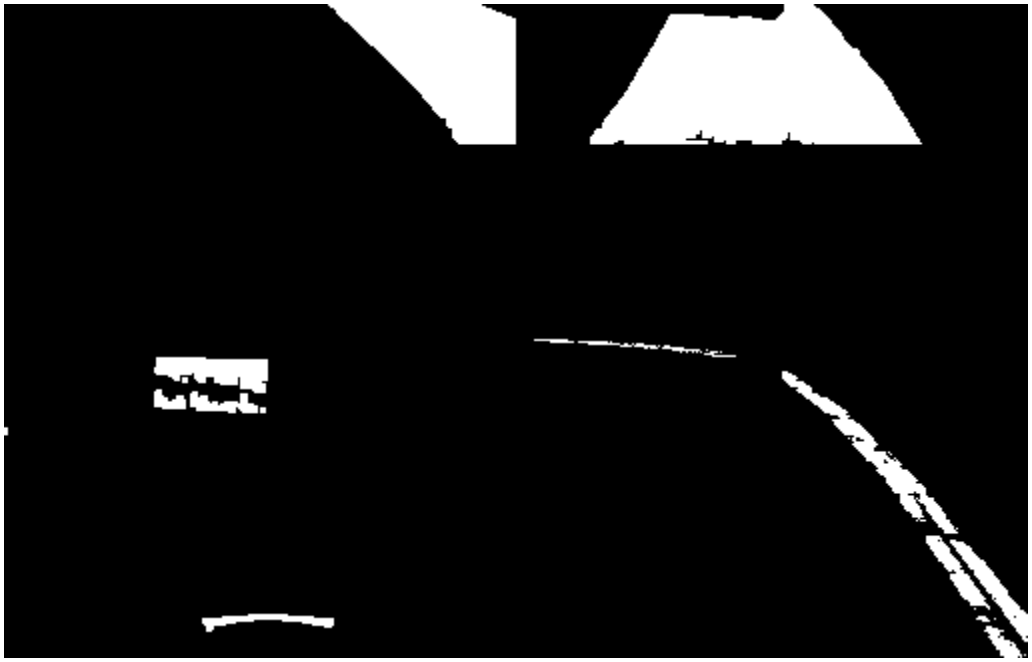
ادامه انجام می شود براساس تصمیم اولیه انتخاب می شوند. در این بخش با درنظر گیری تمام اطلاعات حاصل از زیرسیستم ادراک، قوانینی برای خودرو وضع شده که بر اساس آن اقدام درست انتخاب شود.

حالت زیر یک نمونه از وضعیت خودرو در محیط بین شهری می باشد. در سمت چپ، تصویر دریافتی از دوربین خودرو در محیط بین شهری قرار داده شده است. مستطیل های زیر ناحیه های مطلوب برای استخراج اطلاعات را مشخص می‌کند. در سمت راست ماسک جاده خودرو قرار گرفته است. همان طور که مشاهده می شود، خودرو در لاین چپ قرار گرفته و مانعی در راستای آن قرار دارد. همچنین لازم است بخش کنترلی خودرو برای غلبه بر پیچ جاده، عملکرد درست و سریعی از خود نشان دهد.



شکل ۱۷-۸ نمونه‌ای از وضعیت خودرو

سایر ماسک های استخراج شده از سیستم ادراک نیز به شکل زیر است. لاین های مربوط به رفرنس جاده در قسمت بالا قرار گرفته و ماسک مانع به همراه ماسک خط زرد در بخش پایینی قرار دارد.



شکل ۱۸-۸ ماسک های مرتبط با حرکت

زیرسیستم تصمیم گیری در هر لحظه با بررسی اطلاعات استخراج شده و انتخاب عکس العمل مناسب، رفرنس لازم را برای دنبال کردن، به زیرسیستم کنترل ارسال می کند. در مثال قبل سنسور میانی وجود مانع را شناسایی کرده و زیرسیستم تصمیم گیری رفرنس لاین راست را جهت تغییر لاین برای کنترل انتخاب می کند. در نهایت زاویه فرمان نهایی ۲۰۷ درجه انتخاب شده که به معنای گردش فوری به سمت راست برای جلوگیری از برخورد با مانع می باشد.

Current: 119.44208396560445

Error: -94.45398317664524

Steer: 207.79876298861953

Yellow line: 189.7644305772231

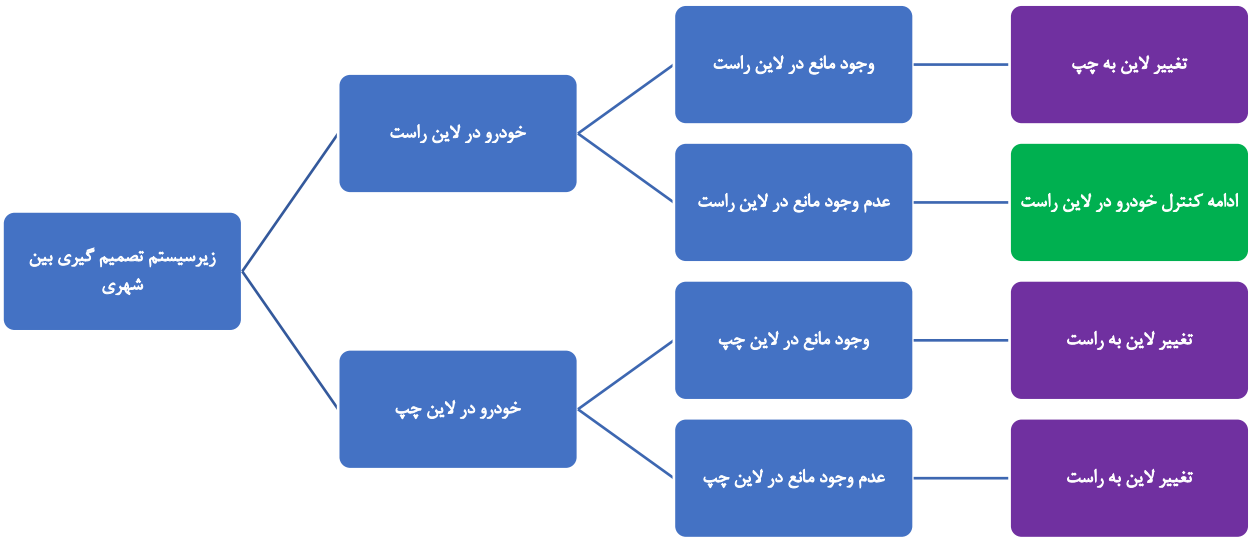
Obstacle yellow: 150.53052917232023

Mean obstacle: 103

Position: left

Sensors: [1499.9, 845.4, 1500.]

نکته مهم در رانندگی بین شهری، حرکت در لاین راست می باشد، لذا زیرسیستم تصمیم گیری تنها در صورت لزوم به لاین چپ تغییر جهت می دهد و در سایر شرایط خودرو را در لاین راست جاده نگه می دارد. دیاگرام زیر نحوه کلی عملکرد زیرسیستم تصمیم گیری را در بخش بین شهری نشان می دهد.



شکل ۱۹-۸ سناریوهای حرکت بین‌شهری

پیاده سازی زیرسیستم تصمیم گیری در پایتون به شکل زیر است.

```
if (position=='left'):
        if ((sensors_array[0]<1450 or sensors_array[1]<1450) and (mean_obstacle<obs_yellow)):
            error = (REFRENCE - SECOND_PXL)
            time1 = time.time()
            steer = -(kp * error)
            steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
            car.setSteering(int(steer_array))

        elif((time.time()-time1)>0.8 and (min(sensors_array) > 1450)):
            error = (REFRENCE - SECOND_PXL)
            steer = -(kp * error)
            steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
            car.setSteering(int(steer_array))

        else:
            error = REFRENCE - CURRENT_PXL
            steer = -(kp * error)
```

```
            steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
            car.setSteering(int(steer_array))

else:
    if ((sensors_array[2]<1450 or sensors_array[1]<1450) and (mean_obstacle>obs_yellow)):
            error = (REFRENCE - SECOND_PXL)
            time1 = time.time()
            steer = -(kp * error)
            steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
            car.setSteering(int(steer_array))

    else:
            error = REFRENCE - CURRENT_PXL
            steer = -(kp * error)
            steer_array = np.round(0.85 * (np.array(steer)) + 0.15 * steer_array, 1)
                car.setSteering(int(steer_array))
```

۸-۶- سیستم کنترل موقعیت خودرو

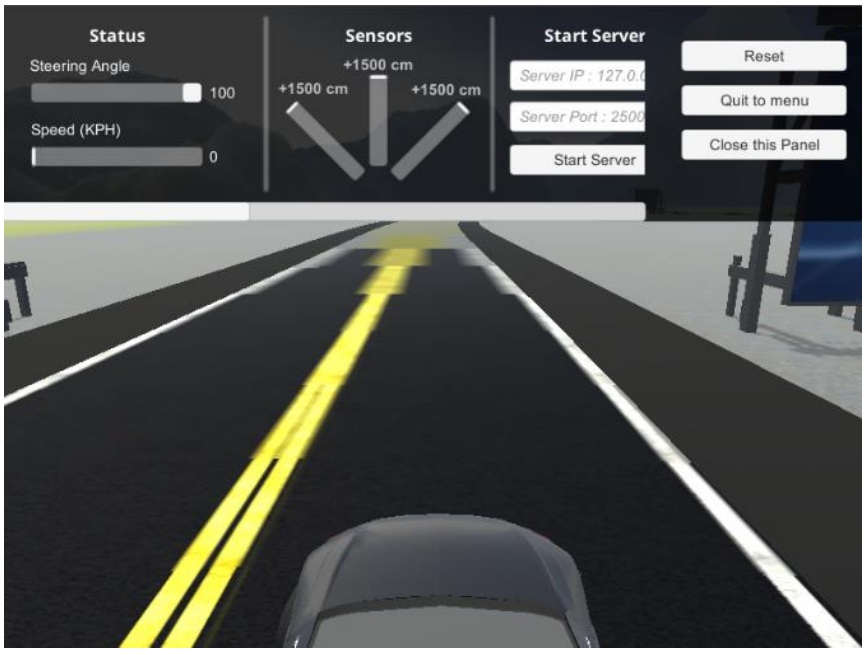
برای تعیین‌موقعیت خودرو و برنامه‌ریزی مسیر کوتاه‌مدت[۳۰۰] نیاز است که خودرو نقطه‌ی مرجع داده‌شده را در هرلحظه دنبال نماید. این مورد نقش بسیار اساسی‌ای در کنترل خودرو برای دور زدن در پیچ‌های مختلف به همراه دور زدن موانع را دارا می‌باشد. برای هر کدام از حالت‌های مختلف (حرکت در مسیر مستقیم، دور زدن مانع و دور زدن پیچ) با توجه به سیاست طراحی‌شده برای حرکت خودرو، یک نقطه‌ی مرجع برای خودرو مشخص می‌شود و نیاز است که خودرو با اعمال سیگنال کنترلی لازم بر روی عملگر حرکت عرضی[۳۰۱] که همان زاویه‌ی چرخ‌ها می‌باشد، حرکت لازم را انجام دهد.

عملگر سیستم که فرمان خودرو می‌باشد، توسط دستور setSteering از کلاس خودرو کنترل می‌شود. اندازه‌ی زاویه‌ی ذکر شده در بازه‌ی ۳۰- الی ۳۰+ درجه قرار می‌گیرد و عملگر خودرو مقدار ۱۰۰- الی ۱۰۰+ درصد را به این زوایا نگاشت می‌کند.

$$Velocity \in [-30, +30] \quad (۸-۱)$$

$$Actuator\ Input \in [-100, +100] \quad (۸-۲)$$

$$Actuator\ input\ to\ steering\ angle: \frac{Actuator\ Input}{100} \times 30 \quad (۸-۳)$$



شکل ۲۰-۸ مقدار حداکثر چرخش چرخ‌ها (۳۰+ درجه و ورودی ۱۰۰ به عملگر)

[300] Short-time motion planning
[301] Lateral movement

برای کنترل حرکت عرضی خودرو، پس از تعیین میزان خطای سیستم با موقعیت مرجع، از یک کنترل‌کننده‌ی PID در فضای گسسته استفاده می‌شود. موقعیت خودرو که Position نامیده می‌شود، اختلاف این مقدار از مقدار مرجع به‌عنوان خطا تعریف می‌شود. با توجه به ثابت بودن زمان نمونه‌برداری (Δt) در هر حلقه، پیاده‌سازی کنترل‌کننده به‌صورت زیر خواهد بود.

$$e(t) = REF - Position \quad (۸-۴)$$

$$I(k) = I(k-1) + K_i \times e(t) \quad (۸-۵)$$

$$D(k) = \frac{e(k) - e(k-1)}{\Delta t} \quad (۸-۶)$$

$$Control\ Signal = K_p \times e(k) + K_i \times I(k) + K_d \times D(k) \quad (۸-۷)$$

شایان ذکر است که مقدار مرجع (REF) با توجه با حالت خودرو دستخوش تغییر می‌شود. و هدف کنترل‌کننده، دنبال‌کردن آن مرجع با سرعت مناسب در همه‌ی حالا می‌باشد. در صورت تغییر کردن مرجع، قسمت انتگرالی کنترل‌کننده که مسئولیت جمع کردن مقادیر خطا را برعهده دارد صفر خواهد شد. متغیر previous_error بیانگر میزان خطا در حلقه‌ی قبلی می‌باشد. همچنین مقادیر اولیه‌ی خطا و انتگرال‌گیر برابر صفر می‌باشد.

```
previous_error = 0
integral = 0.0
error = 0
dt = 0.05
```

برای پیاده‌سازی این قسمت در برنامه‌ی اصلی، پس از محاسبه‌ی خطا در هر مرحله، ابتدا مقدار ذخیره‌شده در انتگرال‌گیر بروزرسانی می‌شود. همچنین می‌بایست مقدار مشتق‌گیر نیز محاسبه شود.

```
integral = integral + error * dt
derivative = (error - previous_error) / dt
```

در ادامه، نیز، مقدار تناسبی محاسبه شده و و هر کدام از ۳ قسمت اصلی کنترل‌کننده، در ضرایب متناظر خود ضرب شده و سیگنال کنترلی را تولید می‌نمایند.

```
steer = (kp * error + ki * integral + kd * derivative)
```

در نهایت سیگنال‌کنترلی که در متغیر steer ذخیره شده‌است توسط دستور setSteering از کلاس خودرو، به خودرو منتقل شده و چرخ‌های خودرو را می‌چرخاند.

```
car.setSteering(steer)
```

با توجه به عدم وجود دینامیک سیستم و همچنین نبود امکان انجام شناسایی، تنظیم پارامترهای کنترل‌کننده‌ی PID مورد استفاده در این بخش به صورت تجربی و با استفاده از دکمه‌های کشویی انجام می‌شود. متغیرهای لازم در پنجره‌ی نوار کشویی به صورت زیر تعریف می‌گردد. برای افزایش دقت، متغیرهای موجود در پنجره بین ۰ الی ۲۰۰ در نظر گرفته شده و در تقسیم بر ۱۰۰ می‌شوند و این باعث دقت تا دو رقم اعشار در تعیین ضرایب کنترل‌کننده می‌باشد.

```
cv2.namedWindow('Controls',cv2.WINDOW_NORMAL)
cv2.createTrackbar('kp','Controls',0 ,200, nothing)
cv2.createTrackbar('ki','Controls',0, 200 ,nothing)
cv2.createTrackbar('kd','Controls',0, 200 ,nothing)

kp = (cv2.getTrackbarPos('kp','Controls')) / 100
ki = (cv2.getTrackbarPos('ki','Controls')) / 100
kd = (cv2.getTrackbarPos('kd','Controls')) / 100
```

شکل ۲۱-۸ پنجره‌ی کشویی برای تعیین ضرایب K_p، K_i و K_d کنترل‌کننده

پس از بررسی‌های متعدد، بهترین مقدار برای ضرایب کنترل‌کننده به صورت جدول ۱-۸ می‌باشد.

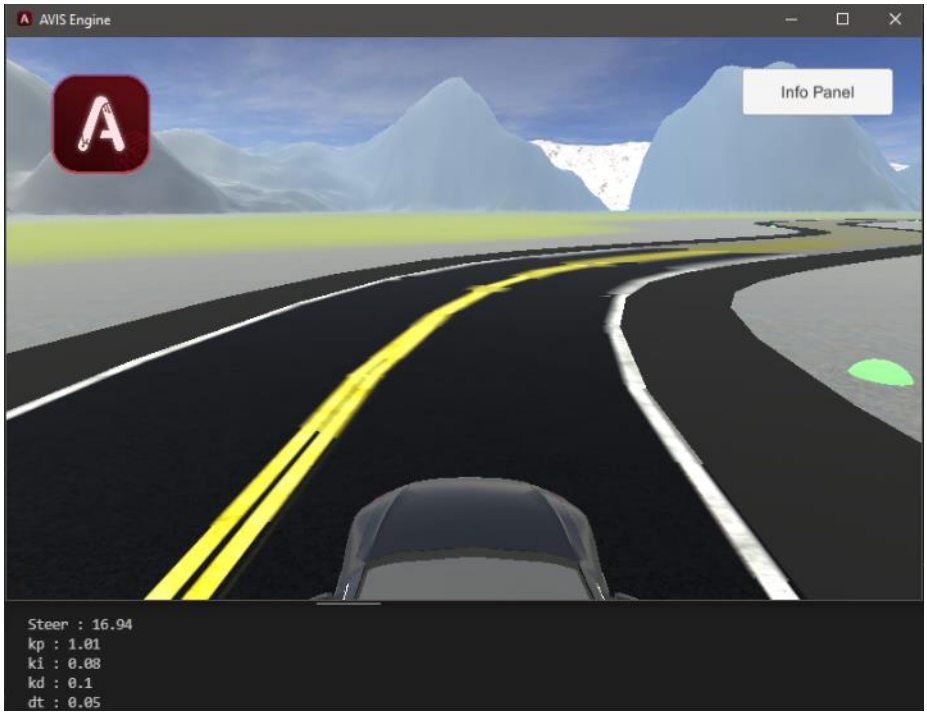
جدول ۱-۸ ضرایب کنترل‌کننده‌ی PID کنترل موقعیت

ضریب	مقدار
K_p	**1.02**
K_i	**0.08**
K_d	**0.1**

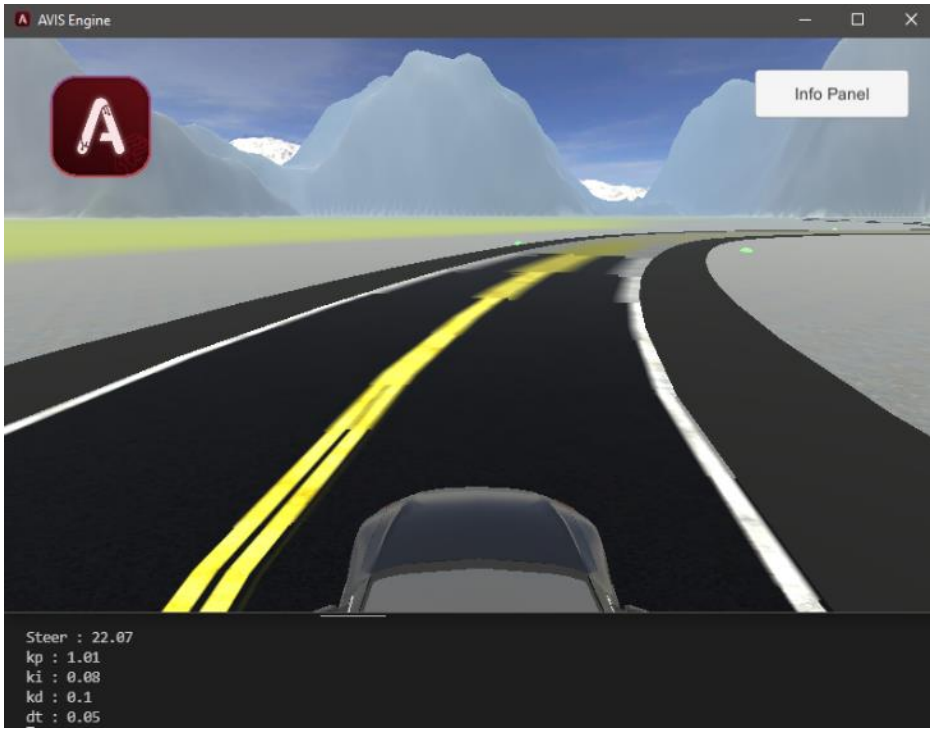
همچنین این مقادیر در برنامه‌ی اصلی وارد می‌شوند.

```
kp = 1.01
ki = 0.08
kd = 0.1
```

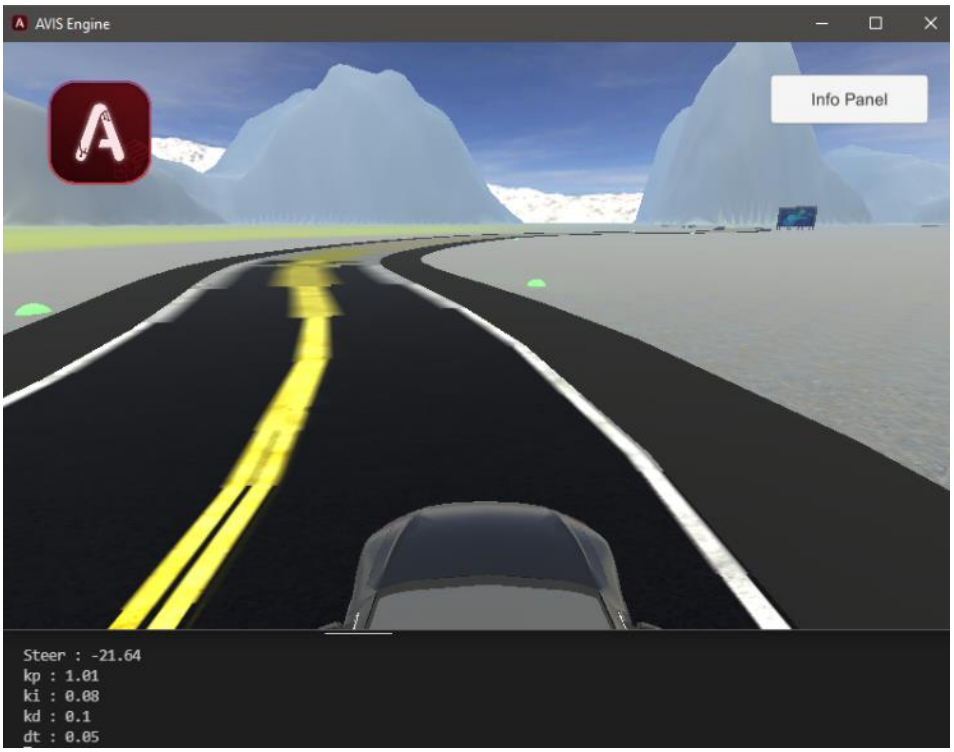
در ادامه، با اجرا کردن برنامه به‌همراه کنترل‌کننده‌ی PID، خودرو قادر خواهد بود که در قسمت‌های مختلف، عملکرد مناسب از خود نشان داده و تصمیمات لازم برای کنترل موقعیت خود را اخذ نماید. در ادامه چند نمونه از عملکرد خودرو در قسمت‌های مختلف به‌همراه چرخش مورد نیاز برای انجام کنترل موقعیت نمایش داده شده‌است.



شکل ۲۲-۸ نمونه‌ی اول از کنترل موقعیت خودرو



شکل ۲۳-۸ نمونه‌ی دوم از کنترل موقعیت خودرو



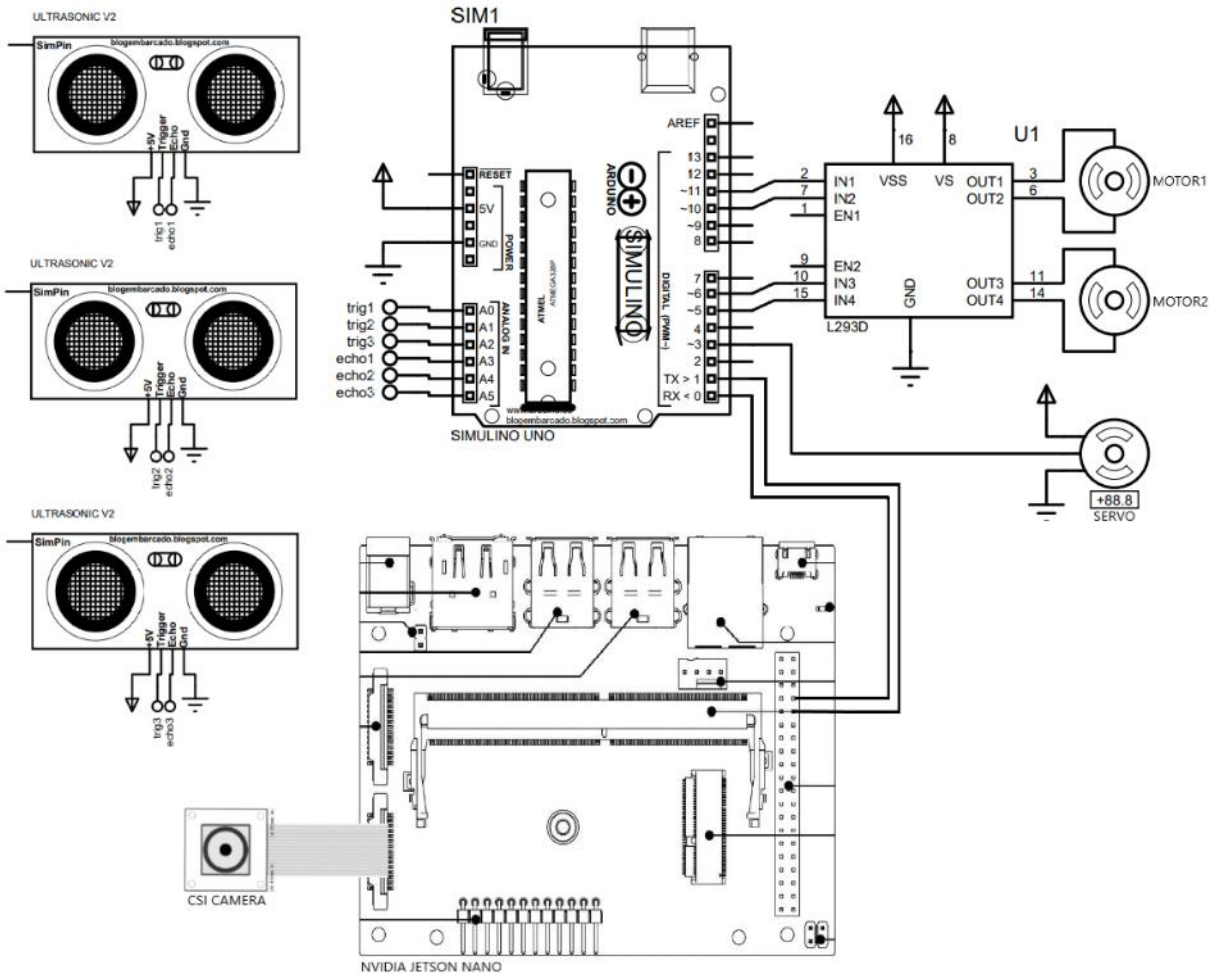
شکل ۲۴-۸ نمونه‌ی سوم از کنترل موقعیت خودرو

۹- توصیف سیستم

در این فصل به بررسی و توصیف کلی عملکرد نرم افزاری و سخت افزاری خودرو پرداخته می‌شود. شرح جزئیات هر بخش نیز در سایر فصل ها آورده شده است.

۹-۱- شماتیک سخت افزاری

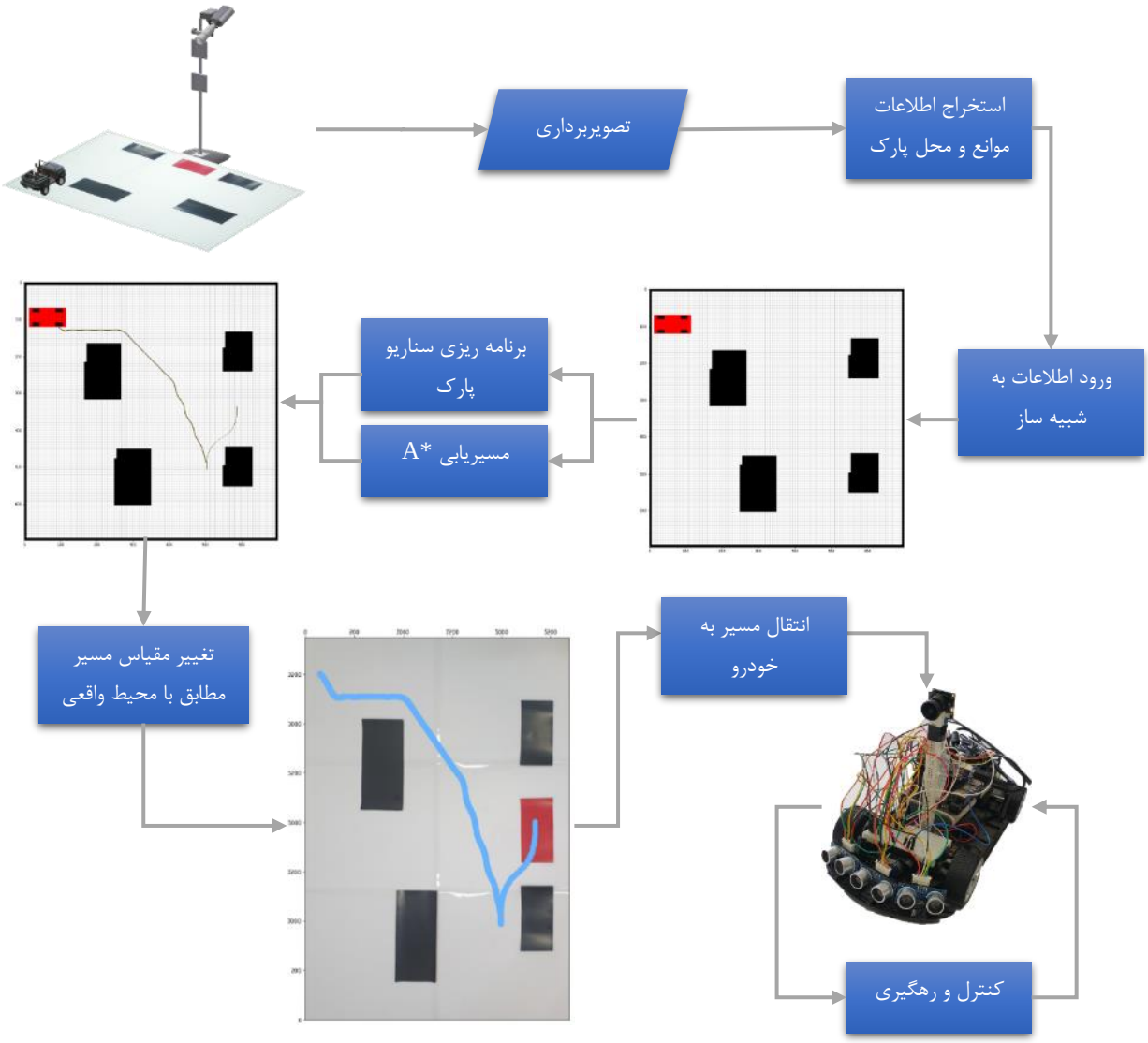
نحوه اتصالات سخت افزار خودرو شامل برد جتسون نانو، برد آردوینو، دوربین، سنسورهای آلتراسونیک، درایور و موتورها در شکل ۹-۱ نشان داده شده است. تغذیه خودرو نیز توسط یک پاوربانک که بر روی خودرو سوار است تامین می‌شود.



شکل ۹-۱ شماتیک سخت افزاری خودرو

۲-۹- عملکرد سیستم در محیط پارک

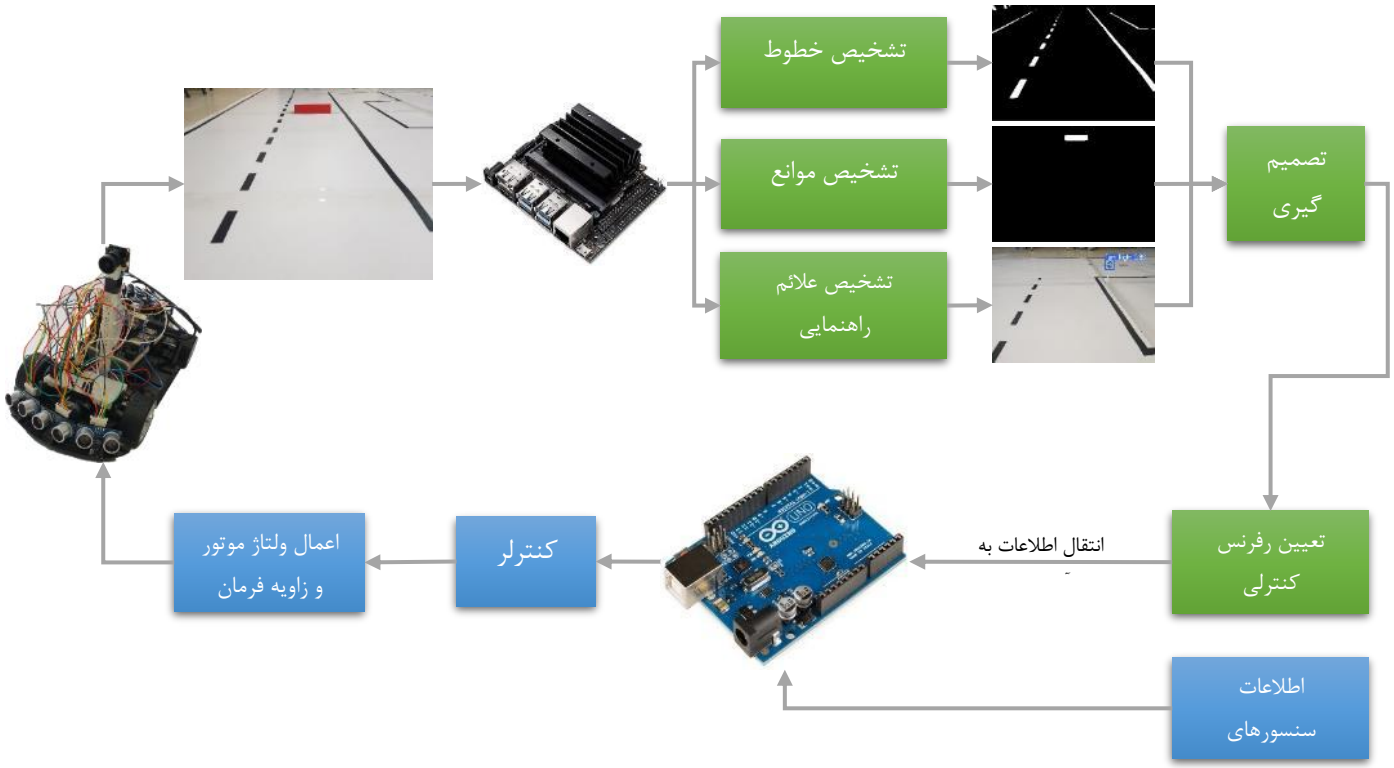
عملکرد خودرو در محیط پارک به دو بخش برنامه ریزی مسیر و رهگیری آن تقسیم می‌شود. بر بخش برنامه ریزی مسیر، ابتدا نقشه با استفاده از الگوریتم های پردازش تصویر، مورد بررسی قرار می‌گیرد. سپس اطلاعات آن استخراج شده و به شبیه ساز داده می‌شود. پس از مسیریابی، رهگیری مسیر توسط خودرو انجام می‌شود. شکل ۲-۹ این فرآیند را به طور کلی نشان می‌دهد.



شکل ۲-۹ دیاگرام عملکردی خودرو در محیط پارک

۳-۹- عملکرد سیستم در محیط شهری و بین شهری

عملکرد سیستم در محیط شهری و بین شهری، توسط ۲ برد جتسون نانو و آردوینو انجام می‌شود. برد جتسون نانو پردازش های سطح بالا مربوط به شبکه های عصبی و الگوریتم های پردازش تصویر را به عهده داشته و آردوینو اجزای سخت افزاری خودرو را کنترل می‌کند. به دلیل اهمیت ادراک، لازم است این مرحله در هر لحظه صورت بگیرد. اطلاعات محیط استخراج شده و پس از تصمیم گیری، رفرنس نهایی جهت کنترل به برد آردوینو ارسال می‌شود. شکل ۳-۹ این فرآیند را به طور کلی نشان می‌دهد.



شکل ۳-۹ دیاگرام عملکردی خودرو در محیط شهری و بین شهری

۱۰ - سخت افزار و پیاده سازی

۱-۱۰- اجزای خودرو

۱-۱-۱۰- سروو موتور

در این بخش، به بررسی سروو موتور پرداخته می‌شود که برای تعیین موقعیت‌های دقیق مورد استفاده قرار می-گیرد. این موتورها انواع مختلفی در شکل و اندازه‌های متفاوت دارند.

شکل ۱-۱۰ یک نوع سروو موتور DC [۳۱۷]

شکل ۲-۱۰ یک نوع سروو موتور AC [۳۱۸]

۲-۱-۱۰- تعریف سروو موتور

سروو موتور یک عملگر چرخشی می‌باشد که با استفاده از آن می‌توان موقعیت زاویه‌ای[۳۰۲] را کنترل کرد. این نوع موتور از یک موتور به همراه حسگر تشکیل شده‌است که به‌هم کوپل شده‌اند و موقعیت موتور توسط سنسور بازخورد[۳۰۳] داده می‌شود. برای تکمیل نمودن سیستم، به یک راه‌انداز یا درایور سروو نیز نیاز است[۳۰۴]. راه‌انداز با استفاده از اطلاع بازخوردشده از حسگر موقعیت موتور را با دقت کنترل می‌نماید.[۳۱۹]

۳-۱-۱۰- مکانیزم سروو موتور

سروو موتور یک مکانیزم حلقه‌بسته می‌باشد که با بازخورد دادن موقعیت، حرکت و موقعیت‌نهایی موتور راکنترل می‌نماید. سیگنال ورودی می‌تواند دیجیتال و یا آنالوگ باشد و نشان‌دهنده‌ی موقعیت شفت خروجی باشد. نحوه‌ی انجام بازخورد موقعیت نیز به این صورت می‌باشد که یک رمزگذار و یا انکودر[۳۰۵] جفت شده‌است که قابلیت تعیین سرعت و موقعیت موتور را دارا می‌باشد. در ساده‌ترین حالت که فقط کنترل موقعیت مدنظر می‌باشد، موقعیت اندازه‌گیری شده‌ی موتور با فرمان موقعیت مقایسه شده و در صورت وجود تفاوت، یک سیگنال خطا تولید می‌شود که باعث حرکت موتور در یکی از دو جهت ساعت‌گرد و یا پادساعت‌گرد خواهد شد تاموقعیت موتور به موقعیت مطلوب برسد. [۳۲۰]

۴-۱-۱۰- کاربرد سروو موتور

یکی از کاربردهای سروو موتور، تعیین زاویه‌ی چرخش چرخ‌ها و یا steering می‌باشد. با توجه به نیاز به این که برای حرکت و تعیین موقعیت نیاز است که چرخ‌های خودرو چرخش مناسب داشته باشند، به یک سروو موتور کوپل شده و و توسط یک سیگنال کنترلی کنترل می‌شوند.

[302] Angular position
[303] Feedback
[304] Servo drive
[305] encoder

۵-۱-۱۰- موتورگیربکس

موتورگیربکس یا الکتروگیربکس همانگونه که از نام آن پیدا است از ترکیب یک موتور الکتریکی و یک گیربکس ساخته می شود که در آن وظیفه گیربکس تغییر دور و گشتاور موتور برای رسیدن به دور و قدرت مورد نظر است. گیربکس قادر است قدرت چرخش را افزایش و سرعت آن را کاهش دهد. هر چه نرخ کاهش سرعت در گیربکس بیشتر شود، قدرت آن بالاتر خواهد رفت. با توجه به شرایط مصرف بهینه انرژی، الکتروگیربکس ها یکی از تجهیزاتی هستند که می توانند مصرف انرژی را به شکل قابل توجهی کاهش دهند.

۶-۱-۱۰- موتورهای الکتریکی

موتور الکتریکی یا همان الکتروموتور ها تجهیزاتی هستند که توانایی کار با جریان برق متناوب و مستقیم را دارند. این تجهیزات قابلیت تبدیل انرژی الکتریسیته به انرژی مکانیکی را دارند. و به دو دسته کلی موتور AC و DC تقسیم بندی میشود که در ادامه به صورت خلاصه توضیح داده میشود:

۷-۱-۱۰- موتور الکتریکی DC

در موتور DC یک کویل وجود دارد که با ایجاد یک میدان مغناطیسی باعث ایجاد حرکت چرخشی در درون موتور می شود. درون موتور یک شفت آهنی وجود دارد که شامل دو قطب با علامت های معکوس است. با تغییر جهت میدان مغناطیسی یک نیروی گشتاوری در اطراف شفت ایجاد می شود که موجب حرکت آن می شود. موتورهای الکتریکی DC در دو مدل خاردار و بدون خار ارائه می شوند.

۸-۱-۱۰- موتور الکتریکی AC

موتورهای الکتریکی AC یا موتور القایی قرار دارد که با جریان متناوب کار می کنند. ساختمان موتورهای الکتریکی القایی از دو قسمت ثابت و دوار که به ترتیب به نام های استاتور و رتور معروف هستند، تشکیل شده اند. وظیفه استاتور ایجاد یک میدان مغناطیسی اطراف رتور است. ساختار استاتور از یکسری ورقه های فلزی تشکیل شده است که یک استوانه توخالی فلزی را تشکیل می دهد.درون ورقه های فلزی موجود در استاتور یکسری شیارهایی وجود دارد که در این شیارها سیم پیچ های الکتروموتور جاسازی شده اند که به کلاف نیز معروف می باشند. رتور نیز همانند استاتور از دو قسمت هسته و سیم پیچ تشکیل شده است که سیم پیچ های اطراف رتور می تواند از جنس مس یا آلمینیوم باشد. جنس هسته رتور معمولا رابطه مستقیمی با قدرت موتور دارد به صورتی که برای موتورهای با قدرت بالا معمولا از قفس مسی و در موتورهای با قدرت کم از قفس آلمینیومی استفاده می شود.

۹-۱-۱۰- گیربکس

گیربکس ها وظیفه انتقال نیرو بین یک مصرف کننده و تولید کننده را دارند. این انتقال می تواند برای افزایش یا کاهش گشتاور و یا سرعت دوران باشد. گیربکس ها به عنوان یک واسط بین مصرف کننده و تولید کننده توان هستند و یک تعادلی را میان این دو طرف ایجاد می کنند. از گیربکس ها همچنین می توان برای تغییر جهت دوران نیز استفاده کرد. خروجی اکثر گیربکس ها به صورت گشتاور است و متناسب با کاربری آن می تواند گشتاور یا سرعت دورانی را کاهش یا افزیش دهد.

۱-۹-۱-۱۰- انواع گیربکس

از جمله انواع گیربکس ها می توان به گیربکس خورشیدی ، حلزونی ، مارپیچ و ترکیبی اشاره کرد. گیربکس ها متناسب با کاربردشان می توانند دارای ضرایب تبدیل مختلفی باشند.

۲-۹-۱-۱۰- الکتروگیربکس

الکتروگیربکس یک سیستم انتقال و افزایش قدرت است که شامل یک منبع نیرو و یک سامانه انتقال قدرت می باشد که باعث می شود سرعت و دور موتور را کاهش داده و آن را تبدیل به قدرت کند به عبارت دیگر باعث می شود دور خروجی موتور کاهش پیدا کند ولی گشتاور (قدرت) آن افزایش یابد.

الکتروگیربکس ها قطعاتی جمع و جور اما با استحکام بالا هستند که همانند گیربکس ها می توانند با کاهش سرعت و دور موتور و تبدیل نمودن آن به یک سیستم انتقال قدرت باعث افزایش قدرت (گشتاور) و بلعکس بشوند.

۳-۹-۱-۱۰- مزایای موتور گیربکس

الکتروگیربکس ها دارای مزایای مختلفی هستند که از جمله آن می توان به موارد زیر اشاره نمود.

- حذف نصب جداگانه موتور الکتریکی و گیربکس اشاره کرد. کاهش هزینه های نصب و طراحی باعث کاهش کلی هزینه های شروع کار یک پروژه مهندسی می شود.
- دقت بالای طراحی دندانه های چرخ دنده الکتروگیربکس ها باعث می شود که امکان انتقال گشتاور و سرعت بالا توسط این تجهیزات فراهم شود. یکی دیگر از مزیت های الکتروگیربکس ها ترکیب مناسب و بهینه گیربکس و الکتروموتور است.

- در حالت عادی معمولا برای انتخاب گیربکس مناسب برای هر الکتروموتور باید وقت زیادی گذاشته شود. اما در الکتروگیربکس ها بهترین حالت تطبیق گیربکس و موتور الکتریکی وجود دارد که موجب مصرف بهینه انرژی می شود.
- همچنین با توجه به اینکه الکتروگیربکس ها نیازی به تراز کردن و کوپلینگ ندارند، تلفات انرژی در این دو مرحله نیز از بین می رود. مشکلات تراز نبودن در صنعت می تواند علاوه بر افزایش هزینه های تمام شده موجب خرابی بلبرینگ ها و کاهش عمر موتور شود.
- طراحی الکتروگیربکس ها در واقع یعنی تحولی عظیم در تمامی جزییات یک الکتروموتور و گیربکس. الکتروگیربکس ها از آلیاژ های به خصوصی استفاده می کنند که دارای مزیت های زیادی نسبت به فلزات معمولی می باشد.
- نوع بلبرینگ و طراحی دندانه های چرخ دنده در الکتروگیربکس ها بهبود پیدا کرده‌اند. تمام این موارد موجب کاهش نویز و افزایش طول عمر این تجهیزات می شود.

۴-۹-۱-۱۰- انتخاب موتور گیربکس

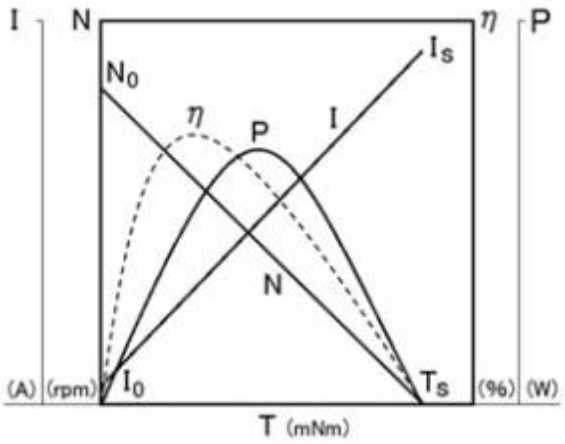
یکی از مهمترین قسمت های طراحی و تولید الکتروگیربکس ها، تطبیق موتور الکتریکی و گیربکس مناسب است. در واقع با توجه به تنوع گیربکس های صنعتی انتخاب گیربکس مناسب کاملا بستگی به نوع کاربری الکتروگیربکس دارد. با توجه به نوع کاربری می توانید از الکتروموتورهای DC یا AC استفاده کنید.

با توجه به مواردی که در قسمت بالا گفته شد، انتخاب بهترین الکتروگیربکس بستگی به کاربری آن دارد. همواره قوی ترین الکتروموتور به همراه بالاترین ضریب تبدیل گیربکس نمی تواند گزینه مناسبی برای هر صنعت باشد. باید تجهیزاتی انتخاب شود که هم از لحاظ هزینه ای و هم از لحاظ بازدهی مناسب باشند.

مراحل ساخت و طراحی یک الکتروگیربکس شامل نیازمندی های کاربر، محاسبات طراحی و مدل های مناسب الکتروموتور و گیربکس است.

در قسمت نیازمندی های کاربری تمامی جزییات مورد نیاز برای نصب الکتروگیربکس در نظر گرفته می شود. در این مرحله پارامتر های مهم طراحی و کاربرد هر مشخصه تعیین می شود.

در قسمت محاسبات طراحی اعداد نهایی برای انتخاب گیربکس و الکتروموتور ارائه می شود. به عنوان مثال توان الکتروموتور یا ضریب تبدیل گیربکس تعیین می شود. پس از این مراحل نوبت به انتخاب بهترین مدل گیربکس و الکتروموتور می باشد.
با توجه به مواردی که گفته شد، یک نمودار برای انتخاب هر چه بهتر الکتروگیربکس ارائه می شود. در تصویر زیر پارامتر های مختلف بر حسب بازدهی، گشتاور و سرعت دوران ارائه شده است.



شکل ۳-۱۰ نمودار پارامترهای تاثیرگذار در انتخاب الکتروگیربکس

۵-۹-۱-۱۰- اجزای اصلی الکتروگیربکس

الکتروگیربکس ها با توجه به نسبت کاهش دورموتور اجزای متفاوتی دارند که این تفاوت شامل فرم و جنس آنها می باشد.

اجزای تمام الکتروگیربکس ها شامل موارد زیر هستند:

- **چرخ دنده ها :**که یکی از مهم ترین قطعات الکتروگیربکس ها چرخ دنده ها هستند که دارای نقش بسیار مهمی درکارکرد الکتروگیربکس ها هستند.
- **پوسته الکتروگیربکس :**این قسمت از الکتروگیربکس تمامی دنده های سیستم را در درون خود جای می دهند و پوسته ها اغلب از جنس چدن ریخته گری می باشند.
- **شکاف یا فرم سوراخ ورودی و خروجی :**که جنس آن شکاف ها از آلومینیوم می باشد که اغلب به فرم سوراخ و یا شکاف وجود دارند.
- **کاسه نمد :**برای محافظت الکتروموتور در مقابل عوامل محیطی مثل گرد و غبار استفاده می شوند.
- **درپوش کنترل روغن**

۶-۹-۱-۱۰- دسته بندی موتور گیربکس

الکتروگیربکس ها به طور کلی به دو دسته تقسیم می شوند.

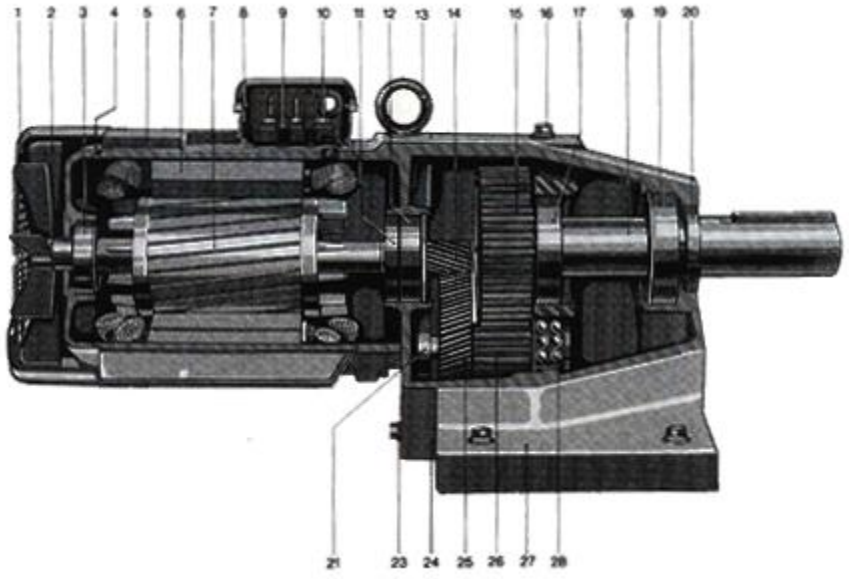
۱-۶-۹-۱-۱۰- الکتروگیربکس کاهنده

گیربکس کاهنده الکتروگیربکس هایی هستند که بین قطعات محرک و متحرک قرار میگیرد و مقدار دور مورد نیاز را تأمین می کنند. این نوع الکتروگیربکس ها در اشکال مختلفی ساخته می شوند، در الکتروگیربکس های کاهنده باید دقت کرد که سرعت در این نوع گیربکس ها کاهش میابد که باعث افزایش سرعت گشتاور می شوند.

۲-۶-۹-۱-۱۰- الکتروگیربکس افزاینده

در صنایعی که نیاز به دور بالا هست مانند صنایع پالایشگاهی و مانند آن از این نوع گیربکس های صنعتی استفاده می شود که برخلاف الکتروگیربکس های کاهنده میزان سرعت را افزایش می دهند و به همان نسبت، میزان گشتاور را کاهش می دهند.

با توجه به توضیحات فوق، از الکتروگیربکس کاهنده که در تصویر ۴-۱۰ آورده شده در خودرو استفاده شده است.



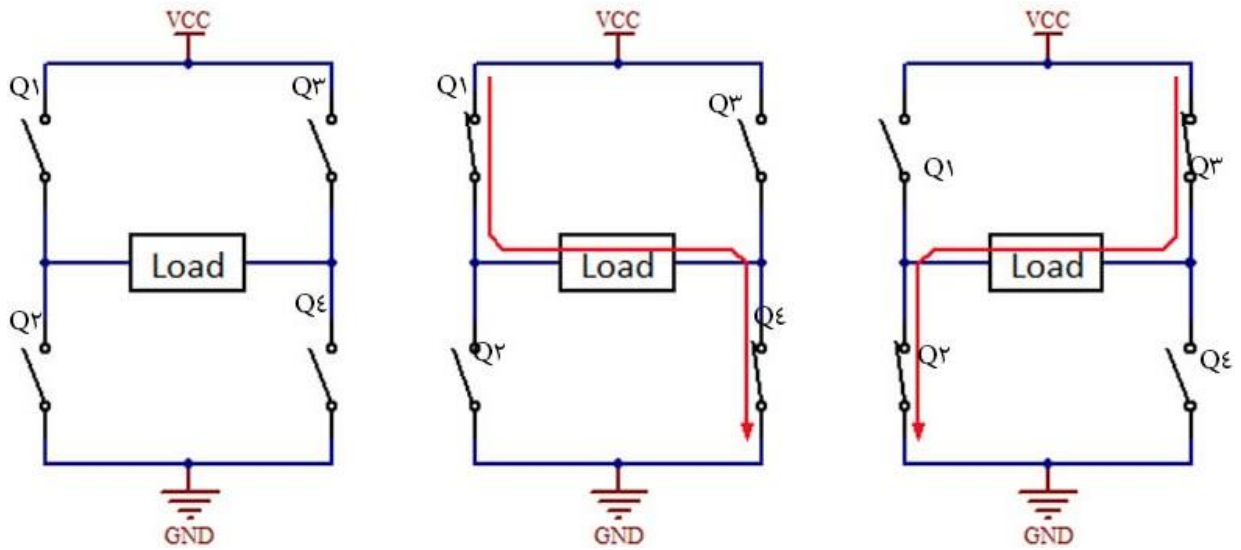
شکل ۴-۱۰ الکتروگیربکس کاهنده [۳۲۱]

۱۰-۱-۱۰- درایو موتور

موتورهای مختلف برای کاربری‌های مختلف بوجود آمده‌اند. هر موتور توانایی متفاوت با ویژگی ساختاری و ظاهری متفاوتی دارد. از اینرو برای فرمان به موتورها نیازمند ولتاژ و جریان منحصر به فرد برای همان موتور می‌باشیم. به‌صورت کلی به دستگاهی که سبب کنترل یک ماشین الکتریکی شود درایو گفته می‌شود به همین منظور از درایور موتور استفاده می‌گردد. درایور موتور یک رابط بین میکروکنترلر و موتور می‌باشد. با این روش میتوان توسط جریان ناچیز میکروکنترلر، به یک موتور با جریان زیادتر و ولتاژ متفاوت فرمان داد. در ادامه درایور موتور L293D که برای خودرو خودران استفاده شده است توضیح داده خواهد شد.

۱-۱۰-۱-۱۰- شیلد درایور موتور L293D

درایور L293D از یک مدار یکپارچه با ولتاژ و جریان بالا با درایور ۴ کاناله تشکیل شده است. درایور L293D به عنوان مدار H Bridge شناخته می‌شود. مدار پل H، به ولتاژ اجازه می‌دهد در مسیر بار Load در خروجی به صورت ساعتگرد و پادساعتگرد حرکت کند. شماتیک مدار H Bridge به صورت زیر می باشد.



شکل ۵-۱۰ شماتیک مدار پل اچ

به عنوان مثال در تصویر بالا با فعال بودن ماسفت (کلید) Q1 و Q4 موتور فعال شده و به سمت راست حرکت می‌کند. همچنین با فعال شدن ماسفت Q2 و Q3 موتور فعال شده و در جهت چپ (معکوس) حرکت خواهد کرد. با فعال شدن همزمان کلید Q1 و Q2 و یا Q3 و Q4 اتصال کوتاه در ولتاژ ورودی به وجود خواهد آمد. با غیر فعال شدن تمامی کلیدها موتور آزاد شده و ثابت خواهد بود. در جدول زیر حالت های مختلف مدار نمایش داده شده است.

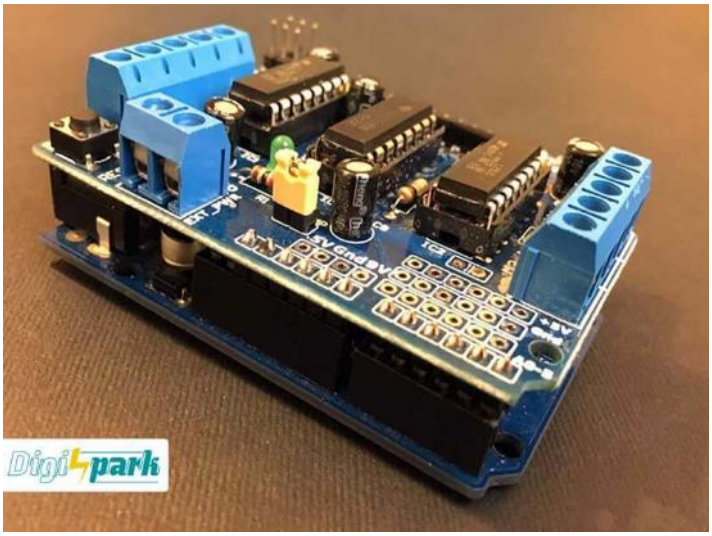
جدول ۱-۱۰ حالت های مختلف مدار

S1	S2	S3	S4	Operation
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor free runs
0	1	0	1	Motor brakes
1	0	1	0	Motor brakes
1	1	0	0	Short Power Supply
0	0	1	1	Short Power Supply
1	1	1	1	Short Power Supply

تراشه L293D توانایی درایو موتور تا ۳۶ ولت را خواهد داشت. توسط این تراشه هر کانال درایور موتور تا ۶۰۰ میلی آمپر جریان دهی خواهد داشت. از ویژگی‌های این شیلد می‌توان به داشتن ۲ کانکتور ۵+ ولت DC اشاره کرد. همچنین دارای ۴ رابط کانکتور برای اتصال استپر موتور (Stepper Motor) و موتور (۲ استپر موتور + ۲ موتور DC) است. شیلد درایور موتور دارای ۶ پین برای اتصال دو سرو موتور (Servo Motor) است که با نام های SER1 و SERVO – 2 بر روی شیلد مشخص شده است. یک کلید ریست Reset پایین شیلد قرار گرفته شده است. ولتاژ ورودی شیلد درایور موتور ۴،۵ تا ۱۲ ولت DC است. این شیلد قابلیت راه اندازی با میکروکنترل های آردوینو Arduino UNO و Arduino Mega2560 را دارد. ابعاد شیلد درایور موتور ۶۹ × ۵۳ × ۱۴،۳ میلی متر است.

شکل ۶-۱۰ تراشه L293D

اتصال شیلد درایور موتور به آردوینو در تصویر زیر آورده شده است.



شکل ۷-۱۰ اتصال شیلد درایور موتور به آردوینو

۲-۱۰- آردوینو

آردوینو یک برد سخت‌افزاری متن‌باز[۳۰۶] می‌باشد که هدف آن استفاده‌ی راحت در پروژه‌های مختلف می‌باشد. این برد علاوه بر یک میکروکنترلر، دارای ورودی-خروجی[۳۰۷] های مناسب دیجیتال و آنالوک می-باشد که با استفاده از آن‌ها می‌توان مجموعه‌ای از کاربردهای مختلف را پیاده‌سازی کرد. یکی از مزایای برد آردوینو نسبت به سایر بردهای موجود، عدم نیاز آردوینو به یک برنامه‌ریز[۳۰۸] مجزا می‌باشد، چرا که با اتصال مستقیم برد به کامپیوتر و استفاده از نرم‌افزار مناسب می‌توان عملیات برنامه‌ریزی را انجام داد.[۳۲۲][۳۲۳]

۱-۲-۱۰- مشخصات آردوینو اونو

شرکت آردوینو بردهای مختلفی و با ویژگی‌های متعدد به بازار عرضه می‌نماید: آردوینو اونو[۳۰۹]، نانو[۳۱۰]، لئوناردو[۳۱۱] و مگا[۳۱۲] تنها چند نمونه از بردهای ارائه شده توسط این شرکت می‌باشند که در نوع سخت‌افزار استفاده‌شده، تعداد ورودی-خروجی ها و فرکانس و شرایط کاری و یا ابعاد متفاوت می‌باشند. نسخه‌ی مورد استفاده در این پروژه، آردوینو اونو (Uno) می‌باشد که از میکروکنترلر ATmega328P استفاده می‌نماید. تصویری از این برد در شکل زیر مشاهده می‌شود.

[306] Open-source
[307] Input/Output (I/O)
[308] Programmer
[309] Uno
[310] Nano
[311] Leonardo
[312] Mega

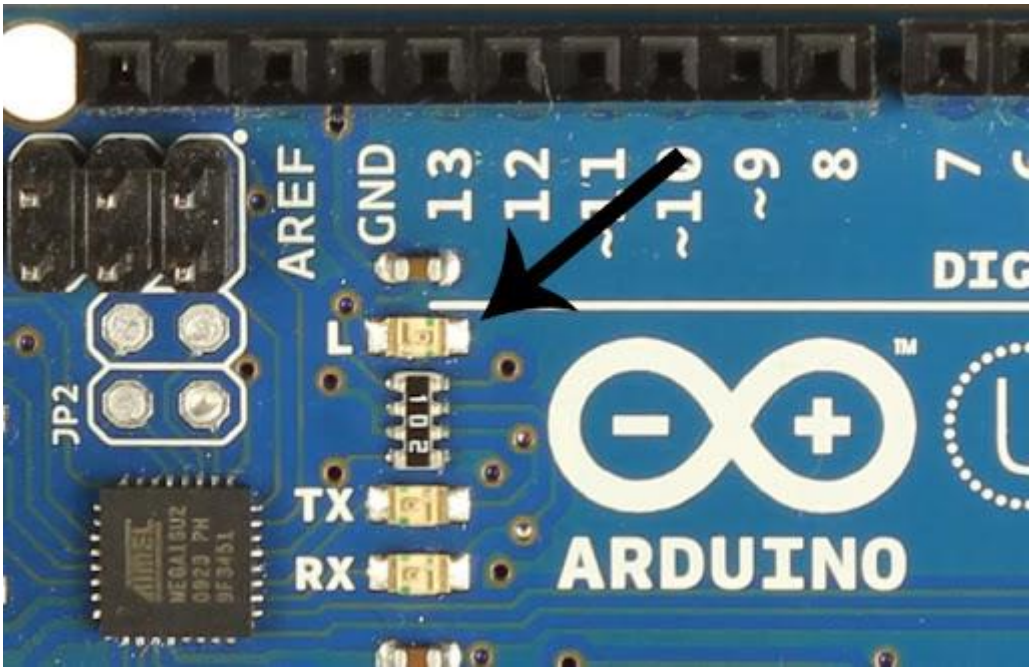
شکل ۸-۱۰ برد آردوینو اونو

مشخصات این برد در جدول زیر قابل ملاحظه است.[۳۲۴]

جدول ۲-۱۰ مشخصات برد آردوینو اونو

مقدار	ویژگی	مقدار	ویژگی
۳۲ کیلوبایت	حافظه‌ی فلش	ATmega328P	میکروکنترلر
۲ کیلوبایت	حافظه‌ی SRAM	۵ – ۱۲ ولت	ولتاژ ورودی
۱ کیلوبایت	حافظه‌ی EEPROM	۵ ولت	ولتاژ کاری
۱۶ مگاهرتز	سرعت ساعت	۱۴	تعداد پین‌های ورودی-خروجی دیجیتال
۶۸.۶ میلی‌متر	طول	۶	تعداد پین‌های دیجیتال با خروجی PWM
۵۳.۴ میلیمتر	عرض	۶	تعداد پین‌های ورودی آنالوگ
۲۵ گرم	وزن	۲۰ میلی‌آمپر	جریان مستقیم هر پین

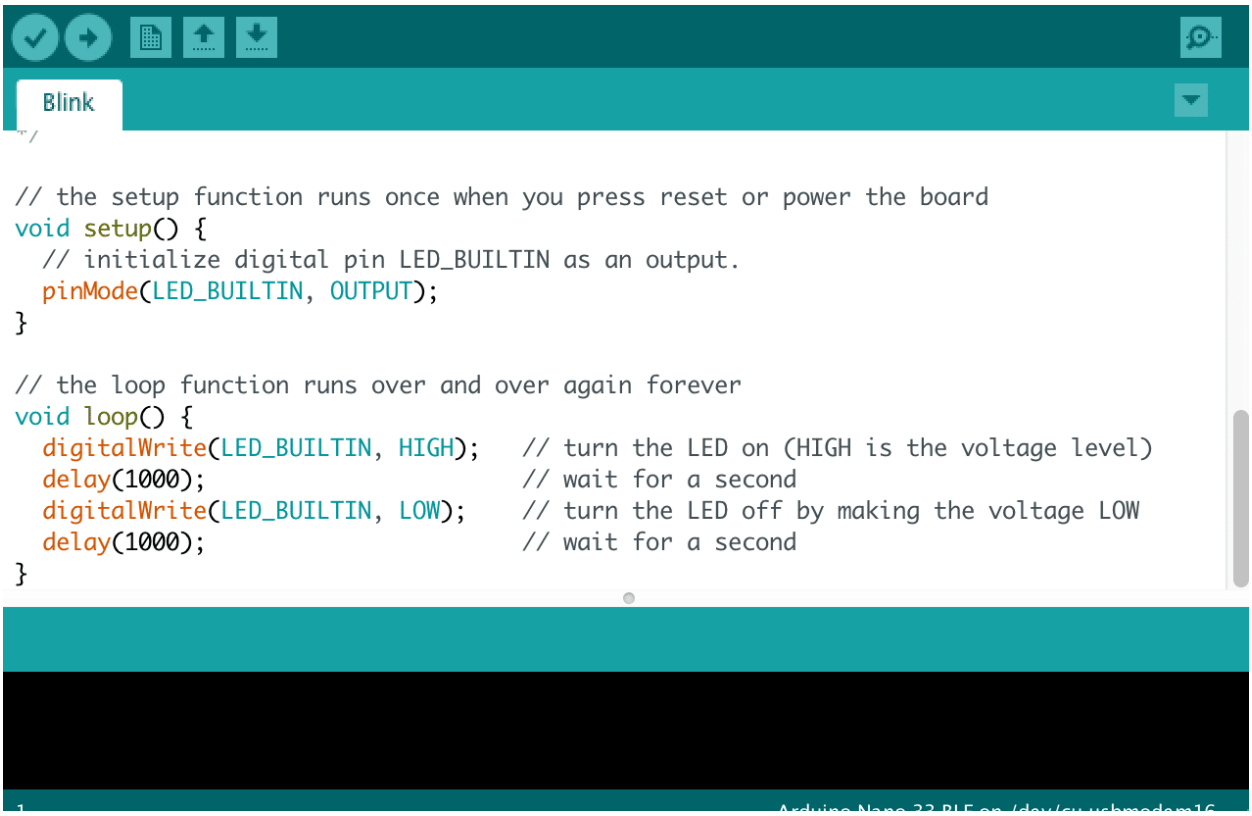
برای تغذیه‌ی این برد نیز، می‌توان از اتصال یک منبع ولتاژ ۵ ولت به صورت مستقیم به برد استفاده کرد. در غیر این‌صورت می‌توان از سوکت USB متصل به خود برد استفاده کرد. نکته‌ی دیگر در مورد این برد آن است که دارای یک چراغ LED متصل به پین شماره‌ی ۱۳ می‌باشد که می‌توان از آن برای کاربردهای نظیر اطمینان از عملکرد برد استفاده کرد. نمایی از این چراغ در شکل زیر قابل مشاهده می‌باشد.



شکل ۹-۱۰ چراغ LED سرخود بورد آردوینو اونو

۲-۲-۱۰- نحوه‌ی کار با آردوینو

برای کار با آردوینو، پیش از راه‌اندازی برد و اتصال قطعات، می‌بایست نرم‌افزار اجرایی را بر روی بورد بارگذاری کرد. برای این‌کار، می‌توان در محیط برنامه‌نویسی (IDE) مخصوص آردوینو، کد نرم‌افزار مورد نظر نوشته می‌شود. این محیط توسعه‌ی نرم‌افزاری توسط خود شرکت آردوینو ارائه می‌شود.[۳۲۵] برنامه‌ی نوشته‌شده برای برد آردوینو به زبان C++ بوده و شامل دو قسمت تنظیمات اولیه (Setup) و همچنین حلقه‌ی اصلی (Loop) می‌باشد.



شکل ۱۰-۱۰ نمایی از محیط Arduino IDE

در قسمت تنظیمات اولیه، تنظیات پیکرندی دستگاه اعم از نحوه‌ی استفاده از پین‌های مختلف، فرکانس کاری، متغیرهای اولیه و غیره مشخص می‌شود. برای مثال تعیین می‌گردد که پین شماره‌ی ۱۳ به عنوان خروجی مورد استفاده قرار گیرد.

در قسمت حلقه، بخش‌هایی از برنامه که نیاز است در طول برنامه به‌طور مداوم اجرا شوند نوشته می‌شود. در هر چرخش این حلقه، تمامی کدهای این قسمت اجرا خواهند شد. استفاده از بخش حلقه برای انجام وظایفی مانند طراحی کنترل‌کننده و خواندن و نوشتن اطلاعات بسیار مفید است.

پس از تکمیل کد مورد نظر، با استفاده از کامپایلر IDE، کد کامپایل شده و سپس به کد ماشین تبدیل می‌شود. این کد ماشین در یک فایل با پسوند hex ذخیره شده و بر روی بورد بارگذاری می‌شود. پس از بارگذاری نرم‌افزار بر روی برد، می‌توان سایر ماژول‌های مورد استفاده را به برد متصل کرده و از آن استفاده کرد.

۳-۲-۱۰- پروتکل‌های ارتباطی پشتیبانی‌شده در آردوینو اونو

برای آن‌که برد آردوینو در انواع پروژه‌های قابل استفاده باشد، نیاز است که دارای تمهیدات ارتباطی گوناگونی باشد که بورد بتواند با سایر میکروکنترلرها، کامپیوترها و ماژول ها ارتباط برقرار کند. پروتکل‌های ارتباطی که در آردوینو اونو وجود دارند به شرح زیر است:

۱. UART
۲. SPI
۳. I2C

۳-۱۰- جتسون نانو

برد توسعه جتسون نانو (Jetson Nano) انویدیا یک کامپیوتر کوچک و قدرتمند است که امکان اجرای الگوریتم های یادگیری عمیق را در بستر سیستم عامل فراهم می کند. این برد با بهره مندی از GPU تعبیه شده، می تواند چندین شبکه عصبی را به موازات اجرا می‌کند و همزمان اطلاعات چندین سنسور را پردازش کند.

شکل ۱۱-۱۰ برد توسعه جتسون نانو

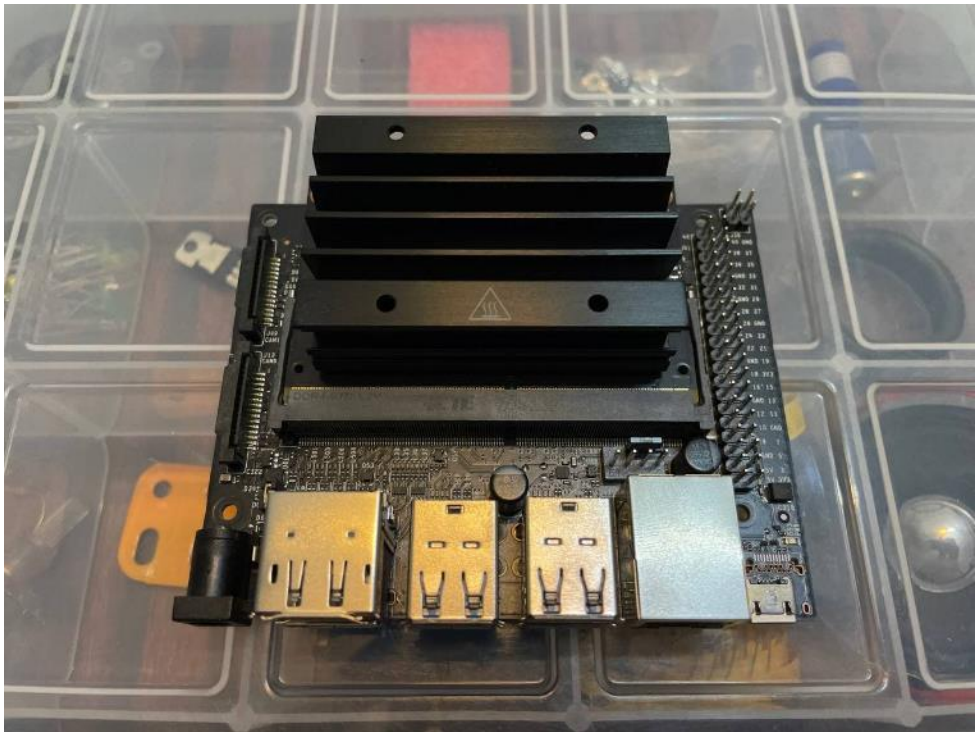
جتسون نانو توسط پکیج نرم افزاری NVIDIA JetPack پشتیبانی می‌شود که شامل سیستم عامل ubuntu به همراه کتابخانه‌های نرم افزاری CUDA، cuDNN و TensorRT برای یادگیری عمیق، بینایی کامپیوتر، TensorFlow، PyTorch، و سایر ابزارهای هوش مصنوعی می باشد. لذا این برد جهت انجام پردازش های سطح بالای موجود در خودروی خودران استفاده شده است.

برد جتسون نانو همانند مغز یک خودرو خودران وظیفه ادراک کامل محیط و استخراج اطلاعات از آن را برعهده دارد. با تجمیع این اطلاعات فرآیند برنامه ریزی و تصمیم گیری در این برد صورت می گیرد و نتجیه آن جهت کنترل خودرو به برد آردوینو ارسال می شود. اطلاعات دریافتی شامل نتیجه پردازش نقشه محیط، تصویر دوربین خودرو و خروجی سنسور فاصله سنج آلتراسونیک می باشد.

۱-۳-۱۰- مشخصات

در مرکز این برد پردازنده مرکزی چهار-هسته‌ای Cortex®-A57 مبتنی بر آرم در کنار پردازنده گرافیکی ۱۲۸-هسته‌ای Maxwell قرار گرفته اند. این برد همچنین دارای ۴ گیگابایت حافظه رم از نوع LPDDR4 بوده و از اسلات کارت حافظه برای ذخیره اطلاعات استفاده می کند.

این برد از رابط اترنت10/100/1000BASE-T با انتقال خودکار برای اتصال به اینترنت پشتیبانی می کند. همچنین دو کانکتور دوربین MIPI-CSI در کنار برد قرار گرفته است. این برد دارای یک پورت HDMI 2.0 برای اتصال نمایشگر بوده و ۴ عدد پورت USB نیز بر روی آن قرار داده شده است. همچنین سایر رابط های GPIO, I2C, I2S, SPI, UART نیز پشتیبانی می شوند.



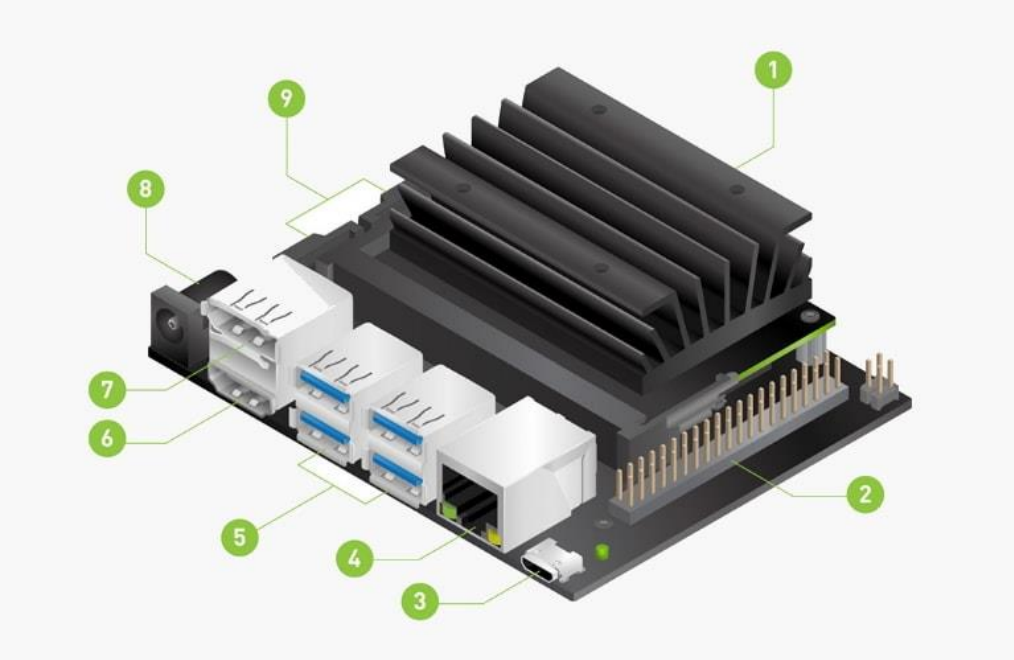
شکل ۱۲-۱۰ برد توسعه جتسون نانو

تغذیه برد از طریق اتصالات میکرو USB و یا جک DC با ولتاژ ۵ ولت صورت می گیرد که استفاده آن را در شرایط مختلف، ممکن می سازد. جدول زیر مشخصات کامل برد جتسون نانو را نشان می دهد.

جدول ۳-۱۰ مشخصات برد جتسون نانو

Hardware	Specification
GPU	**128-core Maxwell**
CPU	**Quad-core ARM A57 @ 1.43 GHz**
Memory	**4 GB 64-bit LPDDR4 25.6 GB/s**
Storage	**microSD**
Video Encode	**4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)**
Video Decode	**4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)**
Camera	**2x MIPI CSI-2 DPHY lanes**
Connectivity	**Gigabit Ethernet, M.2 Key E**
Display	**HDMI and display port**
USB	**4x USB 3.0, USB 2.0 Micro-B**
Others	**GPIO, I^2C, I^2S, SPI, UART**
Mechanical	**69 mm x 45 mm, 260-pin edge connector**

بخش های مختلف برد توسعه جتسون نانو (Jetson Nano) نیز در ادامه آورده شده است.



شکل ۱۳-۱۰ برد توسعه جتسون نانو

۱) اسلات SD کارت: کارت ۱۶ گیگابایت یا بیشتر برای ذخیره‌سازی اصلی و نوشتن فایل سیستم
۲) هدر توسعه ۴۰-پین
۳) پورت میکرو USB: برای ورودی برق 5V یا برای انتقال داده با USB
۴) پورت اترنت: 10/100/1000BASE-T با انتقال خودکار
۵) چهار پورت USB 3.0
۶) پورت خروجی HMDI
۷) اتصال پورت نمایشگر
۸) جک DC: برای ورودی تغذیه ۵ ولت
۹) کانکتور دوربین MIPI CSI

۲-۳-۱۰- سیستم عامل

پکیج نرم افزاری انویدیا، Jetpack نام دارد که شامل جدیدترین بسته درایور Jetson Linux Driver (L4T) به همراه سیستم عامل لینوکس، کتابخانه های CUDA-X و API های یادگیری عمیق و بینایی رایانه می باشد.

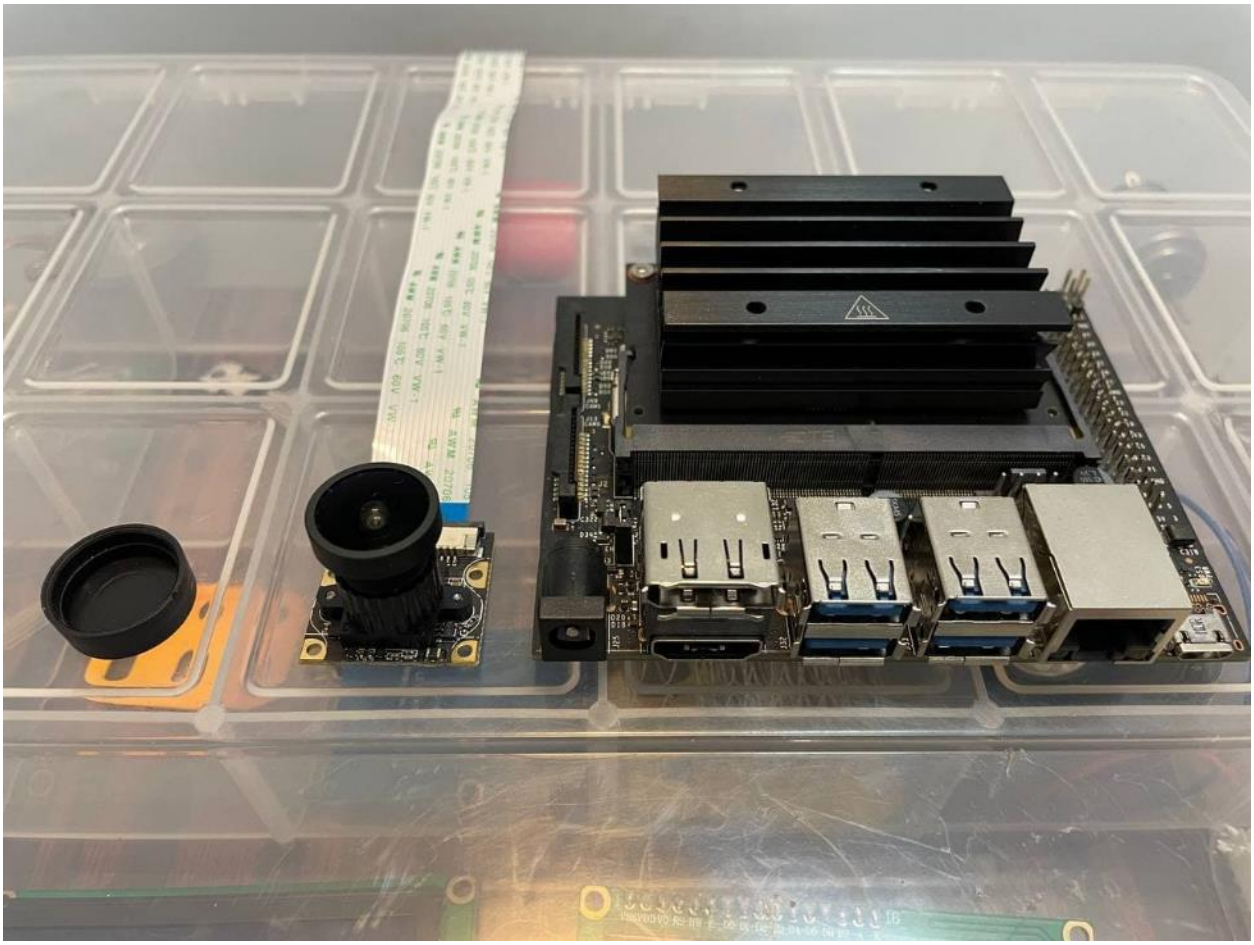
NVIDIA L4T هسته لینوکس ۴.۹، سیستم عامل ubuntu 18.04 ، درایورهای NVIDIA و سایر موارد دیگر را فراهم می کند. همچنین L4T شامل کودا[۳۱۳] و cuDNN بوده که پیاده سازی های بهینه را برای لایه های استاندارد درشبکه های عصبی مانند convolution، pooling، normalization و activation فراهم می کند. کتابخانه OpenCV نیز به عنوان مرجعی برای بینایی رایانه، پردازش تصویر و یادگیری در پکیج نرم افزاری انویدا قرار گرفته است. تصویر زیر محیط سیستم عامل را در برد امبدد جتسون نانو نمایش می دهد.

شکل ۱۴-۱۰ سیستم عامل برد جتسون نانو

[313] CUDA

۳-۳-۱۰- دوربین

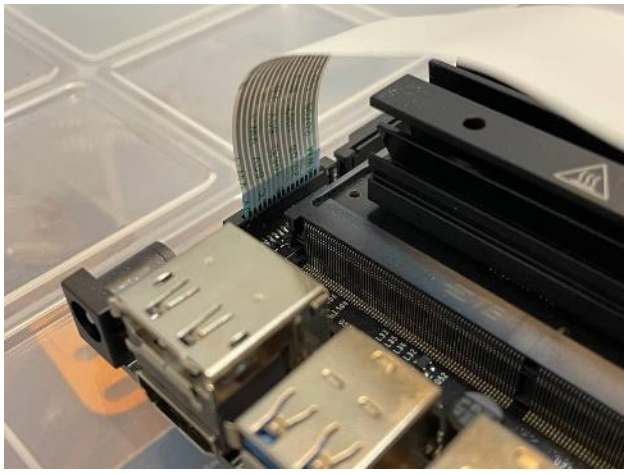
در این پروژه از یک دوربین ۱۲.۳ مگاپیکسلی با رزولوشن ۴۰۵۶ × ۳۰۴۰ پیکسل برای دید خودرو خودران استفاده شده که توسط توسط شرکت Waveshare تولید شده است. این دوربین براساس سنسورSony IMX477 ساخته شده است که وضوح بالاتری نسبت به سنسورهای رایج تر مانند IMX219 دارد و می تواند تا ۹۰ فریم در ثانیه تصویر برداری کند. این دوربین به واسطه سنسور بزرگ آن، تصاویر واضحی ثبت می کند و همچنین قابلیت فوکوس دستی متغیر را فراهم می کند. این دوربین با میدان دید گسترده خود قادر به ثبت اطلاعات زیادی در جلوی خودرو می باشد.



شکل ۱۵-۱۰ برد جتسون نانو در کنار دوربین مورد استفاده

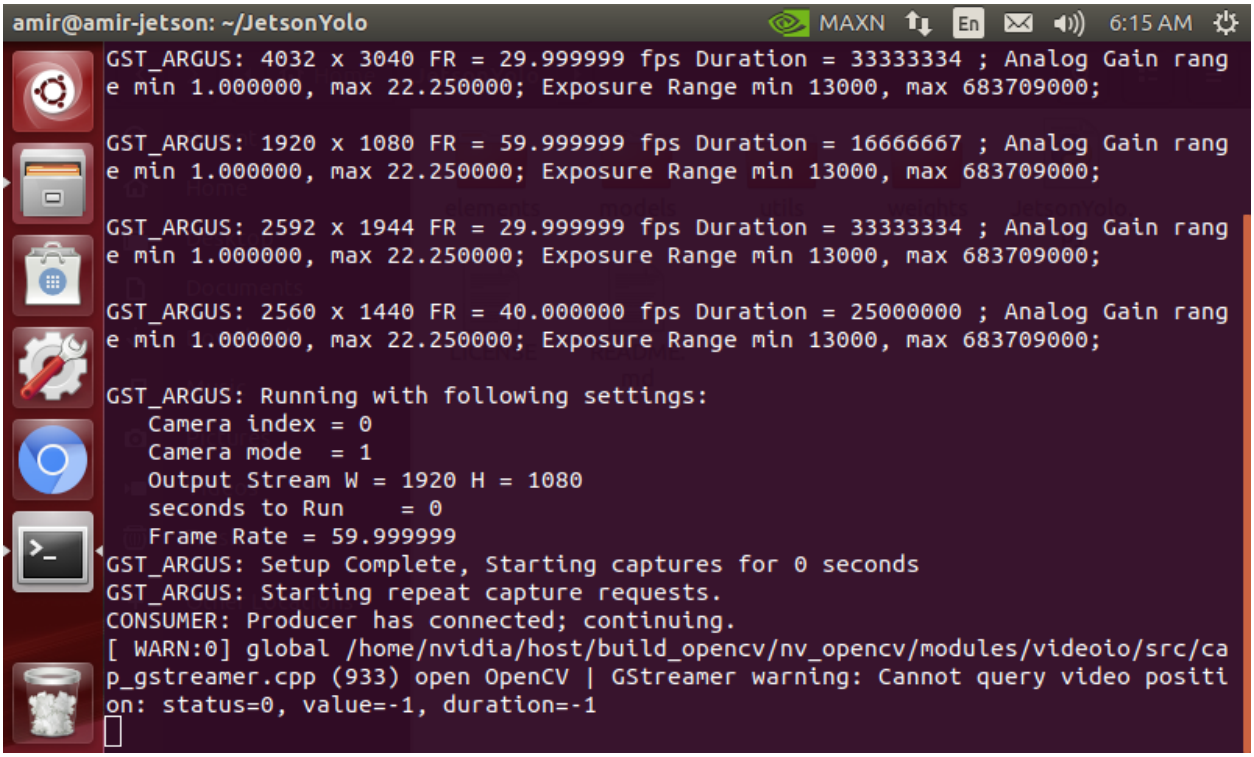
دوربین توسط رابط MIPI-CSI به برد متصل می شود. پهنای باند بالا برای انتقال داده و راحتی اتصال از دلایل استفاده از دوربین با کانکتور ذکر شده است. رابط MIPI-CSI باس پردازنده را به منظور انتقال اطلاعات مشغول نمی کند لذا میزان کار پردازنده را در ثبت تصویر به حداقل می رساند و توان پردازشی برد بر روی فرآیند ادراک خودرو متمرکز می شود[۳۲۶].

شکل ۱۶-۱۰ دوربین



شکل ۱۷-۱۰ ابعاد دوربین در مقایسه با برد

شکل زیر یک پایپلاین مناسب برای انتقال تصویر را نشان می دهد. همان طور که در شکل زیر مشاهده می شود، تنظیمات رزولوشن تصویر بر روی ۱۹۲۰ در ۱۰۸۰ قرار گرفته تا پردازش تصاویر راحت تر شود. نرخ فریم نیز ۶۰ انتخاب شده است.



شکل ۱۸-۱۰ تنظیمات مناسب برای انتقال تصاویر

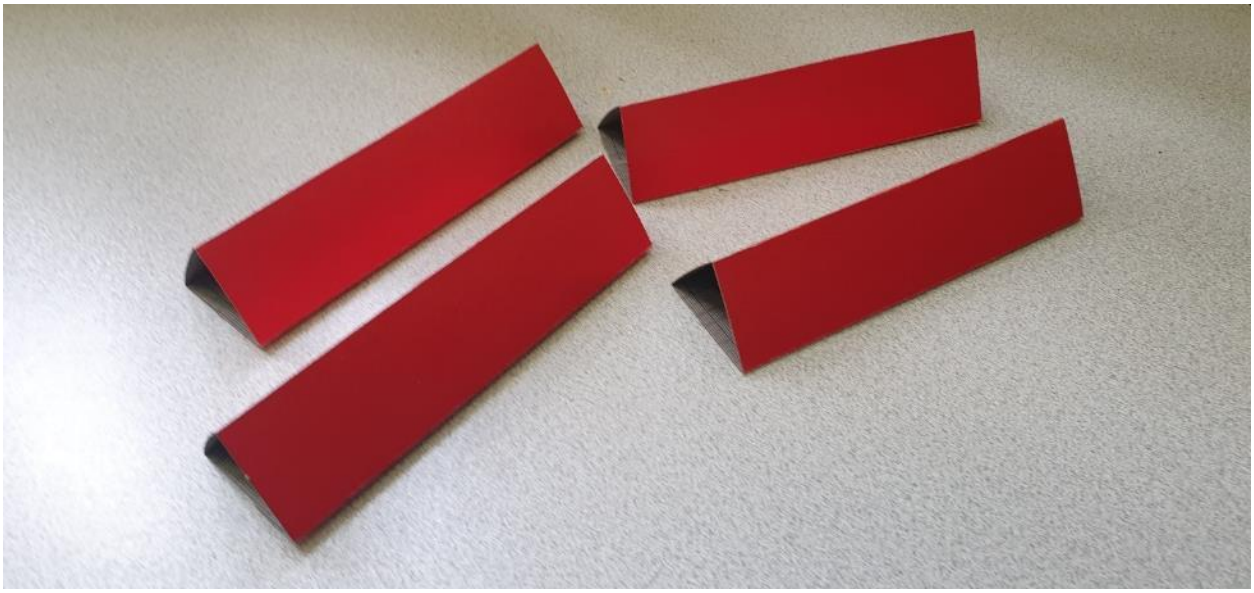
۴-۱۰- محیط تست فیزیکی

برای تست عملکرد های معرفی شده در بخش های قبلی، سه نقشه ی مختلف در ابعاد یک به هشت با محیط واقعی طراحی و پیاده سازی شده اند. جنس استفاده شده برای پیاده سازی جاده ها، کاغذ مخصوص ماکت سازی با ضریب اصطکاک $\mu_s = 0.5$ استفاده شده است تا حداکثر شباهت به محیط جاده های شهری و بین شهری ایجاد گردد.

با توجه به این که رنگ سیاه این جنس از کاغذ ها، بر خلاف رنگ آسفالت خیابان ها جاذب نور اتاق هستند و نوری از سطح آن ها بازتاب نمی گردد، برای آزمایش اثر بازتاب نور روی جاده و عدم حذف شدن این بازتاب، از تکنیک رنگ وارونه[۳۱۴] استفاده شده و برای جاده ها رنگ سفید در نظر گرفته شده است تا اثر بازتاب نور بر سطح جاده نیز آزمایش گردد.

سه نقشه ی معرفی شده در این بخش، در مجموع حدود سی متر مربع مساحت دارند و هر کدام ویژگی های مختص به محیط تست خود را دارا می باشند.

دو نوع مانع در این سه نقشه طراحی و ساخته شده است که به صورت دو بعدی و سه بعدی می باشند. موانع سه بعدی به رنگ قرمز رنگ و مانند نقشه با ابعاد یک به هشت نسبت به موانع حقیقی موجود در جاده ها مانند سطل های زباله، عابرین پیاده و خودرو های مجاور می باشند که در مپ های شهری و بین شهری استفاده شده اند. موانع دو بعدی نیز به رنگ سیاه و برای شبیه سازی محیط های پارک اشغال شده و دیوار ها استفاده شده اند که برای استفاده در مپ پارک خودکار و هنگام پیاده سازی الگوریتم های مسیر یابی مد نظر قرار می گیرند.

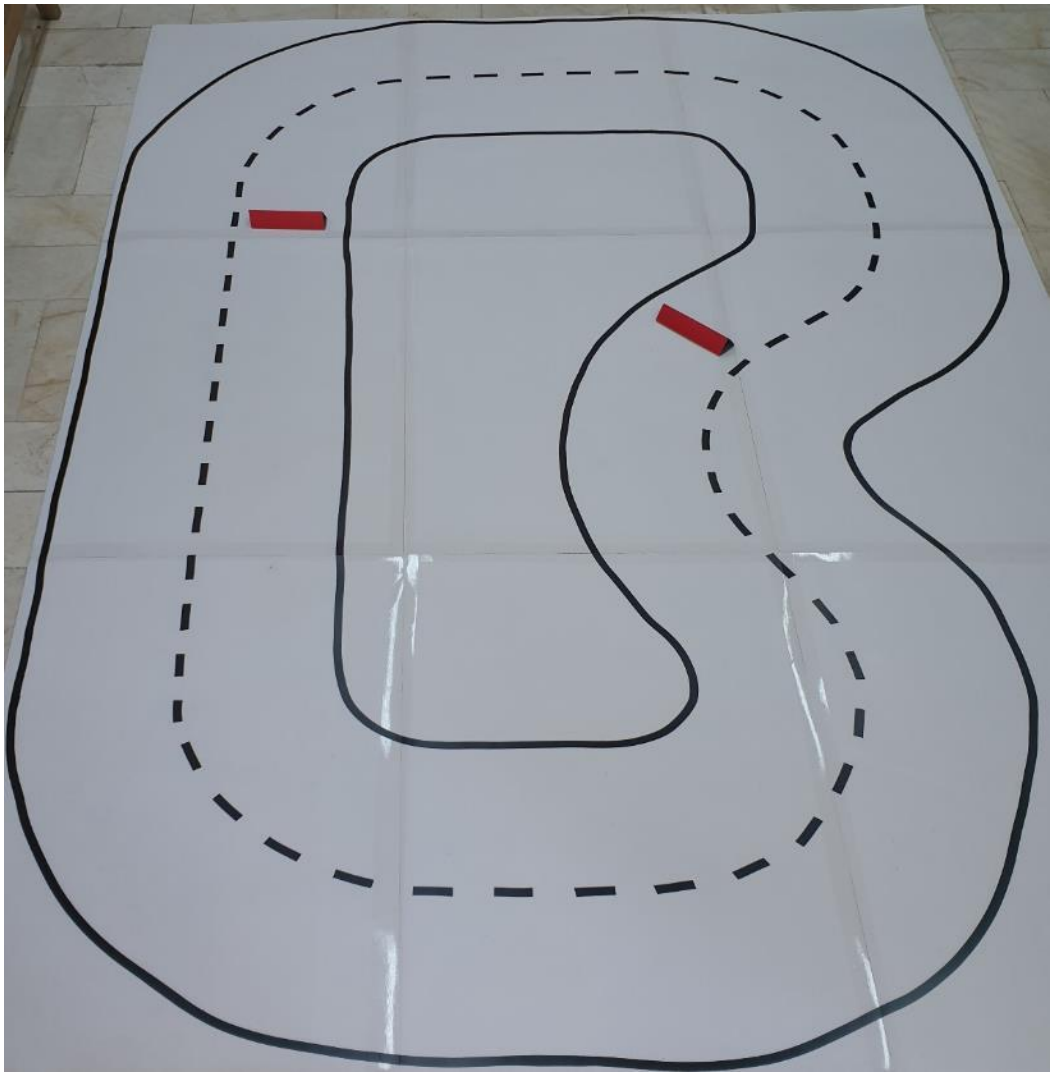


شکل ۱۹-۱۰ نمونه ای از موانع سه بعدی ساخته شده

[314] reverse

۱-۴-۱۰- محیط بین شهری

این محیط در ابعاد ده متر مربع ساخته شده است و شامل خطوط ممتد حاشیه جاده، خطوط مقطع درون جاده و موانع سه بعدی قرمز رنگ در طول جاده می باشد.



شکل ۲۰-۱۰ نقشه ی ساخته شده برای محیط بین شهری

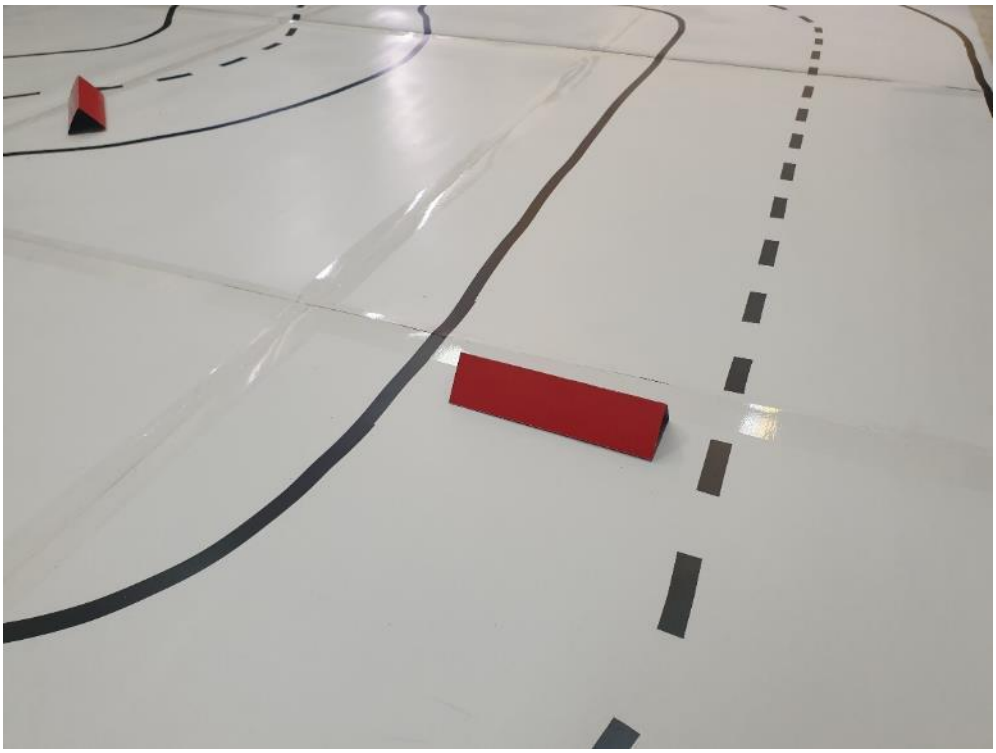
چهار شیب ساعت گرد و پاد ساعتگرد در مپ بین شهری تعبیه شده است تا بحث چرخش زاویه ای چرخ ها توسط سرو موتور در سرعت بالا آزمایش گردد که دو عدد از آن ها در طول مسیر و دو عدد از آن ها در بالا و پایین مسیر قابل مشاهده اند.

شکل ۲۱-۱۰ نمونه ای از شیب های تعبیه شده درون نقشه ی بین شهری

شکل ۲۲-۱۰ نمونه ای از شیب های تعبیه شده درون نقشه ی بین شهری

همچنین موانعی در طول جاده ی بین شهری تعبیه شده اند تا بحث سنسور های اولترا سونیک خودرو و پردازش تصویر مورد آزمایش قرار گیرند.

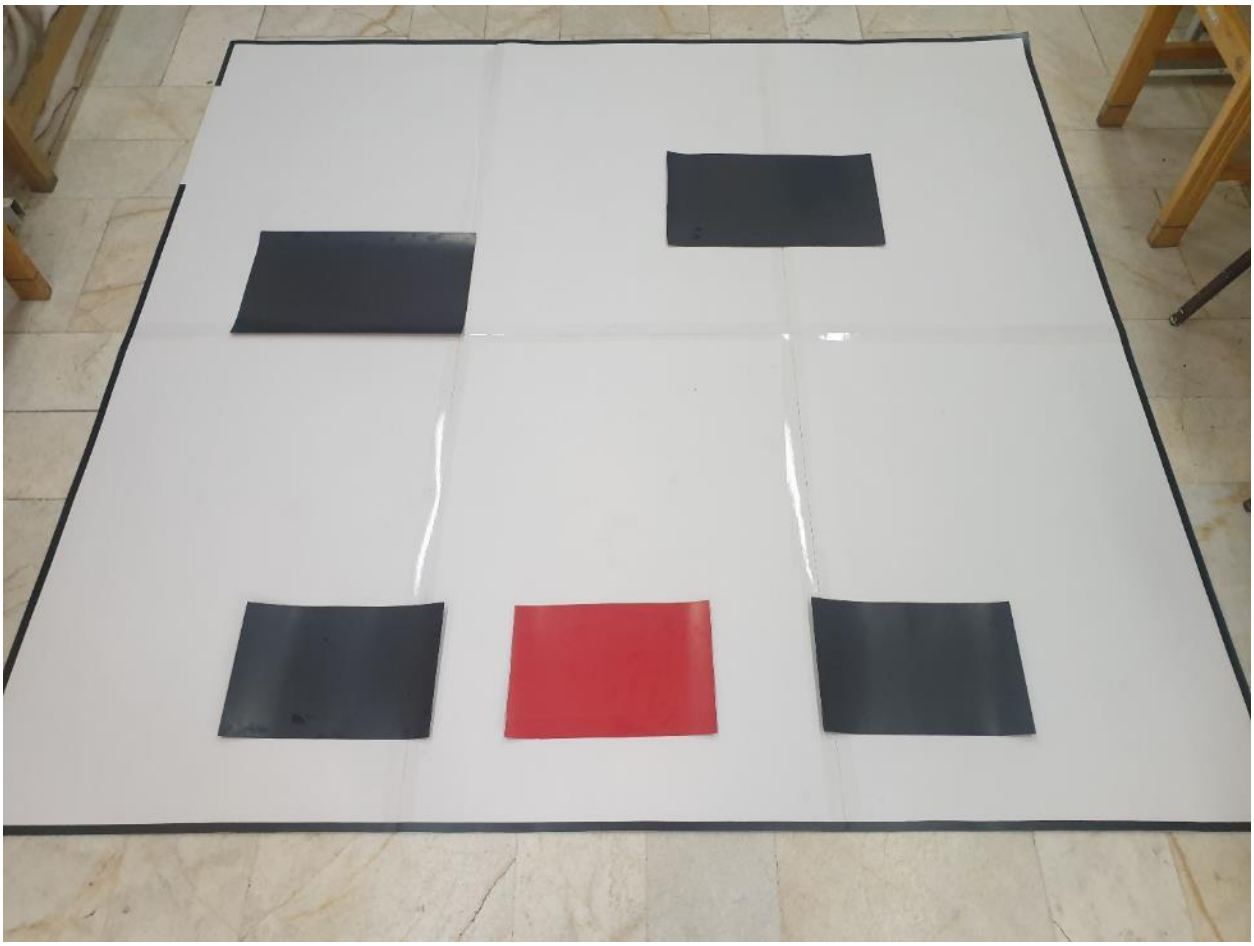
شکل ۲۳-۱۰ نمونه ای موانع سه بعدی قرار داده شده در طول مسیر



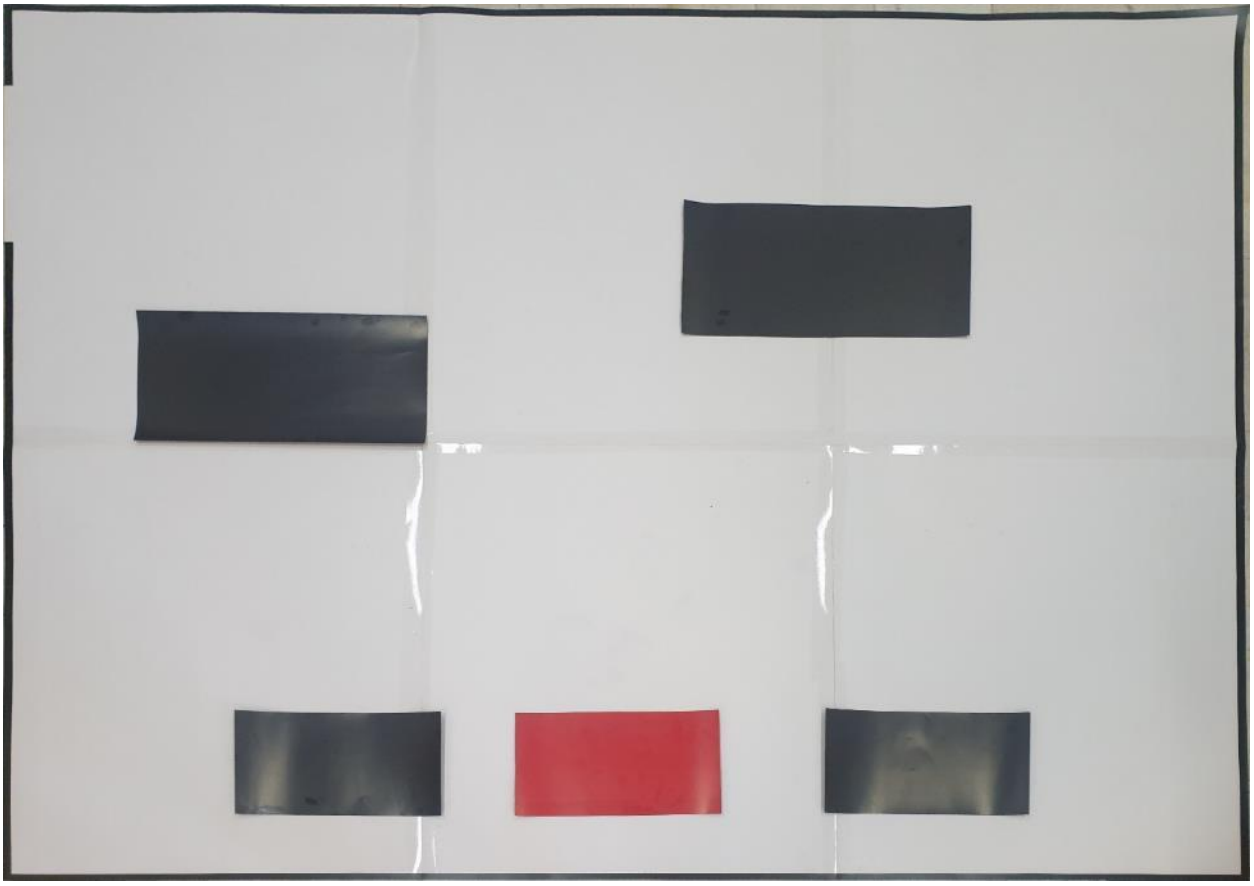
شکل ۲۴-۱۰ نمونه ای موانع سه بعدی قرار داده شده در طول مسیر

۲-۴-۱۰- محیط پارک خودکار

این محیط در ابعاد هشت متر مربع ساخته شده است و شامل خطوط ممتد حاشیه پارکینگ، موانع متعدد سیاه رنگ دو بعدی در داخل پارکینگ، و یک محیط قرمز رنگ هدف برای پارک نهایی خودروست که توسط دو محیط اشغال شده ی سیاه رنگ از بالا و پایین احاطه شده است.



شکل ۲۵-۱۰ نقشه ی ساخته شده برای محیط پارک خودکار



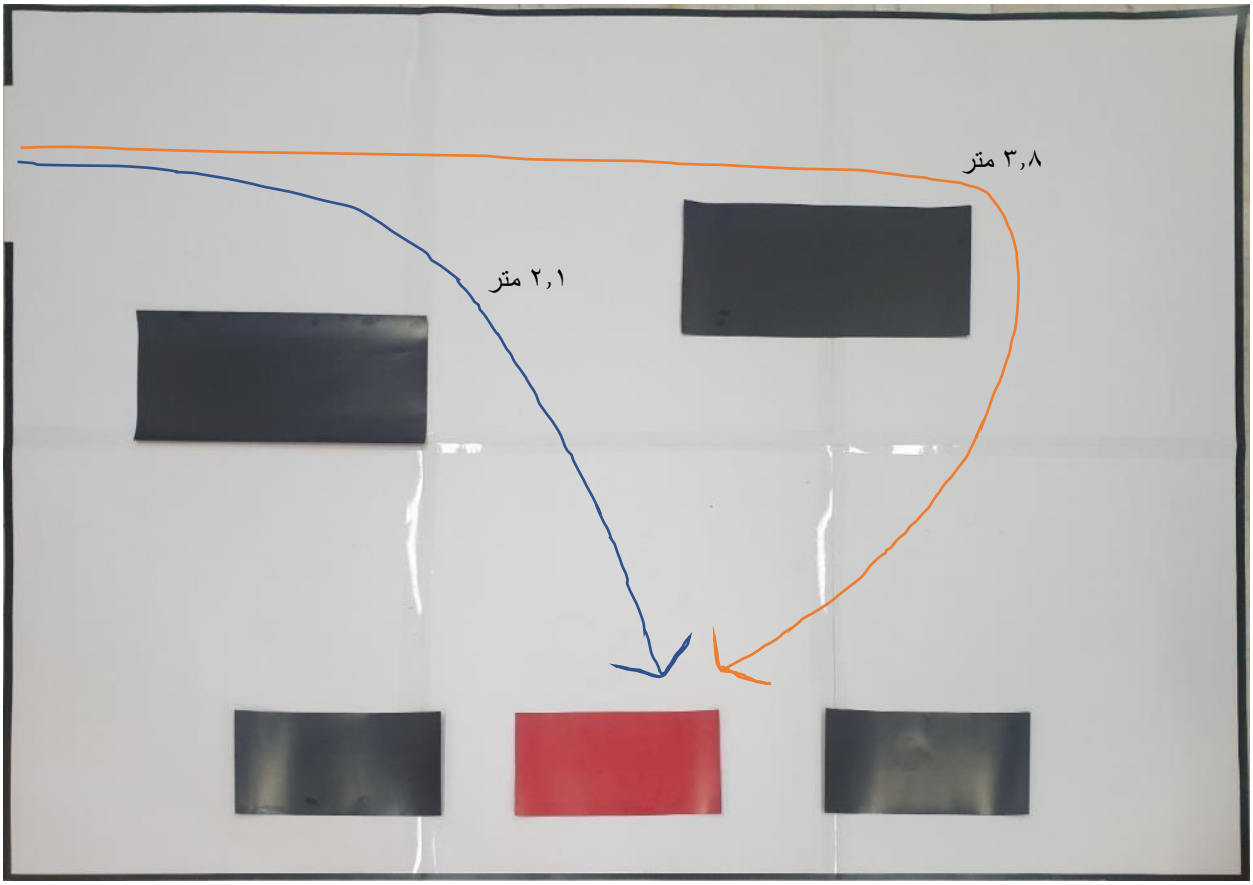
شکل ۲۶-۱۰ نقشه ی استراتژیک محیط پارک خودکار

عملکرد مورد آزمایش در این نقشه، عملکرد مسیریابی، الگوریتم A^* و پارک خودکار است که خودرو باید در بین مسیر های موجود در نقشه، کوتاه ترین آن ها را که مسیر موجود در بین دو مانع است، پیدا کند و از طریق آن عملیات پارک خودکار را انجام دهد.

این فرآیند از طریق اسکن نقشه از زاویه ی استراتژیک یا همان چشم پرنده[۳۱۵] و پیاده سازی الگوریتم های معرفی شده در بخش های قبلی روی نقشه انجام می گردد.

بلوک های سیاه رنگ بزرگتر، نشانگر موانعی مانند دیوار یا مسدود بودن مسیر عبور و بلوک های سیاه رنگ کوچکتر، نمایانگر محل های اشغال شده ی پارک می باشند. همچنین بلوک قرمز رنگ، نشان دهنده ی مقصد نهایی پارک خودرو است.

[315] Eye bird view



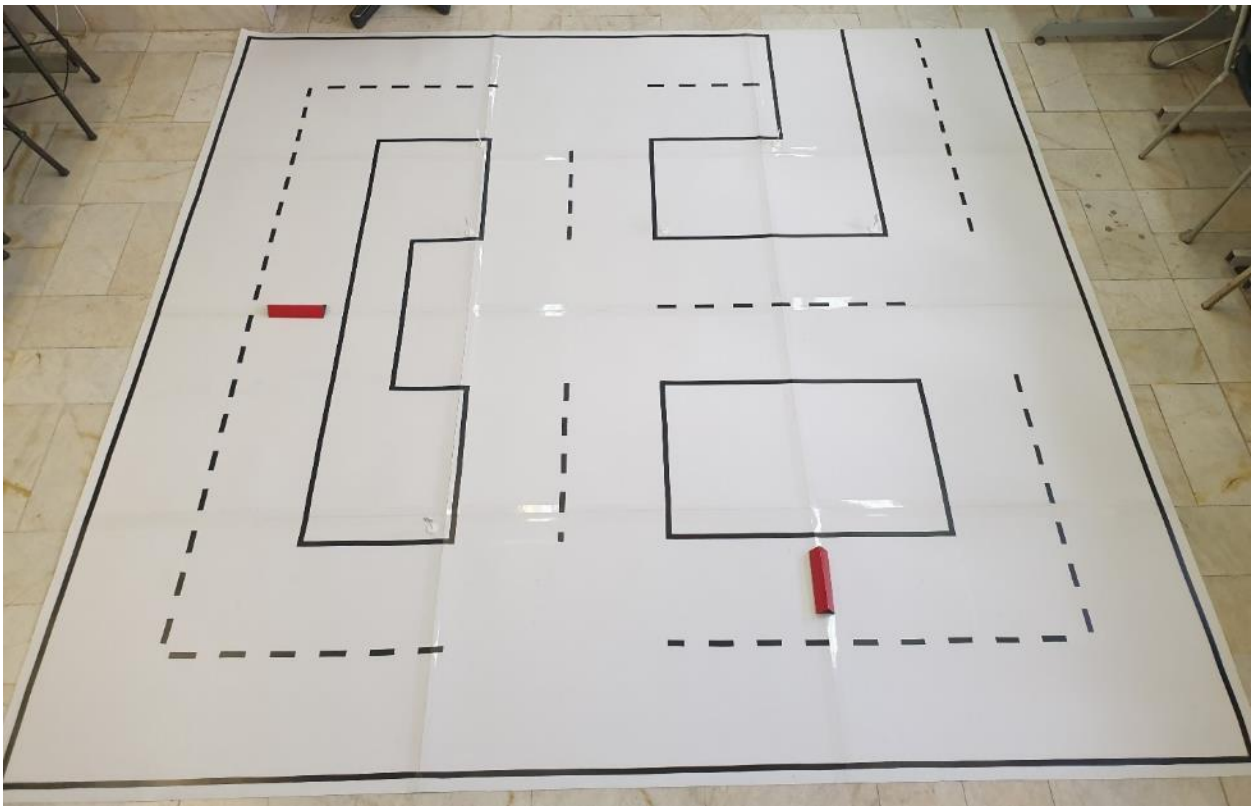
شکل ۲۷-۱۰ مقایسه ی مسیر های موجود برای مسیریابی

همچنین تنها یک ورودی برای محیط پارکینگ در نظر گرفته شده است و موانع موجود در نقشه قابلیت جا به جایی دارند.

شکل ۲۸-۱۰ ورودی نقشه پارکینگ

۳-۴-۱۰- محیط شهری

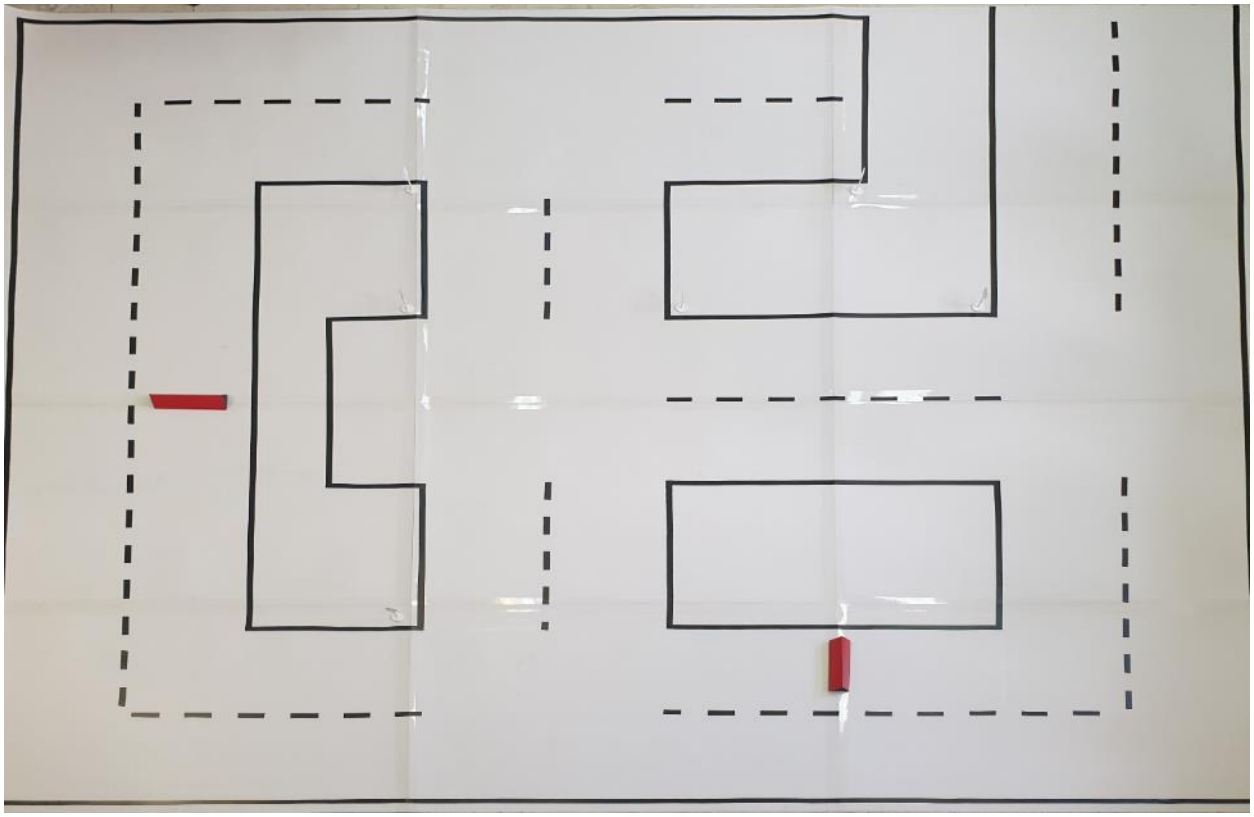
این محیط در ابعاد دوازده متر مربع ساخته شده است و شامل خطوط ممتد حاشیه جاده، خطوط مقطع درون جاده، موانع سه بعدی قرمز رنگ در طول جاده و تابلوهای حاوی علائم راهنمایی و رانندگی در اطراف جاده می باشد.



شکل ۲۹-۱۰ نقشه ی ساخته شده برای محیط شهری

در این نقشه، عملکرد های مورد ارزیابی شامل سیستم ترمز خودکار و هوشمند (هنگام مشاهده ی موانع در محیط شهری)، فرمان های کنترلی خودرو (چرخش به راست و چپ)، کنترلر PID طراحی شده برای خودرو و بررسی درصد اطمینان قانون مندی سیستم و نحوه ی واکنش به قوانین راهنمایی و رانندگی می باشد.

در این نقشه هم از تابلوهای راهنمایی و رانندگی و هم از موانع سه بعدی استفاده شده است که در ادامه در مورد همگی آن ها توضیحات تکمیلی ارائه می گردد.



شکل ۳۰-۱۰ نقشه ی استراتژیک محیط شهری

برای تعبیه ی تابلوهای راهنمایی و رانندگی در نقشه ی شهری، تابلوهای مورد استفاده در محیط تست، طراحی و چاپ گردیدند و بر روی استند های پلاستیکی متناسب با طول خودرو و ارتفاع استاندارد اسکیل شده، نصب گردیدند.

شکل ۳۱-۱۰ نمونه ای از تصاویر طراحی و چاپ شده برای تابلوهای راهنمایی و رانندگی

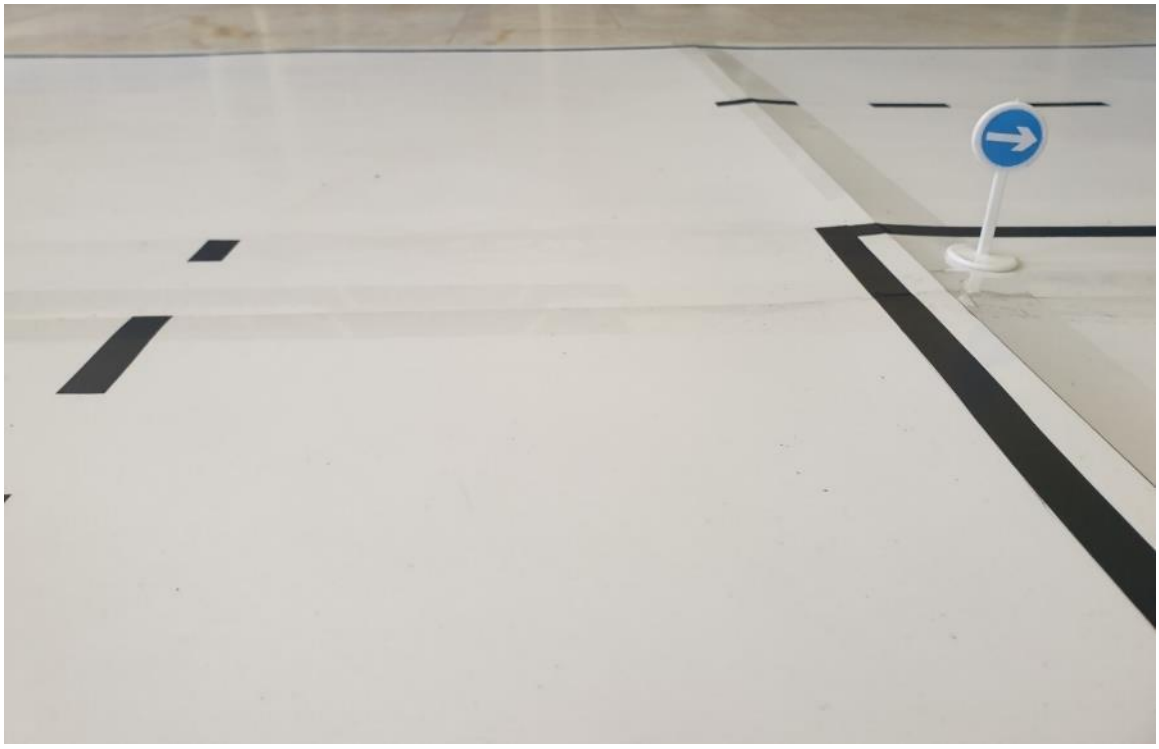
همچنین، تابلوهای متعدد دیگری برای استفاده های آینده و پروژه های پژوهشی در مقاطع بعدی نیز طراحی و ساخته گردیدند که در ادامه تعدادی از آن ها قابل مشاهده می باشند.

شکل ۳۲-۱۰ تعدادی از تابلوهای ساخته شده

در نهایت تابلوهای مورد استفاده در این تست، با توجه به طراحی انجام شده در کنار جاده ها قرار داده شدند و خودرو موظف است تا با استفاده از فرآیند های پردازش تصویری و با کمک الگوریتم های حوزه ی بینایی ماشین[۳۱۶]، تابلو ها را تشخیص و از یکدیگر تفکیک دهد و در ادامه، دستورات کنترلی دریافتی از پردازنده و کنترل کننده را روی سرو موتور و موتور DC پیاده سازی نماید.

[316] Computer vision

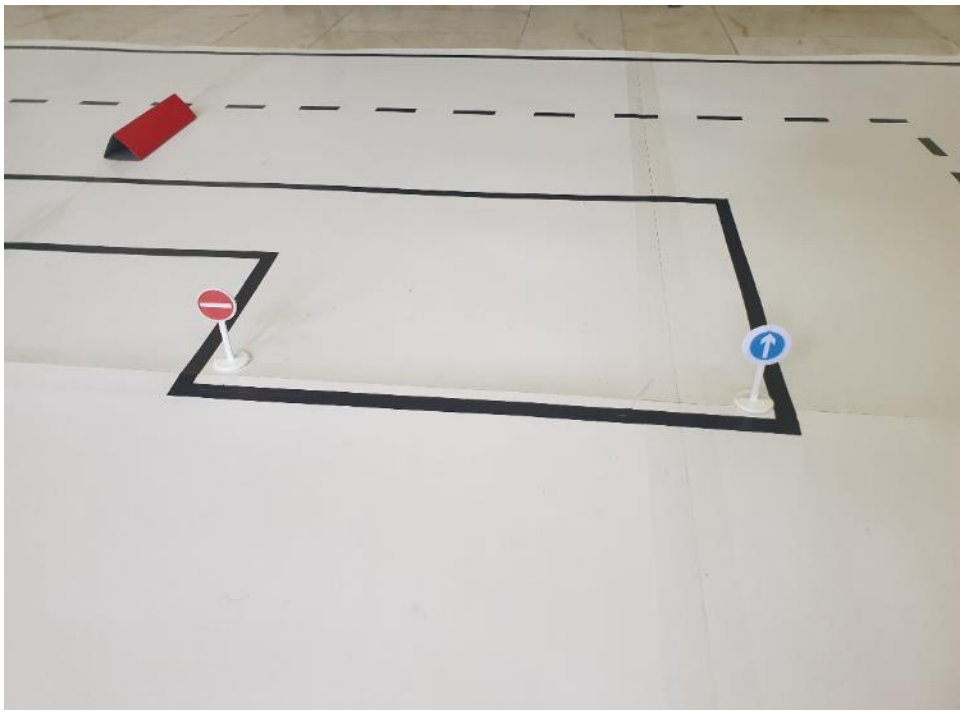
شکل ۳۳-۱۰ نمونه ای از تابلوهای راهنمایی و رانندگی در کنار خطوط جاده



شکل ۳۴-۱۰ نمونه ای از تابلوهای راهنمایی و رانندگی در کنار خطوط جاده

همچنین در ادامه، تعدادی مانع سه بعدی برای شبیه سازی حضور خودروهای مجاور، عابرین پیاده یا هرگونه مانع موجود دیگر در محیط های شهری نیز به نقشه اضافه گردید.

شکل ۳۵-۱۰ نمونه ای موانع سه بعدی قرار داده شده در طول مسیر



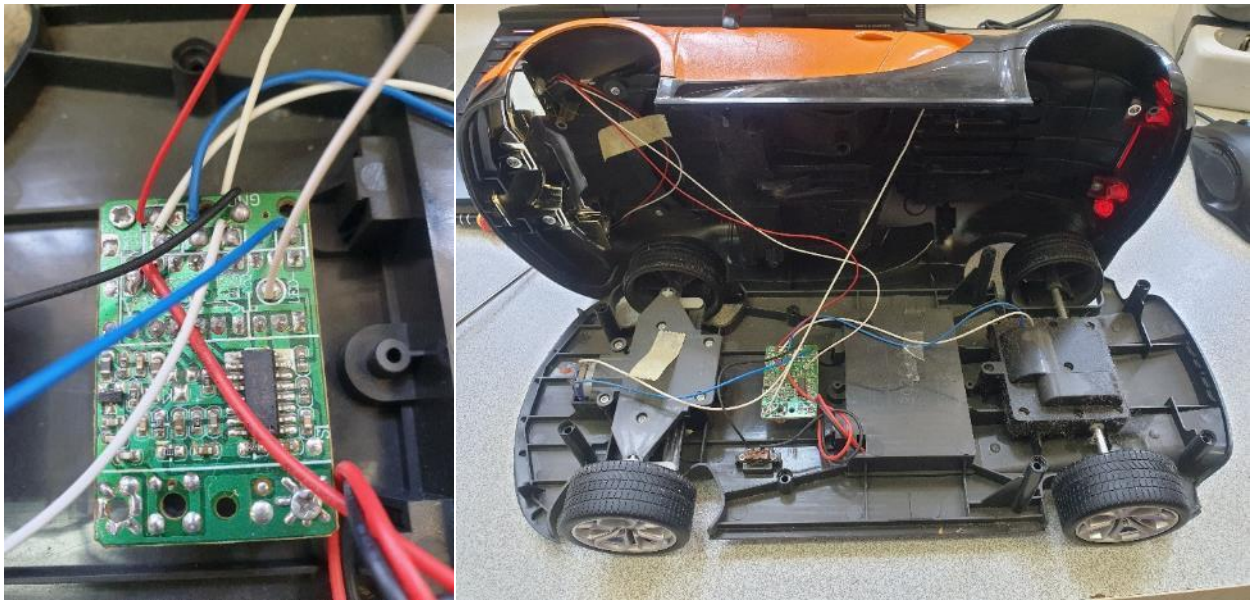
شکل ۳۶-۱۰ نمونه ای از موانع سه بعدی و تابلوهای قرار داده شده در طول مسیر

۵-۱۰- ساخت خودرو

در این بخش به شرح ساخت خودرو و مراحل آن پرداخته می‌شود. نکته قابل توجه این است که با توجه به شرایط کرونا و محدودیت های ایجاد شده، بخشی از کار حضوری و بخشی به صورت از راه دور و حتی با ارسال قطعات به شهرهای مختلف تکمیل شده است.

۱-۵-۱۰- اتصالات موتور

برای ساخت خودرو از یک ماشین کنترلی استفاده شده که در شکل ۳۷-۱۰ نشان داده شده است. کاور خودرو از شاسی آن جدا شده و اتصالات مربوط به LED ها و آنتن خودرو قطع شده است. بخش درایو خودرو که بر روی برد اصلی آن قرار داده شده است حفظ شده و پس از بررسی آن، به برد آردوییِنو جهت کنترل اتصال یافته است.

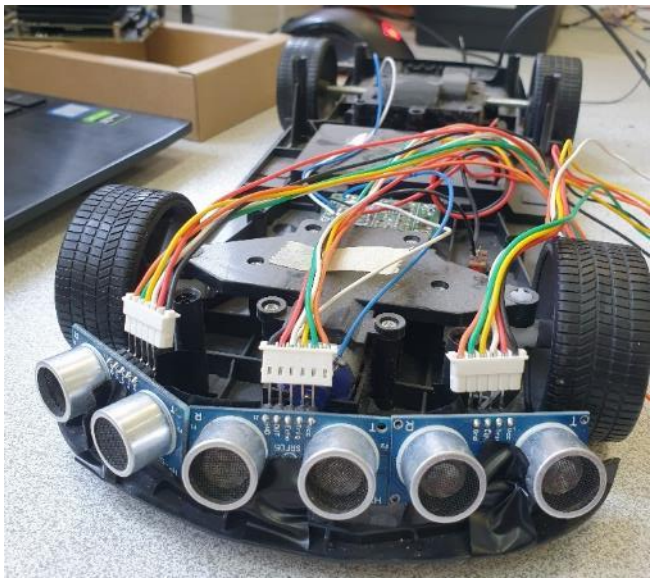


شکل ۳۷-۱۰ شاسی و برد اصلی خودرو جهت کنترل موتورها

یکی از چالش های اصلی، در دسترس نبودن اطلاعات مربوط به برد کنترلی خودرو می باشد که با بررسی سیگنال های ورودی و خروجی آن و اندازه گیری ولتاژها این چالش برطرف شده است.

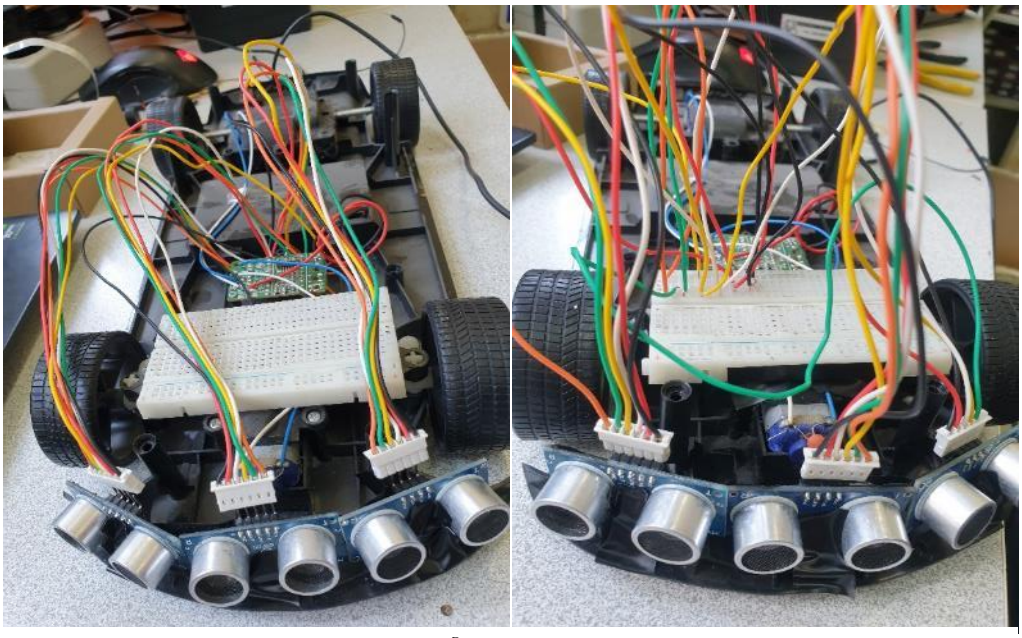
۲-۵-۱۰- اتصال سنسورهای آلتراسونیک

سه سنسور آلتراسونیک درجلوی خودرو قرار گرفته است و اتصالات آن شامل پایه های Vcc، Gnd، Echo و Trig، به برد برد وارد شده است. تغذیه سنسورها از طریق پاوربانک تامین شده و دستورات ورودی و خروجی آن از طریق آردوییینو داده می‌شود. شکل ۳۸-۱۰ محل قرارگیری سنسورها را نشان می‌دهد.



شکل ۳۸-۱۰ محل قرارگیری سنسورهای آلتراسونیک

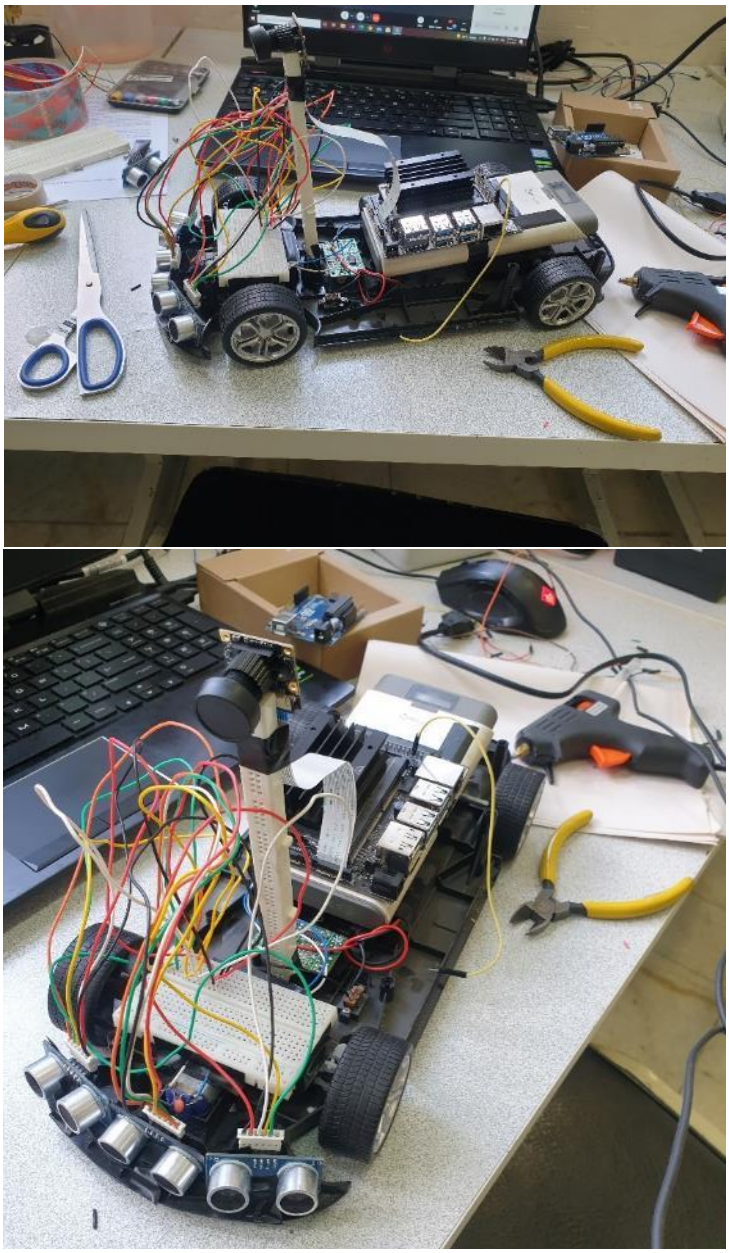
در شکل زیر نیز می توان اتصال تکمیلی سنسورها و نحوه اتصال به برد برد را مشاهده کرد.



شکل ۳۹-۱۰ اتصال سنسورهای آلتراسونیک به برد برد

۳-۵-۱۰- اتصال برد جتسون نانو و دوربین

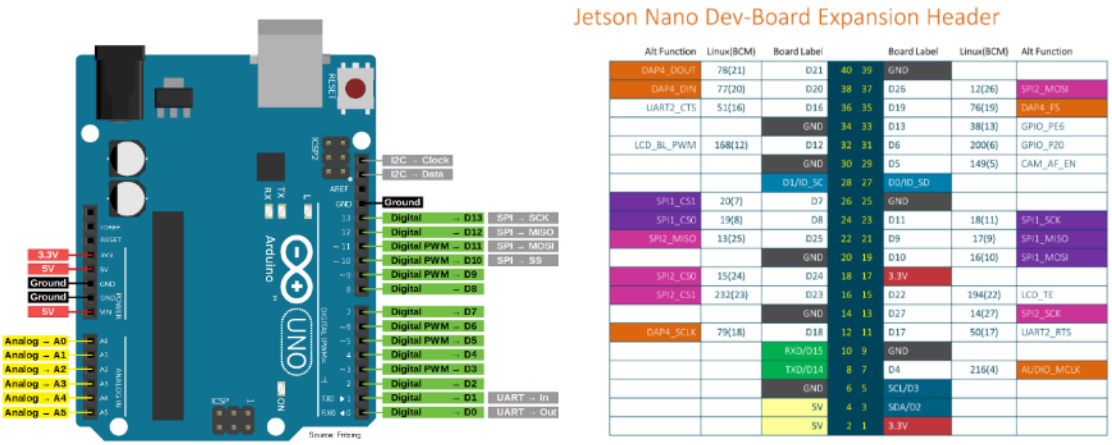
نصب درایور و اتصال دوربین به برد جتسون نانو به وسیله کانکتور CSI در بخش مربوطه اشاره شده است. در ادامه برد جتسون نانو به همراه یک پاوربانک برروی خودرو سوار شده و دوربین نیز بر روی پایه قرار می‌گیرد. تمامی اجزا به کمک چسب برق، محکم شده است.



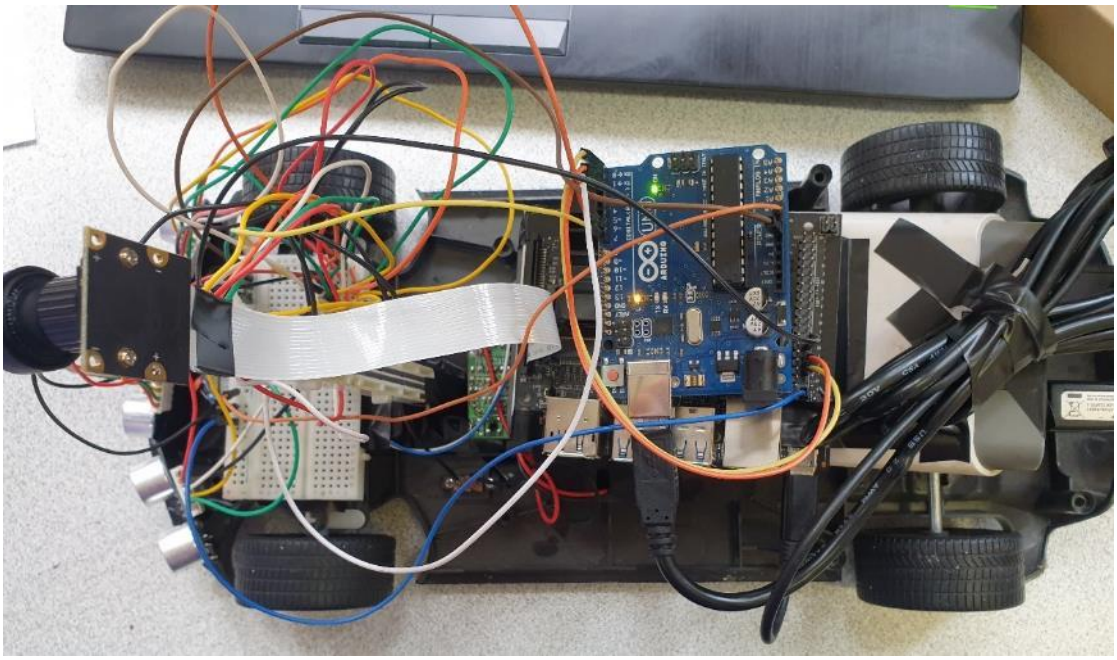
شکل ۴۰-۱۰ قرار گیری برد جتسون نانو به همراه دوربین

۴-۵-۱۰- اتصال برد آردوییِنو

برای برقراری ارتباط بین بردهای جتسون نانو و آردوییِنو Uno از ارتباط سریال استفاده شده است. با توجه به خروجی پین های این دو برد در شکل ۴۱-۱۰، می‌توان در شکل ۴۲-۱۰ مشاهده کرد که پین‌های ۸ و ۱۰ برد جتسون نانو به پین‌ها ۱ و ۲ برد آردوییِنو Uno متصل شده است.



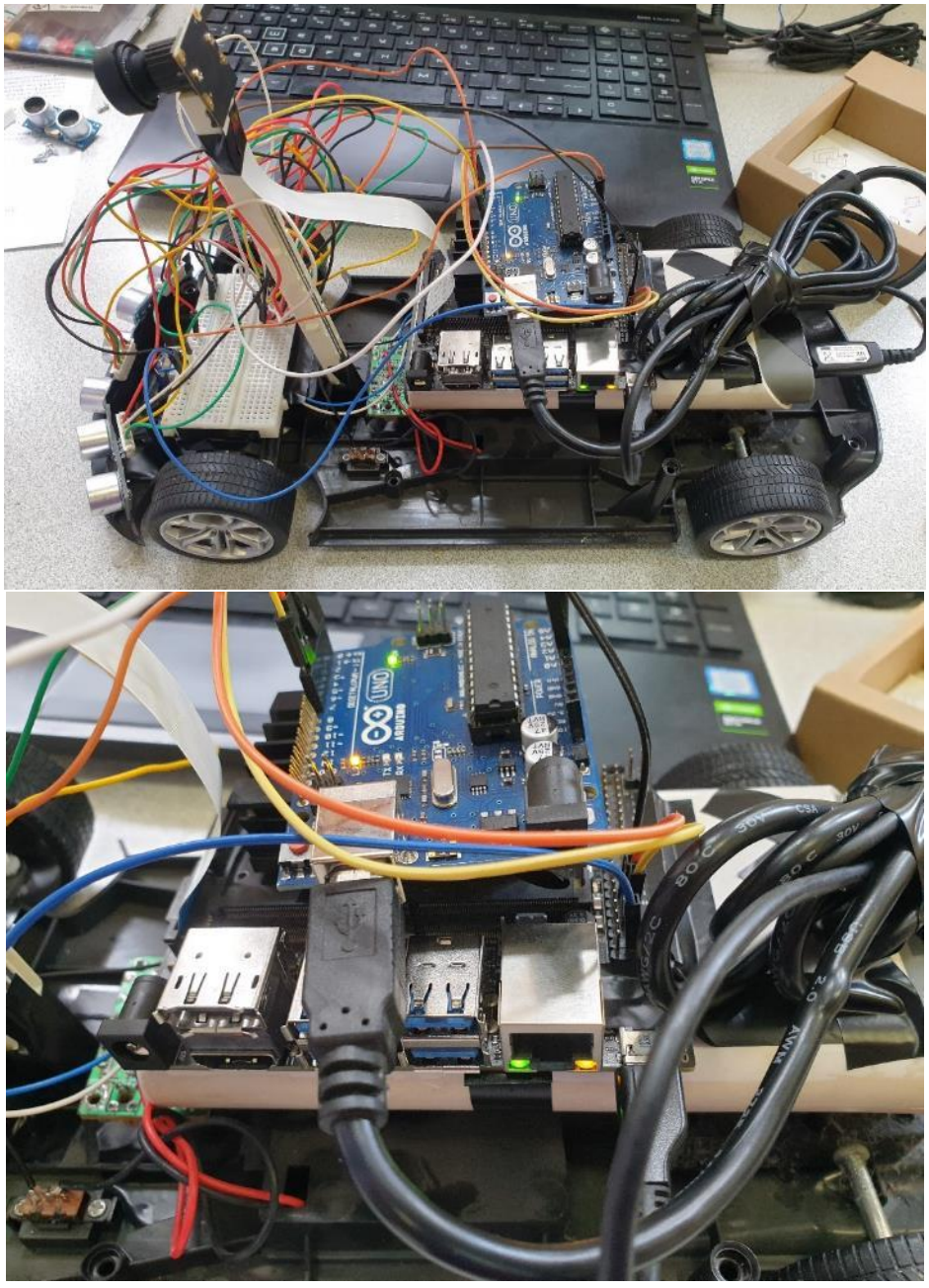
شکل ۴۱-۱۰ خروجی پین‌های برد جتسون نانو و آردوییِنو Uno



شکل ۴۲-۱۰ اتصال سریال (RX/TX) بردهای جتسون نانو و آردوییِنو Uno

۵-۵-۱۰- تغذیه خودرو

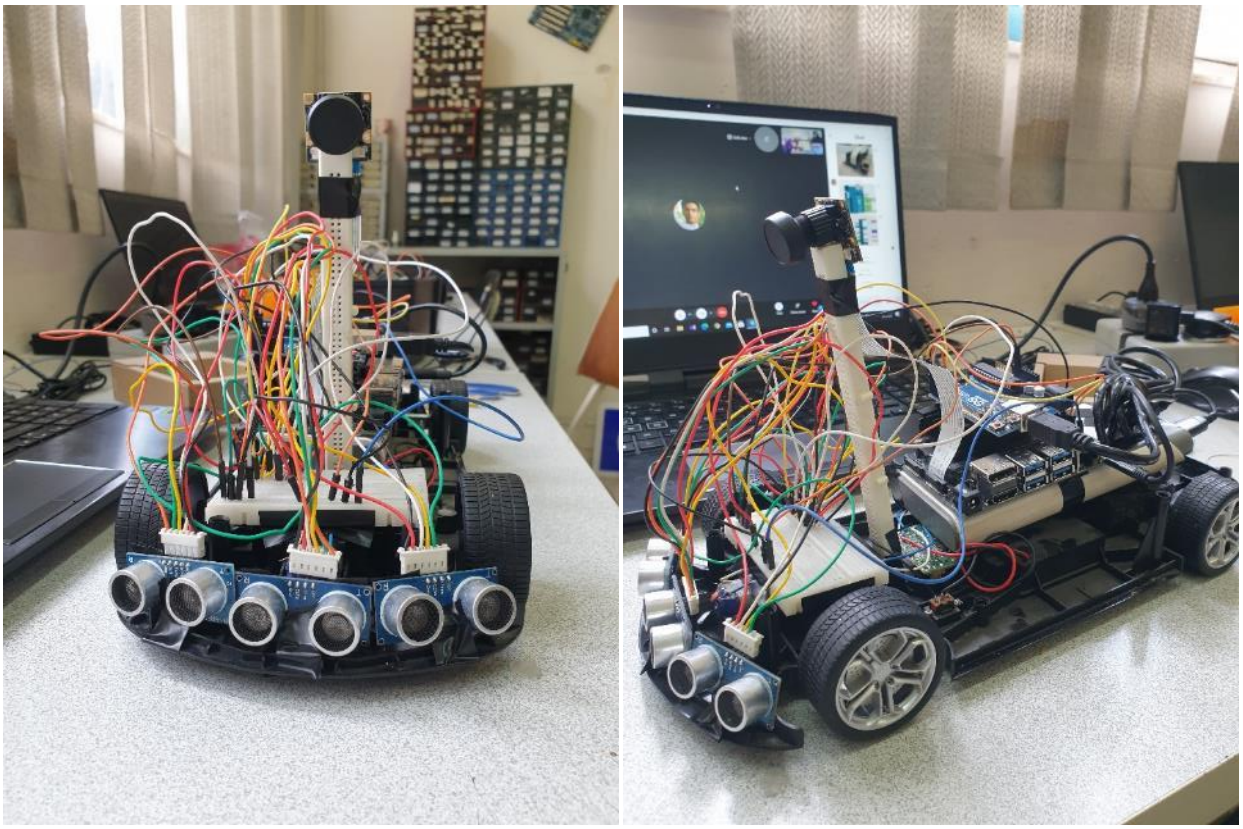
تغذیه موتورهای خودرو توسط محل باتری که در زیر آن تعبیه شده تامین می‌شود. برای تغذیه ۲ برد موجود برروی خودرو و سنسورهای آلتراسونیک از یک پاوربانک استفاده شده که در زیر برد جتسون نانو قرار گرفته است. ولتاژ خروجی این پاوربانک ۵ ولت بوده و توسط ۲ سیم USB به بردها متصل شده است. لازم به ذکر است که تمامی گراندها به یکدیگر متصل شده اند.



شکل ۴۳-۱۰ تغذیه اجزای مختلف خودرو

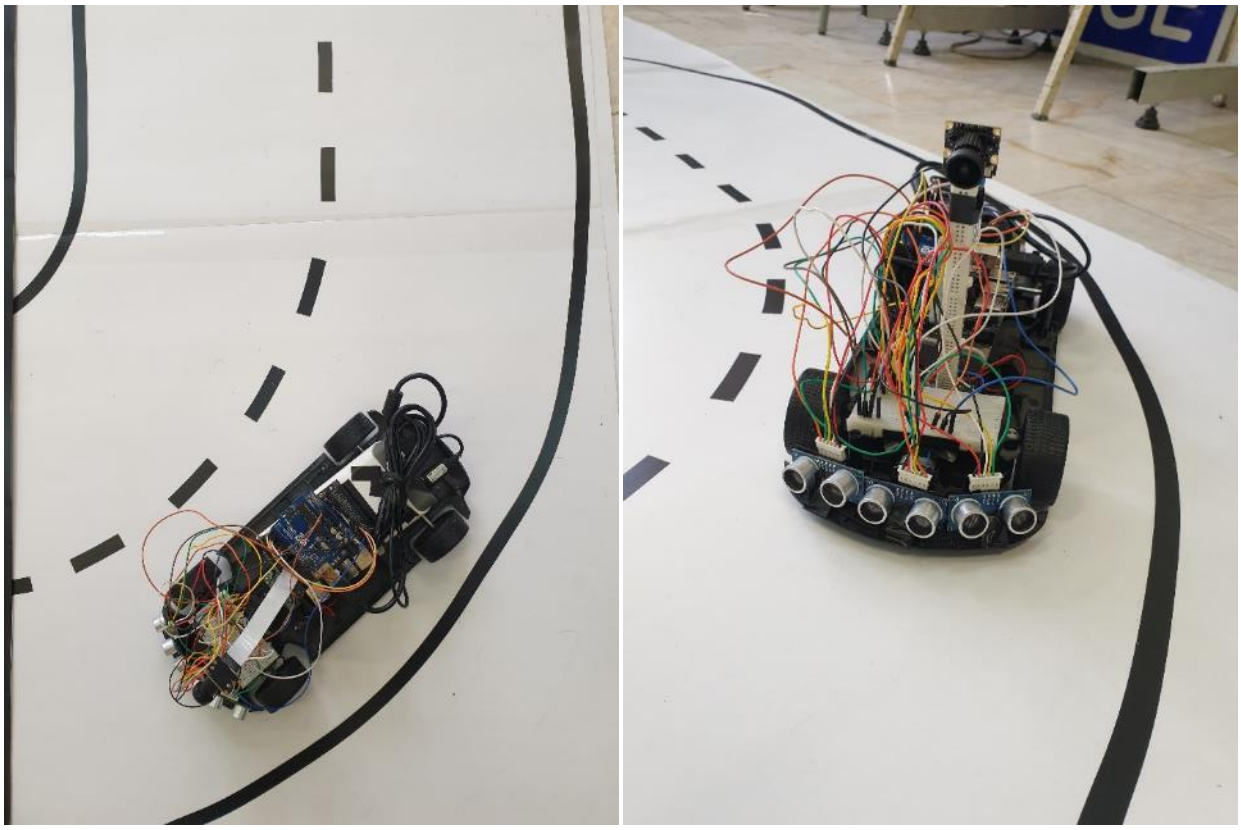
۶-۵-۱۰- نتیجه ساخت

نتیجه خودروی ساخته شده در شکل ۴۴-۱۰ و ۴۵-۱۰ نشان داده شده است. همانطور که در شکل ۴۴-۱۰ مشاهده می‌شود، بخشی از فرآیند ساخت به صورت از راه دور با استفاده از تماس ویدیویی صورت گرفته است و تمامی اعضا در ساخت مشارکت داشته اند.



شکل ۴۴-۱۰ نتیجه نهایی ساخت خودرو

مطابق شکل ۴۵-۱۰، تمامی اجزای خودرو در محل خود محکم شده و با قرارگیری خودرو برروی نقشه مقیاس آن نشان داده شده است.



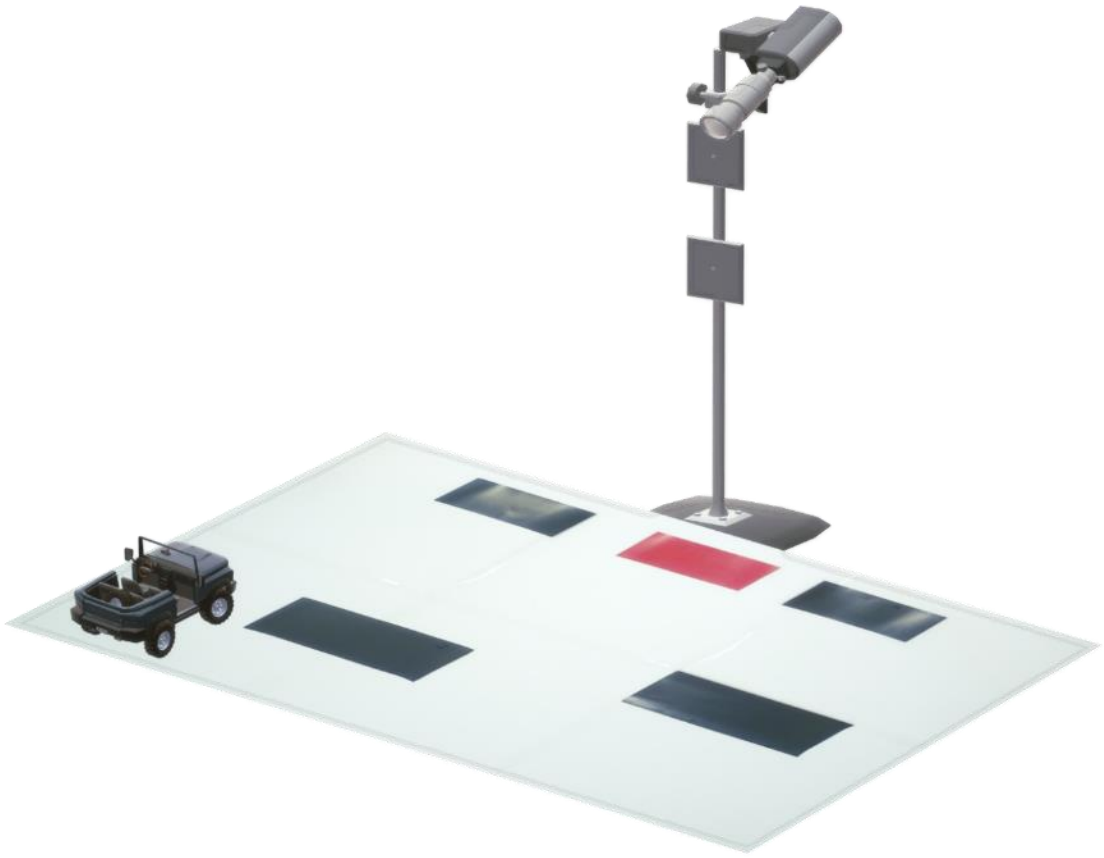
شکل ۴۵-۱۰ محل و مقیاس قرارگیری خودروی ساخته شده بر روی نقشه

۱۱- نتایج

۱-۱۱- برنامه ریزی مسیر در نقشه پارک

در فصل ۵ شبیه سازی پارک خودکار در محیط پایتون مورد بررسی گرفت. در این بخش فرآیند اعمال الگوریتم های پارک خودکار در محیط واقعی مورد بحث قرار می‌گیرد. همانطور که در بخش قبل نیز اشاره شد، نقشه پارکینگ شامل تعدادی بلوک مشکی و یک بلوک قرمز می باشد. بلوک‌های مشکی نشان دهنده موانع و سایر خودروهای موجود در پارکینگ بوده و بلوک قرمز محل پارک نهایی را نشان می‌دهد. هدف این بخش، یافتن کوتاه ترین مسیر مناسب برای رسیدن به محل پارک مورد نظر می باشد به طوری که خودرو با موانع برخورد نکند.

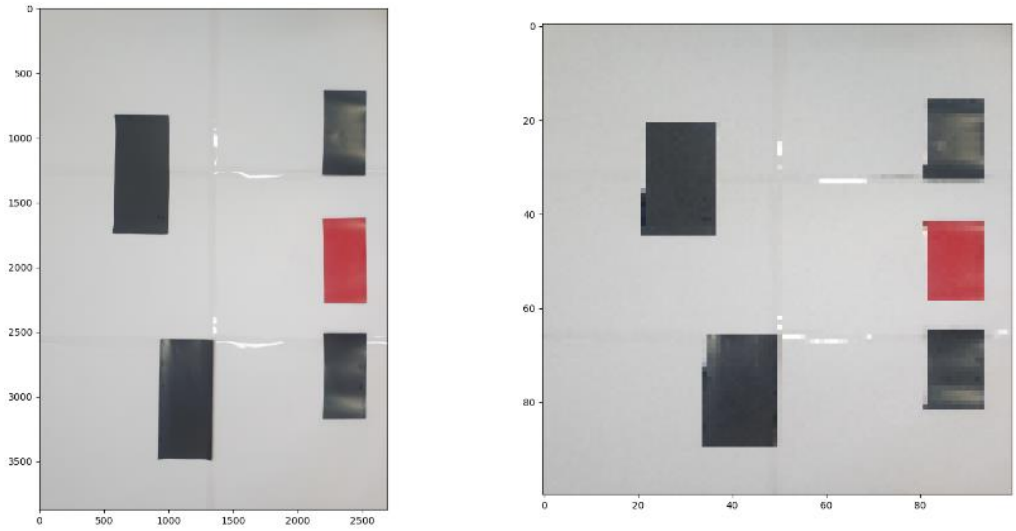
شکل زیر مدل سه‌بعدی ستاپ پارک خودکار را نشان می‌دهد. دوربین از بالای نقشه، تصویری از موانع و محل نهایی تهیه می‌کند. تصویر پردازش شده و مسیر نهایی بر اساس مختصات نقشه به دست می‌آید و در نهایت خودرو از محل ورودی پارکینگ به محل بلوک قرمز هدایت می‌شود.



شکل ۱-۱۱ مدل سه بعدی ستاپ پارک خودکار در محیط واقعی

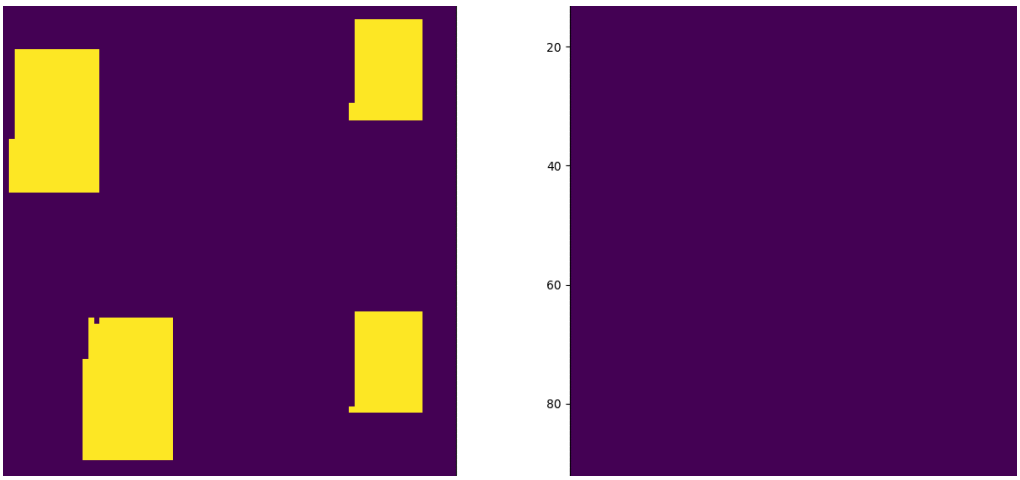
۱-۱-۱۱- پردازش تصویر نقشه

الگوریتم های مسیریابی شبیه ساز، برمبنای محیط ۱۰۰ در ۱۰۰ پیکسل نوشته شده اند. لذا لازم است سایز تصویر دریافتی از دوربین به رزولوشن ۱۰۰ در ۱۰۰ تغییر کند. نتیجه این تغییرات در شکل ۲-۱۱ نشان داده شده است.



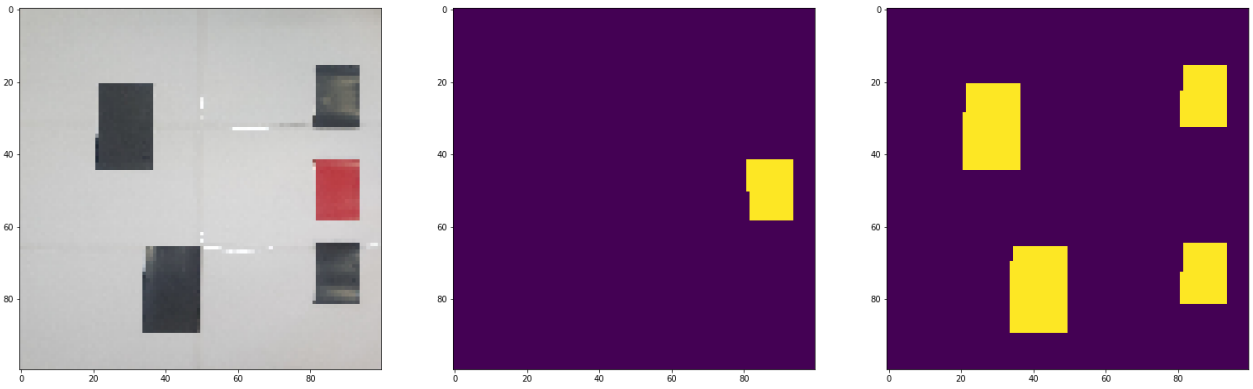
شکل ۲-۱۱ تغییر رزولوشن تصویر دریافتی از دوربین به ۱۰۰ پیکسل در ۱۰۰ پیکسل

در گام دوم، موانع و محل نهایی پارک با استفاده از تصویر نقشه استخراج می‌شود. این مرحله برمبنای رنگ اشیاء بوده و ماسک رنگی موانع و محل نهایی پارک متناسب با محدوده رنگی RGB آن به دست می‌آید.



شکل ۳-۱۱ ماسک رنگی محل نهایی پارک در سمت راست و ماسک رنگی موانع در سمت چپ

با مشاهده ماسک های رنگی می‌توان دریافت که انعکاس نور محیط ممکن است باعث تغییر محدوده رنگ تصویر و ایجاد قسمت های خالی در ماسک شود. برای جلوگیری از ایجاد خطا در مسیریابی به رفع قسمت‌های خالی می‌پردازیم. با استخراج تمامی کانتورهای موجود در ماسک و محدب کردن ساختار آن ها، می توان تظمین کرد که قسمت های خالی هر مانع در ماسک از بین خواهد رفت. خروجی نهایی پردازش تصویر نقشه در شکل ۴-۱۱ نشان داده شده است.



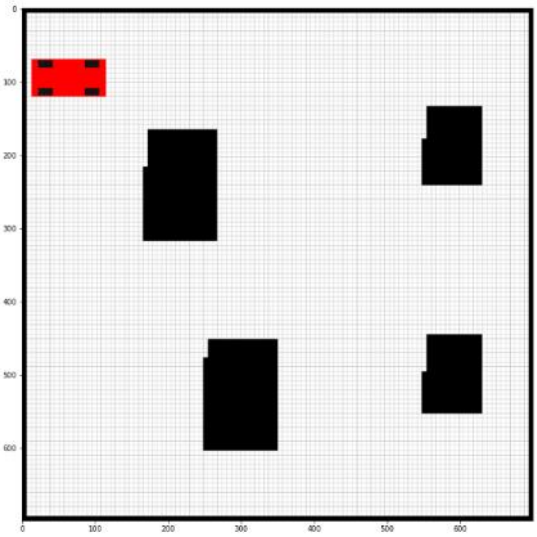
شکل ۴-۱۱ خروجی پردازش تصویر نقشه پارکینگ، تصویر چپ تصویر نقشه ثبت شده توسط دوربین، تصویر وسط ماسک باینری محل پارک و تصویر راست ماسک باینری

دستورات پایتون مربوط به پردازش تصویر نقشه پارکینگ به شرح زیر است.

```
1. map_img = cv2.resize(map_img_org, (100,100))
2.
3. black_mask = cv2.inRange(map_img, np.array([0,0,0]), np.array([140,140,
   140]))
4. red_mask = cv2.inRange(map_img, np.array([0,0,120]), np.array([100,100,
   255]))
5.
6. contours, hierarchy = cv2.findContours(red_mask, cv2.RETR_TREE, cv2.CHA
   IN_APPROX_SIMPLE)
7. parking_mask = np.zeros_like(black_mask)
8. for contour in contours:
9.     hull = cv2.convexHull(contour, False)
10.          cv2.drawContours(parking_mask, [hull], -1, 255, -1)
11.
12.      contours, hierarchy = cv2.findContours(black_mask, cv2.RETR_TREE,
    cv2.CHAIN_APPROX_SIMPLE)
13.      obstacle_mask = np.zeros_like(red_mask)
14.      for contour in contours:
15.          hull = cv2.convexHull(contour, False)
16.          cv2.drawContours(obstacle_mask, [hull], -1, 255, -1)
```

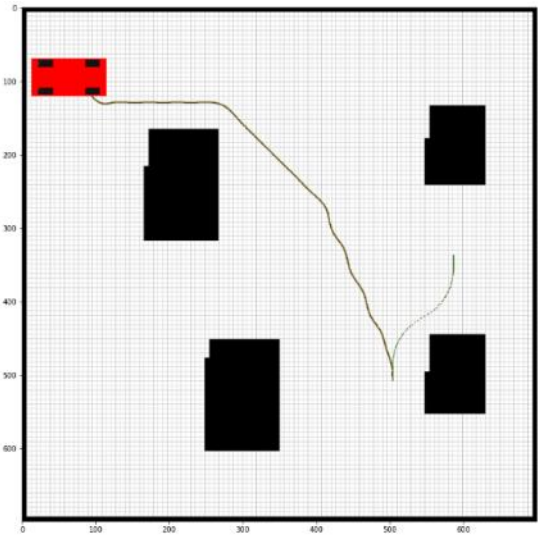
۲-۱-۱۱- مسیریابی

برای برنامه‌ریزی مسیر، لازم است اطلاعات دریافتی از نقشه پارکینگ، به محیط شبیه سازی انتقال یابد. با توجه به وجود تفاوت در اسکیل خودرو موجود در محیط شبیه سازی و واقعی، پارامترهای محیط شبیه سازی دوباره تنظیم می‌شوند. در نهایت با گرفتن خروجی از شبیه ساز، شکل ۵-۱۱ به دست می‌آید.



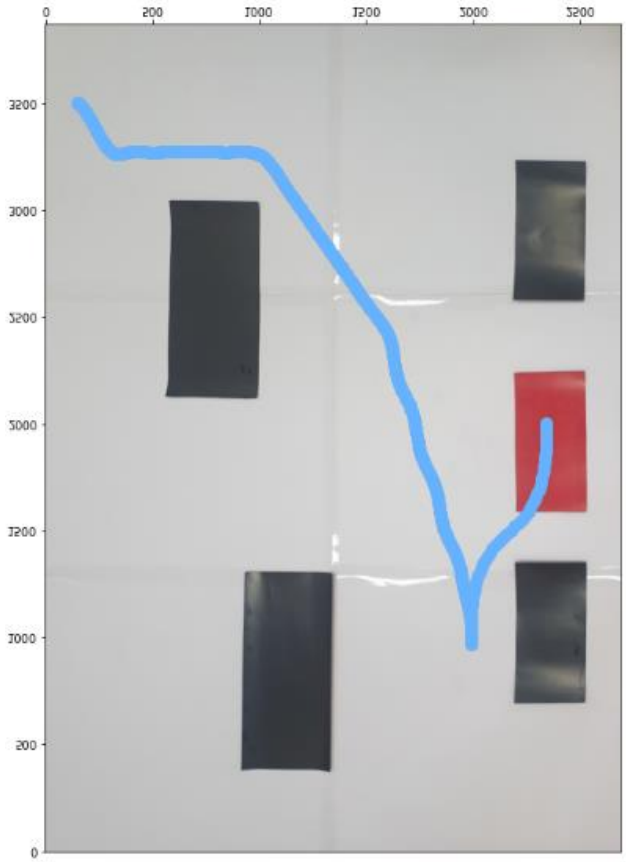
شکل ۵-۱۱ خروجی نقشه پارکینگ واقعی در شبیه ساز پایتون

برنامه‌ریزی مسیر توسط زیرسیستم آن که در شبیه ساز پیاده سازی شده است، صورت می‌گیرد. در نهایت با درنظر گرفتن عرض خودرو و فاصله لازم بین خودرو و موانع، کوتاه ترین مسیر تا نقطه پارک توسط الگوریتم A* به دست می‌آید. درون یابی مسیر نیز بر اساس اسپلاین صورت می‌گیرد تا شکستگی های مسیر کاهش یابد.



شکل ۶-۱۱ برنامه زیری مسیر و درون یابی آن در شبیه ساز

مسیر به دست آمده شامل مجموعه ای از نقاط در فضای ۱۰۰ در ۱۰۰ شبیه ساز می‌باشد. لذا برای تطابق این مسیر با نقشه واقعی لازم است مقیاس نقاط متناسب با رزولوشن تصویر اصلی تغییر کند. شکل ۷-۱۱ مسیر به دست آمده در مقیاس رزولوشن اصلی تصویر نشان می‌دهد.



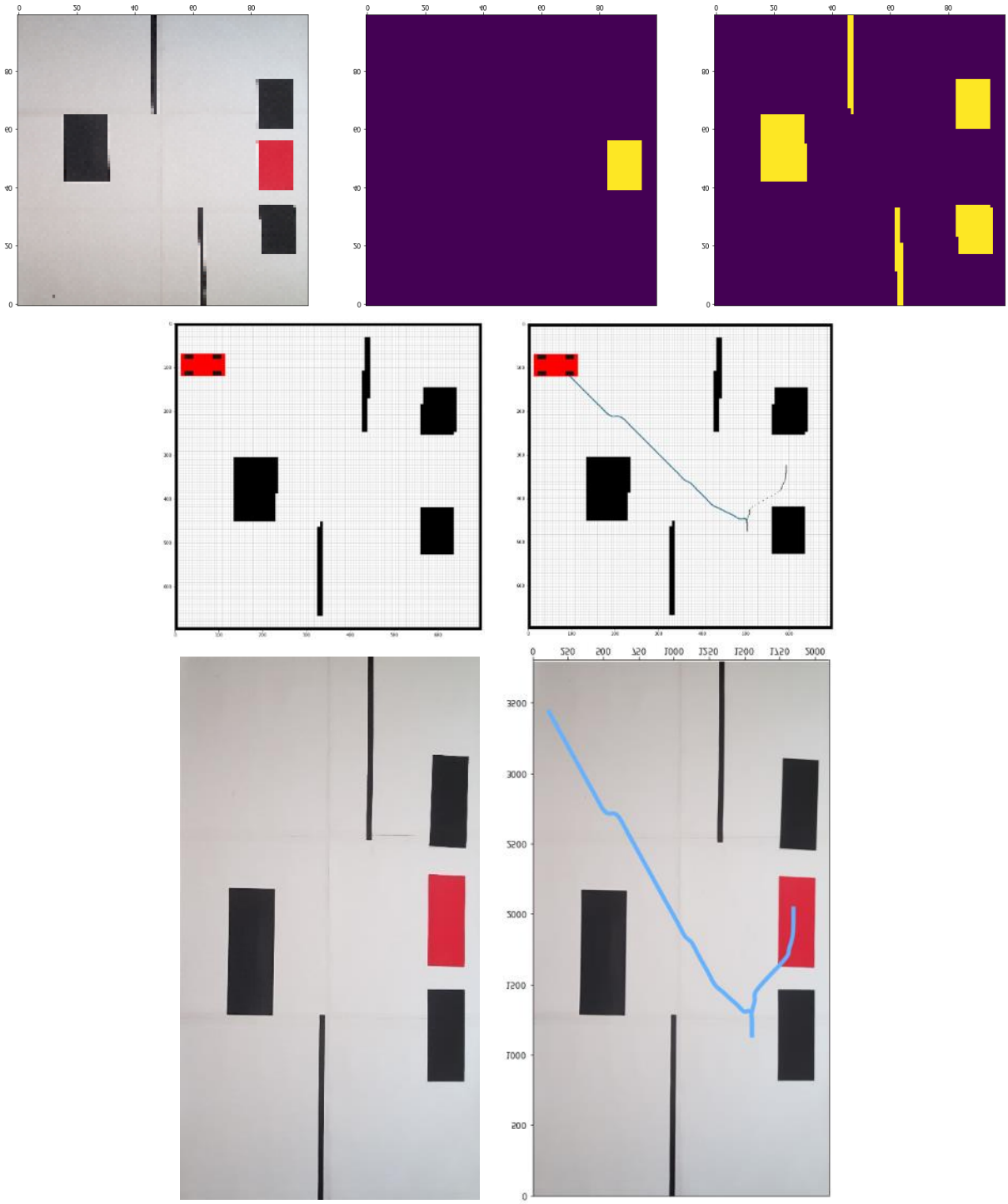
شکل ۷-۱۱ خروجی زیرسیستم برنامه‌ریزی مسیر در مقیاس اصلی تصویر نقشه

سرانجام با داشتن اندازه واقعی و رزولوشن تصویر، نقاط مسیر به مقیاس محیط با واحد سانتی متر تبدیل شده و مسیر نهایی به خودرو ارسال می‌شود. مسیر ارسال شده رفرنس خودرو در هر لحظه می‌باشد و واحد کنترل خودرو را تا رسیدن به مقصد هدایت می‌کند.

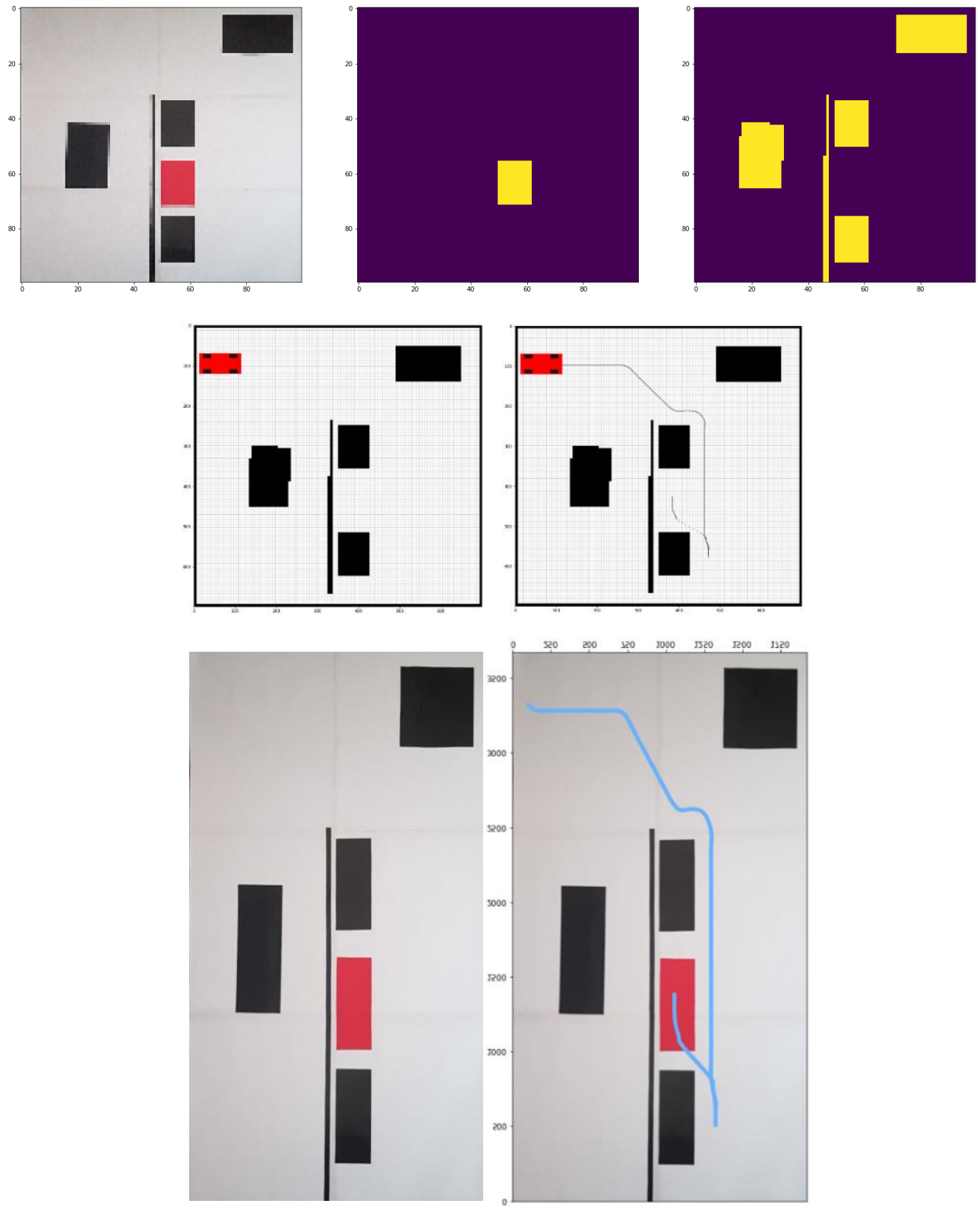
پیاده‌سازی پایتون برنامه‌ریزی مسیر در شرایط واقعی در ادامه آورده شده است.

```
from environment import Environment
from pathplanning import PathPlanning, ParkPathPlanning

obstacle_mask_flip = np.flip(obstacle_mask, axis=0)
obs = np.vstack([np.where(obstacle_mask_flip>100)[1],np.where(obstacle_mask_flip>100)[0]]).T

start = np.array([5, 90])
parking_mask_flip = np.flip(parking_mask, axis=0)
end = [np.where(parking_mask_flip>100)[1].mean().astype(int), np.where(parking_mask_flip>100)[0].mean().astype(int)]

env = Environment(obs)
park_path_planner = ParkPathPlanning(obs)
path_planner = PathPlanning(obs)

print('planning park scenario ...')
new_end, park_path, ensure_path1, ensure_path2 = park_path_planner.generate_park_scenario(int(start[0]),int(start[1]),int(end[0]),int(end[1]))

print('routing to destination ...')
path = path_planner.plan_path(int(start[0]),int(start[1]),int(end[0]),int(end[1]))

path = path_planner.plan_path(int(start[0]),int(start[1]),int(new_end[0]),int(new_end[1]))
path = np.vstack([path, ensure_path1])

print('interpolating ...')
interpolated_path = path_planner.interpolate_path(path)
interpolated_park_path = park_path_planner.interpolate_park_path(park_path)
interpolated_park_path = np.vstack([ensure_path1[::-1], interpolated_park_path, ensure_path2[::-1]])

env.draw_path(interpolated_path)
env.draw_path(interpolated_park_path)

my_car = Car_Dynamics(start[0], start[1], 0, np.deg2rad(0), length=0.4, dt=0.2)
res = env.render(my_car.x, my_car.y, my_car.psi, 0)
```

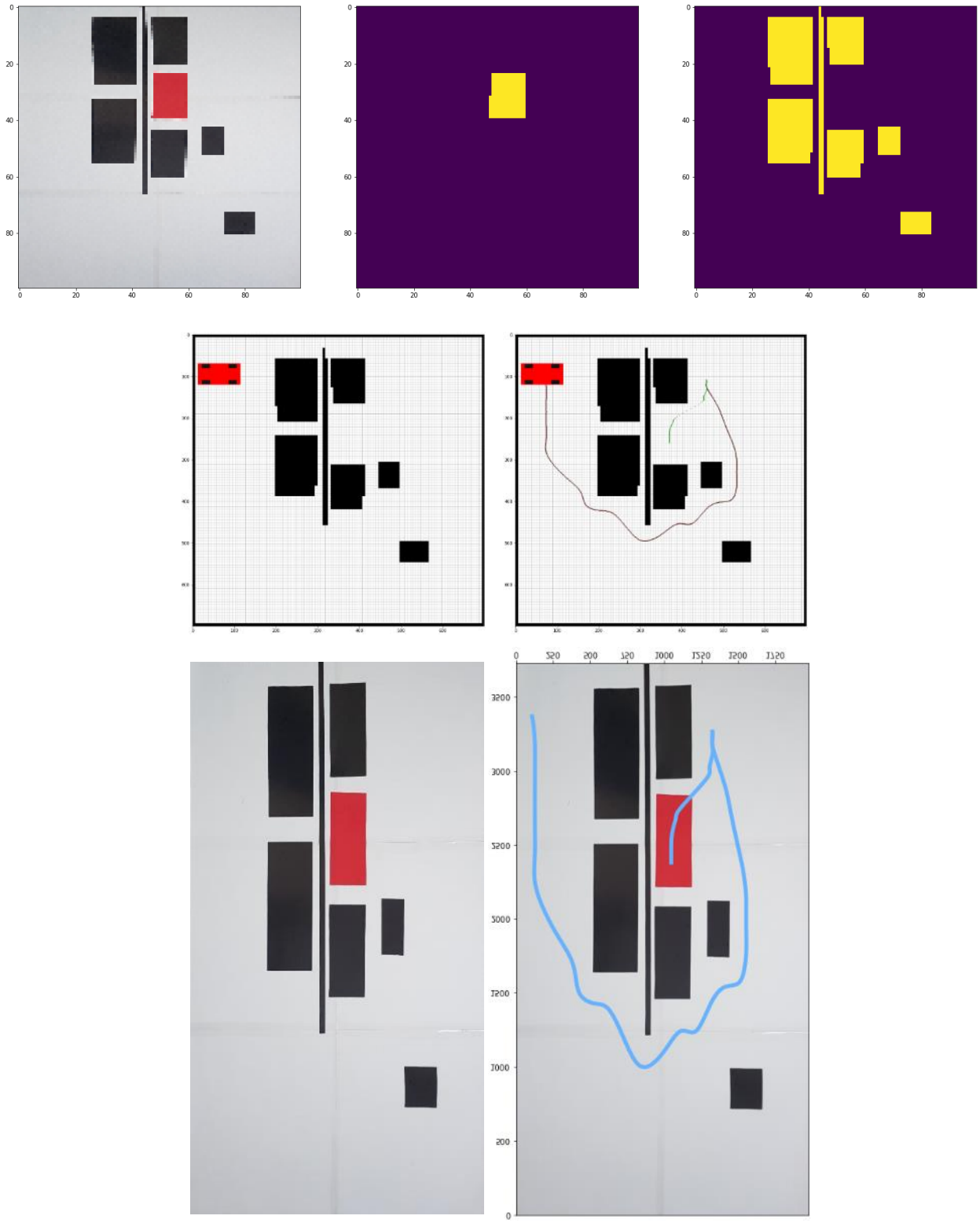
۳-۱-۱۱- بررسی سناریوهای مختلف پارک



شکل ۸-۱۱ نمونه اول پردازش تصویر و مسیریابی در نقشه واقعی



شکل ۹-۱۱ نمونه دوم پردازش تصویر و مسیریابی در نقشه واقعی



شکل ۱۰-۱۱ نمونه سوم پردازش تصویر و مسیریابی در نقشه واقعی

۲-۱۱- الگوریتم‌های پردازش تصویر در نقشه های شهری و بین شهری

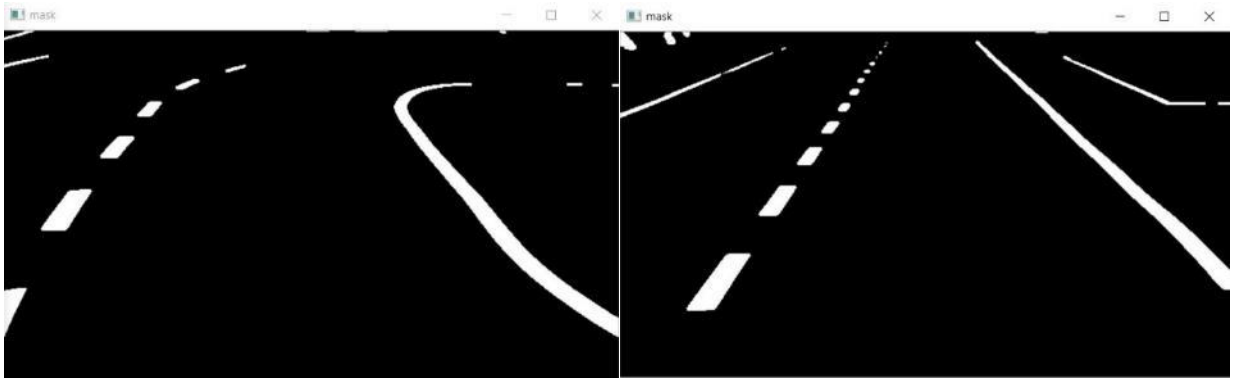
در این بخش نتایج الگوریتم‌های مختلف که صورت شبیه‌سازی در فصول قبل نمایش داده‌شد بر روی نقشه‌های واقعی طراحی شده تست می‌شوند.

۱-۲-۱۱- تشخیص خطوط جاده

در این بخش، لازم است تا خطوط جاده تشخیص داده‌شود، این خطوط در نقشه‌ی طراحی شده به‌رنگ سیاه می‌باشند. یک نمونه از جاده به در شکل ۱۱-۱۱ نمایش داده‌شده است.

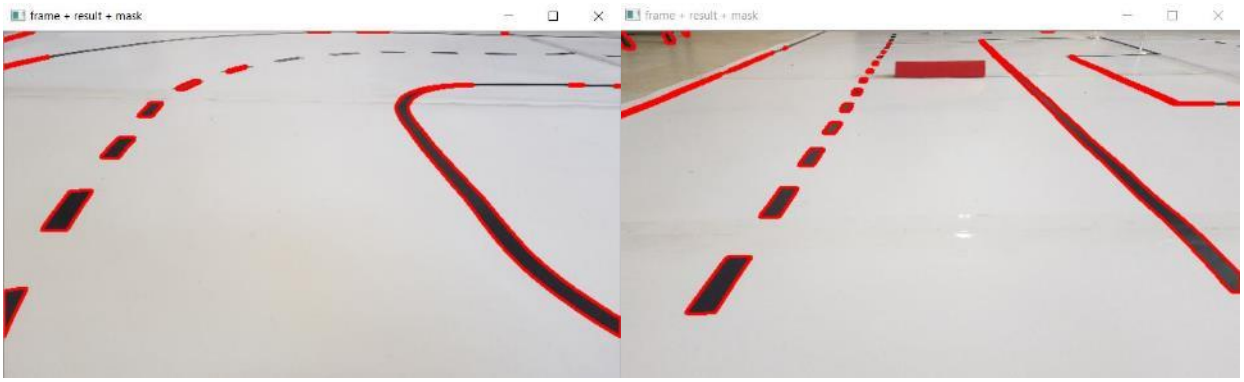
شکل ۱۱-۱۱ نمایی از مسیر طراحی‌شده

برای مسیرهای موجود در شکل ۱۱-۱۱، شکل ماسک جاده به صورت شکل ۱۲-۱۱ می‌باشد که در آن قسمت‌های مختلف جاده مشخص شده است.



شکل ۱۲-۱۱ ماسک جاده برای تشخیص خطوط موجود در شکل ۱۱-۱۱

همچنین، کانتورهای یافت‌شده برای خطوط به‌صورت شکل ۱۳-۱۱ می‌باشد که بر روی تصویر اصلی (شکل ۱۱-۱۱) نمایش داده شده‌است.



شکل ۱۳-۱۱ انتقال کانتورهای یافت‌شده از خطوط مسیر به تصویر اصلی

شایان ذکر است که برای تشخیص ماسک مسیر از مقادیر موجود در جدول ۱-۱۱ در فضای HSV استفاده شده است.

جدول ۱-۱۱ مقادیر پارامترهای HSV برای تشخیص خطوط

پارامتر	کمینه	بیشینه
H	۲۷	**۱۶۰**
S	۱۰	**۶۴**
V	۰	**۲۵۵**

۲-۲-۱۱- تشخیص تابلوهای راهنمایی و رانندگی

از مدل مبتنی بر یادگیری عمیق یولو (Yolo) که در فصل‌های گذشته به تفصیل معرفی شد برای تشخیص تابلوهای راهنمایی و رانندگی موجود در مسیر استفاده می‌شود. دو نمونه از خروجی این الگوریتم از شناسایی تابلوهای موجود در ادامه آورده شده است.

شکل ۱۴-۱۱ شناسایی تابلوی ورود ممنوع توسط الگوریتم یولو

شکل ۱۵-۱۱ شناسایی انواع تابلوهای راهنمایی توسط الگوریتم یولو

۳-۲-۱۱- تشخیص موانع قرمز رنگ

در قسمت آخر تشخیص موانع با استفاده از الگوریتم‌های پردازش تصویر انجام می‌شود. تشخیص موانع به دو روش سنسورهای آلتراسونیک و پردازش تصویر انجام می‌شود که در این جا نتایج تشخیص مانع به این روش نشان داده می‌شود.

برای مثال در تصویر ۱۶-۱۱ خودرو به یک مانع در محیط شهری نزدیک شده است.

شکل ۱۶-۱۱ وجود یک مانع در ادامه‌ی مسیر خودرو

با استفاده از یک فیلتر، مقادیر بین بیشینه و کمینه‌ی پارامترهای HSV تصویر در در جدول ۲-۱۱ مشاهده می‌شود.

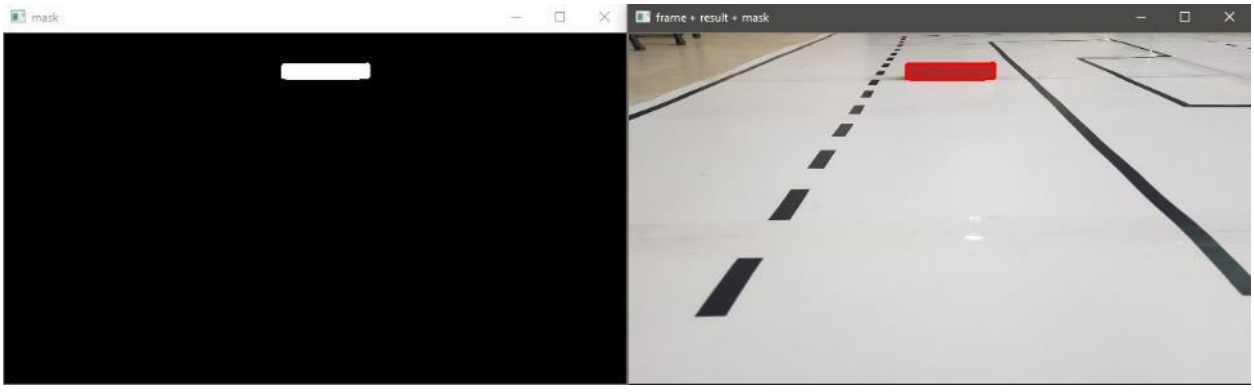
جدول ۲-۱۱ مقادیر پارامترهای HSV برای تشخیص مانع

پارامتر	کمینه	بیشینه
H	۰	۱۷۹
S	۱۲۰	۱۷۳
V	۰	۲۵۵

با استفاده از این فیلتر، ماسک مانع موجود در تصویر ۱۶-۱۱ که به رنگ قرمز می‌باشد تشخیص داده می‌شود و در شکل ۱۷-۱۱ نمایش داده می‌شود.

شکل ۱۷-۱۱ تشخیص ماسک مانع

همچنین کانتور ماسک مورد نظر برروی شکل اصلی انداخته می‌شود و نتیجه‌ی شکل ۱۸-۱۱ حاصل می‌شود.

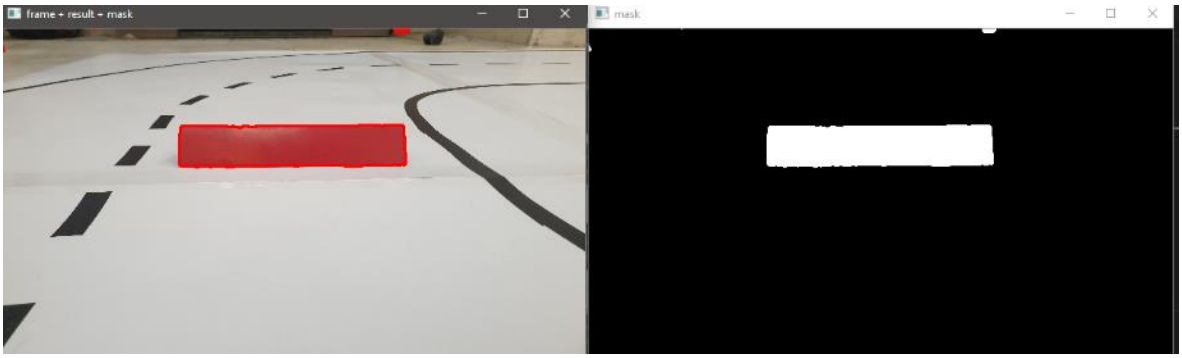


شکل ۱۸-۱۱ ماسک باینری و تصویر ماسک‌شده‌ی شامل مانع

همین عملیات برای موانع موجود در مسیر بین‌شهری نیز تکرار می‌شود. برای مثال شکل ۱۹-۱۱ دارای یک مانع در محیط بین‌شهری می‌باشد.

شکل ۱۹-۱۱ مانع در مسیر بین شهری

با انجام همان فیلترینگ انجام شده در مسیر شهری، ماسک و تصویر ماسک‌شده‌ی شکل ۱۹-۱۱ به صورت شکل ۲۰-۱۱ خواهد بود.



شکل ۲۰-۱۱ ماسک و شکل ماسک‌شده ی مانع در محیط بین‌شهری

۳-۱۱- کنترل خودرو

در این پژوهش از کنترلر PID برای کنترل خودرو در میان دو خط حاشیه ای جاده استفاده شده است. نحوه ی کار این کنترلر در خودروی خودران معرفی شده به این صورت است که هنگام حرکت خودرو در جاده، با استفاده از دوربین خودرو و الگوریتم های مبتنی بر پردازش تصویر که در بخش های قبلی به صورت کامل معرفی گردیده اند، نقطه ی وسط دو خط حاشیه ای جاده محاسبه می گردد و با پیکسل وسط دوربین مقایسه می گردد و ضریبی از این تفاوت که به عنوان خطای کنترلی شناخته می شود، به عنوان فرمان کنترلی به چرخ ها داده می شود تا خودرو با پیاده سازی زاویه ی فرمان مناسب، همواره در بین خطوط جاده باقی بماند و از آن منحرف نگردد.

در ادامه تاثیر هر کدام از ترم ها کنترل کننده روی حرکت خودرو در بین خطوط بررسی می گردد:

۱-۳-۱۱- ترم تناسبی (Proportional)

این ترم بر شدت چرخش چرخ های خودرو تاثیر می گذارد و هر چقدر که خطای کنترلی بیشتر باشد، به کمک این ترم، فرمان کنترلی شدید تری بر روی چرخ ها اعمال می گردد تا روند برگشتن به بین خطوط جاده با سرعت بیشتری انجام گردد.

از این ترم نمی توان به صورت تنها در کنترل یک خودروی خودران استفاده نمود چرا که در آزمایش ها مشخص گردید که نقطه ضعف اساسی استفاده از یک کنترل با ترم تناسبی تنها این است که در هنگام رسیدن به پیچ ها و پیچیدن خودرو در سرعت های بالا، چون خطای کنترلی به صورت ناگهانی افزایش می یابد، فرمان کنترلی باعث چرخیدن حداکثری چرخ ها می گردد که باعث می شود خودرو این بار از جهت مخالف از جاده خارج گردد، و دوباره این فرآیند تکرار می شود که باعث نوسانی شدن حرکت خودرو می گردد و خودرو ی خودران با کنترلر P، به محض رسیدن به اولین پیچ یا چهار راه، شروع به حرکت نوسانی و سینوسی پیرامون یک محور ثابت می کند و این نوسان با گذشت زمان افزایش می یابد به نحوی که خطای کنترلی غیر قابل محاسبه و خودرو به کلی از کنترل خارج می گردد.

۲-۳-۱۱- ترم مشتقی (Derivative)

این ترم در جبران اثرات منفی ترم قبلی استفاده می شود و روی کاهش نوسانات خودرو در جاده پس از تغییرات خطای کنترلی اثر به سزایی دارد.

در واقع عملکرد این ترم به این صورت است که یک ضریب دمپینگ یا میرا شونده به فرمول کنترلر اضافه می کند که به نوعی این هوشمندی را به سیستم کنترلی اضافه می کند که همزمان با کاهش خطا، باید شدت فرمان کنترلی اعمالی به چرخ های خودرو نیز کاهش یابد، پس زاویه ی فرمان دریافتی از ترم تناسبی را به صورت مناسب تعدیل و اصلاح می کند تا با نزدیک شدن خطای کنترلی به صفر، فرمان کنترلی خودرو را به جهت مخالف هدایت نکند و از نوسانات خودرو جلوگیری شود.

۳-۳-۱۱- ترم انتگرالی (Integral)

در نهایت از این ترم برای تصحیح خطای مکانیکی خودروی خودران استفاده می گردد.

ترم تناسبی تنها بر اساس خطای لحظه ای، چرخش فرمان را تنظیم می کند و این موضوع باعث می گردد تا احتمال ایجاد یک خطای ثابت در پاسخ کنترلی سیستم به شدت افزایش یابد. این موضوع در خودروی خودران به این صورت مشاهده می گردد که خودرو با یک آفست و فاصله ی ثابت از نقطه ی میانی جاده به حرکت ادامه می دهد. ترم انتگرالی پس از محاسبه ی مجموع خطاهای کنترلی ایجاد شده در طول مسیر که به علت وجود پیچ و خم های جاده به وجود می آیند، ضریبی از این خطای تجمعی را به فرمان خودرو اعمال می کند که باعث می شود تا خطای مکانیکی و ثابت پاسخ نهایی بخش کنترل خودرو به صفر برسد و خودرو دقیقا در بین دو خط حاشیه ی جاده به حرکت خود ادامه دهد.

۴-۳-۱۱- کنترلر نهایی طراحی شده

در نهایت و پس از انجام تعداد زیادی آزمون صحیح و خطا، با استفاده از روش زیگلر-نیکولز [۳۲۷] ، ضرایب مناسب برای طراحی این خودرو طراحی و به کنترلر اعمال گردید.

جدول ۳-۱۱ روش زیگلر نیکولز برای طراحی کنترل کننده

نوع کنترلر	K_p	K_i	K_d
PID	$0.6 \times K_u$	$2 \times \frac{K_p}{P_u}$	$K_p \times \frac{P_u}{8}$

با توجه به جدول بالا، با داشتن دو ترم K_u و P_u می توان هر سه ضریب کنترل کننده ی PID را به دست آورد. در ادامه نتایج آزمایشات مقادیر مختلف این دو پارامتر را بر اساس روش صحیح و خطا در سرعت ثابت ۶۰ کیلومتر بر ساعت برای رسیدن به ضرایب مطلوب کنترل خودرو قابل مشاهده می باشد:

جدول ۴-۱۱ آزمایشات صحیح و خطا برای یافتن مقادیر کنترلی مطلوب

K_u	P_u	نتیجه ی مشاهده شده
۲۰	۱۲	چرخ ها و فرمان خودرو به شدت سریع می چرخیدند
۱۰	۱۲	کماکان شدن چرخش چرخ ها و فرمان بسیار سریع بود
۵	۱۲	کماکان شدن چرخش چرخ ها و فرمان بسیار سریع بود
۵	۱۳	کماکان شدن چرخش چرخ ها و فرمان بسیار سریع بود
۳.۸	۱۳	در نوسان خودرو افزایش مشاهده شد
۳.۸	۱۲	نوسان مقداری کاهش پیدا کرد اما هنوز نوسان مشاهده می شد
۳.۴	۱۳	وضعیت نوسان خودرو بهتر شد
۳.۴	۱۴	نوسان افزایش پیدا کرد

وضعیت نوسان خودرو بهتر شد	۱۴	۳.۸
وضعیت نوسان خودرو بهتر شد	۱۴.۵	۳.۸
وضعیت نوسان خودرو بهتر شد	۱۵	۳.۸
وضعیت نوسان خودرو بهتر شد و چرخش خودرو ها نیز کند تر(بهتر) شد	۱۶	۳.۸
وضعیت نوسان خودرو بهتر شد و چرخش خودرو ها نیز کند تر (بهتر) شد	۱۶	۳.۷
وضعیت نوسان خودرو بهتر شد و چرخش خودرو ها نیز کند تر (بهتر) شد	۱۷	۳.۶
خودرو کاملا در بین خطوط حرکت می کرد اما هنوز اندکی نوسان داشت	۱۷	۳.۵
خودرو کاملا در بین خطوط حرکت می کرد اما هنوز اندکی نوسان داشت	۱۷	۳.۴
خودرو کاملا در بین خطوط حرکت می کرد اما هنوز اندکی نوسان داشت	۱۷	۳.۲
خطای ثابت در حرکت خودرو مشاهده می شد (از وسط جاده فاصله داشت)	۱۸	۳.۲
خودرو دچار خطای حالت دائم و نوسان شد و نتیجه جالب نبود	۱۸	۳.۱
نوسان کاهش پیدا کرد	۱۹	۳.۲
نوسان کاهش پیدا کرد	۲۰	۳.۲
نوسان کاهش پیدا کرد و در سرعت چرخش فرمان نیز به تعادل رسید	۲۰	۳.۳
نتیجه ی کاملا مطلوبی داشت	۲۰	۳.۴

بدین ترتیب کنترلر PID نهایی طراحی شده برای در وسط جاده نگه داشتن خودرو به صورت زیر می باشد:

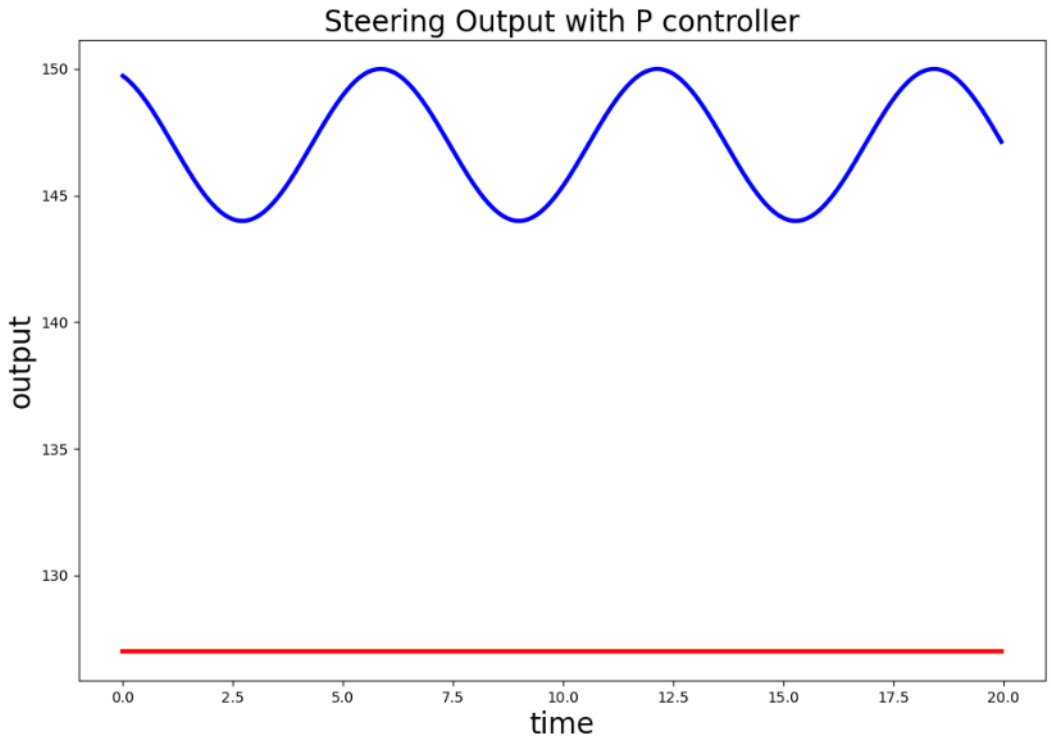
جدول ۵-۱۱ ضرایب نهایی طراحی شده برای خودروی خودران

نوع کنترلر	K_p	K_i	K_d
PID	۲.۰۴	۰.۲۰۴	۵.۱

۵-۳-۱۱- مقایسه ی انواع کنترلر های قابل پیاده سازی

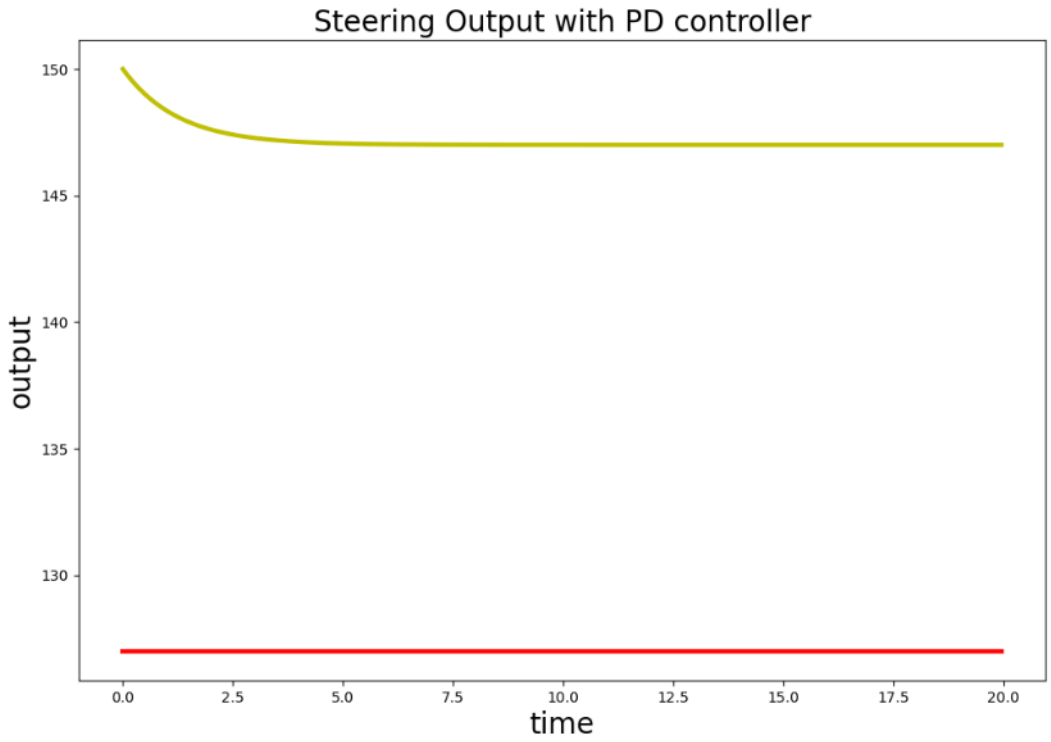
در انتهای آزمایشات این بخش و پس از به دست آمدن ضرایب کنترلر بر اساس روش زیگلر نیکولز، اثر هر کدام از سه کنترلر P، PI و PID بر اساس ضرایب به دست آمده از بخش قبلی نیز به صورت مجزا روی خودرو آزمایش و بررسی گردید.

در این بررسی، پیکسل ۱۲۸ یا همان پیکسل میانی در تصویر دریافتی از دوربین به عنوان رفرنس در نظر گرفته شد و در حین کج بودن زاویه ی اولیه ی فرمان خودرو، سه کنترلر مختلف روی خودرو آزمایش گردید که خودرو باید بتواند نقطه ی بین دو خط حاشیه ی جاده را روی پیکسل ۱۲۸ تنظیم نماید که نتایج آن در تصویر زیر قابل مشاهده می باشد:



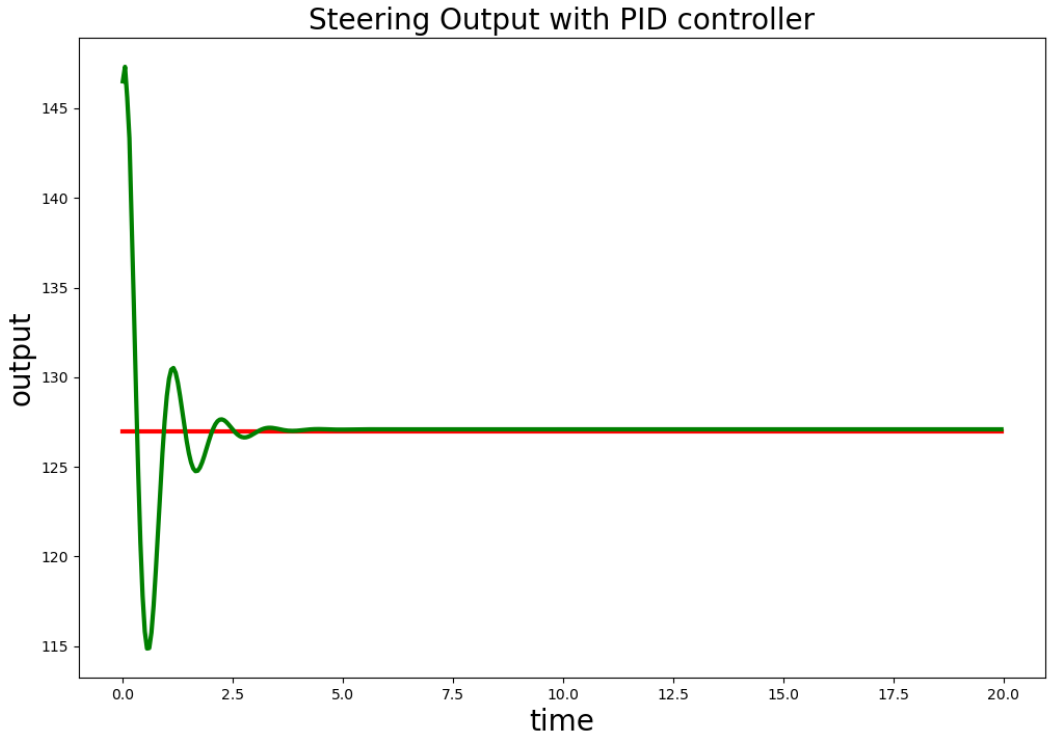
شکل ۲۱-۱۱ مقایسه ی عملکرد کنترلر P در مقایسه با رفرنس کنترلی

همانطور که در تصویر بالا قابل مشاهده است، خروجی کنترلر P دچار نوسان ثابت می گردد و نمیتواند که خودرو را به صورت مناسب کنترل کند.



شکل ۲۲-۱۱ مقایسه ی عملکرد کنترلر PD در مقایسه با رفرنس کنترلی

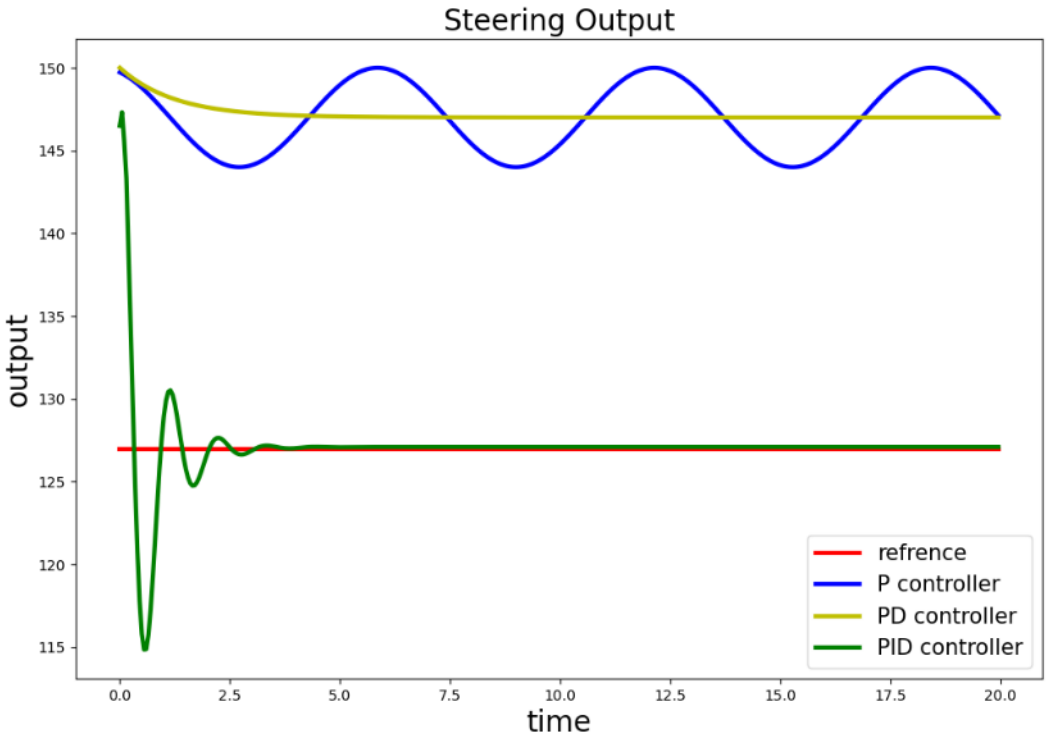
کنترلر PD نیز علی رغم مهار کردن نوسانان خودرو، دارای خطای ثابت تقریبا بیست پیکسلی با نقطه بین دو خط حاشیه ی جاده دارد که معادل حدود ده سانتی متر فاصله ی ثابت از میان جاده است.



شکل ۲۳-۱۱ مقایسه ی عملکرد کنترلر PID در مقایسه با رفرنس کنترلی

و در نهایت مشاهده می گردد که تنها کنترلر PID است که می تواند خودرو را به صورت کامل در میان جاده و بدون خروج یا انحراف از جاده، کنترل نماید.

در تصویر زیر، خروجی تمامی کنترلر ها در کنار رفرنس کنترلی به جهت مقایسه آورده شده است:



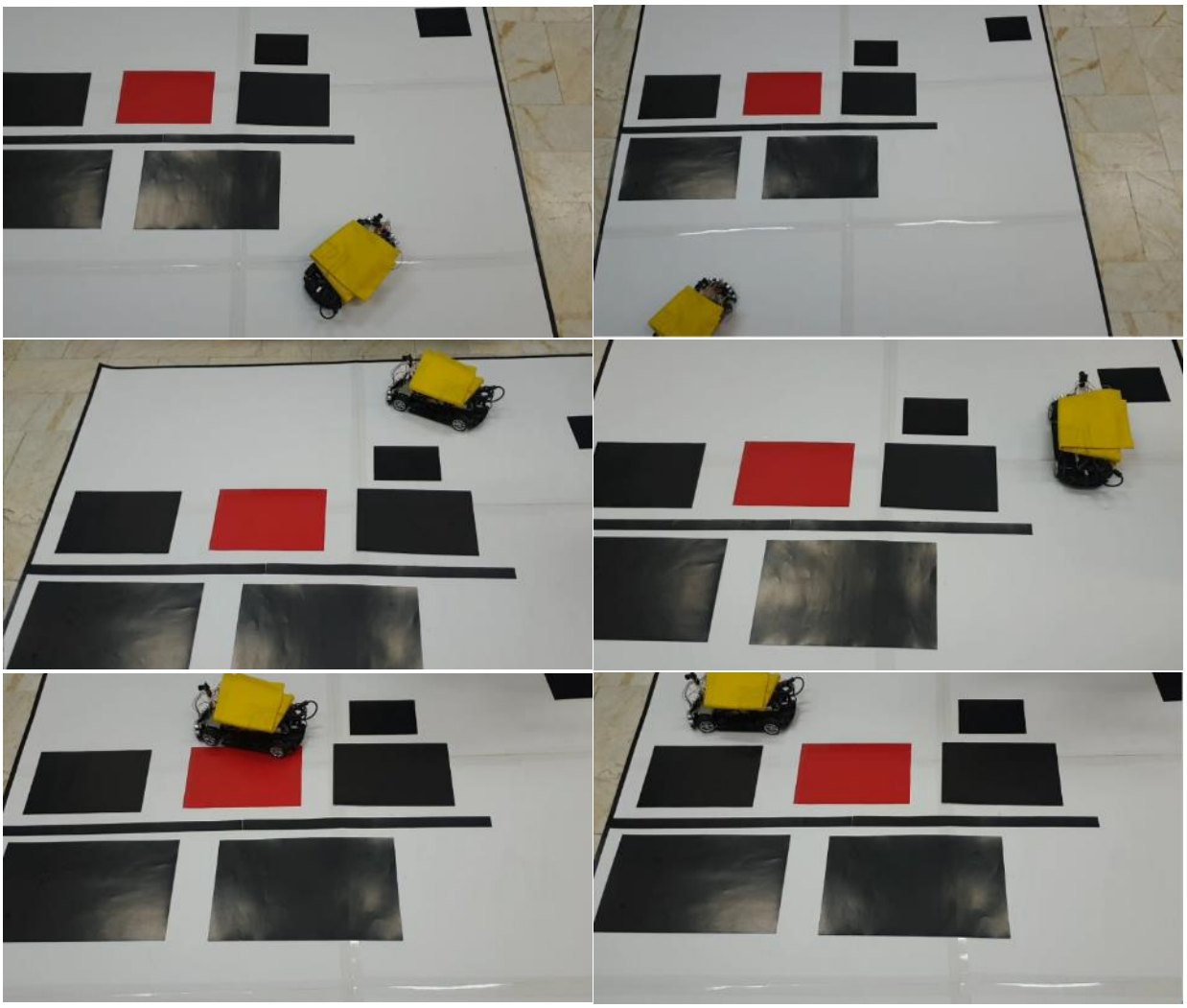
شکل ۲۴-۱۱ مقایسه ی عملکرد همه ی کنترلر های طراحی شده در مقایسه با رفرنس کنترلی

۱۱-۴- عملکرد خودرو در محیط واقعی

در این بخش عملکرد کلی خودرو و الگوریتم‌ها و نرم افزار های توسعه داده شده مورد بررسی قرار می‌گیرد. خودرو در داخل هریک از نقشه های طراحی شده قرار داده شده و خود را تا رسیدن به مقصد هدایت می‌کند.

۱۱-۴-۱- نقشه پارکینگ

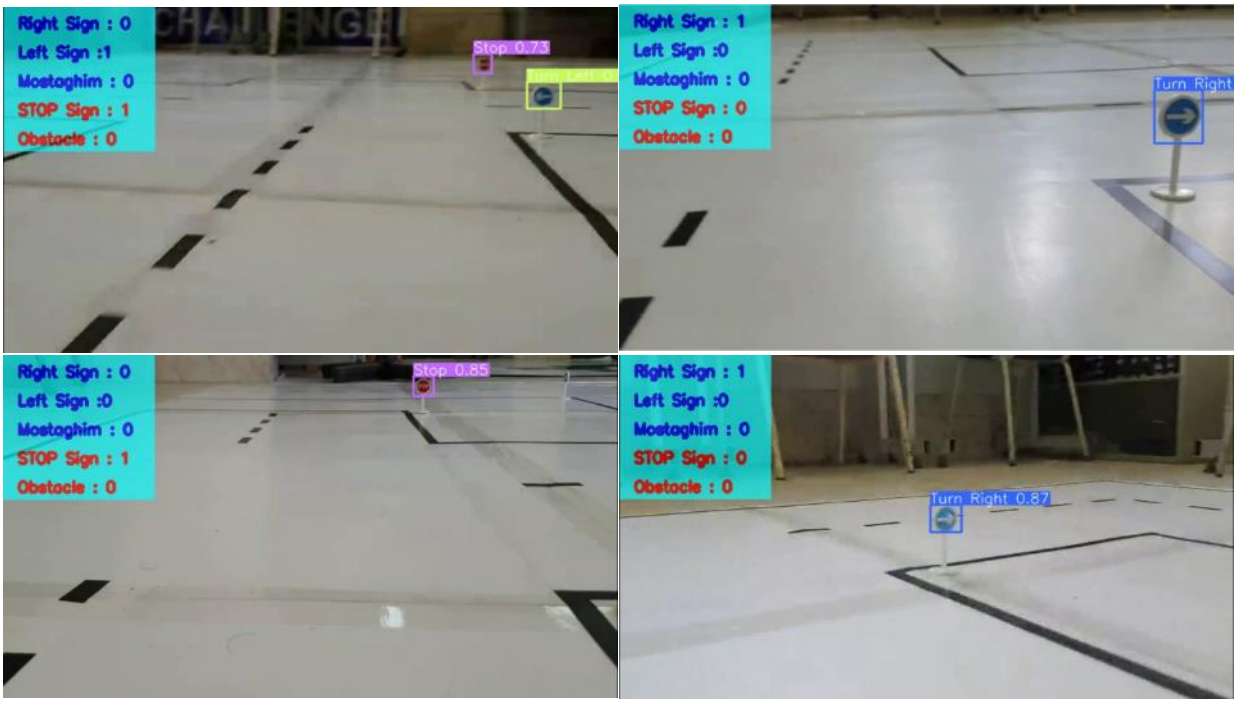
نتایج مسیریابی نقشه پارکینگ، در بخش ۱-۱۱ ارائه داده شده است. در این بخش عملکرد خودرو در دنبال کردن مسیر و انجام پارک موازی مورد بررسی قرار می‌گیرد. همچنین دلیل وجود پارچه زرد، استفاده از فیدبک موقعیت خودرو می‌باشد. با توجه به شکل ۲۵-۱۱ مشاهده می‌شود که خودرو مسیر را به خوبی دنبال کرده و تقریبا در محل مشخص شده قرار می‌گیرد.



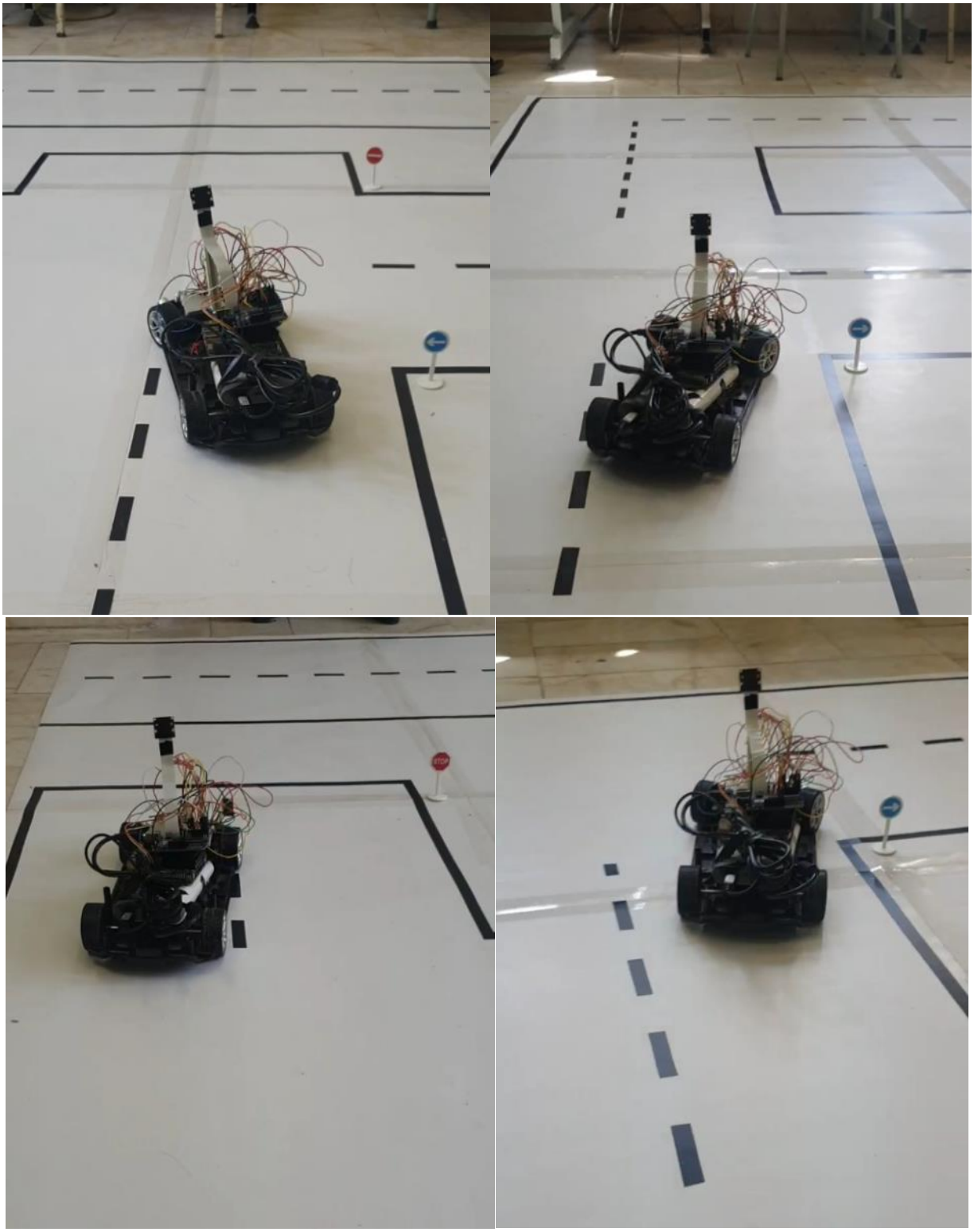
شکل ۲۵-۱۱ مراحل حرکت خودرو در نقشه پارک و انجام پارک موازی

۲-۴-۱۱- نقشه شهری

نتایج عملکرد خودرو در نقشه شهری در ادامه آورده شده است. شکل ۲۶-۱۱ نتایج پردازش محیط از دید خودرو (on board) و شکل ۲۷-۱۱ نتایج عملکرد خودرو در نقشه (off board) را نشان می‌دهند. الگوریتم‌های به کار گرفته شده در این بخش، تشخیص خطوط عمودی و افقی، شناسایی علائم راهنمایی، تشخیص و دور زدن موانع، تصمیم گیری در مواجهه با تقاطع های جاده می باشد.



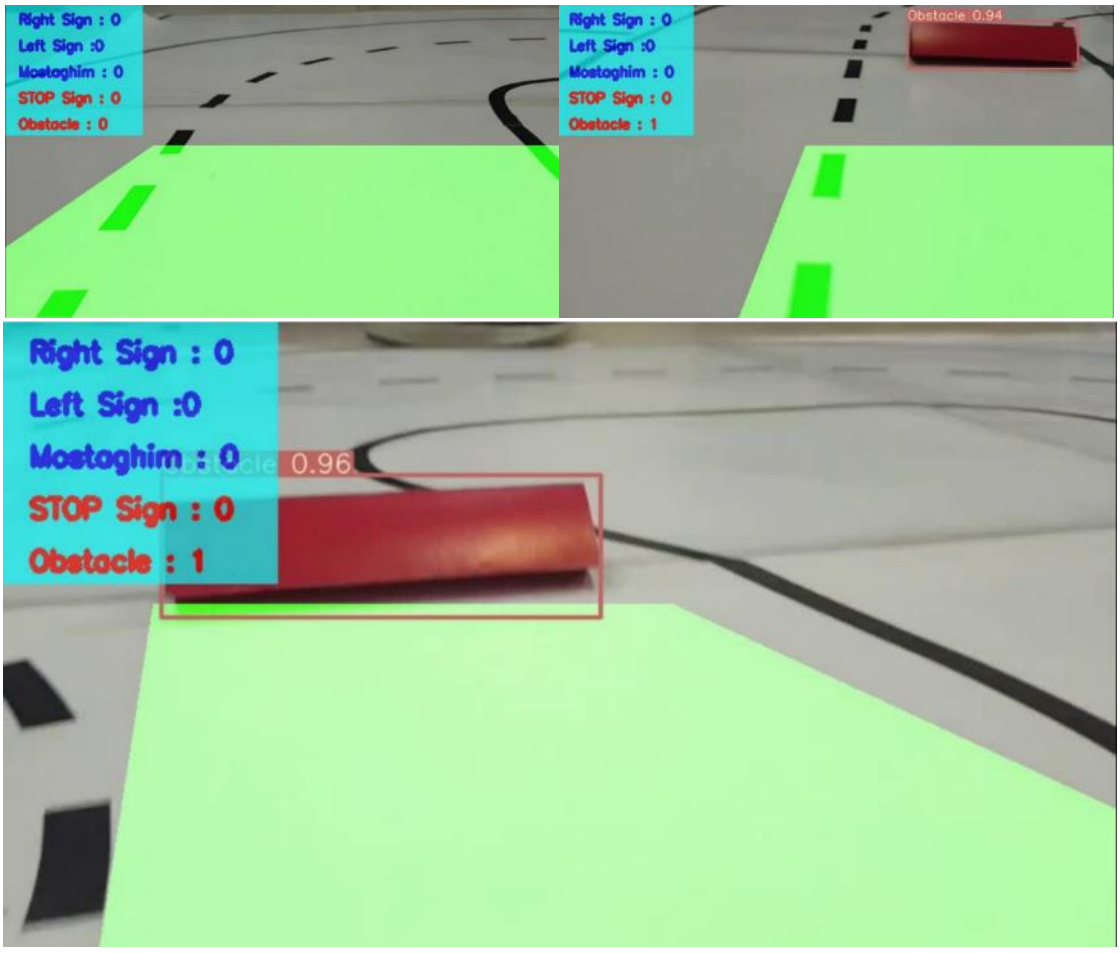
شکل ۲۶-۱۱ نتایج پردازش محیط از دید خودرو (on board) در محیط شهری



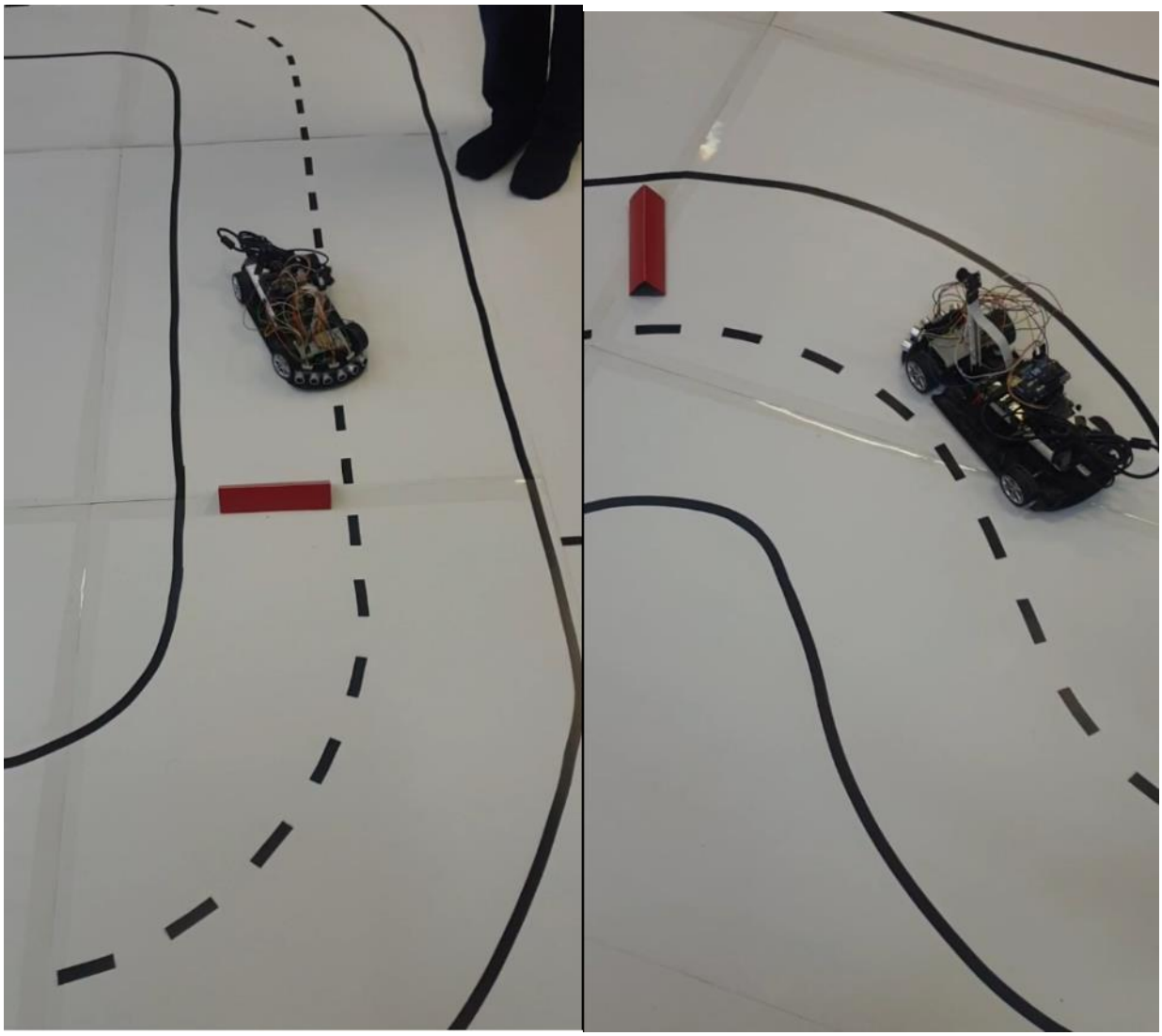
شکل ۲۷-۱۱ نتایج عملکرد خودرو در نقشه (off board) در محیط شهری

۳-۴-۱۱- نقشه بین شهری

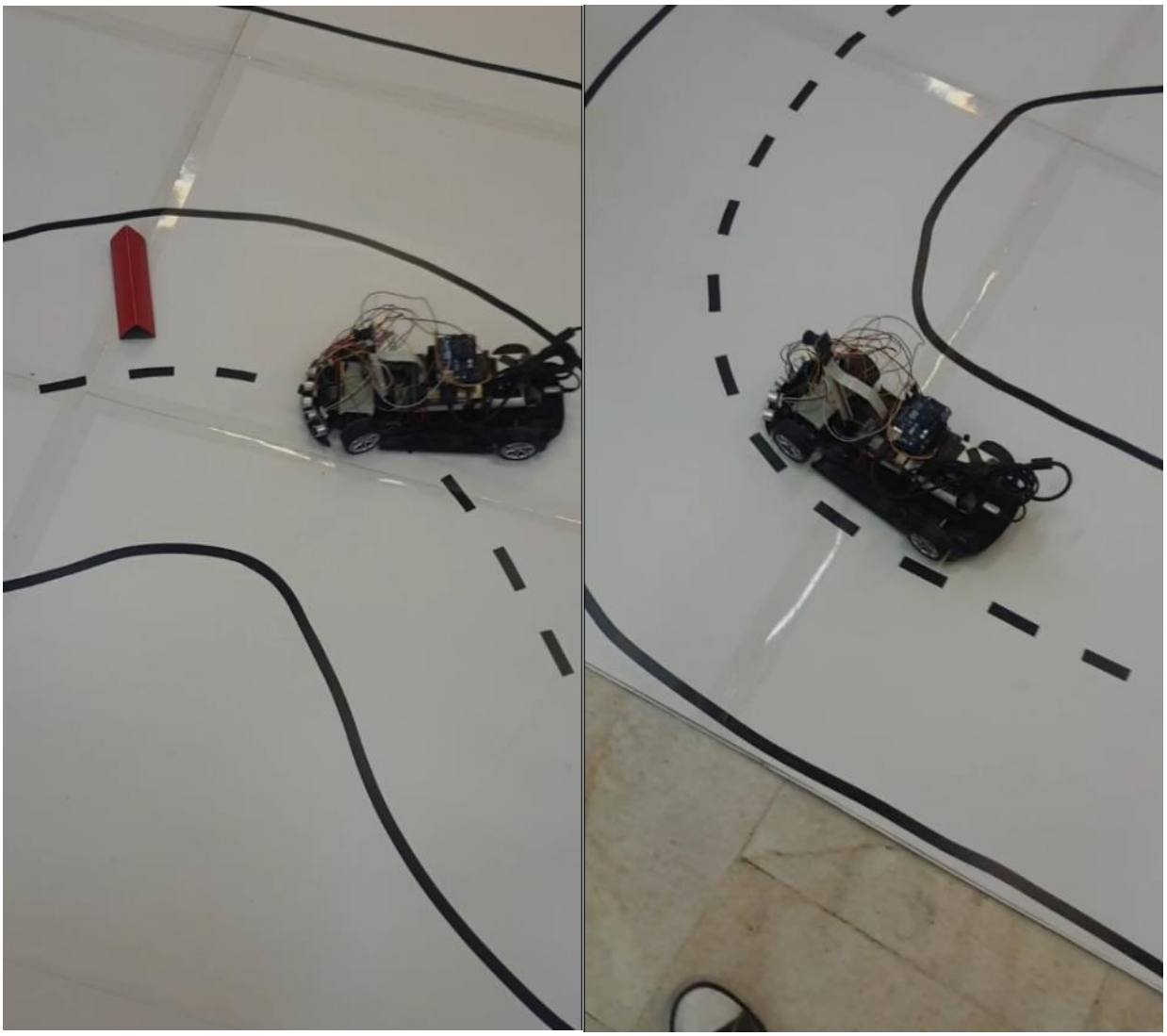
در بخش بین شهری نیز نتایج در دو دسته دید خودرو و عملکردی خودرو نشان داده شده اند. این بخش نیز شامل عملکردهایی از جمله تشخیص خطوط، تشخیص و دور زدن موانع می‌باشد و دراصل چالش این بخش در پیچ و خم جاده است. به این صورت که خودرو باید تصمیم گیری سریع برای تنظیم زاویه فرمان از خود نشان دهد.



شکل ۲۸-۱۱ نتایج پردازش محیط از دید خودرو (on board) در محیط بین شهری



شکل ۲۹-۱۱ نتایج عملکرد خودرو در نقشه (off board) در محیط بین شهری



شکل ۳۰-۱۱ نتایج عملکرد خودرو در نقشه (off board) در محیط بین شهری

۵-۱۱- نتیجه گیری

استفاده از خودروهای خودران در سال های اخیر، اهمیت به سزایی پیدا کرده است چرا که به کمک سطوح متفاوت هوشمندی این خودرو ها، می توان از تصادفات منجر به مرگ و میر، تا حد زیادی جلوگیری نمود. همچنین قابلیت های ارائه شده توسط این گونه خودرو ها مانند پارک خودکار یا تشخیص هوشمد علائم راهنمایی و رانندگی و عمل به دستورات آنها، رانندگی قانون مند و آسوده در محیط های شهری و بین شهری را بسیار تسهیل نموده است.

این مقاله، حاصل یک سال و شش ماه پژوهش، تحقیق، آزمایش و پیاده سازی عملگر های مختلف یک خودروی هوشمند بوده که موفق به کسب چندین مقام کشوری و بین المللی گردیده است از جمله مقام اول در مسابقات کشوری بنیاد ملی نخبگان (رهنشان) و همچنین کسب مقام بهترین گزارش فنی در مسابقات خودروی خودران فیرا و با توجه به نتایج حاصله در رویدادها و مسابقات هجده ماه اخیر، از نظر علمی و عملی، قابل پیاده سازی و بهره برداری در سطح صنعتی و تجاری می باشد.

کلیه ی بخش های مختلف پروژه به صورت پیاده سازی شده ارزیابی گردیده اند و نتایج قابل قبولی را دریافت نموده اند که طی مراحل ارزیابی و آزمایش، الگوریتم *A برای مسیر یابی، کنترلر MPC برای دنبال کردن مسیر، الگوریتم Hough برای تشخیص خطوط افقی، شبکه ی YOLOv5 برای تشخیص علائم و اشیا، دیتاست های Viscos و Tsinghua برای آموزش مدل به جهت تشخیص علائم، دیتاست Coco برای آموزش مدل به جهت تشخیص اشیا، شبکه ی SGDepth برای تشخیص پیاده رو ها و بخش بندی تصویر دریافتی از دوربین به کمک دیتاست Kitti و همچنین تشخیص فاصله با خودرو های اطراف به کمک دیتاست Cityscapes، شبکه ی PINet برای تشخیص خطوط عمودی به کمک دیتاست های CULane و TuSimple و کنترلر PID برای کنترل خودرو درون لاین جاده، بهترین نتایج را در بین روش های متعدد بررسی شده کسب نموده اند.

با توجه به نتایج به دست آمده، می توان نتیجه گیری نمود که پیاده سازی این پروژه به صورت ماژولار روی خودرو های داخلی به کمک یک دوربین، دو سنسور فاصله سنج، یک میکرو کنترلر برای انجام پردازش های مورد نیاز و در نهایت یک نمایشگر برای ارائه ی خروجی های پردازش شده، دور از دسترس نیست و قابل بررسی می باشد.

۶-۱۱- پیشنهادات

در حال حاضر، حجم بالای محاسبات الگوریتم ها و کنترلر های مورد استفاده در بخش های مختلف خودرو و همچنین محدودیت های سخت افزاری، باعث محدود شدن حداکثر سرعت قابل اطمینان خودرو شده اند. البته با توجه به این که خودروهای خودران به جهت افزایش ایمنی سرنشینان در سطح جهانی نیز سرعت محدودی دارند، چالش خاصی برای پروژه ایجاد نمی گردد اما می توان با ارتقای سخت افزار مورد استفاده و بهینه سازی الگوریتم ها و یا استفاده از چندین کنترلر خطی و غیر خطی به صورت آبشاری[۳۱۷] به جای استفاده از دو کنترلر کلی، سرعت پاسخ گویی خودرو به فرمان های کنترلی را افزایش داد. در نتیجه در پروژهش های آتی می توان با اعمال پیشنهادات مطرح شده، سرعت ماکسیمم خودرو را برای داشتن عملکرد بهینه و قابل اطمینان، افزایش داد.

[317] Cascade

مراجع

[1] Chen, T., Li, Z., He, Y., Xu, Z., Yan, Z., & Li, H. (2019). From perception to control: an autonomous driving system for a formula student driverless car. *arXiv preprint arXiv:1909.00119*.

[2] The 6 Levels of Vehicle Autonomy ExplainedRetrieved from https://www.synopsys.com/automotive/autonomous-driving-levels.html

[3] Key Tech Trends Shaping Automotive Industry Retrieved From https://marketpublishers.com/lists/23565/news.html

[4] Impact of COVID-19 on Autonomous Driving and Connected Vehicle Technology Retrieved From http://autonews.gasgoo.com/report/70016920.html

[5] Introduction to Self-Driving Cars Retrieved from https://www.coursera.org/learn/intro-self-driving-cars.

[6] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *J. Mod. Transp.*, vol. 24, no. 4, pp. 284–303, Dec. 2016, doi: 10.1007/s40534-016-0117-3.

[7] "Self-Driving Car Technology: How Do Self-Driving Cars Work? | Landmark Dividend" Retrieved From https://www.landmarkdividend.com/self-driving-car.

[8] "Why LiDAR is important for autonomous vehicle?" Retrieved From https://www.geospatialworld.net/blogs/why-lidar-is-important-for-autonomous-vehicle/#:~:text=This%20device%20is%20LiDAR%20that,surrounding%20objects%20and%20reflect%20back.

[9] "Technology In Autonomous Cars: What, Why, When & How?" Retrieved From https://levelfivesupplies.com/technology-in-autonomous-cars-what-why-when-how.

[10] "LiDAR Boosts Brain Power for Self-Driving Cars" Retrieved From https://eijournal.com/resources/lidar-solutions-showcase/lidar-boosts-brain-power-for-self-driving-cars.

[11] "Ultrasonic Sensors: More Than Just Parking" Retrieved From https://levelfivesupplies.com/ultrasonic-sensors-more-than-just-parking.

[12] "3 types of autonomous vehicle sensors in self-driving cars" Retrieved From https://www.itransition.com/blog/autonomous-vehicle-sensors.

[13] “Meet the self-driving car industry’s most important lobbyist” Retrieved From https://www.theverge.com/2016/4/27/11519680/david-strickland-interview-self-driving-car-lobbyist-nhtsa.

[14] “How RADARs work” Retrieved From https://medium.com/think-autonomous/how-radarswork1eb523893d62#:~:text=In%20self%2Ddriving%20cars%2C%20RADAR,an%20awesome%20sensor%20fusion%20module.

[15] Hussain, R., & Zeadally, S. (2018). Autonomous cars: Research results, issues, and future challenges. IEEE Communications Surveys & Tutorials, 21(2), 1275-1313.

[16] Zhao, J., Liang, B., & Chen, Q. (2018). The key technology toward the self-driving car. *International Journal of Intelligent Unmanned Systems*.

[17] “How DMS is Creating the Future of Mobility - Giving Drivers a Taste of "Omotenashi" “ Retrieved From https://www.mitsubishielectric.com/en/our-stories/article/xx_ww_f_in2001_02.page

[18] Lee, M. H., Park, H. G., Lee, S. H., Yoon, K. S., & Lee, K. S. (2013). An adaptive cruise control system for autonomous vehicles. *International Journal of Precision Engineering and Manufacturing*, *14*(3), 373-380.

[19] “5G network as foundation for autonomous driving” Retrieved From https://www.telekom.com/en/company/details/5g-network-as-foundation-for-autonomous-driving-561986

[20] S. Khvoynitskaya, “3 types of autonomous vehicle sensors in self-driving carse,” 2020. https://bit.ly/2TWlMaw.

[21] “Autoware Tools,” 2020. https://bit.ly/3k0rSRx.

[22] W. Rahiman and Z. Zainal, “An overview of development GPS navigation for autonomous car,” *Proc. 2013 IEEE 8th Conf. Ind. Electron. Appl. ICIEA 2013*, no. June 2013, pp. 1112–1118, 2013, doi: 10.1109/ICIEA.2013.6566533.

[23] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016, doi: 10.1109/TIV.2016.2578706.

[24] Y. Pei, S. Biswas, D. S. Fussell, and K. Pingali, “An elementary introduction to Kalman filtering,” *Commun. ACM*, vol. 62, no. 11, pp. 122–133, 2019, doi: 10.1145/3363294.

[25] C. Stiller, F. Puente León, and M. Kruse, “Information fusion for automotive applications - An overview,” *Inf. Fusion*, vol. 12, no. 4, pp. 244–252, 2011, doi: 10.1016/j.inffus.2011.03.005.

[26] S. J. Babak, S. A. Hussain, B. Karakas, and S. Cetin, “Control of autonomous ground vehicles: A brief technical review,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 224, no. 1, 2017, doi: 10.1088/1757-899X/224/1/012029.

[27] A. Khodayari, A. Ghaffari, S. Ameli, and J. Flahatgar, “A historical review on lateral and longitudinal control of autonomous vehicle motions,” *ICMET 2010 - 2010 Int. Conf. Mech. Electr. Technol. Proc.*, no. Icmet, pp. 421–429, 2010, doi: 10.1109/ICMET.2010.5598396.

[28] T. Gu and J. M. Dolan, “On-road motion planning for autonomous vehicles,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7508 LNAI, no. PART 3, pp. 588–597, 2012, doi: 10.1007/978-3-642-33503-7_57.

[29] R. W. Rbe, “Autonomous Ground Vehicle Prototype via Steering- , Throttle- , and Brake-by Wire Modules,” no. August 2015, 2016.

[30] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, “Autonomous vehicles: challenges, opportunities, and future implications for transportation policies,” *J. Mod. Transp.*, vol. 24, no. 4, pp. 284–303, Dec. 2016, doi: 10.1007/s40534-016-0117-3.

[31] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomous driving: Technical, legal and social aspects*. 2016.

[32] “How Automatic Braking Systems Work” Retrieved from https://auto.howstuffworks.com/

[33] “Lane centering” Retrieved from https://en.wikipedia.org/

[34] http://www.asrekhodro.com/

[35] “How Self-parking Cars Work” Retrieved from https://auto.howstuffworks.com/

[36] Mu, G., Xinyu, Z., Deyi, L., Tianlei, Z., & Lifeng, A. (2015). Traffic light detection and recognition for autonomous vehicles. The Journal of China Universities of Posts and Telecommunications, 22(1), 50-56.

[۳۷] “آمار تولید سال ۹۸ ایران خودرو منتشر شد,” *اسب بخار*. https://asbe-bokhar.com/article/news/آمار-تولید-سال-۹۸-ایران-خودرو-منتشر-شد./

[۳۸] م. برزگر و غ. الهام, “مسئولیّت کیفری کاربر خودروی خودران در قبال صدمات وارده توسط آن,” *پژوهش حقوق کیفری*, ۱۳۹۹.

[۳۹] م. خسروی, “نقش استارت‌آپ ها در آینده صنعت خودرو,” *تدبیر*, ۱۳۹۶.

[۴۰] وزارت صنعت معدن و تجارت - معاونت طرح و برنامه - دفترآمار و فرآوری داده ها - دفتر صنایع خودرو و نیرو محرکه, “گزارش ادواری خودرو سواری.1396, 28–0 .pp ”,

[۴۱] ع. صبحانی, “نقد و بررسی کامل ب ام و سری۷ مدل ۲۰۱۷ *mashin3*, 1395. ”, https://www.mashin3.com/fa/news-view/6915/نقد-و-بررسی-کامل-ب-ام-و-سری۷-مدل-۲۰۱۷.html.

[۴۲] "اتاق بازرگانی، صنایع، مهادن و کشاورزی تهران.www.tccim.ir ".

[43] M. Ryan, "The Future of Transportation: Ethical, Legal, Social and Economic Impacts of Self-driving Vehicles in the Year 2025," *Sci. Eng. Ethics*, vol. 26, no. 3, pp. 1185–1208, 2020, doi: 10.1007/s11948-019-00130-2.

[44] H. Tomita, "Awaiting the realization of fully automated vehicles: Potential economic effects and the need for a new economic and social design," *Voxeu cepr*, 2017. https://voxeu.org/article/potential-economic-and-social-effects-driverless-cars.

[45] L. M. Clements and K. M. Kockelman, "ECONOMIC EFFECTS OF AUTOMATED VEHICLES," *Transp. Res. Rec.*, pp. 1–19, 2017.

[46] "Self-Driving Vehicles And The Environment," *Energy Innovation*, 2019. energyinnovation.org/2019/03/25/self-driving-vehicles-and-the-environment/.

[47] "Autopilot and Full Self-Driving Capability " Retrieved From https://www.tesla.com/support/autopilot

[48] " Model S " Retrieved From https://www.tesla.com/models

[49] " Model 3 " Retrieved From https://www.tesla.com/model3

[50] " Model X " Retrieved From https://www.tesla.com/modelx

[51] " Model Y " Retrieved From https://www.tesla.com/modely

[52] "Waymo" Retrieved From https://waymo.com

[53] "Waymo technology" Retrieved From https://waymo.com/tech

[54] "Lucid Motors" Retrieved From https://www.lucidmotors.com

[55] "Fully automated driving" Retrieved From https://www.volkswagen-newsroom.com/en/fully-automated-driving-3633

[56] "Autonomous." Retrieved From https://www.mercedes-benz.com/en/innovation/autonomous

[57] "The Pass to Autonomous Driving" Retrieved From https://www.bmw.com/en/automotive-life/autonomous-driving.html

[58] "Argo AI: We're building self-driving technology you can trust" Retrieved From. https://www.argo.ai

[59] "Autonomous Driving Unit" Retrieved From https://usa.baidu.com/adu

[60] https://www.zoomit.ir/2017/6/12/167712/iran-autonomous-car/

[61] Levinson, J., & Thrun, S. (2010, May). Robust vehicle localization in urban environments using probabilistic maps. In 2010 IEEE International Conference on Robotics and Automation (pp. 4372-4378). IEEE.
[62] de Paula Veronese, L., de Aguiar, E., Nascimento, R. C., Guivant, J., Cheein, F. A. A., De Souza, A. F., & Oliveira-Santos, T. (2015, September). Re-emission and satellite aerial maps applied to vehicle localization on urban environments. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 4285-4290). IEEE.
[63] Xu, Y., John, V., Mita, S., Tehrani, H., Ishimaru, K., & Nishino, S. (2017, June). 3D point cloud map based vehicle localization using stereo camera. In 2017 IEEE Intelligent Vehicles Symposium (IV) (pp. 487-492). IEEE.
[64] Viswanathan, A., Pires, B. R., & Huber, D. (2016). Vision-based robot localization across seasons and in remote locations. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4815–4821). IEEE
[65] Brubaker, M. A., Geiger, A., & Urtasun, R. (2015). Map-based probabilistic visual self¬localization. IEEE transactions on pattern analysis and machine intelligence, 38, 652–665.
[66] Ziegler, J., Lategahn, H., Schreiber, M., Keller, C. G., Knöppel, C., Hipp, J., Haueis, M., & Stiller, C. (2014). Video based localization for bertha. In 2014 IEEE intelligent vehicles symposium proceedings (pp. 1231–1238). IEEE.
[67] Suhr, J. K., Jang, J., Min, D., & Jung, H. G. (2016). Sensor fusion-based low-cost vehicle localization system for complex urban environments. IEEE Transactions on Intelligent Transportation Systems, 18(5), 1078–1086.
[68] Moravec, H., & Elfes, A. (1985). High resolution maps from wide angle sonar. In Proceedings. 1985 IEEE International Conference on Robotics and Automation (pp. 116–121). IEEE volume 2.
[69] Kim, S., & Kim, J. (2013). Continuous occupancy maps using overlapping local gaussian processes. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4709–4714). IEEE.
[70] Thrun, S., & Montemerlo, M. (2006). The graph slam algorithm with applications to large-scale mapping of urban structures. The International Journal of Robotics Research, 25, 403–429.
[71] Thrun, S., & Montemerlo, M. (2006). The graph slam algorithm with applications to large-scale mapping of urban structures. The International Journal of Robotics Research, 25, 403–429.
[72] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C. et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. Journal of Field Robotics, 25, 425–466
[73] Ramm, F., Topf, J., & Chilton, S. (2011). OpenStreetMap: using and enhancing the free map of the world. UIT Cambridge Cambridge.

[74] Amaral, E., Badue, C., Oliveira-Santos, T., & De Souza, A. F. (2015). Detecc¸ao ˜ e rastreamento de ve´ıculos em movimento para automoveis rob ´ oticos ´ autonomos. In ˆ XII Simp´osio Brasileiro de Automa¸c˜ao Inteligente (SBAI) (pp. 801–806)
[75] Zhang, L., Li, Q., Li, M., Mao, Q., & Nüchter, A. (2013). Multiple vehicle-like target tracking based on the velodyne lidar. IFAC Proceedings Volumes, 46(10), 126-131.
[76] Petrovskaya, A., Perrollaz, M., Oliveira, L., Spinello, L., Triebel, R., Makris, A., ... & Bessière, P. (2012). Awareness of road scene participants for autonomous driving. Handbook of Intelligent Vehicles, 1383-1432.
[77] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., ... & Johnston, D. (2008). Junior: The stanford entry in the urban challenge. Journal of field Robotics, 25(9), 569-597.
[78] Ess, A., Schindler, K., Leibe, B., & Van Gool, L. (2010). Object detection and tracking for autonomous navigation in dynamic environments. The International Journal of Robotics Research, 29(14), 1707-1725.
[79] Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., ... & Kaus, E. (2014). Making bertha drive—an autonomous journey on a historic route. IEEE Intelligent transportation systems magazine, 6(2), 8-20.
[80] Lindner, F., Kressel, U., & Kaelberer, S. (2004, June). Robust recognition of traffic signals. In IEEE Intelligent Vehicles Symposium, 2004 (pp. 49-53). IEEE.
[81] Jang, C., Kim, C., Kim, D., Lee, M., & Sunwoo, M. (2014, June). Multiple exposure images based traffic light recognition. In 2014 IEEE Intelligent Vehicles Symposium Proceedings (pp. 1313-1318). IEEE.
[82] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
[83] Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1,** 269–271 (1959).
[84] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136.
[85] Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., & Werneck, R. F. (2015). Route planning in transportation networks. arXiv preprint arXiv:1504.05140.
[86] Hilger, M., Kohler, E., M ¨ ohring, R. H., & Schilling, H. (2009). Fast point-to- ¨ point shortest path computations with arc-flags. The Shortest Path Problem: Ninth DIMACS Implementation Challenge, 74, 41–72
[87] Delling, D., Holzer, M., Muller, K., Schulz, F., & Wagner, D. (2009). High- ¨ performance multi-level routing. The Shortest Path Problem: Ninth DIMACS Implementation Challenge, 74, 73–92.
[88] Geisberger, R., Sanders, P., Schultes, D., & Vetter, C. (2012). Exact routing in large road networks using contraction hierarchies. Transportation Science, 46, 388–404.
[89] Gutman, R. J. (2004). Reach-based routing: A new approach to shortest path algorithms optimized for road networks. ALENEX/ANALC, 4, 100–111.

[90] Gonzalez, D., P ´ erez, J., Milan ´ es, V., & Nashashibi, F. (2015). A review of ´ motion planning techniques for automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 17, 1135–1145.
[91] Chu, K., Lee, M., & Sunwoo, M. (2012). Local path planning for off-road autonomous driving with avoidance of static obstacles. IEEE Transactions on Intelligent Transportation Systems, 13, 1599–1616.
[92] Arnay, R., Morales, N., Morell, A., Hernandez-Aceituno, J., Perea, D., Toledo, J. T., Hamilton, A., Sanchez-Medina, J. J., & Acosta, L. (2016). Safe and reliable path planning for the autonomous vehicle verdino. IEEE Intelligent Transportation Systems Magazine, 8, 22–32.
[93] Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D. et al. (2008). Odin: Team victortango's entry in the darpa urban challenge. Journal of Field Robotics, 25, 467–492.
[94] Goldberg, A. V., Kaplan, H., & Werneck, R. F. (2006). Reach for a*: Shortest path algorithms with preprocessing. In The Shortest Path Problem (pp. 93– 140).
[95] Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., & Wagner, D. (2010). Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. ACM Journal of Experimental Algorithmics, 15.
[96] Greene, J., Rossi, F., Tasioulas, J., Venable, K. B., & Williams, B. (2016). Embedding ethical principles in collective decision support systems. In Thirtieth AAAI Conference on Artificial Intelligence.
[97] Conitzer, V., Sinnott-Armstrong, W., Borg, J. S., Deng, Y., & Kramer, M. (2017). Moral decision making frameworks for artificial intelligence. In Thirty-first aaai conference on artificial intelligence.
[98] Awad, E., Dsouza, S., Kim, R., Schulz, J., Henrich, J., Shariff, A., Bonnefon, J.-F., & Rahwan, I. (2018). The moral machine experiment. Nature, 563, 59.
[99] Guidolini, R., Scart, L. G., Jesus, L. F., Cardoso, V. B., Badue, C., & OliveiraSantos, T. (2018). Handling pedestrians in crosswalks using deep neural networks in the iara autonomous car. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). IEEE.
[100] Buehler, M., Iagnemma, K., & Singh, S. (2009). The DARPA urban challenge: autonomous vehicles in city traffic volume 56. springer.
[101] Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., Homm, F., Huber, W., & Kaempchen, N. (2015). Experience, results and lessons learned from automated driving on germany's highways. IEEE Intelligent transportation systems magazine, 7, 42–57.
[102] Okumura, B., James, M. R., Kanzawa, Y., Derry, M., Sakai, K., Nishi, T., & Prokhorov, D. (2016). Challenges in perception and decision making for intelligent automotive vehicles: A case study. IEEE Transactions on Intelligent Vehicles, 1, 20–32.
[103] Zhao, L., Ichise, R., Yoshikawa, T., Naito, T., Kakinami, T., & Sasaki, Y. (2015). Ontology-based decision making on uncontrolled intersections and narrow roads. In 2015 IEEE intelligent vehicles symposium (IV) (pp. 83–88). IEEE
[104] Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. MIT press.

[105] Ulbrich, S., & Maurer, M. (2013). Probabilistic online pomdp decision making for lane changes in fully automated driving. In 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013) (pp. 2063– 2067). IEEE.
[106] Brechtel, S., Gindele, T., & Dillmann, R. (2014). Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In 17th International IEEE Conference on Intelligent Transportation Systems (ITSC) (pp. 392–399). IEEE.
[107] Gonzalez, D., P ´ erez, J., Milan ´ es, V., & Nashashibi, F. (2015). A review of ´ motion planning techniques for automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 17, 1135–1145.
[108] Paden, B., Cˇ ap, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). A survey ´ of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles, 1, 33–55.
[109] Pivtoraiko, M., Knepper, R. A., & Kelly, A. (2009). Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics, 26, 308–333.
[110] Gu, T., Atwood, J., Dong, C., Dolan, J. M., & Lee, J.-W. (2015). Tunable and stable real-time trajectory planning for urban autonomous driving. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 250–256). IEEE.
[111] LaValle, S. M., & Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. The international journal of robotics research, 20, 378–400.
[112] Radaelli, R. R., Badue, C., Gonc¸alves, M. A., Oliveira-Santos, T., & De Souza, A. F. (2014). A motion planner for car-like robots based on rapidly-exploring random trees. In Ibero-American Conference on Artificial Intelligence (pp. 469–480). Springer.
[113] Du, M., Mei, T., Liang, H., Chen, J., Huang, R., & Zhao, P. (2016). Drivers' visual behavior-guided rrt motion planner for autonomous on-road driving. Sensors, 16, 102.
[114] Alia, C., Gilles, T., Reine, T., & Ali, C. (2015). Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method. In 2015 IEEE Intelligent Vehicles Symposium (IV) (pp. 674–679). IEEE.
[115] Mouhagir, H., Talj, R., Cherfaoui, V., Aioun, F., & Guillemard, F. (2016). Integrating safety distances with trajectory planning by modifying the occupancy grid for autonomous vehicle navigation. In 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) (pp. 1114– 1119). IEEE.
[116] Ziegler, J., Bender, P., Dang, T., & Stiller, C. (2014). Trajectory planning for bertha—a local, continuous method. In 2014 IEEE Intelligent Vehicles Symposium (pp. 450–457). IEEE.
[117] Howard, T. M., & Kelly, A. (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. The International Journal of Robotics Research, 26, 141–166.
[118] Ferguson, D., Howard, T. M., & Likhachev, M. (2008). Motion planning in urban environments. Journal of Field Robotics, 25, 939–960.
[119] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C. et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. Journal of Field Robotics, 25, 425–466.
[120] Guidolini, R., Badue, C., Berger, M., de Paula Veronese, L., & De Souza, A. F. (2016). A simple yet effective obstacle avoider for the iara autonomous car. In 2016 IEEE 19th

International Conference on Intelligent Transportation Systems (ITSC) (pp. 1914–1919). IEEE.
[121] Cao, H., Song, X., Huang, Z., & Pan, L. (2016). Simulation research on emergency path planning of an active collision avoidance system combined with longitudinal control for an autonomous vehicle. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering, 230, 1624–1653.
[122] He, X., Liu, Y., Lv, C., Ji, X., & Liu, Y. (2019). Emergency steering control of autonomous vehicle for collision avoidance and stabilisation. Vehicle System Dynamics, 57, 1163–1187.
[123] Funke, J., Theodosis, P., Hindiyeh, R., Stanek, G., Kritatakirana, K., Gerdes, C., Langer, D., Hernandez, M., Muller-Bessler, B., & Huhnke, B. (2012). ¨ Up to the limits: Autonomous audi tts. In 2012 IEEE Intelligent Vehicles Symposium (pp. 541–547). IEEE.
[124] Astrom, K. J., & Murray, R. M. (2010). ¨ Feedback systems: an introduction for scientists and engineers. Princeton university press.
[125] Zhao, P., Chen, J., Song, Y., Tao, X., Xu, T., & Mei, T. (2012). Design of a control system for an autonomous vehicle based on adaptive-pid. International Journal of Advanced Robotic Systems, 9, 44.
[126] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V. et al. (2011). Towards fully autonomous driving: Systems and algorithms. In 2011 IEEE Intelligent Vehicles Symposium (IV) (pp. 163–168). IEEE.
[127] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B. et al. (2008). Junior: The stanford entry in the urban challenge. Journal of Field Robotics, 25, 569–597
[128] Paden, B., C˘ ap, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). A survey ´ of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles, 1, 33–55.
[129] Samson, C. (1995). Control of chained systems application to path following and time-varying point-stabilization of mobile robots. IEEE transactions on Automatic Control, 40, 64–77.
[130] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G. et al. (2006). Stanley: The robot that won the darpa grand challenge. Journal of Field Robotics, 23, 661–692.
[131] Guidolini, R., De Souza, A. F., Mutz, F., & Badue, C. (2017). Neural-based model predictive control for tackling steering delays of autonomous cars. In 2017 International Jo
[132] Koga, A., Okuda, H., Tazaki, Y., Suzuki, T., Haraguchi, K., & Kang, Z. (2016). Realization of different driving characteristics for autonomous vehicle by using model predictive control. In 2016 IEEE Intelligent Vehicles Symposium (IV) (pp. 722–728). IEEE.
[133] Kritayakirana, K., & Gerdes, J. C. (2012). Autonomous vehicle control at the limits of handling. Ph.D. thesis Stanford University Stanford, CA.
[134] Brown, M., Funke, J., Erlien, S., & Gerdes, J. C. (2017). Safe driving envelopes for path tracking in autonomous vehicles. Control Engineering Practice, 61, 307–316.
[135] “Valeo” Retrieved from http://www.valeo.com/en/ press-releases/details.html?id556.
[136] “Left Lane, Video: BMW’s automatic parking system” 17 February 2006, Retrieved from http://www.leftlanenews.com/video-bmws-automatic-parkingsystem.html.

[137] Neff, J. Video: self-parking Lexus befuddles automobile editors, Autoblog, 4 December 2006, Retrieved from http://www.autoblog.com/2006/12/04/videoself-parking-lexus-befuddles-automobile-editors/.
[138] Razinkova, A., Cho, H. C., & Jeon, H. T. (2012). An intelligent auto parking system for vehicles. *International Journal of Fuzzy Logic and Intelligent Systems*, *12*(3), 226-231.
[139] Ma, S. and Wang, C. (2011). Parking assist system based on visual perception system and fuzzy logic controller. J. Shenyang Jianzhu University (Natural Science) 27, 5, 1000−1004.
[140] Jiang, Z. and Zeng, W. (2008). Automatic vehicle parking system based on binocular vision and path planning. Highways and Automotive Applications, 4, 69−72.
[141] Paromtchik, I. E. and Laugier, C. (1996). Autonomous parallel parking of a nonholonomic vehicle. Proc. IEEE Intelligent Vehicles Symp., Tokyo, Japan, 13−18.
[142] Laugier, C., Fraichard, T. H. and Gamier, P. H. (1999). Sensor-based control architecture for a car-like vehicle. Autonomous Robot 6, 2, 165−185.
[143] Jiang. K. (2002). A sensor guided parallel parking system for nonholonomic vehicles. Proc. IEEE Conf. Intelligent Transportation Systems, Dearborn, MI, USA, 270−275.
[144] Jeong, S. H., Choi, C. G.., Oh, J. N., Yoon, P. J., Kim, B. S., Kim, M. and Lee, K. H. (2009). Low cost design of parallel parking assist based on an ultrasonic sensor. Int. J. Automotive Technology 11, 3 409−416.
[145] Shi, X. and Wang, C. (2010). An automatic parking system based on laser radar. Mechatronics 16, 3, 72−74.
[146] Bi, Y. and Tang, T. (2011). Auxiliary parking system based on multi-sensor information fusion. Sichuan Ordnance J. 32, 1, 110−112.
[147] Kang, B. and Liang, Y. (2011). Design of intelligent embed based parking assist system. J. Jilin University. Information Science edn. 30, 3, 223−227.
[148] Chou, L. D., Sheu, C. C. and Chen, H. W. (2006). Design and prototype implementation of a novel automatic vehicle parking system. Int. Conf. Hybrid Information Technology, Jeju Island, Korea, 2, 292−297.
[149] Borenstein, J. and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. IEEE Trans. Systems, Man, and Cybernetics 19, 5 1179−1187
[150] Borenstein, J. and Koren, Y. (1991). The vector field histogram - fast obstacle avoidance for mobile robots. IEEE J. Robotics and Automation 7, 3, 278−288.
[151] An, D. and Wang, H. V. (2004). PH: A new laser radar based obstacle avoidance method for intelligent mobile robots. 5th World Cong. Intelligent Control and Automation. Piscataway, NJ, USA, 5, 4681−4685.
[152] Dubins, L. E. (2011). On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal position and tangents. American J. Mathematics, 79, 497−516.
[153] Reeds, J. A. and Shepp, L. A. (1990). Optimal path for a car that goes both forward and backward. Pacific J. Mathematics 145, 2, 367−393.
[154] Kanayama, Y. and Hartman, B. I. (1989). Smooth local path planning for autonomous vehicles. Proc. IEEE Int. Conf. Robotics and Automation, Scottsdale, Arizona, 1265−1270.

[155] Kanayama, Y. and Hartman, B. I. (1989). Smooth local path planning for autonomous vehicles. Proc. IEEE Int. Conf. Robotics and Automation, Scottsdale, Arizona, 1265−1270.
[156] Boer, G. A. D. and Albada, G. D. V. (1993). The MARIE autonomous mobile robots. Proc. Conf. Intelligent Autonomous Systems IAS-s, Pittsburgh, 164−173.
[157] Lee, K., Kim, D. and Chung, W. (2006). Car parking control using a trajectory tracking controller. SICEICASE Int. Joint Conf., Busan, Korea, 2058−2063.
[158] Hsieh, M. F. (2008). A parking algorithm for an autonomous vehicle. IEEE Intelligent Vehicle Symp., Netherlands, 1155−1160.
[159] Ross, I. M. (2006). Issues in the real-time computation of optimal control. Mathematical and Computer Modeling, 43, 1172−1188.
[160] Verma, A. and Junkins, J. (1999). Inverse dynamics approach for real-time determination of feasible aircraft reference trajectories. AIAA Guidance Control and Navigation Conf., Portland, OR, 545−554.
[161] Nieuwstadt, V. (1997). Trajectory Generation for Nonlinear Control Systems. Ph.D. Dissertation. California Institute of Technology
[162] Milam, M. B. (2003). Real-time Optimal Trajectory Generation for Constrained Dynamical Systems. Ph.D. Dissertation. California Institute of Technology.
[163] Milam, M. B., Mushambi, K. and Murray, R. M. (2000). A new computational approach to real-time trajectory generation for constrained mechanical systems. Conf. Decision and Control, Sydney, NSW, 1, 845−851.
[164] Wu, W., Chen, H. and Woo, P. (1999). Optimal motion planning for a wheeled mobile robot. Proc. IEEE Int. Conf. Robotics and Automation, Detroit, Michigan, 1, 41−45.
[165] Laumond, J. P., Jacobs, P. E., Taix, M. and Murray, R. M. (1994). A motion planner for nonholonomic mobile robots. IEEE Trans. Robotics and Automation 10, 5, 577−593.
[166] Paromtchik, I. E. and Laugier, C. (1996). Motion generation and control for parking an autonomous vehicle. Proc. IEEE Int. Conf. Robotics and Automation, Minneapolis, MN, 4, 3117−3122
[167] Paromtchik, I. E. and Laugier, C. (1997). Autonomous parallel parking and returning to traffic maneuvers. Intelligent Robotics and System, 3, 21−23.
[168] Laugier, C., Fraichard, T. H. and Gamier, P. H. (1999). Sensor-based control architecture for a car-like vehicle. Autonomous Robot 6, 2, 165−185.
[169] Langer, D. and Thorpe, C. (1995). Range sensor based outdoor vehicle navigation, collision avoidance and parallel parking. Autonomous Robots, 2, 147−161.
[170] Jiang, K. and Seneviratne, L. D. (1999). A sensor guided autonomous parking system for nonholonomic mobile robots. Proc. 1999 IEEE Int. Conf. Robotics and Automation, Detroit, Michigan, 1, 311−316.
[171] Xu, J., Chen, G. and Xie, M. (2000). Vision-guided automatic parking for smart car. Proc. IEEE Int. Vehicles Symp., Dearborn, USA, 725−730.
[172] Lo, Y. K., Rad, A. B., Wong, C. W. and Ho, M. L. (2003). Automatic parallel parking. 6th IEEE Int. Conf. Inte11igent Transportation Systems, Shanghai, 2, 1190− 1193.
[173] Lee, S., Kim, M., Youm, Y. and Chung, W. (1999). Control of a car-like robot for parking problem. Proc. IEEE Robotics and Automation, Detroit, MI, USA, l−6.

[174] Inoye, T., Dao, M. Q. and. Liu, K. Z. (2004). Development of an auto-parking system with physical limitations. SICE Annual Conf., Sapporo, Japan, 2, 1015−1020.
[175] Sugeno, M. and Murakami, K. (1984). Fuzzy parking control of a model car. Proc. 23rd Conf. Decision and Control, Las Vegas, NV, 902−903.
[176] Sugeno, M., Murofushi, T., Mori, T., Tatematsu, T. and Tanaka, J. (1989). Fuzzy algorithmic control of a model car by oral instructions. Fuzzy Sets System, 32, 207–219.
[177] Nguyen, D. H. and Widrow, B. (1990). Neural networks for self-learning control systems. Control Systems Magazine, IEEE 10, 3, 18−23.
[178] Kong, S. G. and Kosko, B. (1990). Comparison of fuzzy and neural truck backer-upper control. 1990 IJCNN Int. Joint Conf. Neural Networks, San Diego, CA, USA, 349−358.
[179] Yasunobu, S. and Murai, Y. (1994). Parking control based on predictive fuzzy control. Proc. IEEE Int. Conf. Fuzzy System, Orlando, FL, 2, 1338−1341.
[180] Jenkins, R. E. and Yuhas, B. P. (1993). A simplified neural network solution through problem decomposition: The case of the truck backer-upper. IEEE Trans. Neural Network, 4, 718−720.
[181] Gorinevsky, D., Pitanovsky, A. K. and Goldenberg, A. (1996). Neural network architecture for trajectory generation and control of automated car parking. IEEE Trans. Controls Systems Technology, 4, 50−56.
[182] Leitch, D. and Probert, P. J. (1998). New techniques for genetic development of a class of fuzzy controllers. IEEE Trans. Systems, Man and Cybernetics 28, 1, 112− 123.
[183] Leu, M. C. and Kim, T. Q. (1998). Cell mapping based fuzzy control of car parking. Proc. 1998 IEEE Int. Conf. Robotics and Automation, Leuven, 3, 2494−2499.
[184] Gómez-Bravo, F., Cuesta, F. and Ollero, A. (2001). Parallel and diagonal parking in nonholonomic autonomous vehicles. Engineering Applications of Artificial Intelligence 14, 4, 419−434.
[185] Su-Jin, P. P., Lebeltel, O. and Laugier, C. (2002). Parking a car using Bayesian programming. 7th Int. Conf. Control Automation Robotics and Vision, 2, 728−733.
[186] Li, T. S. and Chang, S. J. (2003). Automous fuzzy parking control of a car-like mobile robot. IEEE Trans. Systems, Man and Cybernetics Part A-Systems and Humans, 33, 451−465.
[187] Zhao, Y. and Jr. E. G. C. (2005). Robust automatic parallel parking in tight spaces via Fuzz Logic. Robotics and Autonomous Systems, 51, 111–127.
[188] Muller, B., Deutscher, J. and Grodde, S. (2006). Trajectory generation and feedforward control for parking a car. Proc. 2006 IEEE Int. Conf. Control Applications, Munich, 163−168.
[189] Lee, J. Y. and Lee, J. J. (2007). Multiple designs of fuzzy controllers for car parking using evolutionary algorithm. Proc. Int. Conf. Mechatronics, Kumamoto, 1−6.
[190] Ballinas, E., Montiel, O., Castillo, O., Rubio, Y., & Aguilar, L. T. (2018). Automatic Parallel Parking Algorithm for a Carlike Robot using Fuzzy PD+ I Control. *Engineering Letters*, *26*(4).
[191] Oyama, K., & Nonaka, K. (2013, July). Model predictive parking control for nonholonomic vehicles using time-state control form. In *2013 European Control Conference (ECC)* (pp. 458-465). IEEE.

[192] K.Oyama and K.Nonaka, ”Model Predictive Control for Nonholonomic Vehicles with Travel Range Constraint, ”Proceedings of the 29th Symposium on Guidance and Control, 2012, pp. 65-72 (in Japanese)
[193] L. Li, Z. Jia, T. Cheng and X. Jia, ”Optimal Model Predictive Control for Path Tracking of Autonomous Vehicle, ”Third International Conference on Measuring Technology and Mechatronics Automation, 2004, pp 791-794.
[194] Shin, J. H., Jun, H. B., & Kim, J. G. (2018). Dynamic control of intelligent parking guidance using neural network predictive control. *Computers & Industrial Engineering*, *120*, 15-30.
[195] Ye, H., Jiang, H., Ma, S., Tang, B., & Wahab, L. (2019). Linear model predictive control of automatic parking path tracking with soft constraints. *International Journal of Advanced Robotic Systems*, *16*(3), 1729881419852201.
[196] Kim, W., Kim, D., Yi, K., & Kim, H. J. (2012). Development of a path-tracking control system based on model predictive control using infrastructure sensors. *Vehicle system dynamics*, *50*(6), 1001-1023.Linear model predictive control of automatic parking path tracking with soft constraints
[197] Ye, H., Jiang, H., Ma, S., Tang, B., & Wahab, L. (2019). Linear model predictive control of automatic parking path tracking with soft constraints. *International Journal of Advanced Robotic Systems*, *16*(3), 1729881419852201.
[198] Dong, H. R., Jin, S. T., & Hou, Z. S. (2014, June). Model Free Adaptive Control for automatic car parking systems. In *Proceeding of the 11th World Congress on Intelligent Control and Automation* (pp. 1769-1774). IEEE.
[199] Z. Hou and Z. Wang, “From model-based control to data-driven control: Survey, classification and perspective,” Inf. Sci., vol. 235, pp. 3–35, 2013.
[200] Xu, D., Shi, Y., & Ji, Z. (2017). Model-free adaptive discrete-time integral sliding-mode-constrained-control for autonomous 4WMV parking systems. *IEEE Transactions on Industrial Electronics*, *65*(1), 834-843.
[201] Paul Viola and Michael Jones "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001.
[202] Alvaro Arcos-Garcı´a, Juan A. ´Alvarez-Garcı´a, Luis M. Soria-Morillo, Eval-uation of Deep Neural Networks for traffic sign detection systems, Neurocomputing (2018)
[203] M. Oliveira, V. Santos “AUTOMATIC DETECTION OF CARS IN REAL ROADS USING HAAR-LIKE FEATURES”, Department of Mechanical Engineering, University of Aveiro,2008
[204] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi,” You Only Look Once:Unified, Real-Time Object Detection”,2016
[205] https://github.com/ultralytics/yolov5
[206] Mingxing Tan, Ruoming Pang, Quoc V. Le, EfficientDet: Scalable and Efficient Object Detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition ,2020
[207] Canny, J., A Computational Approach To Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence,1986

[208] Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, 1972
[209] 75. https://paperswithcode.com/sota/graph-classification-on-proteins?p=pinet-a-permutation-invariant-graph-neural
[210] Ko, Y., Jun, J., Ko, D., & Jeon, M. (2020). Key Points Estimation and Point Instance Segmentation Approach for Lane Detection. arXiv preprint arXiv:2002.06604.
[211] Tabelini, L., Berriel, R., Paixão, T. M., Badue, C., De Souza, A. F., & Olivera-Santos, T. (2020). Keep your Eyes on the Lane: Attention-guided Lane Detection. arXiv preprint arXiv:2010.12035.
[212] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, In So Kweon; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1947-1955.
[213] Pan, X., Shi, J., Luo, P., Wang, X., & Tang, X. (2017). Spatial as deep: Spatial cnn for traffic scene understanding. arXiv preprint arXiv:1712.06080.
[214] Ahmad ALI, Abdul JALIL, Jianwei NIU , Xiaoke ZHAO, Saima RATHORE, Javed AHMED, Muhammad AKSAM IFTIKHAR, “Visual object tracking—classical and contemporary approaches”,2015
[215] Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., & He, Z. (2017, May). Spatially supervised recurrent convolutional neural networks for visual object tracking. In 2017 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-4). IEEE.
[216] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., & Torr, P. H. (2019). Fast online object tracking and segmentation: A unifying approach. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1328-1338).
[217] Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP) (pp. 3645-3649). IEEE.
[218] Bergmann, P., Meinhardt, T., & Leal-Taixe, L. (2019). Tracking without bells and whistles. In Proceedings of the IEEE international conference on computer vision (pp. 941-951).
[219] https://towardsdatascience.com/vehicle-detection-and-distance-estimation-7acde48256e1
[220] https://blogs.nvidia.com/blog/2019/06/19/drive-labs-distance-to-object-detection
[221] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zhang, X. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.
[222] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
[223] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).
[224] Levinson, J., & Thrun, S. (2010, May). Robust vehicle localization in urban environments using probabilistic maps. In 2010 IEEE International Conference on Robotics and Automation (pp. 4372-4378). IEEE.

[225] Dong, H. R., Jin, S. T., & Hou, Z. S. (2014, June). Model Free Adaptive Control for automatic car parking systems. In *Proceeding of the 11th World Congress on Intelligent Control and Automation* (pp. 1769-1774). IEEE.

[226] Lee, B., Wei, Y., & Guo, I. Y. (2017). Automatic parking of self-driving car based on lidar. *The International Archives of the photogrammetry, remote sensing and spatial information sciences*, *42*.

[227] Heimberger, M., Horgan, J., Hughes, C., McDonald, J., & Yogamani, S. (2017). Computer vision in automated parking systems: Design, implementation and challenges. *Image and Vision Computing*, *68*, 88-101.

[228] Xu, D., Shi, Y., & Ji, Z. (2017). Model-free adaptive discrete-time integral sliding-mode-constrained-control for autonomous 4WMV parking systems. *IEEE Transactions on Industrial Electronics*, *65*(1), 834-843.

[229] Li, T. H., & Chang, S. J. (2003). Autonomous fuzzy parking control of a car-like mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *33*(4), 451-465.

[230] Gupta, A., Divekar, R., & Agrawal, M. (2010, December). Autonomous parallel parking system for Ackerman steering four wheelers. In *2010 IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1-6). IEEE.

[231] Liang, Z., Zheng, G., & Li, J. (2012, August). Automatic parking path optimization based on bezier curve fitting. In 2012 IEEE International Conference on Automation and Logistics (pp. 583-587). IEEE.

[232] Ballinas, E., Montiel, O., Castillo, O., Rubio, Y., & Aguilar, L. T. (2018). Automatic Parallel Parking Algorithm for a Carlike Robot using Fuzzy PD+ I Control. *Engineering Letters*, *26*(4).

[233] Autonomous Reverse Parking System Based on Robust Path Generation and Improved Sliding Mode Control

[234] Ye, H., Jiang, H., Ma, S., Tang, B., & Wahab, L. (2019). Linear model predictive control of automatic parking path tracking with soft constraints. *International Journal of Advanced Robotic Systems*, *16*(3), 1729881419852201.

[235] Zhang, S., Simkani, M., & Zadeh, M. H. (2011, September). Automatic vehicle parallel parking design using fifth degree polynomial path planning. In *2011 IEEE Vehicular Technology Conference (VTC Fall)* (pp. 1-4). IEEE.

[236] Yurtsever, E., Lambert, J., Carballo, A., & Takeda, K. (2020). A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, *8*, 58443-58469.

[237] Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 390-391).

[238] "Yolo5" Retrieved From https://github.com/ultralytics/yolov5

[239] "YOLOv5 New Version - Improvements And Evaluation" Retrieved From https://blog.roboflow.com/yolov5-improvements-and-evaluation/

[240] Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

[241] Tan, M., Pang, R., & Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10781-10790).

[242] https://towardsdatascience.com/googles-efficientdet-an-overview-8d010fa15860

[243] "A Thorough Breakdown of EfficientDet for Object Detection" Retrieved From https://blog.roboflow.com/breaking-down-efficientdet/

[244] Alvaro Arcos-Garcı´a, Juan A. ´Alvarez-Garcı´a, Luis M. Soria-Morillo, Eval-uation of Deep Neural Networks for traffic sign detection systems, Neurocomputing (2018).

[245] Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., ... & Ang, M. H. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*, *5*(1), 6.

[246] Rangesh, A., & Trivedi, M. M. (2019). No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars. *IEEE Transactions on Intelligent Vehicles*, *4*(4), 588-599.

[247] Almohaimeed, N., & Prince, M. A Comparative Study of different Oject Tracking Methods in a Video. *International Journal of Computer Applications*, *975*, 8887.

[248] "Object Detection and Tracking in 2020" Retrived From https://blog.netcetera.com/object-detection-and-tracking-in-2020-f10fb6ff9af3

[249] Codevilla, F., Santana, E., López, A. M., & Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 9329-9338).

[250] Dong, H. R., Jin, S. T., & Hou, Z. S. (2014, June). Model Free Adaptive Control for automatic car parking systems. In *Proceeding of the 11th World Congress on Intelligent Control and Automation* (pp. 1769-1774). IEEE.

[251] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0.

[252] Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of cnn. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 390-391).

[253] https://github.com/ultralytics/yolov5

[254] yolov5-improvements-and-evaluation Retrived from https://blog.roboflow.com/yolov5-improvements-and-evaluation/.

[255] روش ها و معیار¬های ارزیابی الگوریتم¬های هوش¬مصنوعی، مدل های داده کاوی و یادگیری ماشین – Retrieved From https://bigdata-ir.com/.

[256] https://www.cvl.isy.liu.se/en/research/datasets/traffic-signs-dataset/

[257] https://sid.erda.dk/public/archives/ff17dc924eba88d5d01a807357d6614c/published-archive.html

[258] https://makeml.app/datasets/cars-and-traffic-signs

[259] https://makeml.app/road-signs-detection-tutorial

[260] https://www.vicos.si/Downloads/DFGTSD

[261] https://cg.cs.tsinghua.edu.cn/traffic-sign/

[262] Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1), 11-15.
[263] Ganokratanaa, T., Ketcham, M., & Sathienpong, S. (2013, July). Real-time lane detection for driving system using image processing based on edge detection and Hough transform. In Proceedings of the International Conference on Digital Information and Communication Technology and Its Applications (DICTAP'13).
[264] Hofbauer, M., Kuhn, C. B., Meng, J., Petrovic, G., & Steinbach, E. (2020, September). Multi-View Region of Interest Prediction for Autonomous Driving Using Semi-Supervised Labeling. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-6). IEEE.
[265] [265] Ko, Y., Jun, J., Ko, D., & Jeon, M. (2020). Key Points Estimation and Point Instance Segmentation Approach for Lane Detection. arXiv preprint arXiv:2002.06604.
[266] PINet_New Retrieved from https://github.com/koyeongmin/PINet_new.
[267] https://xingangpan.github.io/projects/CULane.html
[268] https://reposhub.com/python/deep-learning/xbjxh-CurveLanes.html.
[269] https://github.com/TuSimple/tusimple-benchmark/tree/master/doc /lane_detection.
[270] https://artsandculture.google.com/asset/architectural-veduta-francesco-di-giorgio-martini-attributed/kAGQtjZedrY6Xw?hl=en
[271] Magee, M. J., & Aggarwal, J. K. (1984). Determining vanishing points from perspective images. Computer Vision, Graphics, and Image Processing, 26(2), 256-267.
[272] Kunst, F. K., & Dwivedi, V. (2019). Non-Hermitian systems and topology: A transfer-matrix perspective. Physical Review B, 99(24), 245116.
[273] Intwala, C., & Agarwala, A. (2015). U.S. Patent No. 9,117,253. Washington, DC: U.S. Patent and Trademark Office.
[274] Klingner, M., Termöhlen, J. A., Mikolajczyk, J., & Fingscheidt, T. (2020, August). Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *European Conference on Computer Vision* (pp. 582-600). Springer, Cham.
[275] https://github.com/ifnspaml/SGDepth
[276] http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter11.pdf.
[277] Caselles, V., Catté, F., Coll, T., & Dibos, F. (1993). A geometric model for active contours in image processing. Numerische mathematik, 66(1), 1-31.
[278] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html
[279] Schwarz, M. W., Cowan, W. B., & Beatty, J. C. (1987). An experimental comparison of RGB, YIQ, LAB, HSV, and opponent color models. ACM Transactions on Graphics (TOG), 6(2), 123-158.
[280] "Vehicle Dynamics Terminology", SAE J670e, Society of Automotive Engineers (SAE), Inc., Warrendale, PA, July 1976.
[281] M. Guiggiani, "Dinamica del veicolo", CittàStudi Edizioni, Torino, 1998

[282] Fossen, T. I. (2002). Guidance, navigation, and control of ships, rigs and underwater vehicles. Marine Cybernetics. Noruega.
[283] Di Martino, R. (2005). Modelling and Simulation of the Dynamic Behaviour of the Automobile (Doctoral dissertation).
[284] T. D. Gillespie, “Fundamentals of Vehicle Dynamics”, Society of Automotive Engineers (SAE), Inc., Warrendale, PA, 1992.
[285] J. Y. Wong, “Theory of Ground Vehicles”, John Wiley and Sons, Inc., New York, 2001.
[286] M. Mitschke, “Dynamik der Kraftfahrzeuge”, Band A: Antrieb und Brehsung 3. Auflage, Springer, Berlin, 1995. (Relax length)
[287] U. Kiencke and L. Nielsen, "Automotive Control Systems", Society of Automotive Engineers (SAE), Inc., Warrendale, PA, 2000.
[288] G. Genta, Motor Vehicle Dynamics, World Scientific Publishing , Singapore, 1997.
[289] G. Genta, “Meccanica dell'autoveicolo”, Levrotto e Bella, 1997.
[290] A. R. Guido e L. Della Pietra, “Lezioni di meccanica delle macchine”, CUEN, Napoli 1989.
[291] Zeng, W.; Church, R. L. (2009). "Finding shortest paths on real road networks: the case for A*". International Journal of Geographical Information Science
[292] *Edelkamp, Stefan; Jabbar, Shahid; Lluch-Lafuente, Alberto (2005).* "Cost-Algebraic Heuristic Search"
[293] Hansen, Eric A., and Rong Zhou. "Anytime Heuristic Search." J. Artif. Intell. Res.(JAIR) 28 (2007)

[294] Schoenberg, Isaac J. (1946). "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions: Part A.—On the Problem of Smoothing or Graduation. A First Class of Analytic Approximation Formulae"
[295] Schoenberg, Isaac J. (1946). "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions: Part B.—On the Problem of Osculatory Interpolation. A Second Class of Analytic Approximation Formulae"
[296] Paromtchik, I. E., & Laugier, C. (1996, September). Autonomous parallel parking of a nonholonomic vehicle. In Proceedings of Conference on Intelligent Vehicles (pp. 13-18). IEEE.
[297] Lyon, D., “Parallel parking with curvature and nonholonomic constraints.” Proceedings of the IntelligentVehicles '92 Symposium, Jul. 341-346, 1992
[298] Ballinas, E., Montiel, O., Castillo, O., Rubio, Y., & Aguilar, L. T. (2018). Automatic Parallel Parking Algorithm for a Carlike Robot using Fuzzy PD+ I Control. Engineering Letters, 26(4).
[299] https://control.com/technical-articles/what-is-model-predictive-control-mpc/
[300] Ortega, J. G., & Camacho, E. F. (1996). Mobile robot navigation in a partially structured static environment, using neural predictive control. *Control Engineering Practice*, *4*(12), 1669-1679.
[301] Clarke, D. W. (1988). Application of generalized predictive control to industrial processes. *IEEE Control systems magazine*, *8*(2), 49-55.
[302] Richalet, J. (1993). Industrial applications of model based predictive control. *Automatica*, *29*(5), 1251-1274.

[303] https://blog.faradars.org/%DA%A9%D9%86%D8%AA%D8%B1%D9%84-%D9%BE%DB%8C%D8%B4-%D8%A8%DB%8C%D9%86-%D9%85%D8%AF%D9%84/
[304] An Introduction to Model-based Predictive Control (MPC) by Stanislaw H. Z_ ak
[305] Camacho, E. F., & Alba, C. B. (2013). *Model predictive control*. Springer Science & Business Media.
[306] "Path Tracking" Retrieved From https://pythonrobotics.readthedocs.io/en/latest/modules/path_tracking.html
[307] https://unity3d.com/unity/qa/patch-releases/2017.1.3p1.
[308] https://avisengine.com
[309] https://www.manamotor.com/blog/didactic/what-is-encoder
[310] https://www.encoder.com/article-what-is-an-encoder
[311] https://academy.partineh.com/page/341/%D8%A7%D9%86%DA%A9%D9%88%D8%AF%D8%B1-%D8%A7%D9%81%D8%B2%D8%A7%DB%8C%D8%B4%DB%8C-%DA%86%DB%8C%D8%B3%D8%AA-%D9%88-%D8%B7%D8%B1%D8%B2-%DA%A9%D8%A7%D8%B1-%D8%A2%D9%86-%DA%86%DA%AF%D9%88%D9%86%D9%87-%D8%A7%D8%B3%D8%AA%D8%9.
[312] https://wiki.redronic.com/encyclopedia/robotic-and-mechatronics/ultrasonic/
[313] https://www.theengineeringprojects.com/2020/12/myrio-ultrasonic-sensor-interfacing.html
[314] https://pulsedu.ir/%D8%B3%D9%86%D8%B3%D9%88%D8%B1%D9%87%D8%A7%DB%8C-%D8%A2%D9%84%D8%AA%D8%B1%D8%A7%D8%B3%D9%88%D9%86%DB%8C%DA%A9/
[315] https://mosalasezard.com/%D8%B3%D9%86%D8%B3%D9%88%D8%B1-%D8%A7%D9%88%D9%84%D8%AA%D8%B1%D8%A7%D8%B3%D9%88%D9%86%DB%8C%DA%A9/
[316] https://ostovaee.ir/fa/%d8%b9%d9%85%d9%84%db%8c%d8%a7%d8%aa-%d9%85%d9%88%d8%b1%d9%81%d9%88%d9%84%d9%88%da%98%db%8c%da%a9-%da%af%d8%b3%d8%aa%d8%b1%d8%b4
[317] https://robu.in/product/towerpro-sg90-9gm-1-2kg-180-degree-rotation-servo-motor-good-quality/.
[318] https://www.indiamart.com/proddetail/ac-servo-motor-17092280212.html
[319] https://www.kollmorgen.com/en-us/products/motors/servo/servo-motors/
[320] https://en.wikipedia.org/wiki/Servomotor
[321] https://digispark.ir/%D8%B4%DB%8C%D9%84%D8%AF%D8%AF%D8%B1%D8%A7%DB%8C%D9%88%D8%B1%D9%85%D9%88%D8%AA%D9%88%D8%B1-l293d-arduino-servo-motor
[322] https://www.arduino.cc/en/guide/introduction
[323] https://learn.sparkfun.com/tutorials/what-is-an-arduino/all
[324] https://store.arduino.cc/usa/arduino-uno-rev3
[325] https://www.arduino.cc/en/software
[326] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[327] Mazlan, N. N. B. M., Thamrin, N. M., & Razak, N. A. (2020, June). Comparison Between Ziegler-Nichols and AMIGO Tuning Techniques in Automated Steering Control System for Autonomous Vehicle. In *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)* (pp. 7-12). IEEE.