



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده مهندسی برق

پروژه کارشناسی  
گرایش مخابرات

ماشین های خودران -  
با استفاده از یادگیری تقویتی

نگارش  
محمد رضیئی فیجانی

استادان راهنما  
دکتر وحید پوراحمدی و دکتر حمیدرضا امین‌داور

شهریور ۱۳۹۸

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

# صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

## نکات مهم:

- نگارش پایان نامه/رساله باید به **زبان فارسی** و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت **پشت و رو(دورو)** بلامانع است و انجام آن توصیه می شود.



دانشگاه صنعتی امیرکبیر  
(پلی‌تکنیک تهران)

به نام خدا

تاریخ: شهریور ۱۳۹۸

## تعهدنامه اصالت اثر

اینجانب محمد رضیئی فیجانی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی استاد دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مأخذ ذکر گردیده است. این پایان‌نامه قبلًا برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مأخذ بلامانع است.

محمد رضیئی فیجانی

امضا

این پایان نامه را به پدر، مادر و برادرم که گذشتا این پژوهه به اعتمام بر سر تقدیم  
می‌کنم. همچنین امیدوارم این پژوهه گامی نخست برای گام‌های فراتر باشد.

# پاسکزاری

از دکتر پوراحمدی و دکتر امین داور که کمک های فراوانی جهت پیش برد این هدف بزرگ داشته‌اند،  
کمال تشکر را دارم. همچنین از سایر دوستانی که من را در این پروژه همراهی و یاری دادند، بسیار  
متشکر هستم.

محمد رضیئی فیجانی  
شهریور ۱۳۹۸

## چکیده

خودران کردن خودرو و ساختن خودرو های هوشمند یکی از اهداف بزرگی است که محققان فراوانی در دنیا بر روی این موضوع کار می‌کنند و شرکت های بزرگی نیز از آن ها حمایت می‌کنند.

امروزه الگوریتم های یادگیری ماشین در بسیاری از علوم مهندسی و غیر مهندسی مانند کامپیوتر، مخابرات، کنترل، اقتصاد، پردازش تصویر، پردازش صوت و سیگنال، مهندسی پزشکی، علوم اعصاب، خودرو سازی و ... کاربرد فراوان دارد. از این رو محققان این حوزه در تلاش هستند تا آن روش ها را بهبود بخشنند و یا آنها را در علوم یاد شده، پیاده سازند.

یکی از شاخه های یادگیری ماشین، یادگیری تقویتی است. در این شاخه سعی شده است تا یادگیرنده، همان گونه ای یاد بگیرد که انسان یاد می‌گیرد. این روش با الگوهای با ناظر و یا بدون ناظر متفاوت است و چیزی به عنوان ناظر وجود ندارد و صرفا امتیاز ها هستند که تعیین کننده و راهنمای هستند.

ترکیب این دو حوزه یعنی پیاده سازی خودرو های خودران با استفاده از الگوریتم های یادگیری تقویتی، بسیار جالب خواهد بود. فرضیه ای در این حوزه یادگیری تقویتی به نام فرضیه امیتاز ها وجود دارد که بیان می کند هر رخدادی را می توان با بیشینه کردن امید امتیاز تجمعی، توصیف کرد. بنابر این ادعا در این پژوهه سعی شد با تعریف مناسب امتیاز ها و سایر پارامتر ها به خودرو آموخت تا خودران شود. به صورت کلی این پژوهه دارای دولایه می باشد؛ لایه الگوریتم و لایه شبیه ساز. لایه الگوریتم با استفاده از پایتون و لایه شبیه ساز با استفاده از نرم افزار پری اسکن در محیط سیمولینک انجام شده است. نوشتمن محیط شبیه ساز از جمله دستاوردهای بزرگ این پژوهه می باشد.

## واژه های کلیدی:

کلیدواژه اول، ...، کلیدواژه پنجم (نوشتمن سه تا پنجم واژه کلیدی ضروری است)

# فهرست مطالب

صفحه

عنوان

۱	۱	۱ مقدمه
۵	۲	۲ مرور پیشینه پروژه
۶	۱-۲	۱-۲ مقدمه‌ای در مورد ماشین‌های خودران
۶	۱-۱-۲	۱-۱-۲ اتومبیل‌های خودران چگونه حرکت می‌کنند؟
۶	۲-۱-۲	۲-۱-۲ اتومبیل‌های خودران در یک نگاه
۷	۳-۱-۲	۳-۱-۲ تعیین موقعیت و تولید تصویر سه بعدی
۸	۴-۱-۲	۴-۱-۲ اجتناب از موانع
۹	۵-۱-۲	۵-۱-۲ تعیین مسیر
۱۰	۲-۲	۲-۲ بررسی مقدماتی یادگیری تقویتی
۱۰	۱-۲-۲	۱-۲-۲ جایگاه یادگیری تقویتی در یادگیری ماشین
۱۰	۲-۲-۲	۲-۲-۲ وجه تمایز یادگیری تقویتی از دیگر الگوهای یادگیری ماشین
۱۲	۳-۲-۲	۳-۲-۲ عامل و محیط
۱۳	۴-۲-۲	۴-۲-۲ حالت
۱۴	۵-۲-۲	۵-۲-۲ مشاهده پذیری
۱۵	۶-۲-۲	۶-۲-۲ سیاست
۱۶	۳-۲	۳-۲ مطالعه بیشتر
۱۶	۴-۲	۴-۲ پیشنایاز‌های نصب و معرفی قسمت‌های مختلف
۱۶	۱-۴-۲	۱-۴-۲ نرمافزارهای کلی
۱۷	۲-۴-۲	۲-۴-۲ پیشنایاز‌های پایتون
۱۸	۳-۴-۲	۳-۴-۲ معرفی دقیق‌تر اجزای کلی
۱۸	۴-۴-۲	۴-۴-۲ معرفی نرمافزار پری‌اسکن
۲۰	۵-۴-۲	۵-۴-۲ فرمت‌های فایل‌های خروجی
۲۰	۶-۴-۲	۶-۴-۲ نصب موتور متلب
۲۱	۵-۲	۵-۲ معرفی دقیق‌تر پیشنایاز‌های پایتون
۲۱	۱-۵-۲	۱-۵-۲ بسته‌های کمکی

۲۱	.....	۲-۵-۲ بسته gym
۲۳	.....	۳-۵-۲ stable-baseline
۲۵	.....	۳ تعریف مسیله و بررسی الگوریتم
۲۶	.....	۱-۳ معرفی محیط شبیه سازی
۲۸	.....	۲-۳ معرفی رابط برنامه نویسی برنامه و الگوریتم
۳۲	.....	۳-۳ تعریف کردن پارامتر های یادگیری تقویتی
۳۳	.....	۱-۳-۳ معرفی برخی توابع رابط برنامه نویسی برنامه
۳۹	.....	۲-۳-۳ بررسی تابع <code>_next_observation</code>
۴۴	.....	۳-۳-۳ بررسی مقدار <code>γ</code>
۴۶	.....	۴ نحوه پیاده سازی
۴۷	.....	۱-۴ روش پیاده سازی
۴۸	.....	۲-۴ بررسی دقیق فایل سیمولینک
۵۰	.....	۱-۲-۴ معرفی بلوك Environment و بررسی جزیيات آن
۵۴	.....	۲-۲-۴ بررسی ساختار داده های ارسالی و کد آن در سیمولینک
۵۶	.....	۳-۴ بررسی جزیيات بخش پایتون
۵۶	.....	۱-۳-۴ معرفی لایه های کد پایتون
۵۸	.....	۴-۴ بررسی دقیق تر برخی چالش های فنی پروژه
۶۰	.....	۵ شبیه سازی و نتایج
۶۱	.....	۱-۵ راهاندازی
۶۳	.....	۲-۵ نتایج شبیه سازی
۶۶	.....	منابع و مراجع
۶۷	.....	نمایه
۶۸	.....	فهرست اختصارات
۶۹	.....	واژه نامه انگلیسی به فارسی

۷۱ . . . . . واژه نامه فارسی به انگلیسی

# فهرست اشکال

شکل

صفحه

۱-۱	بلوک دیالگرام لایه های کلی	۳
۱-۲	جایگاه یادگیری تقویتی	۱۱
۲-۲	تعامل عامل <sup>۱</sup> و محیط	۱۲
۳-۲	شماتیک تعامل محیط با عامل	۱۳
۴-۲	تقسیم بندی وظایف اصلی کد پایتون	۱۷
۵-۲	آیکون های اضافه شده بر روی محیط دسکتاپ پس از نصب پری اسکن	۱۸
۶-۲	پنل مدیریت نرم افزار پری اسکن	۱۹
۷-۲	صفحه گرافیکی محیط پری اسکن	۱۹
۱-۳	محیط شبیه سازی	۲۶
۲-۳	مدل های موجود در محیط شبیه سازی	۲۷
۳-۳	بررسی دقیق محدوده و مشخصات جاده	۲۷
۴-۳	بررسی تابع action_translate	۳۵
۵-۳	بررسی دقیق تابع افزایش دهنده و کاهنده سرعت ماشین	۳۷
۶-۳	منطق محاسبه done در محیط شبیه سازی	۳۸
۷-۳	نحوه تعریف مشاهده	۴۰
۸-۳	افراز محیط اطراف ماشین بر حسب زاویه	۴۰
۹-۳	نمودار تابع محاسبه امتیاز سرعت نرمال شده	۴۲
۱۰-۳	تصویر نمونه ای از کنار هم شدن دو ماشین از نزدیکی یک دیگر	۴۴
۱۱-۳	تابع امتیاز نزدیکی و محل قرار گیری این تابع در فضای مشاهده <sup>۳</sup>	۴۵
۱-۴	بلوک دیالگرام لایه های کلی	۴۷
۲-۴	فایل سیمولینک ایجاد شده توسط نرم افزار پری اسکن همراه با تغییرات	۴۹
۳-۴	روش صحیح اعمال تغییرات روی فایل سیمولینک	۴۹

<sup>1</sup>Agent

<sup>2</sup>Environment

<sup>3</sup>Observation

۴-۴	فایل سیمولینک - شبیه سازی اتومبیل	۵۰
۴-۵	نگاهی از نزدیک به بلوک Environmnet	۵۲
۴-۶	فایل سیمولینک - داخل بلوک Environment	۵۲
۴-۷	منطق محاسبه تمام شدن و دستور ریست کردن	۵۴
۴-۸	ساختار جیسون برای بلوک های سیمولینک	۵۵
۴-۹	بلوک دیالگرام لایه های پایتون	۵۶
۱-۵	نمودار توصیف رفتار عامل در هر اپیزود <sup>۴</sup>	۶۴
۲-۵	تصویر شبیه سازی نهایی	۶۵

---

<sup>4</sup>Episode

## فهرست جداول

صفحه

جدول

۱-۲ توضیحات فرمت فایل خروجی . . . . .	۲۰
۲-۲ معرفی بسته های کمکی پایتون و علت استفاده از آنها . . . . .	۲۲
۳-۱ بررسی پارامتر های موجود در env_dict . . . . .	۳۰
۳-۲ راهنمای توابع اصلی رابط برنامه نویسی برنامه . . . . .	۳۲
۳-۳ راهنمای توابع کمکی رابط برنامه نویسی برنامه . . . . .	۳۴
۴-۳ تعریف فضای مشاهده و فضای حرکت در پروژه . . . . .	۳۴
۴-۱ بررسی ورودی ها و خروجی های مهم در شکل ۴-۴ . . . . .	۵۱
۴-۲ اطلاعات بلوك های فرستندگی گیرندگی در سیمولینک . . . . .	۵۳
۴-۱ اطلاعات مخزن های پروژه در گیت هاب . . . . .	۶۱

## فهرست نمادها

نماد	مفهوم
$\mathcal{R}_t$	امتیاز آنی در لحظه $t$
$\mathcal{S}_t$	حالت در لحظه $t$
$\mathcal{S}_t^a$	حالت عامل در لحظه $t$
$\mathcal{S}_t^e$	حالت محیط در لحظه $t$
$\mathcal{O}_t$	مشاهده در لحظه $t$
$\mathcal{A}_t$	حرکت در لحظه $t$
$\ \mathcal{O}\ $	اندازه فضای مشاهده
$\ \mathcal{A}\ $	اندازه فضای حرکت
$\mathcal{H}_t$	تاریخچه تا لحظه $t$
$\mathcal{G}_t$	تابع تجمعی امتیاز، تابع بازگشت
$\gamma$	ضریب کاهنده
$F^{\mathcal{C}} _a$	تابع مرتبط با مفهوم $\mathcal{C}$ با پارامتر $a$
$\pi$	سیاست
$\pi^*$	سیاست بهینه

## مقدمه

# فصل اول

یادگیری تقویتی یکی از گرایش‌های یادگیری ماشینی است که از روانشناسی رفتارگرایی الهام می‌گیرد. این روش بر رفتارهایی تمرکز دارد که ماشین باید برای بیشینه کردن پاداشش انجام دهد. این مسئله، با توجه به گستردگی اش، در زمینه‌های گوناگونی بررسی می‌شود. مانند: (آ) نظریه بازی‌ها، (ب) نظریه کنترل، (ج) تحقیق در عملیات، (د) نظریه اطلاعات، (ه) سامانه چندعامله، (و) هوش ازدحامی، (ز) آمار، (ح) الگوریتم ژنتیک، (ط) بهینه‌سازی بر مبنای شبیه‌سازی.

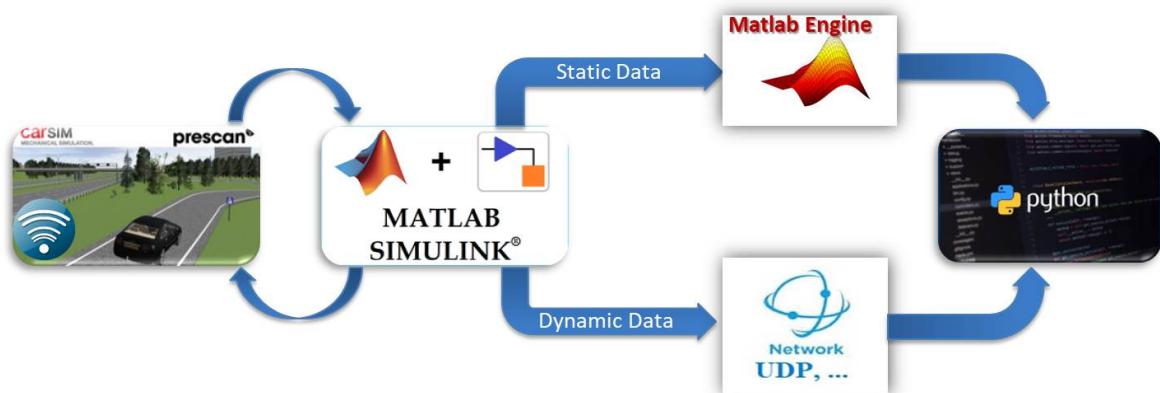
حوزه خودران سازی خودرو ها در سال های اخیر طرفداران زیادی پیدا کرده است. شرکت های بسیاری در دنیا مانند تسلا، رولزرویس، دیپ مایند، گوگل و اخیرا اپل، در حوزه حمایت‌های زیادی از این گونه فعالیت‌ها داشته‌اند. الگوریتم‌های یادگیری تقویتی از آن جهت در این بین محبوب است که یک ماشین سعی می‌کند مانند یک کودک، فعالیت کند و تجربه کسب کند و بیاموزد و حرکت کند. در واقع، قوانین به صورت امتیاز و یا پاداش به این سیستم وارد می‌شوند بدون آن که به ناظر نیاز شود.

این الگوریتم‌ها در دسته سوم شاخه‌های یادگیری ماشین، یعنی یادگیری تقویتی در کنار دو شاخه دیگر یعنی یادگیری با ناظر و یادگیری بدون ناظر مطرح می‌شوند. بر خلاف بخش‌های دیگر این بخش ریاضیات قوی و کامل تری را شامل می‌شود. فرضیه‌ای در این حوزه به نام فرضیه امتیاز‌ها مطرح است که می‌گوید «همه اهداف را می‌توان با بیشینه کردن امید امتیاز‌های تجمعی توصیف کرد.» در این پژوهش سعی شد با استفاده از تعریف مناسب «مشاهده» و «امتیاز»‌ها و همچنین تعیین دیگر پارامترهای الگوریتم، به خودروی مورد مطالعه یاد بدهد که خودران شود. سرانجام این خودرو به صورت شبیه‌سازی شده شروع به حرکت کرد و مسیر را یاد گرفت.

به صورت کلی، این پژوهه دو لایه کلی دارد؛ لایه الگوریتم و لایه شبیه‌ساز. لایه الگوریتم با استفاده از پایتون نوشته شده است اما لایه شبیه‌ساز با بکار گیری نرم افزار پری‌اسکن در محیط سیمولینک مطلب این عمل صورت گرفته است. در حقیقت نوشتن یک محیط شبیه‌سازی (با قابلیت توسعه ساده در آینده) از دستاوردهای مهم و طاقت فرسای این پژوهه می‌باشد.

همان‌طور که گفته شد، در این پژوهه از ابزارهای مختلفی استفاده شده است. برخی ابزارات دیگر نیز جهت ایجاد اتصال بین آن ابزارها استفاده شده‌اند. در این بخش، این اجزا به تفصیل بررسی خواهد شد.

هر کدام از این اجزا کار مشخصی را بر عهده دارند. شکل ۱-۱ این ارتباط را نشان می‌دهد. در شکل ۱-۱ از سمت چپ به راست اجزا یاد شده و نحوه ارتباط آن‌ها با یکدیگر را به خوبی نشان می‌دهد. این بلاک‌ها و ارتباط‌ها عبارتند از:



شکل ۱-۱: بلوک دیالگرام لایه های کلی

- اولین بلاک آن، نرم افزار پریاسکن میباشد. وظیفه اصلی این نرم افزار، شبیه سازی دینامیک یک اتومبیل و یا موتور و ... میباشد. همچنین ایجاد یک محیط گرافیکی زیبا و یک پنل کاربری گرافیکی برای ساخت ماشین ها از دیگر حسن های این نرم افزار است.

فایل های مهم ایجاد شده توسط این بخش، .pex و .pb میباشد.

- بلاک بعدی ترکیبی از متلب و سیمولینک است. چرا که نرم افزار پریاسکن این امکان را دارد که برای کنترل و دسترسی بیشتر به قسمت های کنترلی مختلف، چیزی به نام API ارائه میدهد. این API یک فایل سیمولینک را در اختیار کابران قرار میدهد که در آن بلوک های مشخصی به یکدیگر متصل هستند و با مطالعه و تغییر آن بلوک ها میتوان کنترل سیستم را به دست گرفت. فایل های مهم این بخش نیز در فرمت .slx و .m در دسترس هستند.

همچنین رابط برنامه نویسی برنامه<sup>۱</sup> یاد شده، دستورات دیگری را جهت دریافت داده های استانیک محیط ساخته شده در این نرم افزار را به کابران خویش در محیط متلب می دهد.

- دو بلوک بعدی، مربوط به اتصال بین متلب و یا سیمولینک با پایتون هستند. بلوک بالایی این اتصال را بین داده های استانیک شامل طول جاده و عرض هر لاین، موقعیت اولیه اتومبیل و جاده، و بسیاری اطلاعات دیگر که بسیاری از آن اطلاعات استفاده نشده اند زیرا در این پروژه مفید نبوده اند. این بلوک، فایل سیمولینک را تغییر نمی دهد.

بلوک پایینی نیز با استفاده از روش های شبکه کردن، میتواند داده های پویا را از محیط سیمولینک

<sup>۱</sup>API

به پایتون منتقل کند. این داده های پویا عبارتند از موقعیت و سرعت و اطلاعات دیگری از اتومبیل در حال حرکت، اطلاعات سنسورها و ... باشد.

- بلوک بعدی پایتون است که خود شامل لایه های دیگری است که در شکل ۹-۴ به تفضیل بیان شده است. نکته جالب در آن این است که در آن لایه ها اثری نیز از دو بلوک پیشین آمده است. همچنین بخش اصلی کار، یا به عبارتی مغز و هوش این کار در این قسمت توسعه یافته است.

در فصل ۲ توضیح بسیار مختصری در مورد خود مفاهیم یادگیری تقویتی می شود. فصل ۵ راه اندازی کد و نتایج حاصل این پروژه را نشان می دهد. پیش از راه اندازی باید باتوجه به فصل ۴-۲، پیشنباز های نصب آن تهیه و نصب گردند. همچنین آن فصل توضیح مختصری در مورد چیستی هریک از آن پیشنباز ها ارایه کرده است. در فصل ۳، نحوه تعریف پارامترهای الگوریتم یادگیری تقویتی به صورت کامل بسط داده شده اند. فصل ۴، جزئیات پیاده سازی محیط شبیه سازی را نشان می دهد و بر روی جزئیات فنی آن تمرکز دارد.

## فصل دوم

### مرور پیشینه پروژه

## ۱-۲ مقدمه‌ای در مورد ماشین‌های خودران

### ۱-۱-۲ اتومبیل‌های خودران چگونه حرکت می‌کنند؟

ایلان ماسک، موسس و مدیرعامل شرکت خودروسازی تsla و عده داده که اتومبیل‌های خودران این شرکت از سال ۲۰۲۰ میلادی وارد بازار می‌شوند، شرکت ژاپنی نیسان اعلام کرده که اتومبیل‌های نیمه خودران این شرکت تا سال ۲۰۲۰ آماده فروش می‌شوند و گوگل نیز چند سالی است که سرگرم آزمایش خودروی بدون راننده خود است.

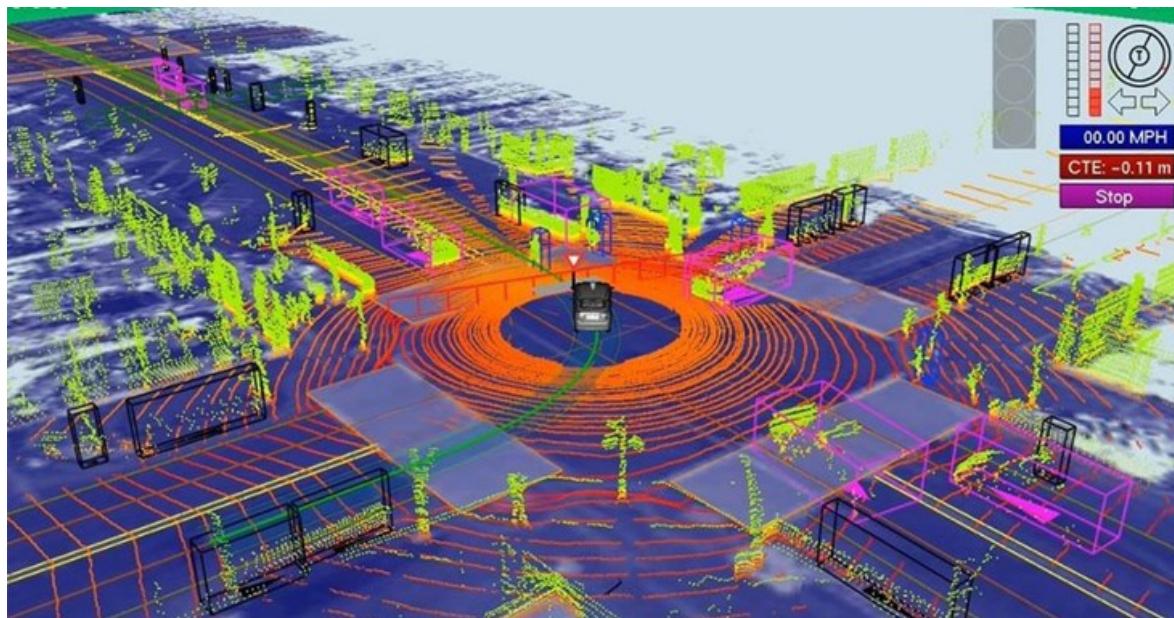
ایده رانندگی خودکار و بدون نیاز به راننده شاید چندان جدید نباشد، چرا که از سال‌ها پیش با آن در فیلم‌های علمی تخیلی آشنا شده‌ایم. اما هدفی که شرکت‌های خودروسازی امروزه دنبال می‌کنند، تحقق بخشیدن به این رویا است. اما از آنجایی که این فناوری هنوز در مراحل ابتدایی خود است و کمتر کسی را پیدا می‌کنید که از طرز کار خودروهای خودران آگاهی داشته باشد.

### ۲-۱-۲ اتومبیل‌های خودران در یک نگاه

اتومبیل‌های خودران قادر هستند تا بدون دخالت عامل انسانی، رویدادهای محیط پیرامون خود را حس کنند و به حرکت درآیند. به این منظور، اتومبیل‌های خودران معمولاً دارای یک تراشه موقعیت یابی یا GPS، یک سیستم ناوبری اینرسی و مجموعه‌ای از حسگرهای هستند که از میان آنها می‌توان به حسگرهای لیزری فاصله یاب، راداری و ویدیویی اشاره کرد. خودرو در واقع از اطلاعات موقعیتی فراهم شده از سوی ماهواره‌های GPS و سیستم ناوبری اینرسی برای تشخیص موقعیت خود، و از اطلاعات حسگرها برای شناسایی جهت و جزئیات محیط اطرافش استفاده می‌کند و در نهایت تصویری سه بعدی از محل کنونی خود و محیط پیرامون خود تهیه می‌کند.

طی این فرایند، اطلاعات بدست آمده از هر حسگر فیلتر می‌شود تا نویزهای آن حذف شود و سپس با یکدیگر ترکیب می‌شوند تا تصویر نهایی تولید شود. حال نوبت به سیستم کنترل خودرو می‌رسد تا براساس این تصویر، نسبت به تعیین مسیر و سایر موارد مرتبط با هدایت خودرو تصمیم گیری کند.

اکثر سیستم‌های کنترلی اتومبیل‌های خودران دارای "معماری مشورتی" هستند، به این معنا که طی دو مرحله دست به تصمیم گیری می‌زنند: ابتدا تولید و بروزرسانی نقشه‌ای داخلی از موقعیت خود و محیط اطراف، و سپس استفاده از این نقشه به منظور پیدا کردن بهترین راه و با حداقل موانع (مانند اجتناب از مسیرهایی که مسدود هستند، دارای ترافیک سنگین هستند و یا عابرین پیاده از آن عبور می-



کنند) از میان راه های موجود برای رسیدن به مقصد.

پس از تعیین بهترین مسیر برای رسیدن به مقصد، تصمیمی که توسط مرکز کنترل خودرو اتخاذ شده به چند دستور جانبی تقسیم شده و هر یک از این دستورات به یکی از عملگرهای مکانیکی خودرو ارسال می شوند. این عملگرهای مکانیکی انجام وظایفی از قبیل گاز دادن، ترمز کردن و تعویض دنده را بر عهده دارند.

فرایند تعیین موقعیت، تولید تصویر سه بعدی، اجتناب از موانع و انتخاب بهترین مسیر طی هر ثانیه بارها توسط پردازنده های قدرتمند کار گذاشته در داخل خودرو تکرار می شود تا اینکه خودرو به مقصد خود برسد.

اگرچه هر یک از شرکت های خودروساز از الگوریتم ها و حسگرهای مختلفی استفاده می کنند، اما منطق بکار رفته در تمام آنها به یک شکل است.

حال به بررسی دقیق تر ابعاد تکنیکی هر یک از فرایندهای بکار رفته در هدایت اتومبیل های خودران می پردازیم.

### ۳-۱-۲ تعیین موقعیت و تولید تصویر سه بعدی

پیش از اتخاذ هرگونه تصمیمی برای تعیین مسیر، خودرو باید ابتدا نقشه ای از محیط اطراف خود بسازد و محل خود در این نقشه را با دقت مشخص کند. رایج ترین ابزارهایی که به خودروها در انجام این کار کمک می کنند، دوربین ها و حسگرهای لیزری فاصله یاب هستند. حسگر لیزری با تاباندن پرتوهای

لیزر به محیط اطراف و تخمین زمان رفت و برگشت هر پرتو، فاصله خود با اشیاء اطرافش را تعیین می کند. مزیت این روش در مقایسه با استفاده از دوربین ها این است که اطلاعات موردنیاز خودرو سریع تر بدست می آید.

اما این حسگرها خالی از عیب نیز نیستند. برد مفید پرتو لیزر صد متر است و اگر از آن برای ارزیابی فاصله اشیاء در فاصله ای دورتر از صدمتر استفاده شود، کارایی چندانی ندارند. پس از دریافت اطلاعات از هر حسگر، نوبت به فیلترسازی آن می رسد. هدف از این کار، ارسال تنها آن دسته از اطلاعاتی است که برای تعیین مسیر اهمیت دارند و به این ترتیب، از ارسال اطلاعات اضافی به پردازشگر جلوگیری می شود که در نهایت، سرعت عمل و بهره وری پردازشگر را افزایش می دهد. در نهایت، این اطلاعات کنار هم گذاشته می شود تا نقشه ای جامع از محیط پیرامونی شکل بگیرد. حال این نقشه آماده استفاده است تا با کمک آن، بهترین مسیر برای رسیدن خودرو به مقصد نهایی اش تعیین شود.

برای اینکه خودرو بتواند موقعیت اش را به نسبت سایر اشیاء در این نقشه پیدا کند، باید از GPS، سیستم ناوبری اینرسی و حسگرهای مختلف استفاده کند. انکای صرف به GPS قابل اطمینان نیست، چرا که احتمال بروز تاخیر در ارسال و دریافت این سیگنال ها بسیار زیاد است. همچنین، برخورد این سیگنال ها با موانع و ساختمان ها نیز موجب کاهش دقت آنها می شود. میزان خطاهای سیستم ناوبری اینرسی نیز در طول زمان زیاد است و به همین دلیل، الگوریتم های مسیریابی اطلاعات موردنیاز خود برای تصمیم گیری را از منابع مختلفی دریافت می کنند. طی هر لحظه ای که خودرو حرکت می کند، اطلاعات جدیدی تولید می شود و در نتیجه، نقشه بروزرسانی می شود.

## ۴-۱-۲ اجتناب از موانع

نقشه درونی هر خودرو شامل موقعیت کنونی تمام موانع ثابت (مانند ساختمان ها، چراغ های راهنمایی و رانندگی، نشانه های توقف) و متحرک (سایر خودروها و عابرین پیاده) اطراف آن می شود. تعیین نوع مانع، ثابت یا متحرک، توسط الگوریتمی تعیین می شود که سعی می کنند تصویر هر مانع را با مجموعه ای از تصاویری که از قبل در آن بارگذاری شده مقایسه کرده و نزدیک ترین شکل به آن را تشخیص دهد. برای مثال، اگر وسیله نقلیه ای با دو چرخ و با سرعت هشتاد کیلومتر در حال حرکت در نزدیکی خودرو باشد، به احتمال زیاد موتور سیکلت است و دوچرخه نیست، و بر همین اساس توسط الگوریتم خودرو دسته بندی می شود.

همچنین، این الگوریتم تلاش می کند تا با اتکا به یک الگوریتم احتمال گرا، مسیر احتمالی اشیاء

متحرک را پیش بینی کند. به این ترتیب، خودرو امکان اتخاذ تصمیمات هوشمندانه تری را به هنگام نزدیکی به مناطق پرتردد را پیدا می کند. تمام موقعیت ها و مسیرهای احتمالی پیش بینی شده در اختیار مرکز پردازش قرار می گیرند تا در بروزرسانی های لحظه ای نقشه، اعمال شوند.

## ۵-۱-۲ تعیین مسیر

هدف از فرایند تعیین مسیر، استفاده از اطلاعات بدست آمده از نقشه خودرو به منظور رسیدن به مقصد با سلامت کامل و بدون مواجهه با موانع جاده و همچنین پیروی از قوانین راهنمایی و رانندگی است. اگرچه ممکن است راه و روشی که هر خودروساز برای اتومبیل خودران خود انتخاب می کند کم و بیش متفاوت باشد، اما منطق بکار رفته در همه آنها یکسان است و در ادامه به شرح آن می پردازیم.

در ابتدا یک راه اصلی و کامل برای خودرو تعیین می شود تا با دنبال کردن آن به مقصد برسد. در عین حال، هر لحظه تغییراتی در مسیرهای مقطعی ایجاد می شود تا عیوب آن برطرف شده و بهترین مسیر بدست آید. این الگوریتم همچنین تلاش می کند تا بهترین مسیر را براساس حرکت خودروهای اطراف پیدا کند. فرض کنید خودرویی در فاصله پنج متری خودروی شما در حال حرکت است و شما قصد دارید تا در اولین خروجی به سمت راست بپیچید. در چنین شرایطی اگر با همین سرعت ادامه مسیر دهید قطعاً امکان دسترسی به خروجی را نخواهید داشت چرا که مسیر شما مسدود است. پس خودرو تصمیم می گیرد تا از سرعت خود کم کند تا خروجی را از دست ندهد. به همین منظور، دستوراتی به تمام عملگرهای مکانیکی داده می شود تا سرعت خودرو کم شده و پس از عبور خودروی جانبی، راه برای رسیدن به خروجی باز شود. جالب است بدانید که تمام این محاسبات طی زمانی کمتر از ۵۰ میلی ثانیه انجام می شود. البته این مدت زمانی به طور متوسط است و ممکن است بنابر شرایط مختلف، این زمان کوتاه تر یا بلندتر نیز بشود. قدرت پردازشگر، حجم اطلاعات دریافت شده و پیچیدگی مسیر پیش رو نیز بر مدت زمان تصمیم گیری خودرو تاثیر می گذارد.

تمام این فرایندها آنقدر ادامه پیدا می کنند تا خودرو به مقصد برسد و مسافران به سلامت از آن پیاده شوند.

خودروسازان طی یک دهه اخیر در زمینه تحقیق بخشیدن به رویای خودروهای خودران دستاوردهای زیادی داشته اند. با این حال، هنوز هم موانع فنی زیادی در این راه وجود دارد که باید پیش از ورود خودروهای خودران به بازار برطرف شوند. سیگنال های GPS هنوز به طور کامل قابل اطمینان نیستند، سیستم های بصری کامپیوتری با مشکلاتی در شناسایی علائم جاده ای روبرو هستند و شرایط آب و

هوایی نیز تا حد زیادی روی عملکرد پردازشگرهای خودروها و تشخیص و تصمیم گیری صحیح آنها اثرگذاراند.

با این حال، این مواعظ قابل حل شدن هستند. هر روز خبرهای تازه‌ای از طراحی و تولید پردازنده‌های قدرتمندتر و الگوریتم‌های بهینه‌تر به گوشی می‌رسد و به همین نسبت، از میزان تصادفات خودروهای خودران کاسته می‌شود. امروزه خودروهایی هستند که به طور نمیه خودران حرکت می‌کنند و راننده از انجام بسیاری از کارها، از جمله پارک کردن خودرو معاف کرده‌اند. پس باید منتظر بود و دید که طی چند سال آینده، چه دستاوردهایی در صنعت خودروسازی حاصل می‌شود و کدام یک از شرکت‌های خودروساز موفق می‌شود تا نام خود را به عنوان نخستین عرضه کننده خودروی خودران در تاریخ ثبت کند.

## ۲-۲ بررسی مقدماتی یادگیری تقویتی

### ۱-۲-۲ جایگاه یادگیری تقویتی در یادگیری ماشین

بسیاری از صاحب نظران یادگیری ماشین<sup>۱</sup> را به سه دسته تقسیم می‌کنند: (۱) یادگیری با ناظر<sup>۲</sup> (۲) یادگیری بدون ناظر<sup>۳</sup> یا خوش‌بندی<sup>۴</sup> (۳) یادگیری شبیه ناظر<sup>۵</sup> در این میان، یادگیری تقویتی<sup>۶</sup> را بعضی‌ها دسته چهارم می‌دانند و بعضی دیگر آن را در دسته سوم قرار می‌دهند. بر اساس دسته‌بندی گروه دوم شکل ۱-۲(آ)<sup>۷</sup> رسم شده است. همچنین شکل ۱-۲(ب)<sup>۸</sup> کابرد یادگیری تقویتی را در علوم مختلف نشان می‌دهد.

### ۲-۲-۲ چه چیزی یادگیری تقویتی را با دیگر الگوهای یادگیری ماشین متمایز می‌کند؟

این سوال از آن جهت حائز اهمیت است که بیان می‌کند چرا ما به سراغ الگوی یادگیری تقویتی رفته‌ایم. پاسخ ملاحظات زیر است.

<sup>1</sup>Machine Learning

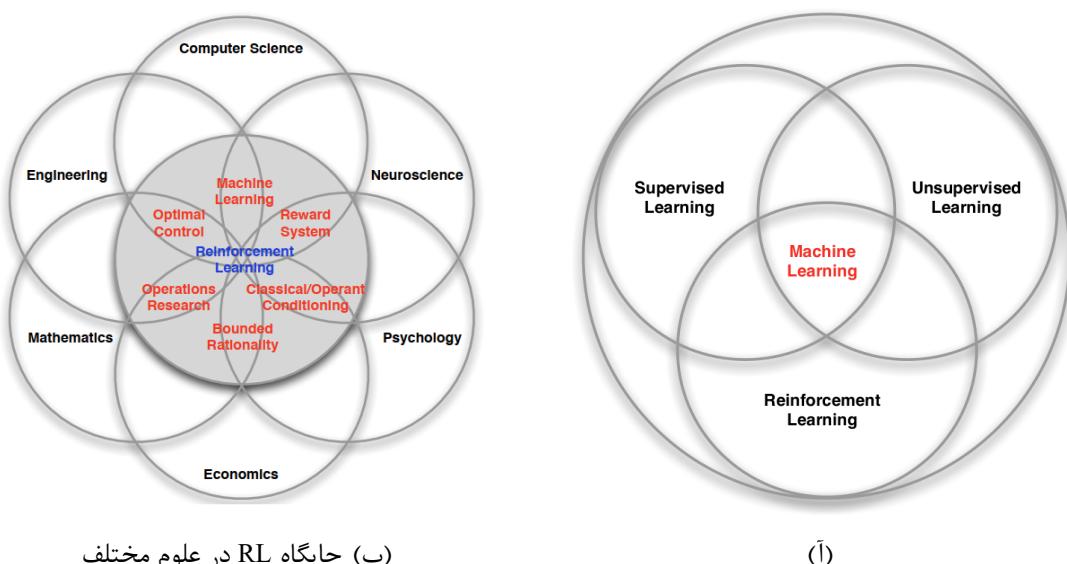
<sup>2</sup>Supervised Learning

<sup>3</sup>Unsupervised Learning

<sup>4</sup>Clustering

<sup>5</sup>Semi-Supervised Learning

<sup>6</sup>Reinforcement Learning



شکل ۱-۲: جایگاه یادگیری تقویتی

آ) هیچ ناظر<sup>۴</sup> وجود ندارد و صرفا امتیاز<sup>۵</sup>ها وجود دارند.

ب) فیدبک<sup>۶</sup> همراه با تاخیر است و به صورت همزمان رخ نمی‌دهد.<sup>۷</sup>

ج) مفهوم زمان واقعا مطرح است و یک ترتیب خاص از داده‌ها داریم. شکل ۳-۲ این توالی زمانی را نشان می‌دهد.

یادگیری تقویتی (RL)<sup>۸</sup> بر اساس فرضیه امتیازها<sup>۹</sup> پایه‌گذاری می‌شود.

تعريف ۱-۲-۲ (فرضیه امتیازها). همه اهداف می‌توانند براساس بیشینه کردن مقدار میانگین تجمعی امتیازها توصیف کرد.

ممکن است این عبارت کمی عجیب بنظر برسد اما در بسیاری از مسایل که به صورت برد و باخت و به نوعی دو حالت مطلوب و نامطلوب دارند، می‌توان در ساده‌ترین حالت مقدار  $+1$  را برای برد و  $-1$  را برای باخت در نظر گرفت.

نکته ۱-۲-۲. در برخی منابع بجائی امتیاز از مفهوم هزینه<sup>۱۰</sup> استفاده می‌کنند و هدف الگوریتم آن

<sup>7</sup>Supervisor

<sup>8</sup>Reward

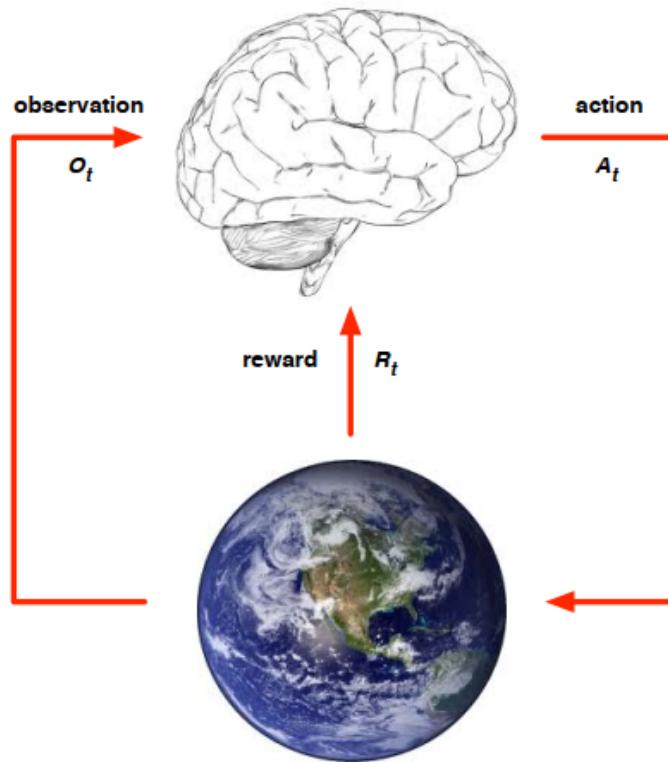
<sup>9</sup>Feedback

<sup>10</sup>در مورد علت تاخیر در ادامه توضیح داده خواهد شد.

<sup>11</sup>Reinforcement Learning

<sup>12</sup>Reward Hypothesis

<sup>13</sup>Cost



شکل ۲-۲: تعامل عامل و محیط

می‌شود که به سمتی حرکت کند که کمترین هزینه را داشته باشد. برای یک پارچه‌سازی این مفاهیم معمولاً یک علامت منفی برای این دو در نظر می‌گیرند یعنی :

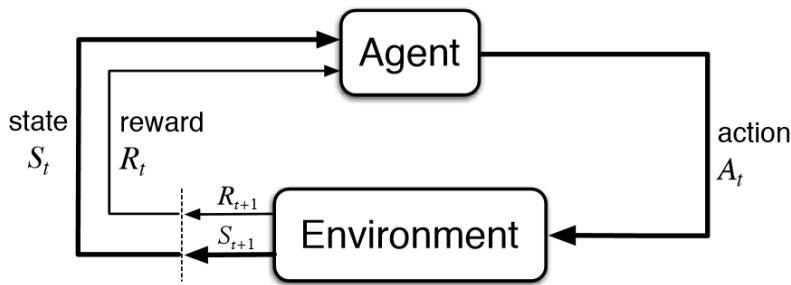
$$\text{هزینه} - \text{امتیاز} = r = -c$$

### ۳-۲-۲ عامل و محیط

این مفهوم بسیار مفهوم مهمی می‌باشد و بارها از آن در این پژوهه یاد شده است. در مسایل یادگیری تقویتی یک عامل وجود دارد که در یک محیط درحال تعامل است. محیط می‌تواند محیط اطراف عامل باشد و یا هر چیزی که عامل با آن در تعامل است. [۱] این تعامل به این صورت است که عامل که در ابتدای یک حالت<sup>۱۴</sup> اولیه دارد، یک حرکت<sup>۱۵</sup> بر روی محیط در زمان  $t$  انجام می‌دهد. محیط مقدار حرکت در زمان  $t$  را دریافت می‌کند و سپس محیط در

<sup>14</sup>State

<sup>15</sup>Action



شکل ۲-۳: شماتیک تعامل محیط با عامل

زمان  $t + 1$  دو اطلاعات مهم را بر می‌گرداند. (آ) مشاهده (ب) امتیاز شکل ۲-۲ و ۳-۲<sup>۱۶</sup> این تعامل را نشان می‌دهد. لازم به ذکر است که در پایان هر مرحله<sup>۱۷</sup> مقدار یک واحد افزایش می‌یابد.

## ۴-۲-۲ حالت

در بخش قبل تعریف مناسبی از حالت ارایه نشد. برای این تعریف ابتدا مفهوم تاریخچه<sup>۱۸</sup> ارایه می‌شود و از روی آن حالت تعریف خواهد شد.

**تعریف ۴-۲-۲** (تاریخچه). به سری شامل مشاهده، حرکت و امتیاز می‌باشد:

$$\mathcal{H}_t = \mathcal{O}_1, \mathcal{R}_1, \mathcal{A}_1, \dots, \mathcal{A}_{t-1}, \mathcal{O}_{t-1}, \mathcal{R}_t$$

با این تعریف حالت را می‌توان به شکل زیر تعریف کرد.

**تعریف ۴-۲-۴.** حالت اطلاعاتی است که در محاسبات برای آن که در بعد چه اتفاقی بیافتد، استفاده می‌شود. به عبارت دیگر حالت تابعی از تاریخچه می‌باشد.

$$S_t = f(\mathcal{H}_t)$$

دو نوع حالت وجود دارد.

<sup>16</sup>Step

<sup>17</sup>History

آ) **حالت محیط**<sup>۱۸</sup> که با علامت  $S_t^e$  نشان داده می‌شود. اطلاعات نهان محیط را نشان می‌دهد و معمولاً برای عامل بهطور کامل دیده نمی‌شود. حتی اگر برای عامل مشاهده‌پذیر نیز باشد، ممکن است اطلاعات کاملاً بی‌ربطی را همراه داشته باشد.

ب) **حالت عامل**<sup>۱۹</sup> که با علامت  $S_t^a$  نشان داده می‌شود. که برابر است با هر اطلاعاتی که عامل برای رسیدن به حرکت بعدی با استفاده از الگوریتم‌های RL استفاده می‌کند.

بنابراین در نعریف ۴-۲-۲ مناسب‌تر است بجای واژه حالت از حالت عامل استفاده شود. بنابراین:

$$\mathcal{S}_t^a = f(\mathcal{H}_t)$$

یادداشت ۲-۲-۵. از این پس در سراسر این پایان‌نامه هرجا صحبت از حالت شد منظور همان حالت عامل است.

تعریف ۲-۲-۶. یک حالت  $s_t$  مارکوف<sup>۲۰</sup> است اگر و تنها اگر:

$$\mathbb{P}[\mathcal{S}_{t+1} | \mathcal{S}_t] = \mathbb{P}[\mathcal{S}_{t+1} | \mathcal{S}_1, \dots, \mathcal{S}_t]$$

در یک حالت مارکوف<sup>۲۱</sup>، آینده از گذشته مستقل است و فقط به زمان حال وابسته است. و این به این معناست که حالت از لحاظ آماری برای توصیف آینده کافی است.

نکته ۲-۲-۷. حالت محیط  $S_t^e$  مارکوف است. همچنین تاریخچه نیز مارکوف است.

## ۵-۲-۲ مشاهده پذیری

### مشاهده پذیری کامل

عامل بهطور مستقیم حالت محیط را مشاهده می‌کند. بنابراین در این حالت داریم:

$$\mathcal{O}_t = \mathcal{S}_t^a = \mathcal{S}_t^e$$

<sup>18</sup>Environment State

<sup>19</sup>Agent State

<sup>20</sup>Markov

<sup>21</sup>Markov State

بنابراین در این حالت عبارت های زیر با یکدیگر برابر هستند.

$$\text{حالت اطلاعاتی} = \text{حالت محیط} = \text{حالت عامل}$$

۲۲ به صورت رسمی، این فرایند یک روند تصمیم‌گیری مارکوف<sup>۲۴</sup> (MDP) می‌باشد. [۲]

### مشاهده پذیری جزئی

عامل به طور غیر مستقیم محیط را مشاهده می‌کند.

نمونه ۲-۸. یک ربات با دید دوربین نمی‌تواند موقعیت مطلق را اعلام کند.

نمونه ۲-۹. یک اتومبیل با سنسور تشخیص فاصله نمی‌تواند اطلاعاتی مانند نوع ماشین و قیمت آن را تشخیص دهد.

### ۶-۲-۲ سیاست

سیاست<sup>۲۵</sup> در حقیقت تابعی احتمالی و یا معین است که تصمیم می‌گیرد که در حالت کنونی چه تصمیمی باید گرفت. در واقع رفتار عامل توسط این تابع، بررسی و نشان داده می‌شود.

تعریف ۲-۱۰. اگر تابع معین باشد این تابع به صورت زیر تعریف می‌شود.

$$a = \pi(s)$$

و اگر تابع احتمالاتی باشد به صورا زیر تعریف می‌شود.

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

۲۲ حالت اطلاعاتی<sup>۲۳</sup> مفهومی مانند حالت مارکوف دارد.

<sup>24</sup>Markov Decision Process

<sup>25</sup>Policy

## ۳-۲ مطالعه بیشتر

جهت مطالعه بیشتر و آشنایی با ادبیات یادگیری تقویتی و همچنین الگوریتم های آن می توانید به مرجع [۱] و [۲] مراجعه کنید.

## ۴-۲ پیشناز های نصب و معرفی قسمت های مختلف

### ۱-۴-۲ نرم افزارهای کلی

در این پژوهه از جهت آنکه نسخه قبلی و پیشینی برای آن نبوده است، به ناچار می بایست که کد آن از صفر تا صد آن به صورت دستی نوشته شود. از این رو، پیچیدگی های بسیار فراوان را به طور خاص در پی داشت. ابزار های زیادی نیز بنابه شرایط در آن استفاده شد که ارتباط بین آن ابزار ها و اجزاء، بر این پیچیدگی پیاده سازی طرح افزووده بود.

ابزار های اصلی و کلی که در این پژوهه استفاده شده بود، عبارتند از:

• نرم افزار پری اسکن<sup>۲۶</sup> ، نسخه 8.5.0

• نرم افزار متلب<sup>۲۷</sup> ، نسخه R2017b

• زبان برنامه نویسی پایتون ، نسخه 3.6.9

بنابراین برای راه اندازی مجدد کد این پژوهه لازم است که موارد بالا روی کامپیوتر شخص به صورت کامل نصب باشد.

همچنین لازم به ذکر است که برخی ابزارات دیگر نیز در این پژوهه استفاده شده است که احتمالاً با نصب موارد بالا دیگر نیازی به نصب آن ها به صورت جداگانه نیست. هدف این ابزار ها ایجاد اتصال بین اجزای اصلی گفته شده است. این گروه شامل موارد زیر هستند:

• سیمولینک<sup>۲۸</sup> ، جهت اتصال بین متلب و پری اسکن

• شبکه UDP<sup>۲۹</sup> ، جهت اتصال داده های پویا<sup>۳۰</sup> بین پایتون و سیمولینک

<sup>26</sup>PreScan

<sup>27</sup>Matlab

<sup>28</sup>Simulink

<sup>29</sup>برای این منظور از مازول socket در پایتون استفاده شده است.

<sup>30</sup>Dynamic Data



شکل ۴-۲: تقسیم بندی وظایف اصلی کد پایتون

#### • موتور متلب<sup>۳۱</sup>، جهت اتصال داده های ساکن<sup>۳۲</sup> بین پایتون و سیمولینک

در این فصل جزئیات بیشتری در مورد لزوم و دلیل استفاده از این ابزارها بررسی می شود.

### ۲-۴-۲ پیشنياز های پایتون

يادداشت ۱-۴-۲. کد پایتون در این پروژه شامل دو قسمت کلی زیر می شود. این دو دسته در شکل ۴-۲ مشخص هستند.

۱. دسته اول مربوط به آن بخش از پروژه است که وظیفه اصلی آن ارتباط پیدا کردن با محیط متلب و پری اسکن و ایجاد یک نوع واسط کاربری است. گرفتن و فرستادن اطلاعات مخصوص این قسمت است.

۲. دسته دوم با محیط و نحوه ارتباط آن کاری ندارد و تمرکز خود را بر روی الگوریتم خود که در اینجا از الگوریتم های یادگیری تقویتی استفاده شده است، قرار داده است.

دسته اول (سمت چپ تصویر ۴-۲) به پکیج های زیر احتیاج دارد:

matlab.engine •	os و time •	numpy •
gym •	pandas •	socket •

اگر از آناکوندا<sup>۳۳</sup> برای پایتون استفاده می کنید غیراز دو بسته gym و matlab.engine به صورت پیشفرض نصب شده اند در صورت عدم نصب آن ها را با استفاده از pip<sup>۳۴</sup> می توان نصب کرد.

<sup>31</sup>Matlab Engine

<sup>32</sup>Static Data

<sup>33</sup>Anaconda

<sup>34</sup>مثلاً بسته numpy را با استفاده از دستور pip install numpy نصب می توان کرد.

بسته gym که در این فصل به تفصیل در مورد آن بحث شده است، به راحتی با همان دستور pip نصب می‌شود. اما نصب matlab.engine یا همان موتور مطلب متفاوت است و نمی‌توان آن را نیز به همان روش نصب کرد.

دسته دوم شامل بسته‌های زیر است:

gym[all] یا gym[atari] •

tensorflow •

stable-baseline •

این بسته‌ها در لایه الگوریتم استفاده شده است.(در مورد این لایه در فصل ۲ بیشتر صحبت خواهد شد). هر سه‌تای این بسته‌ها با همان دستور pip به راحتی نصب می‌شوند.

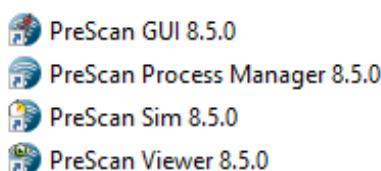
### ۳-۴-۲ معرفی دقیق تر اجزای کلی

در این قسمت میخواهیم سه نرم‌افزار کلی این پروژه را از نگاهی نزدیک تر بشناسیم که عبارتند از :

(۱) نرم‌افزار پری‌اسکن (۲) مطلب (۳) پایتون

### ۴-۴-۲ معرفی نرم‌افزار پری‌اسکن

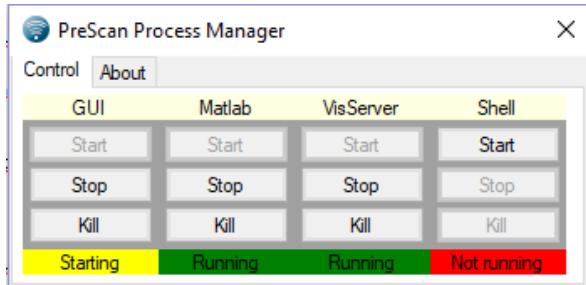
پس از دانلود و نصب نسخه 8.5.0 این نرم‌افزار چهار آیکون مانند شکل ۵-۲ به محیط دسکتاپ اضافه می‌کند. اصلی ترین آن‌ها PreScan Process Manager 8.5.0 نام دارد.



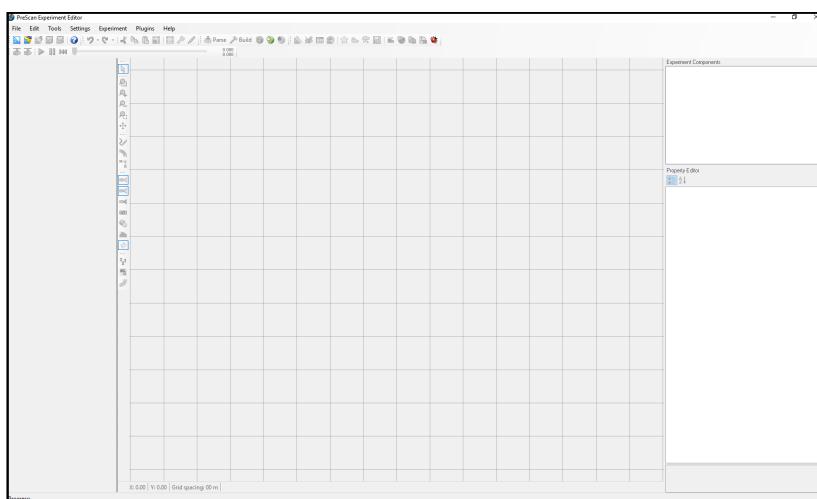
شکل ۲-۵: آیکون‌های اضافه شده بر روی محیط دسکتاپ پس از نصب پری‌اسکن

با انتخاب آن صفحه‌ای مانند زیر باز می‌شود.

این پنجره شامل گزینه‌های زیر است:



شکل ۲-۶: پنل مدربیت نرم افزار پری اسکن



شکل ۲-۷: صفحه گرافیکی محیط پری اسکن

Matlab •

GUI •

Shell •

VisServer •

برای ایجاد یک محیط جدید باید GUI را استارت کرد. پس از مدتی صفحه ای مانند شکل ۷-۲ باز می شود.

پس از ایجاد مدل ها و ذخیره آن، فایل های \*.pex و \*.pb و \*.cs.slx ساخته می شود.<sup>۳۵</sup> جهت استفاده از فایل سیمولینک باید در شکل ۶-۲ مطلب را استارت کنید.

نکته ۴-۲. برای اجرای فایل های سیمولینک خروجی، لازم است که مطلب را فقط و فقط با استفاده از نرم افزار پری اسکن و با استفاده از پنل مدیریت نرم افزار معرفی شده در شکل ۶-۲ باز شود. در صورتی که به صورت مستقیم این کار انجام شود، به مشکل منتهی می شود.

دو قسمت دیگر نیز در شکل ۶-۲ وجود دارد که نیازی به استارت کردن آن ها نیست و خودشان در

<sup>۳۵</sup> علامت \* به معنای یک اسم مشترک در این سه فایل استفاده شده است.

صورت لزوم به صورت خودکار فراخوانی می‌شوند.

## ۵-۴-۲ فرمت های فایل های خروجی

نرم افزار پری اسکن پس از ایجاد یک محیط جدید، فایل ها و پوشه های بسیار زیادی را ایجاد می‌کند. اما در خارج آن پوشه ها ۳ فایل وجود دارد که پسوند آن ها `.pex`, `**.pb` و `**_cs.slx` می‌باشد. علامت `**` همان اسم پروژه‌ای است که ایجاد کرده ایم. هر یک از این فایل ها به یک بلوک از شکل ۱-۴ مربوط می‌شود.

فرمت فایل	توضیحات
<code>**.pex</code>	این فایل مربوط به اولین بلوک شکل ۱-۴ است و ارتباط مستقیم با GUI دارد. برای تغییر محیط گرافیکی باید این فایل را باز کرد.
<code>**.pb</code>	این فایل برخی از اطلاعات فایل <code>pex</code> را در اختیار دارد و با تغییر آن فایل این فایل نیز عوض می‌شود. این فایل حاوی اطلاعات استاتیک محیط ایجاد شده است و مهم ترین کاربرد آن در بلوک موتور مطلب که در شکل ۱-۴ نشان داده شده است می‌باشد. پایتون از طریق این فایل این اطلاعات را دریافت می‌کند.
<code>**_cs.slx</code>	این فایل سیمولینک است که برای کار کردن با آن باید از پنل مدیریت شکل ۶-۲ استفاده کرد. این فایل پس از ایجاد از فایل <code>pex</code> مستقل می‌شود. این فایل خود قابلیت تغییر دارد و می‌توان بلوک‌های آن را در محیط سیمولینک تغییر داد و بلوک‌های دیگری به آن افزود. در صورتی که فایل <code>pex</code> تغییر کند، این امکان را نیز دارد که از داخل خود سیمولینک با فشردن دکمه‌ای این تغییرات جدید اعمال شود بدون آن که به تغییرات خود کاربر لطمه‌ای وارد شود. در این پروژه این فایل، تغییرات بسیاری را تجربه کرد.

جدول ۱-۲: توضیحات فرمت فایل خروجی

جدول ۱-۲ توضیحات لازم را جهت آشنایی با این خروجی‌ها آورده است. همچنین در بخش ۲-۴ در مورد فایل `**_cs.slx` توضیحات دقیق‌تری در مورد جزئیات آن گفته خواهد شد.

## ۶-۴-۲ نصب موتور مطلب

برای نصب موتور مطلب ابتدا نیاز است که به مطلب به طور کامل در سیستم نصب باشد. پس از نصب مطلب، محیط Command prompt (admin) را باز کنید و با توجه به نسخه و محل نصب مطلب خود به آدرس زیر بروید.

```
<matlabroot>\extern\engines\python
```

مثلا برای Matlab R2017b که در محل پیشفرض خود نصب شده باشد این کار با استفاده از دستور زیر انجام می‌شود.

```
cd C:\Program Files\MATLAB\R2017a\extern\engines\python
```

در این پوشه یک فایل به نام setup.py موجود می‌باشد. این فایل را با استفاده از دستور

```
python setup.py install
```

در همان محیط cmd اجرا کنید.

یادداشت ۴-۳. توجه داشته باشید که باید نسخه متلب و پایتون شما باید با یکدیگر سازگار باشند. برای بررسی این موضوع اگر فایل setup.py را با اسنفاده از یک ادیتور باز کنید، یک آرایه به نام \_supported\_versions در آن خواهد دید. مقادیر این آرایه، نسخه هایی از پایتون را نشان می‌دهد که توصیط نسخه متل شما پشتیبانی می‌شود مثلا در این مورد، با توجه به خط زیر نسخه های ۲/۷ ، ۳/۴ و ۳/۵ و ۳/۶ پایتون پشتیبانی می‌شود. در غیر این صورت باید نسخه سازگار متلب و یا پایتون را نصب کنید.

```
_supported_versions = ['2.7', '3.4', '3.5', '3.6']
```

## ۵-۲ معرفی دقیق تر پیشنباز های پایتون

### ۱-۵-۲ بسته‌های کمکی

این بسته‌ها نقش حیاتی ندارند و برای برخی از موارد استفاده شده‌اند. این موارد در جدول ۲-۲ آمده است.

### ۲-۵-۲ بسته gym معرفی

بسته gym که توسط OpenAI توسعه یافته است. این ابزار فوق العاده این امکان را برای محققین علوم کامپیوتر حرفه‌ای و یا آماتور فراهم می‌کند که انواع الگوریتم‌های یادگیری تقویتی (RL) را بر روی کار

نام بسته	دلیل استفاده	روش نصب
numpy	ایجاد ماتریس برای فضای حرکت <sup>۳۶</sup> و فضای مشاهده <sup>۳۷</sup>	pip install numpy
time	جهت ایجاد تأخیر و سقف زمانی <sup>۳۸</sup>	pip install time
os	برای بستن پنجره های باز شده پس از اجرا	pip install os
pandas	برای چاپ اطلاعات آماری امتیاز های بدست آمده در پایان هر اپیزود	pip install pandas
socket	برقراری ارتباط با متلب و فرستن و دریافت کردن داده های پویا	pip install socket
tensorflow	برای لایه الگوریتم و استفاده از الگوریتم های یادگیری تقویتی عمیق <sup>۳۹</sup>	pip install tensorflow

جدول ۲-۲: معرفی بسته های کمکی پایتون و علت استفاده از آنها

خود تست کنند. همچنین پتانسیل این را دارد که محققین محیط خود را ببروی این بسته توسعه دهند. هدف از ایجاد این بسته، استاندارد سازی محیط و نوعی نقطه تراز <sup>۴۰</sup> برای پژوهش های RL محسوب می شود.<sup>[۳]</sup>

در حقیقت می توان این بسته را در وسط شکل ۴-۲ جای داد. جایی که لایه محیط و لایه الگوریتم <sup>۴۱</sup> به یکدیگر می رسانند.

این بسته محیط هایی از پیش ساخته شده دارد. نام این بسته ها در لیست زیر آمده است. <sup>۴۲</sup> اکثر این محیط ها نوعی بازی هستند که عامل سعی در یادگیری آن محیط ها دارد.

Pong-v0 •	CartPole-v0 •
MsPacman-v0 •	Pendulum-v0 •
SpaceInvaders-v0 •	MountainCar-v0 •
Seaquest-v0 •	MountainCarContinuous-v0 •
LunarLanderV2 •	BipedalWalker-v2 •
Reacher-v2 •	Humanoid-V1 •
FrozenLake-v0 •	Riverraid-v0 •
	Breakout-v0 •

<sup>40</sup>Bechmark

<sup>41</sup>Algorithm

<sup>42</sup>جدول کامل در سایت <https://github.com/openai/gym/wiki/Table-of-environments> قرار دارد.

## نصب

برای نصب نسخه کمینه این نرم افزار با همان روش pip به راحتی می‌توان نرم افزار مورد نظر را نصب کرد. [۴] این نسخه کمینه برای لایه محیط کافی می‌باشد. اما اگر بخواهیم بخش الگوریتم را با استفاده از کتابخانه‌های دیگری مانند stable-baseline نوشت. نیازمند نسخه جامع تری از gym می‌باشد.

**یادداشت ۳-۵-۱.** پیشنهاد می‌شود برای نصب نسخه کامل gym و stable-baseline از لینوکس بجای ویندوز استفاده کنید. زیرا در نصب برخی بسته‌ها ممکن است با مشکل رو برو شوید.

برای نصب کامل این بسته از دستور `pip install gym[all]` را استفاده کنید. ممکن است در نصب mujoco به مشکل بخورید در این صورت دستور `pip install gym[atari]` استفاده کنید. اگر موفق به نصب این بسته نشده‌ید می‌توانید مراجل نصب آن را با استفاده از [۳] مراجعه کنید.

## ۳-۵-۲ بسته stable-baseline

این بسته مجموعه‌ای از الگوریتم‌های از پیش تعریف شده در یادگیری تقویتی می‌باشد. در صورتی که محیط در gym登録 شده باشد، می‌توان از آن در پروژه‌های مختلف استفاده کرد. این الگوریتم‌ها عبارتند از:

PPO2 •	DQN <sup>44</sup> •	A2C <sup>43</sup> •
SAC •	GAIL •	ACER •
TD3 •	HER •	ACKTR •
TRPO •	PPO1 •	DDPG •

این کتابخانه، یک کتابخانه بسیار پویا می‌باشد و هر لحظه در حال آپدیت شدن می‌باشد.

**نکته ۳-۵-۲.** برخی از بسته‌هایی که در این پروژه کتابخانه استفاده شده است، صرفا بر روی سیستم عامل لینوکس قابل استفاده هستند. بنابراین برای نصب این بسته، باید از سیستم عامل لینوکس استفاده کرد.

<sup>43</sup>Synchronous Actor Critics

<sup>44</sup>Deep Q-Learning Network

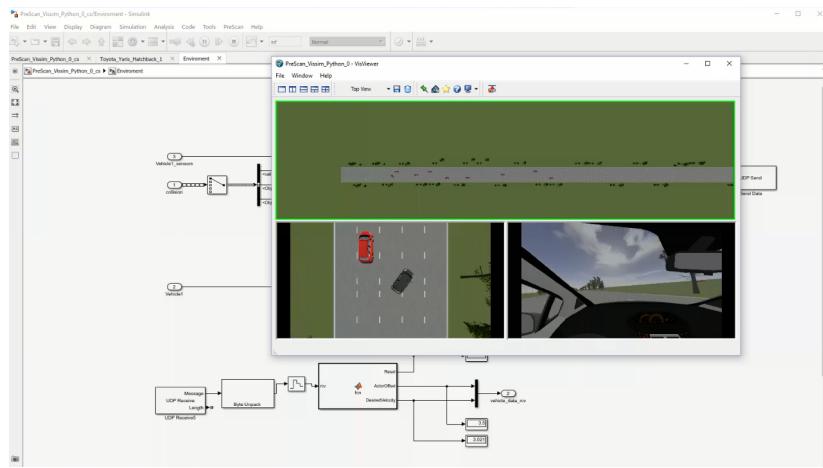
واضح است که این بسته در لایه الگوریتم شکل ۴-۲ استفاده می‌شود. این بسته برای نصب شدن به gym به صورت کامل نیاز دارد. همچنین این بسته برای کار کردن با شبکه‌های عصبی مصنوعی از کتابخانه tensorflow استفاده می‌کند.

پس از نصب tensorflow و gym[all] با دستور زیر این بسته نصب خواهد شد. برای ادامه نصب به مرجع [۵] مراجعه شود.

همچنین بسیاری از پروژه‌های تست شده این کتابخانه در [۶] قابل مشاهده است.

## فصل سوم

### تعریف مسیله و بررسی الگوریتم



شکل ۱-۳: محیط شبیه سازی

در فصل ۲ در مورد مفاهیم یادگیری تقویتی بحث شد. مهمترین مفاهیم عبارتند از:

- |         |         |              |              |           |                              |
|---------|---------|--------------|--------------|-----------|------------------------------|
| ۱. محیط | ۲. عامل | ۳. حالت محیط | ۴. حالت عامل | ۵. امتیاز | ۶. مشاهده پذیری <sup>۱</sup> |
|---------|---------|--------------|--------------|-----------|------------------------------|

هدف در این پروژه این بود که یک ماشین خودران<sup>۲</sup> با استفاده از الگوریتم های یادگیری تقویتی ساخته شود. جزئیات تئوری الگوریتم و جزئیات فنی پروژه به ترتیب در بخش های ۲ و ۴ آورده شده‌اند. در این بخش به شبیه سازی و جزئیات کار و تعریف پارامتر های این پروژه برداخته می‌شود.

### ۱-۳ معرفی محیط شبیه سازی

در ابتدا محیط شبیه سازی را معرفی می‌کنیم. جزئیات فنی این محیط در ۴ و همچنین نحوه راهاندازی آن در بخش ۱-۵ به صورت کامل مورد بحث قرار گرفته است. اگر آن محیط را باز کنید محیط مانند شکل ۱-۳ باز خواهد شد. این محیط دو آبجکت مهم دارد؛ (آ) ماشین(اتومبیل) (ب) جاده (شکل ۲-۳) چیزی که اهمیت دارد اندازه ها و نحوه تعریف محدوده هاست. شکل ۳-۳ اندازه ها و محدوده ها را مشخص کرده است. شکل ۳-۳(ب) نشان می‌دهد که این محدوده ها کاملاً برروی یک دیگر منطبق نیستند. دلیل اصلی این موضوع عدم اهمیت تطبیق دقیق این دو می‌باشد. در بخشی که پشت ماشین قرار دارد این محدوده از ۴ – (کمی بیشتر از اندازه عرض لاین ها) شروع می‌شود. زیرا نیازی نیست بیشتر

<sup>1</sup>Observability

<sup>2</sup>Autonomous Vehicle



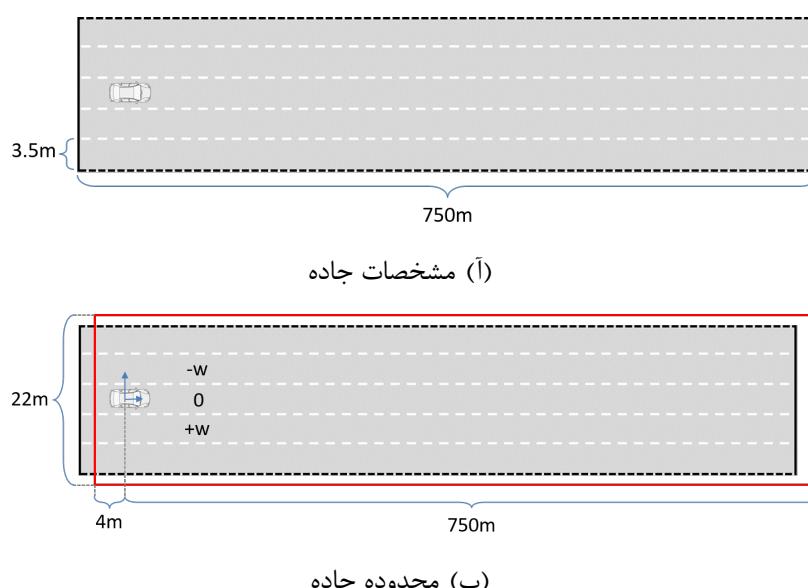
شکل ۳-۲: مدل های موجود در محیط شبیه سازی

از این مقدار ماشین مورد بررسی به عقب بروود تا متوجه شویم اشتباه در حال رفتن است. در حقیقت این مورد کمک می کند تا تعداد مرحله ها را در هر اپیزود اشتباه کاهش یابد. بخش های کناری نیز از  $-11$  تا  $+11$  محدود شده اند (بیشتر از عرض خود جاده) تا اگر نوسانی یافت به ماشین این اجازه داده شود تا به مسیر اصلی برگردد.

**یادداشت ۱-۱-۳.** ماشین در مبدا صفحه قرار دارد. از این رو اعداد منفی نسبت به همین ماشین نیز سنجیده می شوند.

۳ مقدار  $w$  ،  $0$  و  $-w$  که در شکل ۳-۳(ب) بر روی جاده نوشته شده است در حقیقت مرتبط با بحث فنی ماجرا می باشد اما مفهوم آن این است که عامل مورد بررسی می تواند این سه لاین را به عنوان حرکت اختیار کند. در حقیقت می توان آنها را به عنوان اسم برای هر لاین در نظر گرفت. در مورد حرکت بیشتر صحبت خواهد شد.

**یادداشت ۱-۱-۳.** راه اندازی این محیط کمی در دسر خواهد داشت از این رو نیاز است پیش از راه اندازی



شکل ۳-۳: بررسی دقیق محدوده و مشخصات جاده

بخش ۱-۵ به طور دقیق مطالعه شود.

## ۲-۳ معرفی رابط برنامه‌نویسی برنامه و الگوریتم

در این پروژه دو الگوریتم DQN و A2C بهتر از سایر الگوریتم‌ها عمل کردند اما در نهایت با توجه به آزمایش‌ها و ملاحظاتی که انجام شد، الگورینم DQN از لحاظ سرعت همگرایی بهتر از الگوریتم A2C پاسخ داد. بنابراین صرفاً بر روی این الگوریتم بحث خواهد شد.

```

1 import gym, gym_prescan
2
3 from stable_baselines.common.vec_env import DummyVecEnv
4 from stable_baselines.deepq.policies import MlpPolicy
5 from stable_baselines import DQN
6
7 save_load = "deepq_prescan"
8 env_dict = {
9     'id':      'prescan-without-matlabengine-v0',
10    'verbose': True,
11    'host':    '172.21.217.140',
12    # 'delay':  1,
13    'nget':    150
14 }
15
16 env = gym.make(**env_dict)
17 env = DummyVecEnv([lambda: env])
18 model = DQN(MlpPolicy, env, verbose=1, gamma=0.8,
19             prioritized_replay=True)
20 print('Model created!')
21 try:
22     model.learn(total_timesteps=50000)
23 except:
24     print('Error!')
25 model.save(save_load)

```

این کد بخش آموزش<sup>۳</sup> را نشان می‌دهد. بخش تست<sup>۴</sup> در تمامی الگوریتم‌ها مشابه یک دیگر است و

<sup>3</sup>Train

<sup>4</sup>Test

از جایی که مدل تعریف می‌شود (در اینجا خط ۱۸) شروع خواهد شد.  
بخش تست در تمامی الگوریتم‌ها کد زیر است.

```

1 model = DQN.load(save_load)
2 obs = env.reset()
3 while True:
4     print('-----TEST-----')
5     action, _states = model.predict(obs)
6     obs, rewards, dones, info = env.step(action)
7     env.render()
```

از روی چند خط آغازین کد DQN می‌توان دریافت که این کد با استفاده از [stable-gym](#) [۴] و [baseline](#) [۵] نوشته شده است.

بخش مهم بعدی متغیری از جنس دیکشنری به نام env\_dict است. این متغیر برای ساختن متغیر env در دستور `env = gym.make(**env_dict)` به کار می‌رود.<sup>۵</sup> توضیح این متغیر و اجزای آن در جدول ۱-۳ آمده است.

همان‌طور که در جدول ۱-۳ توضیح داده شده است، دو متغیر nget و delay هر دو از جنس تاخیر می‌باشند. محل تاخیر در تابع step می‌باشد. کد زیر محل تاخیر را نشان می‌دهد.

```

1 def step(self, action):
2     self.send(action)
3     # ----- BEGIN DELAY -----
4     if self.delay > 0 :
5         sleep(self.delay)
6     for _ in range(self.nget):
7         self.render_()
8         if self.done:
9             break
10    # ----- END DELAY -----
11    observation = self._next_observation()
12    reward = self.calc_reward()
13    done = self.done
14    info = {'Collision':self.collision
15            , 'Position':self.agent['data']['Position']}
```

<sup>۵</sup> متغیر env در حقیقت نقش محیط را در الگوریتم دارد.

متغیر	توضیحات
id	در این پروژه این متغیر دو حالت بیشتر ندارد که هردو از جنس رشته هستند. اگر این کد با استفاده از موتور متلب استفاده شود، 'prescan-v0' خواهد بود و اگر از موتور متلب استفاده نشده باشد مقدار آن 'prescan-without-matlabengine-v0' خواهد بود. این متغیر مقدار پیشفرض ندارد.
verbose	این متغیر که از جنس بولین میباشد، در صورتی که یک باشد اطلاعات جامعی را در هر مرحله را چاپ میکند. علاوه برآن اطلاعات آماری امتیازهای بدستآمده در پایان هر اپیزود را نیز چاپ میکند. به طور کلی اجازه گزارش دادن و ندادن اطلاعات درونی الگوریتم توسط این متغیر کنترل میشود.
host	این متغیر برای اتصال شبکه بین دو کامپیوتر به کار میرود و در حقیقت IP کامپیوتری است که ویندوز بر روی آن نصب است. مقدار پیشفرض این متغیر 'localhost' میباشد.
delay	همان طور که مشخص است این متغیر مقدار تاخیر را مشخص میکنم و مورد کاربرد آن لحظه ای است که action در تابع step فرستاده شده است و پس از گذشت مقداری تاخیر بر حسب ثانیه سعی در دریافت اطلاعاتی مانند مشاهده بعدی و محاسبه امتیاز و ... داریم. مقدار پیشفرض این متغیر نیز صفر است.
nget	این متغیر نیز به نوعی متفاوت تاخیر را شکل میدهد. این متغیر از نوع عدد صحیح میباشد و هنگامی که مقدار آن ۱۵۰ است یعنی محل تاخیر، ۱۵۰ بار دادهها را دریافت میکند و مقدار آنها را میخواند. در حالت عادی تا پایان ۱۵۰ ثانیه دریافت هیچ کاری نمیکند مگر این که مقدار done برابر یک شود؛ در این صورت حلقه را متوقف کرده و باقی عملیات را انجام میدهد. مقدار پیشفرض این متغیر یک میباشد.
experimant_name	این متغیر مربوط به تنظیمات موتور متلب میباشد و به صورت عادی نیازی به تغییر مقدار پیشفرض آن نیست.
close_window	این پارامتر درصورتی قابل اجراست که کد پایتون و نرمافزار پریاسکن هردو بر روی یک کامپیوتر باشند و وظیفه آن این است که محیط گرافیکی را پس از اجرا شدن کد میبندد و مقدار پیشفرض آن صفر میباشد.

جدول ۳-۱: بررسی پارامتر های موجود در env\_dict

این کد که در حقیقت هسته اصلی <sup>۶</sup>تابع step می‌باشد. در بین محدوده مشخص شده، تاخیر صورت می‌گیرد. همان طور که مشخص است این تاخیر بین فرستادن حرکت و محاسبه پارامترهایی مانند امتیاز و مشاهده(حالت) می‌باشد.

یادداشت ۱-۲-۳. علت اصلی وجود تاخیر، مهلت دادن به عامل برای انجام حرکت است.

دو متغیر delay و nget هردو وظیفه ایجاد تاخیر دارند که نحوه ایجاد این تاخیر با یکدیگر کاملاً متفاوت است. همچنین این امکان نیز وجود دارد به صورت ترکیبی نیز این تاخیر را ایجاد کرد. هر کدام از این روش‌ها مزايا و معایب خاص خود را دارند.

مزیت مهم استفاده از nget این است که به صورت مداوم در حال دریافت اطلاعات از محیط شبیه‌سازی است. بنابراین در صورت رخ دادن اتفاق خاصی مانند تصادف کردن و یا از مسیر خارج شدن می‌تواند آنرا به موقع تشخیص دهد و تصمیمات لازم را انجام دهد. در صورتی که در زمانی که تاخیر ناشی از delay است، عمل در آن مدت ارتباط با محیط شبیه سازی قطع شده است و ممکن است رخ دادن موارد گفته شده یا بسیار دیر متوجه شود و یا اصلاً متوجه نشود.

به طور مثال در صورتی که دو ماشین با یکدیگر تصادف داشته باشند؛ اگر این رخداد سریعاً تشخیص داده نشود، در این صورت احتمال دارد این دو ماشین از روی یک دیگر عبور کنند! و پس از عبور کردن این اطلاعات دریافت شود و برخوردی تشخیص داده نشود. از آن جا که برخورد بیشترین میزان تاثیر در امتیاز دارد؛ بنابراین، این اتفاق تاثیرات خیلی محربی می‌تواند بر الگوریتم بگذارد.

عیب اصلی روش nget نیز این است که یک مقدار مشخص ندارد و به پارامترهایی از جمله سرعت شبکه نیز وابسته است. بنابراین اگر از یک شبکه به شبکه دیگر منتقل شود می‌تواند مقدار کاملاً متفاوتی به خود گیرد که شاید مطلوب نباشد. اما به راحتی با عوض کردن مقدار این متغیر در لایه الگوریتم می‌توان این مشکل را حل کرد. بنابراین توصیه می‌شود از این متغیر استفاده شود.

به کد DQN برگردیم. خط ۱۸ این کد مدل را می‌سازد. چیزی که اهمیت دارد این است که پارامتر ۷ چه مقداری انتخاب شود. در نسخه نهایی این مقدار روی  $80\%$  تنظیم شده است. در مورد این متغیر در بخش ۲ و در مراجع [۲] و [۱] بحث شده است. در ابتدا این متغیر مقدار پیش‌فرض  $90\%$  را داشت. پس از بررسی‌های انجام شده این مقدار به  $80\%$  کاهش یافت.

---

<sup>۶</sup>این کد از آن جهت که کاملاً با تابع اصلی برابر نمی‌باشد، واژه "هسته اصلی" برای آن در نظر گرفته شده است. تفاوت آن با کد اصلی برخی عملیات است که مرتبط با چاپ شدن اصلاحات در حال اجرا می‌باشد که به مقدار verbose مرتبط می‌شود.

نام تابع	توضیحات
<code>__init__</code>	این تابع علاوه بر تنظیم کردن برخی پارامترهای مرتبط به کلاس، اتصال کد پایتون به نرم افزار متلب را نیز برعهده دارد. همچنین تعیین فضای مشاهده و فضای حرکت نیز برعهده این بخش می باشد.
<code>step</code>	این تابع همان مرحله است که در بخش ۲ مطرح شد. محل اصلی اجرای این تابع در داخل یک حلقه متناهی می باشد. این تابع <code>action</code> را به صورت ورودی می گیرد و متغیرهایی مانند <code>info</code> ، <code>done</code> ، <code>reward</code> ، <code>observation</code> را خروجی می دهد. در مورد نحوه محاسبه این خروجی ها صحبت خواهد شد.
<code>reset</code>	این تابع <code>env</code> را ریست می کند و به عنوان خروجی حالت اولیه را برمی گرداند. موارد استفاده از این تابع معمولاً در اول کد و در آخر هر اپیزود می باشد. آخر هر اپیزود هنگامی فرا می رسد که متغیر <code>done</code> که یکی از خروجی های تابع <code>step</code> است، یک شود.
<code>render</code>	این تابع به صورت معمول کارهای گرافیکی را برعهده دارد. اما از آنجایی که عمل در پس زمینه طرح وجود دارد، پس کار اصلی آن گرفتن داده ها و منظم کردن آن ها می باشد. برای این کار از یک تابع کمکی به نام <code>render_</code> استفاده می کند.

جدول ۳-۲: راهنمای توابع اصلی رابط برنامه نویسی برنامه

### ۳-۳ تعریف کردن پارامتر های یادگیری تقویتی

منظور از پارامترهای یادگیری تقویتی از متغیرهای حرکت و حالت و امتیاز و ... تا تعریف برخی توابع می باشد. ابتدا کلیات توابع را بررسی کنیم و سپس وارد جزئیات آن پارامتر ها می شویم.

توابع استفاده شده به دو دسته تقسیم می شوند. (آ) توابع اصلی (ب) توابع فرعی یا کمکی.

توابع اصلی آن دسته از توابعی هستند که مختص به کتابخانه `gym` هستند و قرار دادن آن ها به شکل صحیح آن، اجباری است. توابع کمکی آن دسته از توابعی هستند که در این توابع نقش های مشخصی را ایفا کردن و لی استفاده کردن از آن ها اجباری نداشته است.

یادداشت ۳-۳. در صورت لزوم کاربر می تواند توابع فرعی را تغییر دهد تا خروجی مطلوب خود را حاصل کند اما در لایه الگوریتم صرفا از توابع اصلی استفاده می شود. زیرا هدف هم که استاندارد سازی کد می باشد با این موضوع سازگار است.

جدول ۳-۲، توابع اصلی را نشان می دهد و جدول ۳-۳ نیز توابع کمکی را نشان می دهد. همچنین لازم به ذکر است که **کد تست استاندارد** این پروژه در به صورت زیر است:

```

1 import gym, gym_prescan
2
3 env_dict = {
4     'id':      'prescan-without-matlabengine-v0',
5     'host':    '172.21.217.140',
6     'verbose': True,
7     'nget':    152
8 }
9 env = gym.make(**env_dict)
10 for i_episode in range(20):
11     observation = env.reset()
12     for t in range(100):
13         env.render()
14         print(observation)
15         action = env.action_space.sample()
16         observation, reward, done, info = env.step(action)
17         if done:
18             print("Episode finished after {} timesteps".format(t+1))
19             break
20 env.close()

```

یادداشت ۳-۲-۳. این کد صرفاً صحت عملکرد و نحوه استفاده از رابط برنامه‌نویسی برنامه را نشان می‌دهد و شامل هیچ گونه الگوریتمی نمی‌باشد.

### ۱-۳-۳ معرفی برخی توابع رابط برنامه‌نویسی برنامه

بررسی تابع `__init__`:

این تابع سه وظیفه مهم دارد.

آ) بخشی از وظایف آن، وظایف مشخص آن در کد پایتون است.

ب) ست کردن برخی پارامتر های مهم که در جدول ۱-۳ نیز آمده‌اند.

ج) این تابع فضای مشاهده و فضای حرکت را مشخص می‌کند.

اطلاعات فضای مشاهده و فضای حرکت در جدول ۴-۳ آمده‌است.

نامتابع	محل استفاده	توضیحات
make	__init__	این تابع لایه ای را ایجاد می کند که هرگونه ارتباط با محیط شبیه سازی به آن لایه مرتبط می شود.
render_	render و reset و step	این تابع وظیفه دریافت اطلاعات از محیط شبیه سازی و استخراج داده های مفید از آن است.
send	step	این تابع برای فرستادن حرکت به محیط شبیه سازی استفاده می شود.
calc_reward	step	این تابع امتیاز را محاسبه می کند.
_next_observation	step و reset	این تابع حالت بعدی عامل را محاسبه می کند.
action_translate	send	این تابع مانند یک دسته بازی در تابع send حرکت را تفسیر می کند و دستورات کنترلی قبل اجرا برای محیط شبیه سازی ایجاد می کند.

جدول ۳-۳: راهنمای توابع کمکی رابط برنامه نویسی برنامه

نام متغیر	مفهوم	جنس متغیر	توضیحات
action_space	فضای حرکت	spaces.Discrete(6)	عدد صحیح ۶
observation_space	فضای مشاهده	spaces.Box(shape=(1, 38), dtype=np.float16)	ماتریس $1 \times 38$ تایی

جدول ۴-۳: تعریف فضای مشاهده و فضای حرکت در پروژه

#### بررسی تابع action\_translation

این کد دقیقاً پیاده سازی یک دسته بازی<sup>۷</sup> می باشد. شکل ۴-۲ اطلاعات کامل این موضوع به همراه تفسیر آن ها دارد.

از آنجایی که در این پروژه فضای حرکت طبق جدول ۴-۳ مقدار عدد صحیح ۶ را دارد و این به آن معناست که ۶ حالت گسسته بین صفر تا ۵ برای حرکت وجود دارد همچنین نشان می دهد که جنس action عدد صحیح int است. بنابراین، می بایست که آن را تفسیر کرد. وظیف اصلی این تابع نیز تفسیر مقدار مختلفی است که action می تواند بگیرد، می باشد.

<sup>7</sup>Joystick

```

1 def action_translate(self,action):
2     lanewidth =
3         self.enviroment.road.laneWidth
4     self.__action_old__ = self.__action__
5     vel = self.agent['data']['Velocity']
6     offset = self.__action__[0]
7
8     if action == 0 :
9         offset = -lanewidth
10    if action == 1 :
11        offset = 0
12    if action == 2 :
13        offset = lanewidth
14    if action == 3 :
15        vel = action_velocity(vel,True)
16    if action == 4 :
17        vel = action_velocity(vel,False)
18
19    self.__action__ = [offset,vel]
20
21    return self.__action__

```



شماره	وظیفه
۰	رفتن به لاین -w
۱	رفتن به لاین ۰
۲	رفتن به لاین +w
۳	زياد کردن سرعت
۴	کم کردن سرعت
۵	بدون تغییر

شكل ۴-۳: بررسی تابع action\_translate

یادداشت ۳-۳-۳. دستور کنترلی اصلی یک بردار دوتایی است (خط ۱۸ کد شکل ۴-۳) که مقدار اولی آن لاین را نشان می‌دهد و مقدار دوم آن سرعتی می‌باشد که انتظار داریم که عامل، سرعت خود را به آن برساند.

یادداشت ۳-۳-۴. اگر دقت کنید در کد مذکور دو مقدار کنونی و قدیمی تر action نگهداری شده است.

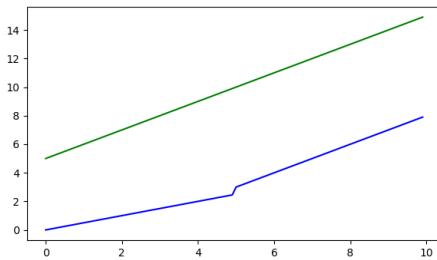
یادداشت ۳-۳-۵. همچنین دقت شود که در این کد، مقدار سرعت، همان مقدار حقیقی سرعت است که از محیط شبیه‌سازی می‌آید. و مقداری که در بردار action قرار می‌گیرد مقدار کنترلی سرعت است که با یکدیگر تفاوت داردند.

نکته دیگری که حائز اهمیت است این است که هنگامی که گزینه ۳ و یا ۴ انتخاب می‌شوند، سیاستی برای افزایش و کاهش سرعت اتخاذ شده است. این سیاست در قالب یکتابع در تصاویر شکل ۵-۳ ظاهر شده است. همانطور که کد ۵-۳(آ) و نمودار متناظر آن در شکل ۵-۳(ب) نشان می‌دهد، سیاست‌های مختلفی برای زیاد کردن و کم کردن سرعت اتخاذ شده است.

برای زیاد کردن سرعت، سرعت واقعی که از محیط شبیه‌سازی دریافت شده است با ۵ واحد جمع می‌شود. بنابراین انتظار داریم سرعت پس از سه بار افزایش به ۱۵ برسد (شکل ۵-۳(ج)، نمودار قرمز) اما این اتفاق نمی‌افتد. زیرا این افزایش سرعت، کار زمان‌بری است و نیاز به حوصله دارد که اگر حوصله و تحمل و به عبارت دیگر تاخیر را از یه حدی بالاتر ببریم عملاً در کنترل عامل به مشکل خواهیم رسید. همچنین مورد مشابه آن چه که گفته شد، در شکل ۵-۳(د) نیز برقرار است. نقاطی که رنگشان قرمز است نمودار ایدآلی مفروض خواهند بود که معادل تاخیر کم سیستم جهت اعمال سرعت نهایی است. و نمودار دیگر معادل رخدادی است که ۳۰٪ به آن عمل شده است و ۷۰٪ تحت تاثیر مقدار قبلی خواهد بود و به این صورت یک میانگین وزن‌دار گرفته شده است.

مشاهده می‌شود که مسیری که به واقعیت نزدیک‌تر است آرامتر از مسیر مدنظر است. همچنین تفاوت تغییر روند کاهشی سرعت‌های کمتر از ۵ واحد، این است که هیچ گاه منفی نشود ولی به صورت نمایی کاهش یابد و نزدیک صفر شود.

تفاوت دیگر آن است که به هنگام افزایش مقدار ۵ واحد به سرعت افروده می‌شود و در هنگام کاهش مقدار دو واحد (برای سرعت‌های بالای ۵ واحد) از آن کسر می‌شود. این تفاوت در مقایسه فاصله نقاط بین دو نمودار شکل‌های ۵-۳(ج) و ۵-۳(د) نیز ظاهر شده است. علت اصلی این تفاوت در بررسی



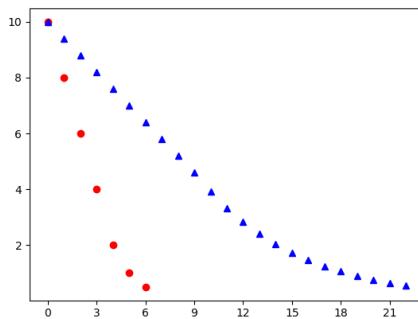
(ب) شکل کلی نمودار افزایش سرعت

```

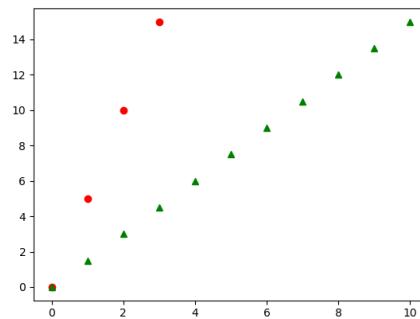
1 def action_velocity(vel, increase):
2     if increase:
3         return vel+5
4     else:
5         if vel < 5:
6             return vel/2
7         else:
8             return vel - 2

```

(آ) کد اصلی روند افزایشی و یا کاهشی سرعت



(د) حرکت در جهت افزایش سرعت با شروع از صفر



(ج) حرکت در جهت کاهش سرعت با شروع از صفر

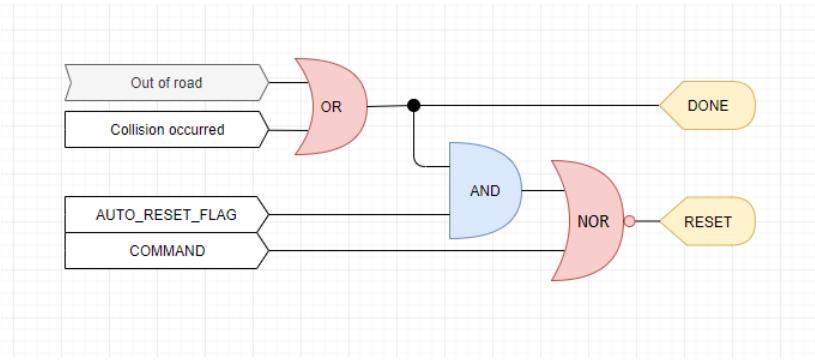
شکل ۳-۵: بررسی دقیق تابع افزایش دهنده و کاهنده سرعت ماشین

خود را نشان می‌دهد. اما پیشتر در نظر داشته باشید که یک سیاست چهت‌دار و تشویقی جهت قرار گرفتن سریع‌تر در مسیر درست می‌باشد.

#### بررسی تابع :step

کد این تابع را پیش تر بررسی شد. جایگاه این تابع داخل کد تست استاندارد در داخل یک حلقه است هنگامی که حلقه تمام شود به اصلاح یک اپیزود تمام شده است. دلیل تمام شدن حلقه یا رسیدن به سقف تعداد مرحله در هر اپیزود است و یا یک شدن مقدار done. این مقدار یکی از خروجی‌هایی است که این متغیر می‌تواند اختیار کند.

این تابع مقدار حرکت را می‌گیرد. سپس آن را برای محیط شبیه سازی می‌فرستد. در کد زیر این کار توسط تابع send انجام می‌پذیرد. در تابع send ابتدا با استفاده از تابع action\_translate صورت دستور کنترلی در خواهد آمد و پس از این تبدیل برای محیط شبیه سازی ارسال می‌شود. پس از مقداری تاخیر، متغیر های reward و observation که به ترتیب مشاهده و امتیاز می‌باشند، محاسبه می‌شود. در کنار این دو خروجی، خروجی‌های دیگری وجود دارند. done که از محیط شبیه



شکل ۳-۶: منطق محاسبه done در محیط شبیه‌سازی

سازی می‌آید و نشان می‌دهد که اپیزود تمام شده است. متغیر info می‌تواند اطلاعات اضافه‌ای را خروجی می‌دهد که در دیباگ کردن مفید خواهد بود. این اطلاعات اضافی در این پروژه، اطلاعات برخورد و اطلاعات موقعیت عامل انتخاب شده اند.

یادداشت ۳-۳-۶. نحوه محاسبه done وظیفه محیط شبیه‌سازی می‌باشد. منطق محاسبه آن در شکل ۳-۶ آمده است. نبایراین اطلاعات دو حالت وجود دارد که موجب شدن یک اپیزود و به‌طور معادل یک شدن done می‌شود:

آ) خارج شدن از محدوده جاده<sup>۴</sup>

ب) تصادف کردن با ماشین دیگر

یادداشت ۳-۳-۷. مدار منطقی شکل ۳-۶ علاوه بر محاسبه متغیر done، در مورد نحوه ریست شدن محیط نیز تصمیم‌گیری می‌کند که مرتبط به بخش فنی است و در این لایه بررسی نمی‌شود.

کد این تابع در زیر آمده است. دو تابع calc\_reward و next\_observation به ترتیب مشاهده و امتیاز را محاسبه می‌کنند که به صورت جداگانه بررسی خواهند شد.

```

1 def step(self, action):
2     self.send(action)
3     # ----- BEGIN DELAY -----
4     if self.delay > 0 :
5         sleep(self.delay)
6     for _ in range(self.nget):
7         self.render_()
8         if self.done:
  
```

<sup>۴</sup>منتظر از محدوده جاده، محدوده‌ای است که در شکل ۳-۳(ب) مشخص شده است.

```

9         break
10        # ----- END DELAY -----
11        observation = self._next_observation()
12        reward = self.calc_reward()
13        done = self.done
14        info = {'Collision':self.collision
15                  , 'Position':self.agent['data']['Position']}

```

### ۲-۳-۳ بررسی تابع : \_next\_observation

این تابع مشخص می‌کند که چه چیزی به عنوان مشاهده اعلام شود. همان‌طور که در جدول ۴-۳ آمده است، خروجی نهایی این تابع دارای ابعاد  $38 \times 1$  می‌باشد. شکل ۷-۳ این روند را به‌طور کامل نشان می‌دهد.

بنابراین مشاهده یک بردار ۳۸ تایی است که ۳۶ داده اول آن بر اساس زاویه و اندازه محاسبه شده است و ۲ داده دیگر آن مقدار  $u$  و مقدار سرعت( $v$ ) می‌باشد.

بر روی ماشین یک سنسور وجود دارد که دو بردار  $10^{\circ}$  تایی شامل فاصله و زاویه از  $10^{\circ}$  ماشین کنار خود را می‌دهد. به این ترتیب برای تعریف مشاهده فضای اطراف ماشین به ۳۶ قسمت افزایش شد به طوری که هر افزای آن یک محدوده به زاویه  $10^{\circ}$  درجه را شامل می‌شود. شکل ۸-۳ طرز تعریف و مقدار دهی این بردار ۳۶ تایی را نشان می‌دهد. اگر در آن افزای ماشینی قرار نگرفت مقدار آن صفر است و در غیر این صورت فاصله آن با نزدیکترین ماشین خواهد بود.

یادداشت ۳-۳-۸. از آنجایی که مقدار زاویه در بازه  $(-\theta, \theta) \in [180^\circ, 180^\circ]$  قرار دارد، برای از بین بردن بازه منفی کل زوایا ابتدا با  $180^\circ$  جمع شد و سپس افزای صورت گرفت. روش دیگر می‌توانست با استفاده از همنهشتی به پیمانه  $360^\circ$  باشد که به هنگام آزمایش روش اول کمک می‌کرد که بهتر یاد بگیرد.

### بررسی تابع calc\_reward

این تابع امتیازهای رابط برنامه‌نویسی برنامه را تنظیم می‌کند. بنابراین مهم‌ترین بخش‌های آن محسوب خواهند شد. کد این تابع در زیر آمده است.

```

1 def calc_reward(self):
2     lanewidth = self.enviroment.road.laneWidth
3     vel_sim = self.agent['data']['Velocity']

```

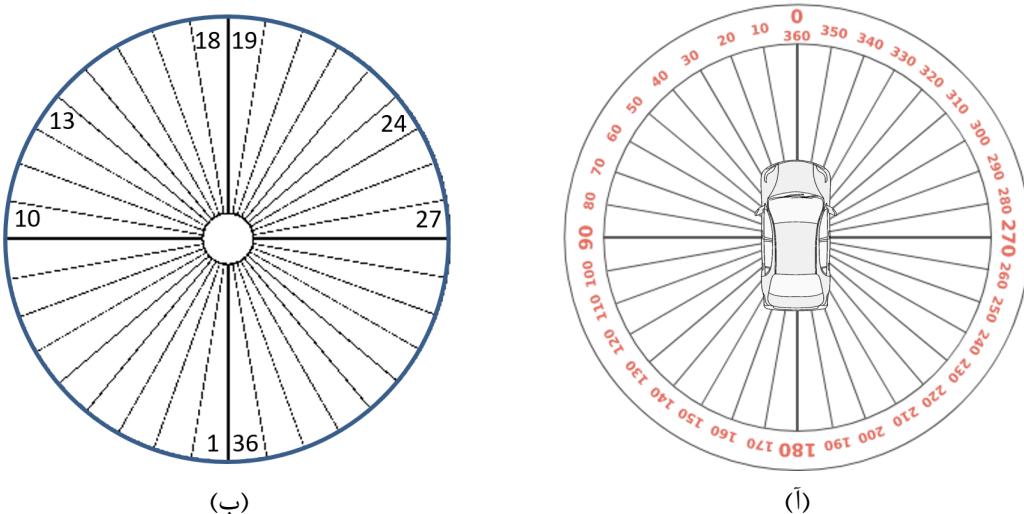
```

1 def _next_observation(self):
2     self.render_()
3
4     obs = np.zeros((1,36), dtype=np.float)
5
6     car = self.agent
7
8     theta = car['Sensors'][0]['data']['theta']
9     Range = car['Sensors'][0]['data']['Range']
10
11
12     for i in range(len(theta)):
13         t = int((theta[i] + 180)/10)
14
15         r = Range[i]
16
17         if obs[0,t] != 0:
18             obs[0,t] > r
19
20             continue
21
22         obs[0,t] = r
23
24     extra = [car['data']['Velocity'],
25             car['data']['Position']['y']]
26
27     self.__obs__ = np.append(obs, extra)
28
29     return self.__obs__
30
31
32
33
34
35
36
37

```

Range & theta

شکل ۳-۷: نحوه تعریف مشاهده



شکل ۳-۸: افزار محیط اطراف ماشین بر حسب زاویه و تعریف نحوه قرار دادن آن در ماتریس مشاهده

```

4     vel_cmd = self.__action__[1]
5     Vel = 0.8 * vel_sim + 0.2 * vel_cmd
6
7     Longitudinal_reward = reward_velocity(Vel,28) *1.5
8     Collision_reward = -25 if self.collision['Occurred'] else 0
9     Violation_reward = -0.75 * (np.abs(self.__action__[0] -
10           self.__action_old__[0])/lanewidth)
11    Nearby_reward = nearby_reward_linear(self.__obs__[12],-2.5,1.5) +\
12          nearby_reward_linear(self.__obs__[23],-2.5,1.5)
13    reward_T = Longitudinal_reward + Collision_reward +\
14          Violation_reward + Nearby_reward
15
16    return reward_T

```

همان‌طور که کد نیز نشان می‌دهد، امتیاز نهایی به صورت مجموع ۴ امتیاز دیگر می‌باشد. هر یک از این امتیازها مربوط به یک بخش خاص است. بنابراین،

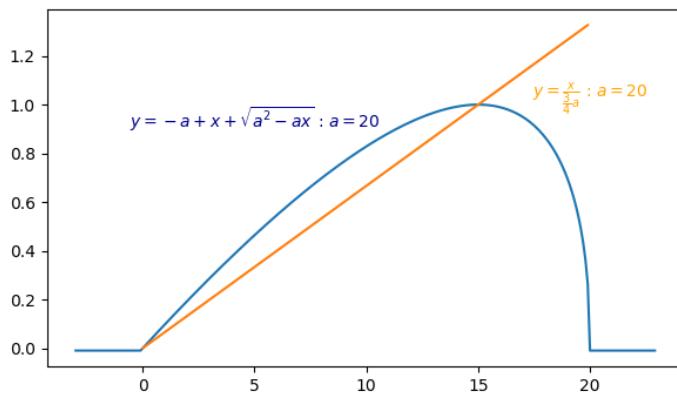
$$\mathcal{R}_T = \mathcal{R}_{\text{Collision}} + \mathcal{R}_{\text{Violation}} + \mathcal{R}_{\text{Longitudinal}} + \mathcal{R}_{\text{Nearby}}$$

$\mathcal{R}_{\text{Collision}}$  در صورتی که تصادف رخ دهد، مقدار آن ۲۵ – و در این صورت صفر خواهد بود. بنابراین هنگام تصادف علاوه بر تمام شدن اپیزود یک امتیاز منفی با اندازه زیاد دریافت می‌کند.  $\mathcal{R}_{\text{Violation}}$  تعریف شد تا عامل متوجه شود که تغییر لاین هزینه خواهد داشت. مقدار این هزینه ۷۵ – برابر مقدار تغییر لاین است. یعنی اگر یک واحد لاین خود را تغییر دهد، مقدار آن  $75^\circ$  – خواهد شد و اگر دو واحد لاین عوض کند، مقدار آن  $1/5$  – خواهد شد.

یادداشت ۳-۹. اندازه مقدار  $\mathcal{R}_{\text{Violation}}$  به ازای یک واحد تغییر لاین، نصف مقدار بیشینه  $\mathcal{R}_{\text{Longitudinal}}$  یادداشت ۳-۳-۳. انتخاب شده است.

یادداشت ۳-۱۰. تنها امتیاز مثبت فقط  $\mathcal{R}_{\text{Longitudinal}}$  با مقدار بیشینه  $1/5$  می‌باشد.

برای محاسبه  $\mathcal{R}_{\text{Longitudinal}}$  از یکتابع به نام  $\text{reward\_velocity}$  با ویژگی‌های مشخص استفاده شده است. این تابع در ورودی اول خود، ورودی تابع و در ورودی دوم خود پارامتر تابع را دریافت می‌کند که این پارامتر مقدار ثابتی دارد. تابع مذکور، از لحاظ مقدار بیشینه بسیار خوش‌تعریف است و دارای ضابطه زیر است.



شکل ۹-۳: نمودار تابع محاسبه امتیاز سرعت نرمال شده

$$F^{\mathcal{R}_v}|_a = \begin{cases} -a + x + \sqrt{a^2 - ax} & : x \in [0, a] \\ 0 & : \text{سایر نقاط} \end{cases}$$

این بیشینه مقدار خود را در نقطه  $\sqrt{a^2 - ax}$  با مقدار بیشینه  $a/2$  اختیار می‌کند. اگر این تابع را بر مقدار بیشینه اش تقسیم کنیم، نرمال خواهد شد. بنابراین:

$$F_n^{\mathcal{R}_v}|_a = \frac{F^{\mathcal{R}_v}|_a}{\max\{F^{\mathcal{R}_v}|_a\}} = \begin{cases} \frac{4}{a} \left( -a + x + \sqrt{a^2 - ax} \right) & : x \in [0, a] \\ 0 & : \text{سایر نقاط} \end{cases}$$

تابع reward\_velocity به عنوان ورودی سوم تصمیم می‌گیرد که خروجی نرمال شده باشد یا نه. اگر normal=True باشد خروجی این تابع برابر خروجی تابع  $F_n^{\mathcal{R}_v}|_a$  همین تابع خواهد بود و اگر این مقدار False باشد، خروجی برابر خروجی تابع  $F^{\mathcal{R}_v}|_a$  خواهد بود. به صورت پیشفرض این تابع خروجی نرمال شده خواهد داشت.

نمودار این تابع در ۹-۳ آمده است که با تابع خطی که از مقدار بیشینه آن عبور می‌کند مقایسه شده است.

آ) تا قبل از مقدار بیشینه مقدار این دو تابع نزدیک یک دیگر می‌باشد و شبیه آن‌ها نیز به هم نزدیک است.

ب) این تابع پس از مقدار رسیدن به مقدار بیشینه خود، به صورت نزولی افت می‌کند.

ج) اگر از نقطه‌ای که مقدار بیشینه در آن رخ می‌دهد، به سمت مثبت حرکت کنیم سریع‌تر از هنگامی که به سمت منفی حرکت می‌کند.

د) دیگر ویژگی آن این است که در حدود مقدار بیشینه خود، تقریباً هموار است که این هموار بودن آن موجب آن خواهد شد که عامل بتواند در حوالی سرعت بیشینه خود بدون تفاوت زیادی در مقدار امتیاز تصمیم بگیرد که سرعت را در صورت لزوم کم و یا زیاد کند.

نکته ۱۱-۳ (خیلی مهم). چیزی که به عنوان متغیر مستقل به تابع ارسال می‌شود سرعت است اما با سرعت واقعی متفاوت است. در حقیقت میانگین وزن‌داری از سرعت واقعی و سرعت دستوری می‌باشد. منظور از سرعت دستوری همان سرعتی است که با استفاده از دستورات کنترلی برای محیط شبیه سازی فرستاده می‌شود.

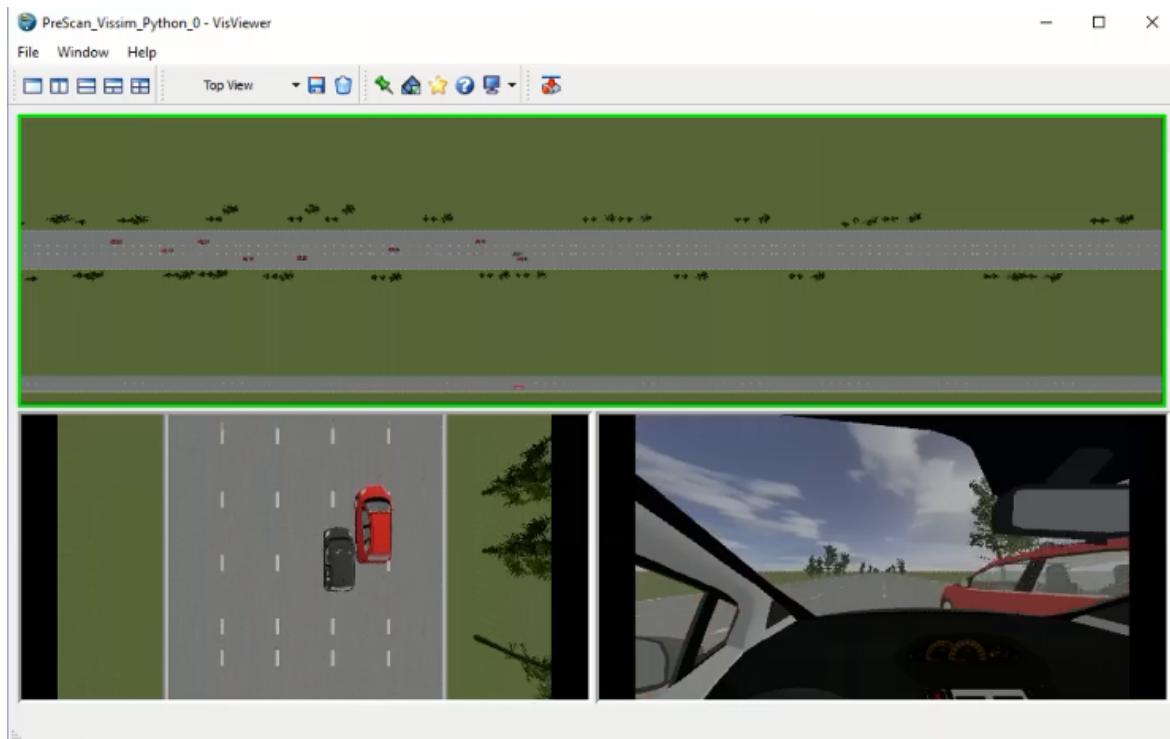
$$V = {}^\circ \Delta V_{\text{sim}} + {}^\circ \Delta V_{\text{cmd}}$$

علت این تفاوت نیز به سیاست‌های تشویقی مربوط می‌شود. فرض کنید عامل تصمیم می‌گیرد سرعت را افزایش دهد. با این تصمیم مقدار سرعت ۵ واحد افزایش می‌یابد. ولی سرعت بعد از گذشت از یک مرحله به مقدار کمی سرعت آن افزایش یافته است از این رو امتیاز کمی می‌گیرد. با این میانگین‌گیری در حقیقت برای تصمیم و نیت خوب عامل مقداری امتیاز در نظر گرفته می‌شود تا مقدار امتیاز بیشتری را بگیرد.

در صورتی که عامل تصمیم بگیرد که سرعت خود را کم کند، بابت این تصمیم که نامطلوب است جرمیه می‌شود. مقدار این جرمیه کمتر از مقدار تشویق است. این موضوع در نزدیکی‌های سرعت بیشینه کمک کننده خواهد بود. زیرا بالا بردن سرعت در آن حوالی مطلوب نیست بنابراین در صورتی که نیت کند که سرعت زیاد شود، این تصمیم عامل همراه با افت امتیاز بیشتری (نسبت به حالتی که نیت در نظر گرفته نشده است) خواهد بود. بنابراین این تفاوت‌ها مطلوب و سازنده خواهد بود.

آخرین امتیاز،  $R_{\text{Nearby}}$  می‌باشد. بخاطر مشکلات شبیه‌سازی در اثر تغییر لاین، این تغییر به تمامی عوامل نمی‌شود و ممکن است عامل در فاصله بسیار نزدیک از یک ماشین دیگر رانندگی کند به گونه‌ای که تصادف رخ ندهد. (شکل ۱۱-۳(b) را مشاهده کنید). هدف از ایجاد این امتیاز نیز ایجاد یک حس خطر است. مسلماً باید علامت آن منفی باشد.

برای این منظور به نوعی دو سنسور مجازی در کناره‌های عامل کار گذاشته شد. در صورتی که ماشین دیگر در فاصله ۱/۵ متری از عامل در جهت‌های داده شده در شکل ۱۱-۳(b) قرار داشته باشد،



شکل ۳-۱۰: تصویر نمونه‌ای از کنار هم شدن دو ماشین از نزدیکی یک دیگر

مقدار آن صفر خواهد بود و در صورتی که فاصله از این مقدار کمتر شود به صورت خطی تا  $2/5$ -امتیاز منفی دریافت می‌کند.

نمودار تابع در شکل ۱۱-۳(ب) آمده است و ضابطه آن به صورت زیر می‌باشد:

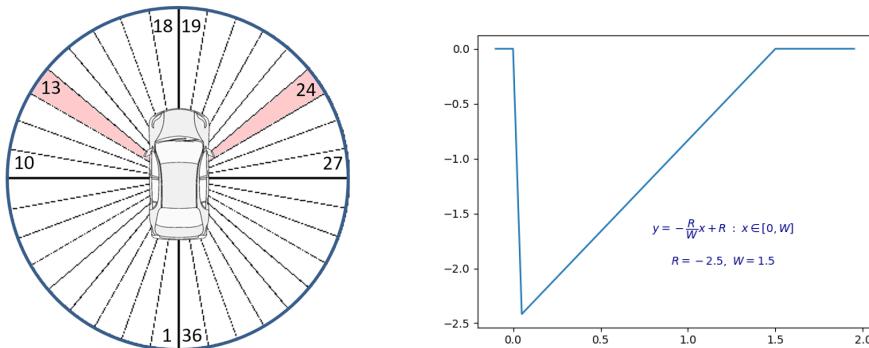
$$F^{\mathcal{R}_{\text{Nearby}}}|_{R,W} = \begin{cases} -\frac{R}{W}x + R & : x \in [0, W] \\ 0 & : \text{سایر نقاط} \end{cases}$$

که در رابطه فوق، مقدار  $W$  طول سنسور مجازی و اندازه  $R$  بیشترین امتیاز (اما در جهت منفی) که در این نمودار اتفاق می‌افتد.

### ۳-۳-۳ بررسی مقدار $\gamma$

مقدار  $\gamma$  در لایه الگوریتم مطرح می‌شود. در الگوریتم DQN این پارامتر، مقدار دید روبه جلو را نشان می‌دهد. این پارامتر در ابتدا مقدار پیشفرض  $99^\circ$  داشت و این یعنی تقریباً عامل کل مسیر آینده را نگاه می‌کرد تا ارزش حرکت خود را نشان دهد.

با توجه به تعریف نحوه زیاد و کم شدن سرعت و همچنین تعریف سرعت ورودی تابع محاسبه امتیاز



(ب) محل قرار گیری سنسورهای مجازی بر روی عامل

(آ)تابع امتیاز در زمان نزدیکی

شکل ۱۱-۳: تابع امتیاز نزدیکی و محل قرار گیری این تابع در فضای مشاهده

سرعت، در صورتی که عامل، سرعت خود را زیاد کند و بلافاصله سرعت خود را کم کند و دوباره همین سیاست را پیش بگیرد اتفاق بدی می‌افتد.

این اتفاق موجب آن خواهد شد که امتیاز اضافی و غیر واقعی دریافت کند. این موضوع در الگوریتم های یادگیری تقویتی زیاد مطرح نیست چون این الگوریتم ها به سمت بیشینه مقدار امتیاز در حرکت هستند. بنابراین گرچه امتیاز اضافی دریافت می‌کند اما باید یاد بگیرد که سرعت را کم نکند و به صورت مداوم آن را تا حد بیشینه افزایش دهد؛ اما این اتفاق نیفتاد.

اتفاقی که در واقعیت افتاد این بود که ماشین سرعت خود را به صورت نوسانی کم و زیاد می‌کند به طوری که روی سرعت ۲ واحد تقریباً سرعت خود را نگه می‌دارد.

با کمی تفکر این نتیجه بدیهی می‌شود. چرا که در این صورت بجای حدوداً  $70^\circ$  مرحله رفتن در هر اپیزود، آن اپیزود را در حدود  $25^\circ$  مرحله می‌گذراند (به خاطر سرعت بسیار پایین آن). و چون هم امتیاز اضافی دریافت می‌کند و هم بخاطر طولانی شدن مسیر، در مجموع امتیاز زیادی دریافت می‌کند. بنابراین سیاست بھینه خود را این گونه پیدا کرده بود.

برای جلوگیری از این پدیده سعی شد تا با کاهش مقدار  $\gamma$  به ازای مقادیر مختلف، دید روبه جلو را کاهش دهد و به دو و یا سه حرکت آینده محدود کند. مقادیر مختلف مورد آزمایش و شبیه سازی قرار گرفتند و در  $0.8^\circ$  مقدار آن بھینه شد.

بنابراین این پارامتر اهمیت فراوانی در جلوگیری از قرار گرفتن عامل در مسیر اشتباه دارد و با آن انتخاب از این گونه اشتباهات جلوگیری شده است.

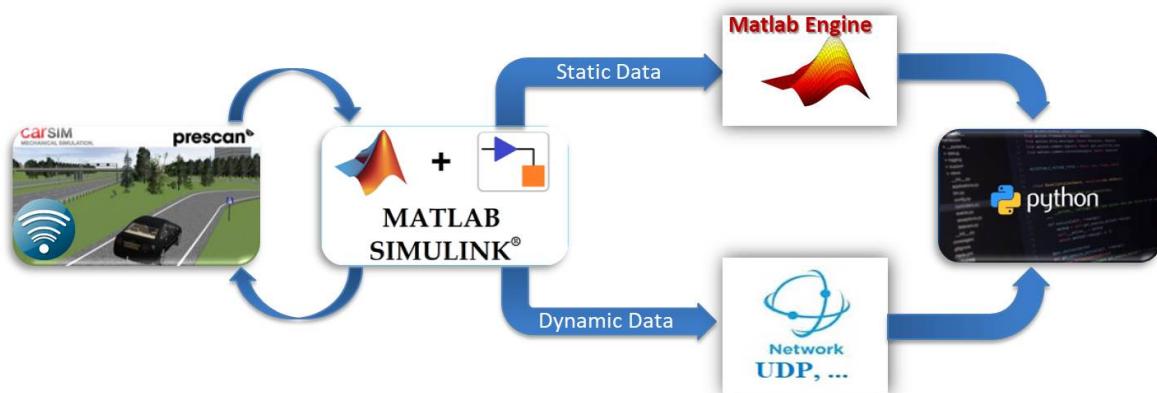
# فصل چهارم

## نحوه پیاده سازی

این بخش لایه هایی را مطرح می کند که جهت پیاده سازی پروژه از آن استفاده شده است. لایه ای که به الگوریتم مرتبط می شود در بخش ۳ به صورت کامل بررسی خواهد شد و در این قسمت نحوه کار کردن محیط شبیه سازی مطرح خواهد شد.

## ۱-۴ روش پیاده سازی

همان طور که گفته شد، در این پروژه از ابزار های مختلفی استفاده شده است. برخی ابزارات دیگر نیز جهت ایجاد اتصال بین آن ابزار ها استفاده شده اند. در این بخش، این اجزا به تفصیل بررسی خواهد شد. هر کدام از این اجزا کار مشخصی را بر عهده دارند. شکل ۱-۴ این ارتباط را نشان می دهد.



شکل ۱-۴: بلوک دیالگرام لایه های کلی

در شکل ۱-۴ از سمت چپ به راست اجزا یاد شده و نحوه ارتباط آنها با یکدیگر را به خوبی نشان می دهد. این بلاک ها و ارتباط ها عبارتند از:

- اولین بلاک آن، نرم افزار پری اسکن می باشد. وظیفه اصلی این نرم افزار، شبیه سازی دینامیک یک اتومبیل و یا موتور و ... می باشد. همچنین ایجاد یک محیط گرافیکی زیبا و یک پنل کاربری گرافیکی برای ساخت ماشین ها از دیگر حسن های این نرم افزار است.

فایل های مهم ایجاد شده توسط این بخش، pex. و pb. می باشد.

- بلاک بعدی ترکیبی از متلب و سیمولینک است. چرا که نرم افزار پری اسکن این امکان را دارد که برای کنترل و دسترسی بیشتر به قسمت های کنترلی مختلف، چیزی به نام API ارائه می دهد. این API یک فایل سیمولینک را در اختیار کابران قرار میدهد که در آن بلاک های مشخصی به یکدیگر متصل هستند و با مطالعه و تغییر آن بلاک ها می توان کنترل سیستم را به دست گرفت.

فایل های مهم این بخش نیز در فرمت `s1x`. و `m`. در دسترس هستند.

همچنین API یاد شده، دستورات دیگری را جهت دریافت داده های استاتیک محیط ساخته شده در این نرم افزار را به کاربران خویش در محیط مطلب می دهد.

- دو بلوک بعدی، مربوط به اتصال بین مطلب و یا سیمولینک با پایتون هستند.

بلوک بالایی این اتصال را بین داده های استاتیک شامل طول جاده و عرض هر لاین، موقعیت اولیه اتومبیل و جاده، و بسیاری اطلاعات دیگر که بسیاری از آن اطلاعات استفاده نشده اند زیرا در این پروژه مفید نبوده اند. این بلوک، فایل سیمولینک را تغییر نمی دهد.

بلوک پایینی نیز با استفاده از روش های شبکه کردن، می تواند داده های پویا را از محیط سیمولینک به پایتون منتقل کند. این داده های پویا عبارتند از موقعیت و سرعت و اطلاعات دیگری از اتومبیل در حال حرکت، اطلاعات سنسورها و ... باشد.

- بلوک بعدی پایتون است که خود شامل لایه های دیگری است که در شکل ۹-۴ به تفضیل بیان شده است. نکته جالب در آن این است که در آن لایه ها اثربخشی نیز از دو بلوک پیشین آمده است. همچنین بخش اصلی کار، یا به عبارتی مغز و هوش این کار در این قسمت توسعه یافته است.

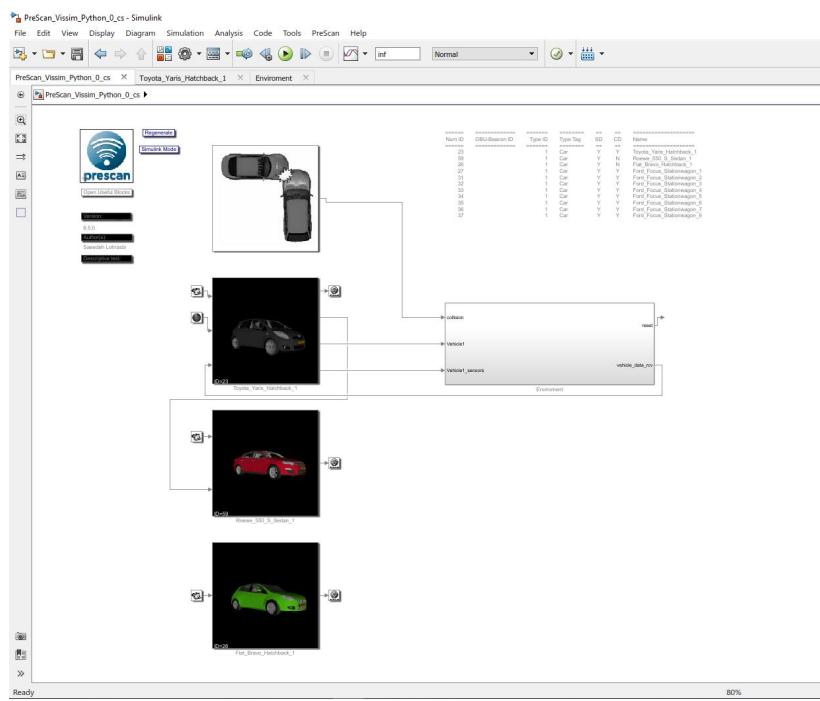
## ۲-۴ بررسی دقیق فایل سیمولینک

فایل سیمولینک ایجاد شده توسط نرم افزار پری اسکن، قابلیت تغییر به دست کاربر را دارد. شکل ۲-۴ فایل تغییریافته مربوط به این پروژه را نشان می دهد.

بلوک های سمت راست نشان داده شده در سمت راست شکل ۲-۴ توسط نرم افزار پری اسکن ایجاد شده است که البته دستخوش تغییراتی نیز بوده اند.

در صورتی که با استفاده از محیط گرافیکی GUI فایل `pex`.\*\* تغییر کند، فایل سیمولینک تغییر نمی کند. در برخی موارد این تغییرات ممکن است منجر به پیغام خطأ شود.

نکته ۴-۲-۱. در صورتی که فایل `pex`.\*\* تغییر کند، برای اعمال این تغییرات، باید روی کلمه Regenerate که در شکل ۳-۴ آمده است، کلیک کرد. با این کار، تغییرات جدید اعمال می شود بی آن که تغییرات کاربر تحت تاثیر قرار بگیرد.



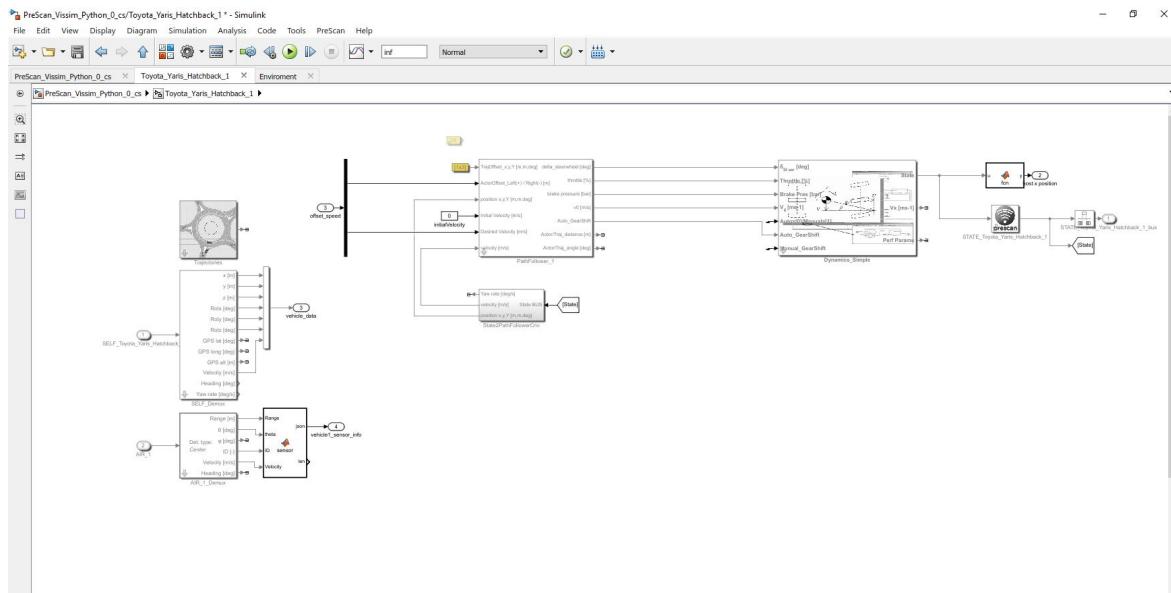
شکل ۲-۴: فایل سیمولینک ایجاد شده توسط نرم افزار پریاسکن همراه با تغییرات



شکل ۳-۴: روش صحیح اعمال تغییرات روی فایل سیمولینک

در شکل ۳-۴ همانطور که در نکته ۱-۲-۴ به آن اشاره شد، دکمه ای تحت عنوان Regenerate وجود دارد که استفاده از آن در همان نکته مشخص شده است. همچنین در این تصویر در زیر لوگوی برنامه پری اسکن، اطلاعاتی مانند شماره نرم افزار (که در اینجا ۸.۵.۰ می باشد)، نام نویسنده مشاهده می شود.

در شکل ۲-۴ اولین بلوک سمت راست همان ماشینی است که ما آن را تحت کنترل گرفته ایم. اگه به آن وارد شویم، شکل ۴-۴ را مشاهده می کنیم. در این تصویر ورودی و خروجی ها نقش خیلی مهمی دارند. این اطلاعات در جدول ۱-۴ آمده اند.



شکل ۴-۴: فایل سیمولینک - شبیه سازی اتومبیل

## ۱-۲-۴ معرفی بلوک Environment و بررسی جزئیات آن

در جدول ۱-۴ صرفا ورودی خروجی های مهم مورد بررسی قرار گرفته است. این ورودی ها و خروجی های مهم به یک بلوک دیگر منتقل می شود. بلوک Environment همان بلوک است که در شکل ۲-۴ نیز مشخص است و در شکل ۵-۴ نیز از زاویه نزدیک تر با جزئیات بیشتر می توان آن را مشاهده نمود. وظیفه اصلی بلوک Environment جمع آوری اطلاعات محیط سیمولینک و همچنین ارسال دستورات کنترلی به آن هاست. اطلاعات جمع آوری شده از محیط سیمولینک شامل موارد زیر است.

- اطلاعات ماشینی که نقش «عامل» را در الگوریتم دارد.

<sup>14</sup>json

۲بعدا خواهید دید که مشخص بودن طول بسیار اهمیت دارد!

نوع	عنوان	بلوک مربوط	توضیحات
خروجی	اطلاعات ماشین	SELF_Demux	اطلاعات ماشین، شامل اطلاعات موقعیت( $y, x$ ) و اطلاعات چرخش (حول $y, x$ و $z$ ) به همراه سرعت ماشین را خروجی می دهد. این ۷ داده قبل از خروجی توسط یک mux با یکدیگر ادغام می شوند.
خروجی	اطلاعات سنسور ماشین	AIR_1_Demux	اطلاعات سنسور V2C را خروجی می دهد. از آنجا که این سنسور فاصله و زاویه و ... تا ده ماشین نزدیک خود را می دهد. بنابراین هریک از این اطلاعات یک بردار ده تایی است. برای فرستادن آن اطلاعات به خروجی، ابتدا آنها را به طریقی به فرمت جیسون تبدیل می کند و یک رشته کاراکتر با طول مشخص را خروجی می دهد.
ورودی	کنترل لاین و سرعت ماشین	PathFollower_1	دستورات کنترلی اتومبیل، شامل کدام خط بودن و مقدار سرعت نهایی، به صورت ورودی وارد یک demux می شود و پس از جدا سازی، به بلوک مربوط متصل می شود.

جدول ۴-۴: بررسی ورودی ها و خروجی های مهم در شکل ۴-۴

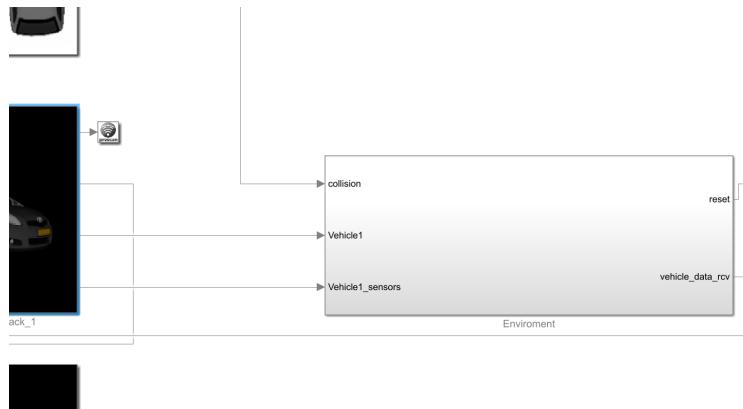
- اطلاعات سنسور های ماشین «عامل»

- اطلاعات مربوط به تصادف کردن و کدوم ماشین با کدوم ماشین تصادف کرده است.

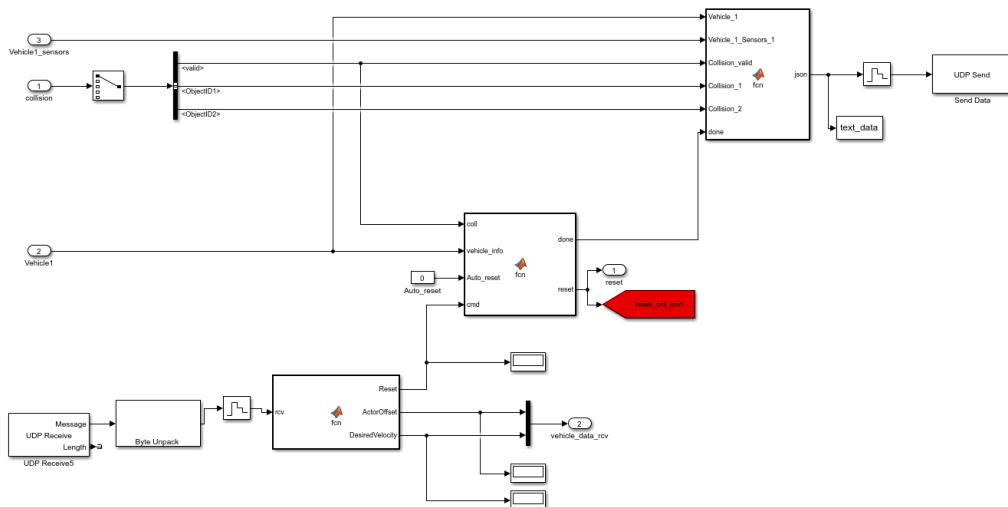
این اطلاعات به صورت ورودی به این بلوک وارد می شود و دستورات کنترلی شامل کنترل لاین ماشین به همراه مقدار سرعت نهایی ماشینی که نقش «عامل» در الگوریتم دارد، از این بلوک خارج می شود.

پیشتر صحبت شد که وظیفه تصمیم گیری بر عهده کد پایتون است. با این حساب سیمولینک قدرت تشخیص و تصمیم گیری ندارد. از این رو وظیفه بلوک Environment نیز تصمیم گیری نمی باشد بلکه با تکنیک هایی سعی بر برقراری ارتباط با کد پایتون دارد. در حقیقت این بلوک نقش واسط بین سیمولینک و پایتون را دارد. این بلوک علاوه بر دستورات کنترلی لاین و سرعت، دستور «شروع کردن دوباره» و یا ریست را نیز دریافت می کند. با این دستور تمامی اطلاعات به حالت اول بر خواهد گشت و ماشین «عامل» به جای اول بر خواهد گشت و منتظر دستورات جدید می ماند.

شکل ۱-۴ توضیح بسیار کلی در مورد این نحوه ارتباط نشان داده است و در شکل ۶-۴ جزئیات



شکل ۴-۵: نگاهی از نزدیک به بلوک Environment



شکل ۴-۶: فایل سیمولینک - داخل بلوک Environment

بیشتری را در مورد داخل این بلوک را نشان می‌دهد.

اگر به شکل ۴-۶ دقت شود دو بلوک UDP Send در بالا سمت راست و بلوک UDP Receive در پایین سمت چپ شکل ۶-۴) این دو بلوک برای فرستادن داده های مورد نیاز به پایتون و گرفتن دستور های کنترلی از پایتون در سیمولینک تعبیه شده اند.

یادداشت ۲-۲-۴. اگر نمودار شکل ۱-۴ در نظر داشته باشیم، خواهیم یافت که این بلوک در شاخه پایینی ارتباط بین سیمولینک و پایتون قرارا دارد که از روش های شبکه کردن بین این دو انجام می شود. این بلوک صرفا داده های پویا را از مطلب به سیمولینک انتقال می دهد و با داده های استاتیک کاری ندارد.

در جدول ۲-۴ اطلاعات دقیق شبکه برای این دو بلوک دیده می شود. گفتنی است که این داده ها

نام بلوک	نوع بلوک	IP	Port
UDP Receiver	گیرنده	localhost	۸۰۷۰
Send Data	فرستنده	localhost	۸۰۳۱

جدول ۴-۲: اطلاعات بلوک های فرستنده‌گی گیرنده‌گی در سیمولینک

به نحوه مناسب کد شده‌اند تا تنها با استفاده از این دو بلوک بتوان داده‌ها را منتقل کرد. روش کد شدن<sup>۳</sup> قبل از ارسال و یا دریافت در این دو بلوک کاملاً با یکدیگر متفاوت هستند. این کار توسط بلوک‌های قبل و بعد دو بلوک مذکور انجام می‌شود. در شکل ۶-۴ سه بلوک تابع متلب<sup>۴</sup> وجود دارد که از مهم‌ترین نقش را دارند. این نقش‌ها در ادامه توضیح توضیح مفصل داده شده‌اند.

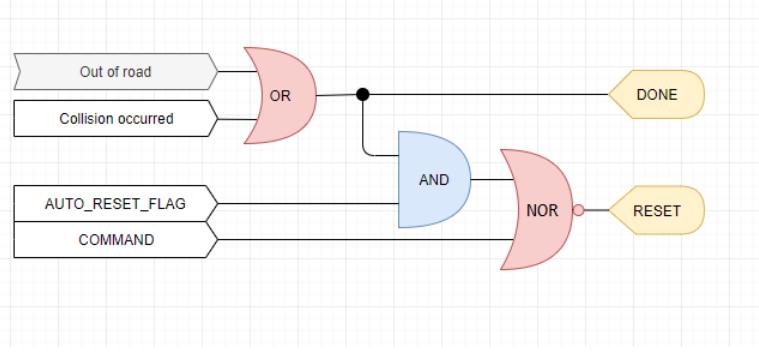
خلاصه این وظایف در زیر آمده است:

- **بلوک فرستنده**: این بلوک در سمت بالا سمت راست تمام اطلاعاتی که به بلوک کلی Environment وارد می‌شود را به نحوی مناسب به همراه متغیر done که از یک تابع متلب دیگر خارج می‌شود را برای پایتون طی ساختار مشخصی (جیسون) ارسال می‌کند. موضوع به صورت بسیط در بخش ۲-۴ مورد بررسی قرار گرفته است.

- **بلوک گیرنده** : این بلوک تنها وظیفه آن این است که داده‌هایی که پس از بلوک Byte Unpacking آمده است، را انتخاب می‌کند که هر یک از این دیتا‌ها چه مفهومی هستند. از آن جا که ۳ داده کنترلی (لاین و سرعت و ریست) از طرف پایتون ارسال می‌شود، بلوک Byte Unpacking وظیفه این ۳ داده را به فرمت double در می‌آورد (این فرمت در سمت پایتون نیز به همین شکل قرار داد شده‌است). بنابراین این بلوک Byte Unpacking است که کار اصلی را انجام Byte Unpacking می‌دهد و تنها وظیفه این بلوک تفکیک و اسم گذاری بر روی خروجی بلوک Byte Unpacking است.

- **بلوک محاسبه تمام شدن و ریست کردن** : این بلوک با دو خروجی مهم می‌دهد. یکی از آن خروجی‌ها done است که توسط بلوک فرستنده نیز برای پایتون ارسال می‌شود. و مقدار دیگری

<sup>3</sup>Encoding<sup>4</sup>Matlab-function block



شکل ۷-۴: منطق محاسبه تمام شدن و دستور ریست کردن

را که خروجی می‌دهد دستور ریست کردن است. این دستور در لبه مثبت (از صفر به یک برسد) ریست می‌کند. منطق این بلوک یک منطق باینری است که در شکل ۷-۴ روش محاسبه آن نیز آمده است. همچنین این بلوک یک مقدار به اسم `Auto_reset_flag` نیز به عنوان ورودی دریافت می‌کند. این مقدار که می‌تواند صفر و یا یک باشد، تصمیم می‌گیرد که در صورتی که `done = 1` شد آیا ریست کند و یا ریست نکند. اگر یک باشد ریست می‌کند و گرنه منتظر رسیدن دستور از پایتون می‌ماند.

## ۲-۲-۴ بررسی ساختار داده‌های ارسالی و کد آن در سیمولینک

بلوک فرستنده در شکل ۶-۴ اطلاعات دریافت شده را طی ساختار مشخصی به ساختار جیسون در می‌آورد. نمونه ای از این ساختار در شکل ۱-۴ آمده است.

شکل ۸-۴(ب) همان ساختاری است که پیشتر در مورد آن صحبت شد. با توجه به این که این ساختار (در ادامه خواهید دید که) در اطلاعات سنسور ماشین به کار گرفته می‌شود و هر سنسور ممکن است اطلاعات خاص خود را داشته باشد. برای یکسان‌سازی این اطلاعات، در اولین لایه دو گزینه `data` و `name` را به طور قراردادی بین همه سنسورها یکسان است تا در کد پایتون بتوان با توجه به اسم آنها ادامه ساختار که در `data` می‌باشد، قابل تشخیص باشد. اما از آنجا که صرفاً از اطلاعات یک سنسور استفاده شده است این شکل ساختاردهی تنها یک گزینه برای توسعه‌های آتی آن است.

در این ساختار ۴ اطلاعات نشان داده شده که هر یک از آن اطلاعات خود یک بردار ۱۰ تایی است، برای پایتون ارسال می‌شود.

<sup>۶</sup> این عکس با استفاده از سایت <http://jsonviewer.stack.hu/> تهییه شده است.



(ج) شکل ادغام شده

(ب) جیسون سنسور

(آ) اطلاعات خام جیسون کلی

شکل ۴-۴: تصویر (آ) نحوه ساختاردهی کلی در بلوک تابع متلب واقع در زیرسیستم Environment را نشان می‌دهد. در این قسمت اطلاعات ماشین شامل دو قسمت data و Sensors است. اطلاعات سنسور های این ماشین از تصویر (ب) تامین می‌شود. این ساختار در همان زیرسیستم ماشین تبدیل به جیسون شده و برای زیرسیستم Environmnet ارسال می‌شود. قسمت سنسور یک آرایه است تا بتوان چندین سنسور مختلف را برای آن ارسال کرد. بنابراین ساختاری مانند تصویر (ج) باید را در نهایت برای پایتون ارسال می‌کند.<sup>۶</sup>

شکل ۴-۴(آ) ساختار کلی است که محیط Environmnet از آن استفاده می‌کند. در این بلوک علاوه بر داده‌هایی مانند اطلاعات ماشین و سنسورش، اطلاعات تصادف و اطلاعات تمام شدن و یا نشدن شبیه سازی که پیش‌تر در مورد آن اطلاعات صحبت شد، دو اطلاعات اضافه دیگر نیز می‌فرستد.

• Time : زمان شبیه سازی را همراه با دیتا دیگر ارسال می‌کند و به اصلاح یک ساختار زمانی ایجاد می‌شود.<sup>۷</sup>

• Object : این عبارت کمک به کد پایتون می‌کند که ماشین هدف و یا عامل را از بین ماشین های موجود در لیست Vehicles بیابد.<sup>۸</sup>

در نهایت با ادغام شدن اطلاعات سنسور نیز، ساختار کلی به شکل ۴-۴(ج) در خواهد آمد. در بخش

۴-۴ از برخی از چالش‌های ساختاردهی کردن به این شکل، صحبت خواهد شد. همچنین خروجی نهایی

<sup>7</sup>Time-struct

از آنجایی که در این پروژه لیست Vehicles یک آجکت بیشتر ندارد، پس این بخش نیز صرفاً ظرفیت توسعه‌پذیری این کد را بالا می‌برد.



شکل ۴-۹: بلوك دیالگرام لایه های پایتون

آن نیز را می توانید در همان بخش بباید.

### ۳-۴ بررسی جزئیات بخش پایتون

بارها اشاره شد که پایتون بخش اصل تصمیم‌گیری را برعهده دارد. در بخش ۲۴ سعی شد تا به نحو مناسبی دیتای محیط شبیه‌سازی را به پایتون منتقل کند و دستورات کنترلی را نیز از پایتون به محیط شبیه‌سازی ارسال کند. در این بخش بر روی مفاهیم پایتونی آن مانور می‌دهیم.

از آن جا که کد پایتون باید به سیمولینک متصل شود و دیتا را به الگوریتم کنترلی خود (که از الگوریتم های یادگیری تقویتی استفاده شده است)، برساند و از آن جا دستورات را دریافت کند و به سیمولینک برساند، نیازمند لایه هایی است که هر لایه بخشی از این کارها انجام دهد.

### ۱-۳-۴ معرفی لایه های کد پایتون

کد پایتون از لایه های مختلف تشکیل شده است. از هر لایه به لایه بعدی سطح زبان بالاتر می‌رود. این لایه ها در شکل ۱-۳-۴ مشخص شده اند. در لایه های ابتدایی، سطح استفاده از دستورات بسیار ابتدایی است و در هر لایه با تعریف توابع و کلاس هایی این امکان را ایجاد کرده اند که بدون در نظر گرفتن این که در سطوح پایین‌تر چه اتفاقاتی می‌افتد، در سطوح بالاتر از آن امکانات استفاده کرد. در ادامه این بحث مفصل توضیح داده خواهد شد.

در شکل ۴-۹ لایه برنامه نویسی شده با ۵ عدد نشان داده شده است. توضیحات زیر متناسب با

هریک از این شماره ها در نظر گرفته شده اند:

**Model ۱**: در این لایه، با استفاده از موتور متلب با مطلب و سیمولینک ارتباط برقرار می کند و با این ارتباط دو وظیفه بسیار مهم را انجام می دهد.

- آ) ساخت مدل هایی مانند اتومبیل و جاده و دریافت اطلاعات ضروری آن ها در متلب
- ب) تعریف کلاس sim و ارسال دستوراتی مانند شروع کردن، توقف کردن، مکث کردن، ادامه دادن و ... به سیمولینک.

این لایه صرفا از اطلاعات استاتیک سیمولینک استفاده می کند.

**Environment ۲**: این لایه برای ارتباط با زیرسیستم Environment که پیشتر آنرا در شکل ۴-۵ مشاهده کردید، ساخته شده است. در این لایه با استفاده از ابزار هایی که لایه Model در اختیار آن قرار می دهد، به سادگی آبجکت هایی ماند ماشین و جاده را می سازد و با استفاده از اطلاعات شبکه که در جدول ۴-۲ آمده است، با سیمولینک ارتباط برقرار می کند و آن اطلاعات پویا را دریافت می کند و دستورات کنترلی را برای آن ارسال می کند.

این لایه از موتور متلب استفاده نمی کند بلکه از روش های شبکه ای استفاده می کند.

**Env ۳**: این لایه یک آبجکت از کلاس Environment را دریافت می کند و تمامی نیاز های خود را در مورد هر مسیله ای که به ارتباط با محیط متلب و یا سیمولینک به آن واگذار می کند و تمرکز آن بر روی مفاهیم یادگیری تقویتی مانند (۱) فضای مشاهده (۲) فضای حرکت (۳) تعریف حرکت (۴) تعریف حالت (۵) تعریف امتیاز و ... می باشد.

**gym ۴**: در نهایت یک مجموعه داریم که هر یک وظایف مربوط به خود را دارند. در این قسمت کد ما که در لایه Env به gym نزدیک شده است، با فراخوانی لایه های پیشین به ترتیب از بالا به پایین (از نظر سطح) فراخوانی می شود و در هر لایه آبجکت هایی از کلاس های موجود در لایه پایین تر فراخوانی می شود. از این رو جهت پیکان بر عکس است.

در مورد gym در فصل ۴-۲ به صورت مفصل بحث شده است.

**Algorithm ۵**: در لایه های پیشین سینتکس کد gym ایجاد شده است و این لایه بر توسعه الگوریتم های یادگیری تقویتی که در فصل ۲ مورد بررسی مفصل قرار گرفته است، تمرکز دارد.

می توان گفت این لایه نقش ریسیس و یا مغز کار را دارد و باقی لایه ها کارگرانی هستند که وظیفه دارند که اطلاعات را به شکلی مناسب این لایه، برای آن منتقل کنند.

## ۴-۴ بررسی دقیق تر برخی چالش های فنی پروژه

یکی از مهمترین چالش های این پروژه برقرار ارتباط محیط شبیه سازی سیمولینک با پایتون بود. با استفاده از بلوک UDP Send می توان یک یا چند داده هم نوع را منتقل کرد. این کار از لحاظ فنی، موضوع انتقال را دشوار کرده بود. زیرا یا باید برای هر دیتا، یک بلوک فرستنده قرار داد و یا باید همه دیتا ها را به یک نوع تبدیل کرد و سپس اطلاعات به ترتیب خاصی برای فرستنده فرستاد.

روش دوم دو تا مشکل دارد یکی آن که به شدت به ترتیب قرار گیری دیتا وابسته خواهد شد و دیباگ آن بسیار سخت می شود. دوم آنکه از دیتای ارسالی هیچ مفهومی را نمی توان بدون داشتن آن ترتیب استخراج نمود و به عبارت دیگر اصلاً ماژولار نخواهد بود و کد بسیار نامفهوم و سخت خواهد شد و در صورت توسعه بخش فنی عملکار از اول شروع می شود.

برای حل این مشکلات بلوکی به نام Environment در نظر گرفته شد که هرگونه ارتباط با پایتون توسط این بلوک رخ می دهد. سپس در قسمت فرستنده یک تابع پیش پردازنده تعییه شد تا داده ها را جمع آوری و ساختار دهی کند. ساختار استفاده شده json است که در شکل ۸-۴ معرفی شد. این ساختار، کد را به شدت ماژولار می کند.

نکته خیلی مهم در استفاده از این ساختار این است که این ساختار به شکل رشته می باشد. از آنجایی که در سیمولینک باید ابعاد ماتریس ورودی و خروجی مشخص باشد و از آنجایی که متلب رشته را مانند یک بردار با طول متغیر می بیند بنابراین محیط سیمولینک به محض آن که خروجی و یا ورودی داده ای از جنس کاراکتر باشد، پیغام خطای دهد.

روشی که برای دور زدن این مورد به کار رفت، به شرح زیر است.

۱. طول ساختار جیسون را بر حسب کاراکتر ثابت کشت. برای این کار لازم شد که طول اعداد نیز ثابت شود.

۲. خروجی به uint8 تبدیل شد. زیرا دیگر جنس آن کاراکتر نیست ولی از لحاظ حافظه دقیقاً برابر می باشد.

خروجی تابع پیش پردازنده به صورت زیر خواهد بود.

در این ساختار اعداد هر محدوده‌ای که داشته باشد با طول و دقیق مشخص در کنار یک دیگر قرار می‌گیرند. بنابراین این مشکل نیز حل می‌شود.

## فصل پنجم

# شبیه سازی و نتایج

نیاز به موتور متلب	توضیحات	نام مخزن
✓	پروژه کامل در این مخزن قرار دارد. در آن بر بخش الگوریتم از کتابخانه هایی استفاده شده است که بر روی سیستم عامل لینوکس قابل اجرا هستند بنابراین در صورتی که از لایه الگوریتمی که این پروژه از آن استفاده می کند استفاده نشود، در هر سیستم عاملی می توان از آن استفاده کرد.	gym-Prescan
✗	در اینجا بخش فایل های پری اسکن به همراه فایل هایی که به موتور متلب نیاز دارد حذف شده است.	gym-Prescan-minimal

جدول ۱-۵: اطلاعات مخزن های پروژه در گیت هاب

در فصل های گذشته در خصوص ابزار هایی که در این پروژه استفاده شده اند، صحبت شد و توضیح مفصلی بر چیستی آن ابزار ها و ضرورت استفاده از آنها داده شد. اما سوالات بی جوابی نیز ماند که در این فصل به آنها خواهیم پرداخت.

یکی از آن سوالات نحوه راه اندازی کد پروژه می باشد و سوال دیگر نتیجه حاصل از شبیه سازی نهایی چگونه است می باشد.

## ۱-۵ راه اندازی

این کد در گیت هاب در دو مخزن<sup>۱</sup> قرار دارد. جدول ۱-۵ اطلاعات این مخزن ها<sup>۲</sup> را نشان می دهد.<sup>۳</sup> ابتدا پیش نیاز های لازم را که در بخش ۴-۲ توضیح داده شدند، را نصب کنید. پیشنهاد می شود که تمامی نسخه های لازم توضیح داده شده در بخش ۴-۲ را در سیستم عامل لینوکس استفاده کنید. اگر به توصیه استفاده از لینوکس عمل نکرده باشید، می توانید با استفاده شبکه که کد در اختیارتان قرار می دهد کد را روی دو کامپیوتر ران کنید به طوری که در یک کامپیوتر ویندوز و نرم افزار های گفته شده نصب باشد و روی دیگری لینوکس و پایتون و پیش نیاز های پایتون که در بخش ۵-۲ بررسی شدند

<sup>1</sup> Repository

<sup>2</sup> غیر از این دو مخزن، مخزن دیگری به آدرس زیر وجود دارد که تاریخ چه و روند رسیدن به این نتیجه نهایی را نشان می دهد. [https://github.com/MohammadRaziei/Prescan\\_test](https://github.com/MohammadRaziei/Prescan_test)  
<sup>3</sup> آدرس این مخزن به این شکل بدست می آید: <https://github.com/mohammadraziei/<REPOSITORY-NAME>>

نصب باشد.

**یادداشت ۱-۵.** بهتر است این دو کامپیوتر در یک شبکه داخلی به یک دیگر متصل شده باشند. حال وارد کامپیوتری شوید که ویندوز بر روی آن نصب است. این کامپیوتر قرار است نقش محیط را برای ما ایجاد کند.

**یادداشت ۲-۵.** کدهای پایتون صرفا بر روی کامپیوتری که لینوکس دارد اجرا کنید.  
که دستور زیر را در ترمینال<sup>۴</sup> خود وارد کنید.

```
1 git clone https://github.com/MohammadRaziei/gym-Prescan.git
2 pip install -e gym-Prescan
```

خط دوم این کد، اختیاری می‌باشد و صرفا کار را ساده می‌کند. همچنین می‌توان آن را به صورت زیر نیز نوشت:

```
1 pip install git+https://github.com/MohammadRaziei/gym-Prescan
```

سپس وارد مسیر زیر شوید.

gym-Prescan/gym\_prescan/envs/PreScan

سپس با استفاده از آیکون<sup>۵</sup> کلیک کنید. در این صورت بروی نوار Toolbar این آیکون نیز ظاهر می‌شود. با فشردن آن، پنجره شکل ۶-۲ باز می‌شود. بروی Matlab کلیک کنید تا محیط متلب باز شود. اچرای فایل startup.m برای هنگامی که از کد پایتون بر روی همان سیستم استفاده نمی‌کنید، اختیاری است.

فایل سیمولینک را باز کنید و به صورت دستی IP بلوک فرستنده را به IP مورد نظر تغییر دهید و یا با استفاده از کد زیر در کامندهاین متلب تغییرات لازم را انجام دهید.

```
1 ExperimentName = 'PreScan_Vissim_Python_0';
2 send_ip = 'localhost';
3 sys = load_system([ExperimentName '_cs']);
4 set_param([ExperimentName '_cs/Environment/Send Data'],
           'remoteURL',[''' send_ip '''']);
5 save_system(sys);
```

<sup>4</sup>Terminal

<sup>5</sup> این آیکون پس از نصب نرم‌افزار پریاسکن بر روی دستکتاب تشکیل می‌شود.

در این کد کافیست مقدار send\_ip را مطابق با IP کامپیوتر دیگر تنظیم کنید. پس از تنظیمات فایل سیمولینک را اجرا کنید برای این کار می‌توانید آن را باز کرده و اجرا کنید و یا با استفاده از دستور زیر در کامندلاین مطلب آن را انجام دهید.

```
1 ExperimentName = 'PreScan_Vissim_Python_0';
2 sys = load_system([ExperimentName '_cs']);
3 set_param(bdroot, 'SimulationCommand', 'start');
```

حال در سیستم لینکوس خود می‌توانید بسته زیر را دانلود کنید.

```
1 git clone https://github.com/MohammadRaziei/gym-Prescan-minimal.git
2 cd gym-Prescan-minimal
```

در این پوشه تعدادی از الگوریتم‌های معروفی که در حوزه یادگیری تقویتی عمیق (DRL<sup>6</sup>) نوشته شده است، قرار دارد. در بین این الگوریتم‌ها دو الگوریتم DQN و A2C نسبت به بقیه بهتر جواب داده اند. این الگوریتم‌ها در بخش ۲ و در [۷] توضیح کامل داده شده‌اند.

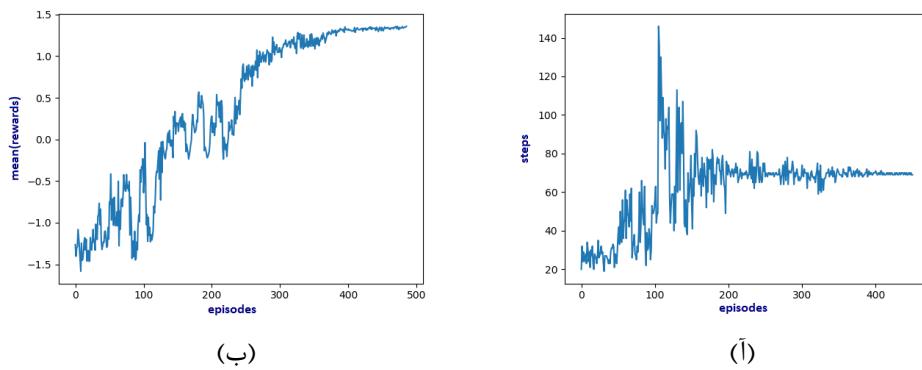
برای اجرای الگوریتم DQN باید ابتدا IP کامپیوتر ویندوزی را در قسمت مشخص در متغیر env\_dict که در جدول ۱-۳ به طور کامل بررسی شده است، بنویسید و سپس دستور python dqn.py را در ترمینال لینوکس وارد کنید. بدین صورت محیط شبیه‌سازی روی هر دو کامپیوتر شروع به کار می‌کند.

## ۲-۵ نتایج شبیه‌سازی

الگوریتم استفاده شده در این پروژه DQN می‌باشد. مقدار  $\gamma$  در این الگوریتم  $0.8$  انتخاب شده است. یا توجه به حالت و امتیاز‌های تعریف شده در فصل ۲ پس از  $400$  بار تلاش به سیاست بهینه<sup>7</sup> می‌رسد. اما برای بالا رفتن دقت،  $25000$  بار عامل مسیر را طی کرده است. دلیل آن بالا رفتن تعداد اپیزودها می‌باشد. با این کار مقدار  $\epsilon$  کاهش یافته و احتمال جست و جوی سیاست جدید آن کاهش می‌یابد. نمودار شکل ۱-۵(ب)، میانگین بدون وزن امتیاز در پایان هر اپیزود می‌باشد. در حوالی تا قبل از حدود  $200$  میانگین امتیاز‌ها منفی است. زیرا تا قبل از آن، تصادف رخ می‌داد و در کل منفی بودش. شبیه تقریباً صعودی آن در این مرحله بخاطر افزایش مرحله‌ها می‌باشد که تاثیر منفی تصادف را کاهش داده است. در کل مشاهده می‌شود که روندی صعودی تا رسیدن به مقدار بیشینه امتیاز دارد.

<sup>6</sup>Deep Reinforcement Learning

<sup>7</sup>Optimal Policy



شکل ۵-۵: نمودار توصیف رفتار عامل در هر اپیزود

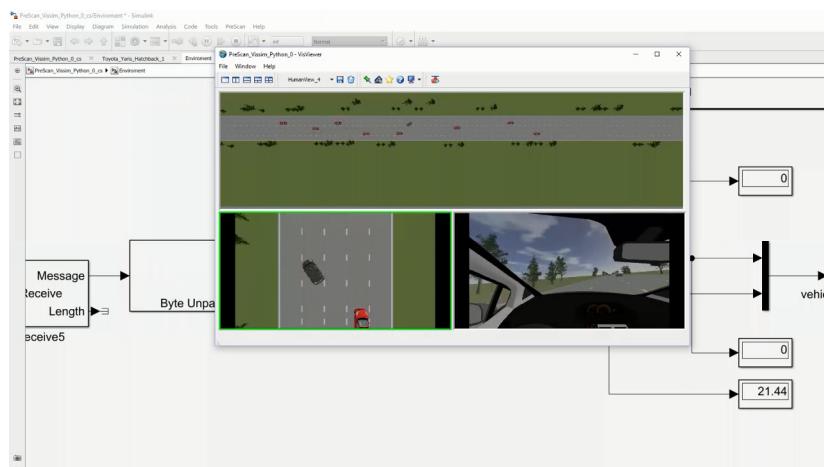
نمودار ۵-۵(آ)، تعداد مراحل را در هر اپیزود نشان می‌دهد. انتظار آن است که در ابتدا به دلیل تصادف کم باشد و در انتهای روی مقدار مشخصی ثابت شود. حال ممکن است در این بین اتفاقاتی نیز افتاده باشد. یکی از اتفاقات جالب کاهش مقدار قله در شکل ۵-۵(آ) می‌باشد. این مقدار با تنظیم مناسب پارامتر  $\gamma$  کنترل شده است و مشاهده می‌شود که یک قله بیشتر ندارد و این یعنی آن سیاست را ادامه نداده است.

در کل پس یافتن سیاست بهینه، تعداد مرحله‌ها به ۷۳ رسید. سرعت عامل حول ۲۱ در حال نوسان است و میانگین بدون وزن امتیازها برابر  $1/33821$  می‌باشد. از آنجایی که نتایج این پروژه به صورت تصویر قابل بیان نیستند و نمی‌توان حتی چندین تصویر مختلف از وضعیت‌های مختلف آن گرفت و به عنوان نتیجه منتشر کرد، از نتایج این پروژه، فیلمی تهیه شده است که در آدرس زیر بارگذاری شده است.

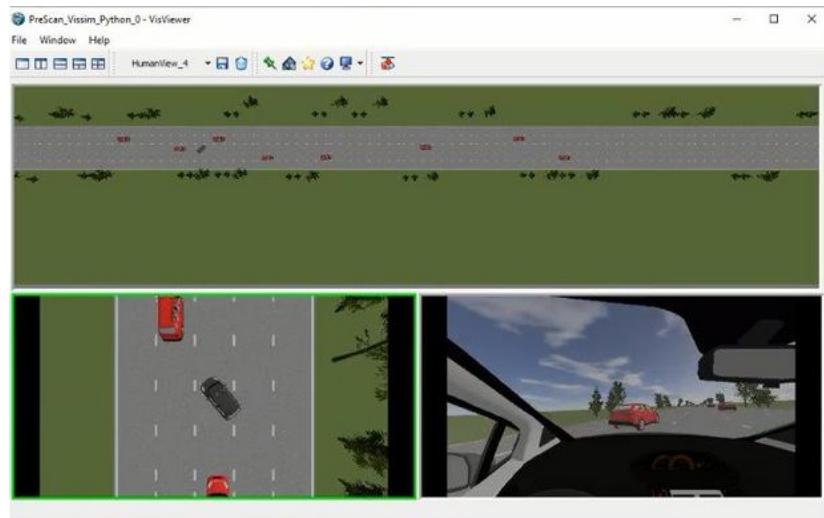
<https://youtu.be/chgpBWU9XCA>

همچنین فیلم دیگری نیز از روند اجرای این الگوریتم در سه بازه زمانی آماده شده است که آدرس آن نیز در زیر قرار دارد.

<https://youtu.be/JrDWWDWqv1Hg>



(ا)



(ب)

شکل ۲-۵: شبیه سازی نهایی، شکلی مانند این دارد. از این رو فیلمی از این محیط در حال اجرا تهیه شده است که در آدرس <https://youtu.be/chgpBWU9XCA> بارگذاری شده است.

# مراجع و مراجع

- [1] D. Silver, “UCL course on RL.” <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>, 2015.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.
- [3] G. Hayes, “How to install openai gym in a windows environment.” <https://medium.com/p/338969e24d30>, 2018.
- [4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.
- [5] A. Hill, “Stable baselines: a fork of openai baselines — reinforcement learning made easy.” <https://stable-baselines.readthedocs.io/en/master/>.
- [6] A. Hill, “Stable baselines: a fork of openai baselines — reinforcement learning made easy.” <https://medium.com/p/df87c4b2fc82/>.
- [7] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Stable baselines.” <https://github.com/hill-a/stable-baselines>, 2018.

# نمايه

مخزن، ۲۷	حرکت، ۲۱، ۸۶
امتیاز، ۲۰، ۷۵، ۲۲	عامل، ۲۰، ۱۷، ۹۶
فرضیه امتیازها، ۵	حالت عامل، ۸، ۲۰
یادگیری تقویتی، ۶۴، ۱۶، ۲۰، ۲۲	الگوریتم، ۱۶، ۱۷
یادگیری شبه ناظر، ۴	ماشین خودران، ۲۰
حالت، ۲۲، ۸۶	نقطه تراز، ۱۶
مرحله، ۶، ۲۱	خوشه بندی، ۴
ناظر، ۵	هزینه، ۵
یادگیری با ناظر، ۴	یادگیری تقویتی عمیق، ۱۷، ۲۸
ترمینال، ۲۷	محیط، ۷، ۶، ۱۶، ۹، ۱۷، ۲۰
تست، ۲۲، ۲۳	حالت محیط، ۷، ۸، ۲۰
آموزش، ۲۲	اپیزوود، ۲۱
یادگیری بدون ناظر، ۴	فیدبک، ۵
	تاریخچه، ۷، ۸
	حالت اطلاعاتی، ۹
	مارکوف، ۸
	حالت مارکوف، ۸، ۹
	متلب، ۱۷
	یادگیری ماشین، ۴
	مشاهده، ۶، ۷
	مشاهده پذیری، ۸، ۲۰
	پریاسکن، ۲۸

# فهرست اختصارات

## A

A2C ..... Synchronous Actor Critics

## D

DQN ..... Deep Q-Learning Network

DRL ..... Deep Reinforcement Learning

## R

RL ..... Reinforcement Learning

# واژه‌نامه انگلیسی به فارسی

## E

Environment .....	محیط .....
Environmnet State .....	حالت محیط .....
Episode .....	اپیزود .....

## F

Feedback .....	بازخورد .....
Full observability .....	مشاهده‌پذیری کامل .....

## H

History .....	تاریخچه .....
---------------	---------------

## I

Information State .....	حالت اطلاعاتی .....
-------------------------	---------------------

## J

Joystick .....	دسته‌بازی .....
----------------	-----------------

## M

Machine Learning .....	یادگیری ماشین .....
------------------------	---------------------

## A

Action .....	حرکت .....
Action Space .....	فضای حرکت .....
Agent .....	عامل .....
Agent State .....	حالت عامل .....
Algorithm .....	الگوریتم .....
Anaconda .....	نرم‌افزار آناکوندا .....
API .....	رابط برنامه‌نویسی برنامه .....
Autonomous Vehicle .....	ماشین خودران .....

## B

Bechmark .....	نقطه تراز .....
----------------	-----------------

## C

Clustering .....	خوشه بندی .....
Cost .....	هزینه .....

## D

Deep .....	یادگیری تقویتی عمیق .....
	Reinforcement Learning
Dynamic Data .....	داده‌های پویا .....

Simulink .....	سیمولینک .....	Markov .....	مارکوف .....
State .....	حالت .....	Markov Decision	رونده تصمیم‌گیری مارکوف
Step .....	مرحله .....	Process	Process
Supervised Learning .....	یادگیری با ناظر .....	Markov State .....	حالت مارکوف .....
Supervisor .....	ناظر .....	Matlab .....	نرم‌افزار متلب .....
		Matlab Engine .....	موتور متلب .....

## T

Terminal .....	ترمینال .....
Test .....	تست .....
Timeout .....	سقف زمانی .....
Train .....	آموزش .....

## O

Observability .....	مشاهده‌پذیری .....
Observation .....	مشاهده .....
Observation Space .....	فضای مشاهده .....
Optimal Policy .....	سیاست بهینه .....

## U

Unsupervised Learning .....	یادگیری بدون ناظر .....
-----------------------------	-------------------------

## P

Partial observability .....	مشاهده‌پذیری جزئی .....
Policy .....	سیاست .....
PreScan .....	نرم‌افزار پرسن .....

## R

Reinforcement Learning .....	یادگیری تقویتی .....
Repository .....	مخزن (گیت‌هاب) .....
Reward .....	امتیاز .....
Reward Hypothesis .....	فرضیه امتیازها .....

## S

Semi-Supervised Learning .....	یادگیری شبه ناظر .....
	Learning

# واژه‌نامه فارسی به انگلیسی

خ

Clustering ..... خوشبندی

Train ..... آموزش

Episode ..... اپیزود

Algorithm ..... الگوریتم

Reward ..... امتیاز

د

Dynamic Data ..... داده‌های پویا

Feedback ..... بازخورد

ر

API ..... رابط برنامه‌نویسی برنامه

ت

History ..... تاریخچه

Terminal ..... ترمینال

Test ..... تست

س

Timeout ..... سقف زمانی

State ..... حالت

Policy ..... سیاست

Information State ..... حالت اطلاعاتی

Optimal Policy ..... سیاست بهینه

Agent State ..... حالت عامل

Simulink ..... سیمولینک

Markov State ..... حالت مارکوف

ح

Agent ..... عامل

Environment State ..... حالت محیط

ع

Action ..... حرکت

## ی

Supervised Learning یادگیری با ناظر . . . . .  
 Unsupervised Learning یادگیری بدون ناظر . . . . .  
 Reinforcement Learning یادگیری تقویتی . . . . .  
 Deep یادگیری تقویتی عمیق . . . . .  
 Reinforcement Learning

Semi-Supervised یادگیری شبه ناظر . . . . .  
 Learning  
 Machine Learning یادگیری ماشین . . . . .

## ف

Reward Hypothesis فرضیه امتیازها . . . . .  
 Action Space فضای حرکت . . . . .  
 Observation Space فضای مشاهده . . . . .

## م

Markov مارکوف . . . . .  
 Autonomous Vehicle ماشین خودران . . . . .  
 Environment محیط . . . . .  
 Repository مخزن (گیت‌هاب) . . . . .  
 Step مرحله . . . . .  
 Observation مشاهده . . . . .  
 Observability مشاهده‌پذیری . . . . .  
 Partial observability مشاهده‌پذیری جزئی . . . . .  
 Full observability مشاهده‌پذیری کامل . . . . .  
 Matlab Engine موتور متلب . . . . .

## ن

Supervisor ناظر . . . . .  
 Anaconda نرم‌افزار آناکوندا . . . . .  
 PreScan نرم‌افزار پرسکن . . . . .  
 Matlab نرم‌افزار متلب . . . . .  
 Bechmark نقطه تراز . . . . .

## ه

Cost هزینه . . . . .