



دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیووتر

پروژه کارشناسی

دوقلوی دیجیتالی صحنه ترافیکی مبتنی بر دوربین و لایدار با استفاده از AWSIM و ROS

نگارش

رهام زنده‌دل نوبری

استاد راهنما

دکتر مهدی جوانمردی

شهریور ۱۴۰۲

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

تقدیم به
آن که جز به فضلش امیدی نیست...

سپاس‌گزاری

از پدر و مادرم که همواره در مواجهه با سختی‌های این دنیا دلسوزانه همراهم بوده‌اند؛
از استاد بزرگوارم جناب آقای دکتر مهدی جوانمردی که با حسن خلق و گشاده‌رویی، رهنماهای
شبانه‌روزی خود را از من دریغ نکرده‌اند؛
و از سایر عزیزانی که در کنارشان این نتیجه حاصل آمد کمال تشکر و قدردانی را دارم.

رهام زنده‌دل نوبری
شهریور ۱۴۰۲

چکیده

در حوزه‌ی سیستم‌های مدرن حمل و نقل، توسعه و آزمون سیستم‌های حمل و نقل هوشمند^۱، الگوریتم‌های مدیریت ترافیک^۲ و خودروهای خودران^۳ جزو مهم‌ترین موضوع‌های در حال تحقیق هستند. اطمینان از کارایی و ایمنی این سیستم‌ها، به توانایی شبیه‌سازی دقیق سناریوهای ترافیکی دنیای واقعی وابسته است. این نیازمندی، به ایجاد مفهومی به نام دوکلوبی دیجیتال^۴ منجر شده است که به عنوان یک ابزار قدرتمند در شبیه‌سازی‌های کامپیوترازی به طور گسترده پیاده‌سازی می‌شود. دوکلوبی دیجیتال به عنوان یک نسخه مجازی پویا از محیط فیزیکی عمل می‌کند و به عنوان یک سکوی بینظیر در سرعت بخشیدن به پیشرفت فناوری خودروهای خودران و مدیریت ترافیک، خدمت می‌کند. علاوه بر کاربرد در خودروهای خودران، دوکلوبهای دیجیتال در دامنه‌های گسترده‌تری از زمینه‌ها، از جمله سیستم‌های حمل و نقل هوشمند و سیستم‌های مدیریت ترافیک نیز اهمیت دارند. این اهمیت به ویژه در هنگام ارزیابی عملکرد الگوریتم‌ها و استراتژی‌ها در یک سناریوی ترافیکی شبیه‌سازی شده اما نزدیک به واقعیت نمایان می‌شود. شبیه‌سازی‌های ترافیک، نقش بسزایی را در آموزش منطق رانندگی خودکار ایفا می‌کنند و به دانشمندان اطمینان می‌دهند که این خودروها قادر به سازگاری با شرایط ترافیکی پویا و ویژه هر منطقه هستند. به طور سنتی، ایجاد شرایط ترافیکی در شبیه‌سازی‌ها از طریق روش‌های دستی یا سنتز با استفاده از مدل‌های ریاضی صورت می‌گیرد. با این حال، این رویکردها اغلب در بازتولید دقیق پیچیدگی‌های منحصر به فرد و ویژه مناطق واقعی ترافیک‌خیز، ناکام می‌مانند. علاوه بر این، این روش‌ها نیاز به تلاش‌های کاری فراوان دارند. این پژوهش به رویکردی نوآورانه و عملی برای مواجهه با این چالش‌ها پرداخته است و یک روش خودکار را پیشنهاد می‌دهد که با بهره‌گیری از داده‌های حسگری لایدار^۵ و استفاده از الگوریتم‌های هوش مصنوعی تشخیص اشیاء سه‌بعدی^۶، سناریوهای مختلف ترافیک واقعی جهان را با دقت شبیه‌سازی کند.

واژه‌های کلیدی:

هوش مصنوعی، دوکلوبی دیجیتال، شبیه‌سازی ترافیک، خودروی خورдан، سیستم مدیریت ترافیک

¹ Intelligent Transportation Systems (ITS)

² Traffic Management

³ Autonomous Vehicle

⁴ Digital Twin

⁵ Lidar Sensor

⁶ 3D Object Detection

فهرست مطالب

صفحه

عنوان

۱	۱ مقدمه
۲	۱-۱ انقلاب صنعتی چهارم و شهرهای هوشمند
۲	۲-۱ سیستم‌های حمل و نقل هوشمند
۳	۳-۱ شبیه‌سازی محیط برای سیستم‌های هوشمند مدیریت ترافیک
۴	۴-۱ دوکلوبی دیجیتال
۵	۵-۱ دوکلوبی دیجیتال در صحنه‌های شهری و ترافیکی
۶	۶-۱ ضرورت استفاده از دوکلوبی دیجیتال
۷	۷-۱ تعریف مسئله
۷	۷-۱-۱ هدف اصلی
۷	۷-۱-۲ اهداف فرعی
۸	۸ مفاهیم پایه و نگاهی بر کارهای پیشین
۹	۹-۱ مفاهیم پایه
۹	۹-۱-۱ حسگرها
۹	۹-۱-۱-۱ حسگر لایدار
۱۱	۱۱-۱-۲ دستگاه RSU
۱۱	۱۱-۲ شبکه‌های عصبی
۱۲	۱۲-۱-۲ شبکه‌های عصبی پیچشی
۱۳	۱۳-۲-۱-۲ لایه پیچش
۱۶	۱۶-۲-۱-۲ ابر نقاط و تشخیص اجسام سه‌بعدی
۱۷	۱۷-۲-۱-۲ تشخیص اجسام سه‌بعدی
۱۹	۱۹-۲-۳-۱-۲ روش‌های مبتنی بر ابر نقاط
۲۰	۲۰-۳-۱-۲ روش‌های تشخیص مبتنی بر واکسل
۲۱	۲۱-۴-۳-۱-۲ روش‌های تشخیص مبتنی بر نقطه
۲۲	۲۲-۵-۳-۱-۲ روش‌های تشخیص مبتنی بر واکسل-نقطه
۲۲	۲۲-۶-۳-۱-۲ تشخیص اجسام سه‌بعدی در دوکلوبهای دیجیتال
۲۳	۲۳-۴-۱-۲ مجموعه داده مدل‌های تشخیص سه‌بعدی
۲۳	۲۳-۱-۴-۱-۲ اندازه
۲۳	۲۳-۲-۴-۱-۲ تنوع
۲۴	۲۴-۳-۴-۱-۲ مزایا و معایب
۲۵	۲۵-۵-۱-۲ معیارهای ارزیابی
۲۵	۲۵-۱-۵-۱-۲ ماتریس درهم ریختگی

۲۶	دقت	۱-۲
۲۶	پوشش	۱-۲
۲۶	نسبت اشتراک بر روی اجتماع	۱-۲
۲۸	میانگین دقต و میانگین دقت درونیابی شده	۱-۲
۳۰	mAP	۱-۲
۳۱	KITTI	۱-۲
۳۱	nuScenes	۱-۲
۳۲	Waymo	۱-۲
۳۳	معیار فاصله اقلیدسی	۱-۲
۳۴	معیار فاصله ماهالانوبیس	۱-۲
۳۵	خوشه‌بندی اقلیدسی	۱-۲
۳۶	مدل‌سازی سه‌بعدی	۱-۲
۳۷	نگاهی بر کارهای پیشین	۲-۲
۳۷	طراحی سیستماتیک دوکلوبی دیجیتال	۲-۲
۳۸	نحوه‌ی بهینه مدل‌سازی سه‌بعدی دوکلوهای دیجیتال	۲-۲
۳۹	چرخه‌ی شبیه‌سازی خودکار اجسام پویا در دوکلوهای دیجیتال	۲-۲
۴۱	ابزارها و کتابخانه‌ها	۳
۴۲	ROS	۱-۳
۴۲	اکوسیستم	۱-۳
۴۴	نسخه‌ها	۱-۳
۴۵	معماری بسته توسعه نرم‌افزار ROS2	۱-۳
۴۵	گراف	۱-۳-۱-۳
۴۵	گره	۱-۳-۱-۳
۴۶	رابطها	۱-۳-۱-۳
۴۸	کتابخانه‌های کلاینت	۱-۳-۱-۳
۴۹	Autoware	۱-۳
۵۰	معماری خودمختار در ابعاد میکرو	۱-۲-۳
۵۰	مدول Core	۱-۲-۳
۵۱	مدول Universe	۱-۲-۳
۵۱	مولفه Sensing	۱-۲-۳
۵۳	مولفه Map	۱-۲-۳
۵۴	مولفه Localization	۱-۲-۳
۵۶	مولفه Perception	۱-۲-۳

۵۷	۱-۷-۲-۳	گره پیشپردازش
۵۸	۲-۷-۲-۳	گره تشخیص و ردیابی اجسام سه بعدی
۶۱	۸-۲-۳	مولفه Planning
۶۳	۳-۳	شبیهساز AWSIM
۶۴	۱-۳-۳	معماری
۶۴	۲-۳-۳	مولفه محیط
۶۵	۱-۲-۳-۳	مولفه های ترافیک (Traffic)
۶۶	۳-۳-۳	مولفه خودروی خودران
۶۷	۴-۳-۳	Autoware و ترکیب آن با شبیهساز AWSIM
۶۹	۵-۳-۳	افزونه ROS2ForUnity
۷۰	۴-۳	موتور بازی Unity
۷۱	۵-۳	ابزار Blender
۷۲	۱-۵-۳	افزونه Blosm
۷۲	۶-۳	ابزار OpenStreetMap
۷۳	۷-۳	کتابخانه OusterSDK
۷۴	۸-۳	ابزار Ouster Studio
۷۵	۴	پیادهسازی و طراحی
۷۶	۱-۴	نگاهی به مسیر طی شده
۷۶	۲-۴	سناریو اول: استفاده از داده های نمونه
۷۷	۱-۲-۴	پیدا کردن داده های نمونه لایدارهای شرکت Ouster
۷۷	۳-۴	انتخاب مدل هوش مصنوعی تشخیص سه بعدی و ردیاب
۷۸	۱-۳-۴	مدل SFA3D
۷۸	۲-۳-۴	مخزن MMDetection3D
۷۹	۱-۲-۳-۴	مدل تشخیص اجسام سه بعدی CenterPoint
۸۱	۳-۳-۴	ایده: نگاه انداختن به معماری Autoware
۸۱	۴-۳-۴	مولفه ادارک نرم افزار Autoware
۸۳	۵-۳-۴	نتیجه گیری
۸۳	۴-۴	اعمال مدل هوش مصنوعی به ورودی لایدار
۸۴	۱-۴-۴	دستکاری فایل های اجرایی Autoware
۸۷	۲-۴-۴	نتیجه گیری
۹۰	۵-۴	پردازش ویژگی های اجسام تشخیص داده شده در شبیهساز
۹۳	۶-۴	شبیهسازی ترافیک در شبیهساز
۹۶	۱-۶-۴	ایجاد صحنه جدید

۹۹	۲-۶-۴ نتیجه‌گیری سناریو اول	۴
۱۰۰	۷-۴ سناریو دوم: داده‌برداری از منطقه خیابان رشت و طراحی سه‌بعدی	۴
۱۰۰	۱-۷-۴ داده‌برداری از خیابان رشت	۴
۱۰۳	۲-۷-۴ اعمال مدل هوش مصنوعی به ورودی لایدار	۴
۱۰۶	۸-۴ طراحی مدل سه‌بعدی منطقه دانشگاه صنعتی امیرکبیر	۴
۱۰۹	۹-۴ شبیه‌سازی در خیابان رشت طراحی شده	۴
۱۱۲	۵ ارزیابی و بررسی	۵
۱۱۵	۶ جمع‌بندی، نتیجه‌گیری و پیشنهادات برای کارهای آتی	۶
۱۱۶	۱-۶ جمع‌بندی و نتیجه‌گیری	۶
۱۱۶	۲-۶ پیشنهادات برای کارهای آتی	۶
۱۱۶	۱-۲-۶ طراحی نقشه ابر نقاط	۶
۱۱۶	۲-۲-۶ تهیه نقشه برداری	۶
۱۱۷	۳-۲-۶ مدل‌سازی کامل منطقه دانشگاه امیرکبیر	۶
۱۱۷	۴-۲-۶ استفاده از مدل‌های جدید	۶
۱۱۸	کتابنامه	

فهرست تصاویر

شکل

صفحه

- ۱-۱ تصویری از دوقلوی دیجیتال یک اتوبان در بیجینگ که به صورت زنده در حال شبیه‌سازی وضعیت کنونی اتوبان است. [۱] ۵
- ۱-۲ تصویری از نحوه محاسبه فاصله اجسام توسط حسگر لایدار ۱۰
- ۲-۱ حسگر لایدار OS1:64 از شرکت Ouster [۲] ۱۰
- ۳-۲ مدل استاندارد شبکه‌ی عصبی عمیق [۳] ۱۲
- ۴-۲ نگاهی کلی به معماری استاندارد شبکه‌های پیچشی [۴] ۱۳
- ۵-۲ عملیت پیچش با فیلترهای 3×3 و گام ۱ [۴] ۱۴
- ۶-۲ یک عملیات پیچش با لایه‌گذاری صفر [۴] ۱۵
- ۷-۲ اسکن رنگی لایدار از ساختمان معروف نوتردام پاریس که با استفاده از آن این ساختمان را بازسازی کردند. ۱۶
- ۸-۲ اسکن لایدار OS1:64 از شرکت Ouster که بر روی یک ماشین نصب شده است. ۱۷
- ۹-۲ بررسی اجمالی وظیفه‌ی تشخیص اجسام سه‌بعدی از تصاویر و ابر نقاط [۵] ۱۸
- ۱۰-۲ سیر زمانی پیشرفت روش‌های مبتنی بر ابر نقاط [۵] ۲۰
- ۱۱-۲ ماتریس در هم ریختگی [۶] ۲۵
- ۱۲-۲ نسبت اشتراک بر روی اجتماع [۶] ۲۶
- ۱۳-۲ سنجش واقعی یا کاذب بودن براساس حد آستانه [۶] ۲۷
- ۱۴-۲ نسبت اشتراک بر روی اجتماع در فضای سه‌بعدی و دو بعدی [۷] ۲۷
- ۱۵-۲ مرتب سازی براساس امتیاز اطمینان کادرهای پیش‌بینی شده ۲۸
- ۱۶-۲ جدول محاسبه دقیق و پوشش هر تخمین به صورت تجمعی ۲۹
- ۱۷-۲ نمودار دقیق-پوشش ۲۹
- ۱۸-۲ نمودار دقیق-پوشش درون‌یابی شده ۳۰
- ۱۹-۲ فاصله‌ی دو نقطه بر اساس فاصله‌ی اقلیدسی در دو بعد ۳۳
- ۲۰-۲ فاصله‌ی ماهalanobis حول یک نقطه مرکزی [۸] ۳۴
- ۲۱-۲ مدل سه‌بعدی یک ساختمان ۳۶
- ۲۲-۲ معماری دوقلوهای دیجیتالی و سلسله مراتب آنها [۹] ۳۷
- ۲۳-۲ خلاصه‌ای از روش پیشنهاد شده برای ایجاد دوقلوی دیجیتال یک منطقه (در فصل چهارم به تحلیل دقیق این شکل می‌پردازیم) [۱۰] ۳۸
- ۲۴-۲ چرخه پردازش داده و تشخیص پارامترهای پویا و ساختن دوقلوی دیجیتال به صورت خودکار [۱۱] ۳۹
- ۴-۳ تصویری از اجزای ROS [۱۲] ۴۲
- ۴-۴ تفاوت قابلیت‌های ROS2 نسبت به ROS1 [۱۳] ۴۴

۴۵	۳-۳ چهار توزیع اصلی ROS2
۴۶	۴-۳ یک گراف ROS2 ساده [۱۴]
۴۷	۵-۳ رابطهای یک گره ROS2: مبحث‌ها، سرویس‌ها، اعمال [۱۳]
۴۸	۶-۳ کتابخانه‌های کلاینت ROS2 [۱۳]
۵۰	۷-۳ دو مدول اصلی Autoware [۱۵]
۵۱	۸-۳ انواع ورودی‌های قابل پذیرش توسط مولفه Sensing [۱۵]
۵۲	۹-۳ گراف گرهی مولفه Sensing [۱۵]
۵۳	۱۰-۳ معماری مولفه Map [۱۵]
۵۴	۱۱-۳ گراف گرهی مولفه Map (مستطیل‌های سبز) [۱۵]
۵۵	۱۲-۳ معماری پیشنهادی Autoware برای مولفه Localization [۱۵]
۵۶	۱۳-۳ گراف گرهی مولفه Localization [۱۵]
۵۷	۱۴-۳ معماری رابطهای مولفه Perception [۱۵]
۵۷	۱۵-۳ گراف گرهی پیش‌پردازش مولفه Perception [۱۵]
۵۹	۱۶-۳ گراف گرهی تشخیص اجسام و ردیابی مولفه Perception [۱۵]
۶۱	۱۷-۳ معماری سطح بالای مولفه Planning [۱۵]
۶۲	۱۸-۳ معماری سطح بالای نرم‌افزار Autoware [۱۵]
۶۳	۱۹-۳ شبیه‌ساز AWSIM [۱۶]
۶۴	۲۰-۳ معماری سطح بالای شبیه‌ساز AWSIM [۱۶]
۶۵	۲۱-۳ معماری مولفه محیط شبیه‌ساز AWSIM [۱۶]
۶۶	۲۲-۳ معماری مولفه خودروی خودران شبیه‌ساز AWSIM [۱۶]
۶۷	۲۳-۳ معماری ترکیب Autoware و AWSIM [۱۶]
۶۸	۲۴-۳ تصویری از همکاری شبیه‌ساز AWSIM (سمت چپ) و نرم‌افزار Autoware
۷۰	۲۵-۳ شبیه‌ساز AWSIM در بستر موتوبازی Unity [۱۶]
۷۱	۲۶-۳ محیط ابزار Blender
۷۳	۲۷-۳ نقشه دانشگاه صنعتی امیرکبیر در ابزار OSM
۷۴	۲۸-۳ ابزار نمایشگر کتابخانه OusterSDK برای ابر نقاط [۱۷]
۷۶	۱-۴ فلوچارت پروژه
۷۷	۲-۴ داده‌ی نمونه که با لایدار OS1:128 ضبط شده است.
۷۸	۳-۴ تشخیص بلادرنگ مدل SFA3D، با سرعت تشخیص بیشتر از ۹۰ فریم در ثانیه [۱۸]
۷۹	۴-۴ تعداد مدل‌های پیاده‌سازی شده در مخزن MMDetection3D [۱۹]
۸۰	۵-۴ معماری مدل CenterPoint [۲۰]
۸۲	۶-۴ نحوه کارکرد گره ردیاب همزمان چند جسم
۸۴	۷-۴ روند تبدیل داده‌های ضبط شده لایدار به پیام PointCloud2 و انتشار آن

۸۵	۸-۴	معماری فایل اجرایی اصلی autoware_launch
۸۵	۹-۴	منفی گذاشتن تمامی مولفه‌ها بجز مولفه ادراک
۸۶	۱۰-۴	معماری فایلهای اجرایی [۲۱]perception.launch
۸۶	۱۱-۴	معماری فایل اجرایی detection.launch
۸۷	۱۲-۴	معماری فایل اجرایی lidar_based_detection.launch
۸۸	۱۳-۴	دستورهای اجرایی تشخیص دهنده سه بعدی
۸۸	۱۴-۴	۱۴-۴ اجرایی از تشخیص اجسام سه بعدی توسط مدل CenterPoint در داده‌ی نمونه یک چهارراه
۸۹	۱۵-۴	۱۵-۴ اجرایی از تشخیص اجسام سه بعدی توسط مدل CenterPoint در داده‌ی نمونه یک ماشین در حال رانندگی در سطح شهر
۸۹	۱۶-۴	۱۶-۴ مشاهده خروجی بخش‌بندی موائع، که نقاط زمین و نویزها را از ابر نقاط فیلتر می‌کند.
۹۰	۱۷-۴	۱۷-۴ مشخصات مبحث خروجی گره ردیاب همزمان چند جسم
۹۱	۱۸-۴	۱۸-۴ کد مربوط به ساختار پیام خروجی ردیاب همزمان چند جسم
۹۱	۱۹-۴	۱۹-۴ کد مربوط به ساختار پیام جسم در حال ردیابی
۹۲	۲۰-۴	۲۰-۴ کد مربوط گرفتن اشتراک برای مبحث ردیاب و انتشار به یک مبحث جدید
۹۳	۲۱-۴	۲۱-۴ مبحث منشر شده پیام‌های ردیاب، توسط افزونه R2FU در بستر Unity
۹۳	۲۲-۴	۲۲-۴ یک خودروی نمونه پیش‌ساخته در پروژه AWSIM
۹۴	۲۳-۴	۲۳-۴ شبیه‌ساز AWSIM، چندین مدل خودرو برای ایجاد تنوع دارد.
۹۴	۲۴-۴	۲۴-۴ صحنه نمونه یک خودرو از جنس NPCVehicle
۹۵	۲۵-۴	۲۵-۴ بخشی از کد NPCVehicleController که به خودروهای NPCVehicle دستورهای حرکتی متنوع می‌دهد.
۹۶	۲۶-۴	۲۶-۴ بخشی از کد NPCVehicleController که در آن نحوه شبیه‌سازی حرکت خودروها نمایش داده شده است.
۹۶	۲۷-۴	۲۷-۴ صحنه جدید ساخته شده برای شبیه‌سازی ترافیک
۹۷	۲۸-۴	۲۸-۴ بخشی از کد DetectionSubscriber
۹۸	۲۹-۴	۲۹-۴ بخشی از کوروتین کد DetectionSubscriber
۹۸	۳۰-۴	۳۰-۴ شبیه‌سازی موفق از ترافیک مشاهده شده در داده نمونه
۹۹	۳۱-۴	۳۱-۴ پرسه شبیه‌سازی بلادرنگ سناریو اول با داده نمونه
۱۰۰	۳۲-۴	۳۲-۴ نقشه پردهیس دانشگاه امیرکبیر
۱۰۱	۳۳-۴	۳۳-۴ تصاویر خیابان رشت از زاویه دید درب رشت
۱۰۱	۳۴-۴	۳۴-۴ محل پیشنهادی داده‌برداری
۱۰۲	۳۵-۴	۳۵-۴ تصاویری از عملیات داده‌برداری
۱۰۲	۳۶-۴	۳۶-۴ تصویری از خیابان رشت و ابر نقاط متناظر آن
۱۰۳	۳۷-۴	۳۷-۴ تصویری از لایدار OS1:64 که بر روی سه‌پایه نصب شده است.

۳۸-۴	اقدام ناموفق تشخیص بر روی ابر نقاط خیابان رشت	۱۰۳
۳۹-۴	نمای نزدیک از کالیبره نبودن ابر نقاط خیابان رشت.	۱۰۴
۴۰-۴	تشخیص موفق اجسام با ورودی ابر نقاط کالیبره شده	۱۰۵
۴۱-۴	تصویری از خیابان رشت و ابر نقاط پردازش شده متناظر آن	۱۰۶
۴۲-۴	روش پیشنهاد شده برای ایجاد دوقلوی دیجیتال یک منطقه [۱۰]	۱۰۷
۴۳-۴	نمایی از افزونه Blosm در ابزار مدلسازی سه بعدی Blender	۱۰۸
۴۴-۴	منطقه انتخاب شده در افزونه Blosm	۱۰۸
۴۵-۴	مدل سه بعدی منطقه دانشگاه صنعتی امیرکبیر که توسط افزونه Blosm تولید شده است.	۱۰۹
۴۶-۴	نقشه وارد شده در شبیه ساز AWSIM	۱۰۹
۴۷-۴	شیء بازی SpawnPoint که نسبت به موقعیت تبدیل یافته لایدار مستقر شده است.	۱۱۰
۴۸-۴	دوقلوی تقریبی دیجیتال از خیابان رشت	۱۱۰
۴۹-۴	سه تصویر مختلف از یک لحظه زمانی (تصویر، تشخیص، شبیه سازی)	۱۱۱
۱-۵	تصویری از مثبت کاذب تشخیص داده شده	۱۱۳
۲-۵	تصویری از تشخیص اشتباه دو میله به عنوان عابرین پیاده	۱۱۴
۳-۵	تصویری از تشخیص اشتباه موتوسیکلت به عنوان عابر پیاده	۱۱۴

صفحه **فهرست جداول** جدول

- ۱-۲ توزیع داده در داده‌های آموزش KITTI. کلاس خودرو ۸۲.۹۹٪ سه کلاس اصلی (خودرو، عابرپیاده، دوچرخهسوار) را تشکیل می‌دهد [۵].
۲۴
۲-۲ توزیع داده در داده‌های آموزش nuScenes [۵]
۶۹
۱-۳ جدول پیام‌های پشتیبانی شده توسط R2FU [۱۶]
۸۰
۱-۴ جدول مقایسه مدل‌های سه‌بعدی براساس ارزیابی مجموعه داده Waymo Open [۲۰]
۸۰
۲-۴ جدول مقایسه مدل‌های سه‌بعدی براساس ارزیابی مجموعه داده nuScenes [۲۰]

فصل اول

مقدمه

۱-۱ انقلاب صنعتی چهارم و شهرهای هوشمند

انقلاب صنعتی چهارم^۱، جدیدترین فاز پیشرفت فناوری در صنعت و تولید است که با ادغام سیستم‌های دیجیتالی، فیزیکی و بیولوژیکی شناخته می‌شود. این انقلاب بر روی انقلاب صنعتی سوم که با بهره‌گیری از تکنولوژی کامپیوترها و اتوماسیون همراه بود، بنا شده است و با بهره‌گیری از تکنولوژی‌هایی نظیر اینترنت اشیا^۲، هوش مصنوعی، رباتیک و آنالیز پیشرفت‌هه داده یک قدم به سمت جلو برداشته است.

انقلاب صنعتی چهارم به طور اساسی روش زندگی و کار ما را تغییر می‌دهد و مدل‌های جدید کسب‌وکار، صنایع و روش‌های تعامل آنها با یکدیگر را ایجاد می‌کند. این انقلاب به شرکت‌ها این امکان را می‌دهد که فرایندهای خود را دیجیتالی کنند، عملیات خود را بهینه کنند و محصولات و خدمات جدیدی را ایجاد کنند که قبلاً غیرقابل تصور بوده است. انقلاب صنعتی چهارم همچنین ماهیت کار کردن را تغییر می‌دهد و با پتانسیل اتوماسیون بسیاری از وظایف، کارهای روزمره و افرایش قابلیت انسان در تعامل با فناوری همراه است.

شهرهای هوشمند یکی از هیجان‌انگیزترین و تحول‌بخش‌ترین توسعه‌های انقلاب صنعتی چهارم است. با تکیه بر تکنولوژی‌های اخیر نظری اینترنت اشیاء، هوش مصنوعی و آنالیز داده؛ شهرهای هوشمند شیوه زندگی، کار و تعامل انسان‌ها با محیط را متحول کرده‌اند. در اصل، شهرهای هوشمند برای بهبود کیفیت زندگی شهروندان و بهبود کارایی و پایداری زیرساخت‌های شهری طراحی شده‌اند. شهرهای هوشمند در مسائلی از جمله کاهش ترافیک و آلودگی هوای شهری تا بهینه‌سازی مصرف انرژی و بهبود ایمنی عمومی نقش کلیدی دارند. شهرهای هوشمند از داده‌های مختلف و آنالیز آن‌ها برای تصمیم‌گیری‌های بهتر و بهبود تجربه زندگی شهری بهره می‌برند.

یکی از ویژگی‌های کلیدی شهرهای هوشمند، قابلیت استفاده آن‌ها از حسگرهای پیشرفت‌ه و شبکه‌های ارتباطی برای جمع‌آوری و آنالیز داده‌ها به صورت بلاذرنگ^۳ است. این داده‌ها می‌توانند برای ایجاد سیستم‌های حمل و نقل هوشمند که جریان ترافیک را بهینه می‌کنند و ترافیک را کاهش می‌دهند، استفاده شوند.

شهرهای هوشمند یکی از بهترین مثال‌های انقلاب صنعتی چهارم هستند و تاثیر آن‌ها به مرور زمان رشد می‌کند، زیرا شهرهای بیشتری در دنیا به دیجیتالی شدن و قدرت تحول‌بخش فناوری‌های پیشرفت‌ه روی می‌آورند.

۲-۱ سیستم‌های حمل و نقل هوشمند

سیستم‌های حمل و نقل هوشمند فناوری‌ها و سیستم‌های پیشرفت‌هایی هستند که برای بهینه‌سازی عملکرد، امنیت و پایداری شبکه‌های حمل و نقل طراحی شده‌اند. سیستم‌های حمل و نقل هوشمند،

¹Fourth Industry Revolution

²Internet of Things (IoT)

³Real-time

شامل یک مجموعه گسترده از فناوری‌ها مانند سیستم‌های مدیریت ترافیک، سیستم‌های ارتباطی خودرو به خودرو، خودرو به زیرساخت، خودروهای خودران و سیستم‌های کمک راننده پیشرفته است.

هدف اصلی سیستم‌های حمل و نقل هوشمند، بهبود کارایی و ایمنی شبکه‌های حمل و نقل است که کاهش تاثیرات زیست محیطی را نیز به دنبال دارد. با بهره‌گیری از داده‌های بلاذرنگ و الگوریتم‌های پیشرفته، این سیستم می‌تواند جریان ترافیک را بهینه کند، حمل و نقل را بهبود بخشد و ایمنی کاربران جاده را تقویت بدنهند. همچنین، این سیستم‌ها می‌توانند به کاهش تراکم ترافیک کمک کنند، که به کاهش جریان ترافیک، کاهش آلودگی و کیفیت هوای محیط کمک می‌کند.

یکی از مهم‌ترین اجزای سیستم‌های حمل و نقل هوشمند، سیستم‌های مدیریت ترافیک هستند که برای نظارت و مدیریت جریان ترافیک به صورت بلاذرنگ طراحی شده‌اند. این سیستم‌ها از انواع منابع مانند حسگرها، دوربین‌ها و دستگاه‌های موقعیت‌یاب^۱، داده‌ها را جمع‌آوری می‌کنند و از آن برای تنظیم پویای زمان‌بندی سیگنال‌های ترافیکی، تغییر مسیر ترافیک و ارائه اطلاعات ترافیکی بلاذرنگ به کاربران جاده استفاده می‌کنند. در سال‌های اخیر، برای شبیه‌سازی بلاذرنگ صحنه‌های ترافیکی از فناوری دوکلوبی دیجیتال استفاده می‌شود تا بتوان سیستم‌های هوشمند ترافیکی را در محیطی نزدیک به واقعیت آزمود.

۳-۱ شبیه‌سازی محیط برای سیستم‌های هوشمند مدیریت ترافیک

شبیه‌سازی یکی از پراستفاده‌ترین روش‌ها برای تسهیل طراحی سیستم‌های مدیریت ترافیک است. شبیه‌ساز ماشین‌های خودران یک فضای دنیای فیزیکی را به شکل یک فضای دیجیتالی بازسازی می‌کند و این بستر آزمایشی را در اختیار سیستم‌های هوشمند مدیریت ترافیک قرار می‌دهد تا بتواند شرایط شبیه‌سازی شده را زیر نظر بگیرد، تصمیم درست در انتخاب مسیر ترافیک و سیگنال‌های ترافیکی بگیرد و عمل مناسب را انجام دهد [۱۱]. شبیه‌سازی، یک بستر کنترل شده و امن را برای پیاده‌سازی و آزمایش سیستم‌های مدیریت ترافیک فراهم می‌کند. در این روش ابتدا نرمافزار در فضای شبیه‌سازی شده پیاده‌سازی می‌شود و پس از آزمون‌های گوناگون و اطمینان از کارکرد آن، در دنیای فیزیکی واقعی آزموده می‌شود [۱۱]. اما اگر فضای شبیه‌سازی ما ایستا باشد، یعنی در شبیه‌سازی از داده‌های پویا همانند عابر پیاده، ماشین‌های دیگر، تغییرات آب و هوا و... استفاده نشود، آزمون‌های ما قابل اطمینان نخواهند بود و نمی‌توانیم سیستم طراحی شده خود برای مدیریت ترافیک را در دنیای واقعی بیازماییم. پس نیاز است از شبیه‌سازی‌های پیشرفته استفاده کنیم که در آن فضای شبیه‌سازی شده پویا باشد و مشابه دنیای واقعی عمل کند.

^۱GPS

۴-۱ دوکلوبی دیجیتال

دوکلوبی دیجیتال، یک بازنمایی پویا و دیجیتال از یک سیستم یا یک فضا است که توسط مایکل گریوز^۱ در سال ۲۰۰۳ معرفی شد [۲۲]. پس از آن این مفهوم توسط ناسا^۲ بازبینی شد و آن را یک شبیه‌سازی بسیار دقیق و چند لایه‌ای از یک پدیده یا یک سیستم معرفی کرد که براساس داده‌های بلادرنگ شبیه‌سازی می‌شود [۲۳]. روزن و همکاران^۳ در مقاله‌ای منتشر شده در سال ۲۰۱۵، دوکلوبی دیجیتال را یک بازنمایی مجازی از دنیای فیزیکی معرفی می‌کند که با استفاده از داده‌ها و شبیه‌سازی‌ها؛ کنترل و بهینه سازی را مقدور می‌سازد [۲۴]. از دوکلوبی دیجیتال برای شبیه‌سازی، نظارت و مدیریت سیستم‌ها و پدیده‌ها استفاده می‌شود، زیرا قادر به متصل کردن دنیای فیزیکی به دنیای مجازی آنها است، به گونه‌ای که چرخه زندگی^۴ سیستم‌ها را به صورت کامل شبیه‌سازی می‌کند. با بکارگیری از اینترنت اشیا و قابلیت متصل کردن حسگرهای گوناگون با استفاده از اینترنت، می‌توانیم از داده‌های بلادرنگ جمع آوری شده از حسگرهای توسط شبکه‌های ارتباطی استفاده کنیم و دوکلوبی دیجیتال از سیستم‌ها و پدیده‌های گوناگون تشکیل دهیم. دوکلوبی دیجیتال به تازگی در بازارهای بزرگ صنعت و شهرسازی استفاده می‌شود و شبیه‌سازی‌های بسیار دقیق آن از دنیای واقعی، پیاده‌سازی انواع سیستم‌ها و مانیتورینگ را بسیار کارآمد کرده است [۲۵].

دوکلوبی دیجیتال، یکی از مولفه‌های اصلی انقلاب صنعتی چهارم است، به گونه‌ای که بیشتر شرکت‌ها و کارخانه‌ها، چرخه‌های زندگی سیستم‌های خود و مدل‌های مختلفی از پدیده‌های گوناگون را دیجیتالیزه می‌کنند، یا به عبارتی دوکلوبی دیجیتال می‌سازند، و تمامی طراحی‌ها و آزمایش‌های گوناگون را در این بستر انجام می‌دهند. همچنین شرکت‌های شهرسازی و دولت‌ها، در حال ساختن دوکلوبی دیجیتالی از شهرهای خود هستند تا بتوانند طرح‌های جدید را در سطح شهری بیازمایند و نتایج بدست آمده را تحلیل کنند.

دوکلوبی دیجیتال و شبیه‌سازی هر دو از مدل‌های دیجیتالی برای بازنمایی پروسه‌های مختلف یک سیستم استفاده می‌کنند. اما دوکلوبی دیجیتالی یک محیط و فضای مجازی براساس واقعیت است که آن را برای تحقیقات بسیار ارزشمندتر می‌کند. یکی از تفاوت‌های دوکلوبی دیجیتال و شبیه‌سازی در مقیاس آنهاست؛ با استفاده از شبیه‌سازی، عموماً در مورد یک پروسه خاص تحقیق و بررسی می‌شود، اما دوکلوبی دیجیتال بررسی چندین شبیه‌سازی بر روی چندین پروسه را می‌سر می‌کند. تفاوت دیگر شبیه‌سازی و دوکلوبی دیجیتال این است که شبیه‌سازی عموماً از داده‌های بلادرنگ استفاده نمی‌کند، در حالی که دوکلوبی دیجیتال از یک جریان دوطرفه اطلاعات بهره می‌برد به طوری که در ابتدا از حسگرهای داده‌های بلادرنگ را می‌گیرد و در دوکلوبی دیجیتال شبیه‌سازی می‌کند و نتیجه حاصل را به دنیای فیزیکی و سیستم یا شئ مورد نظر برمی‌گرداند.

¹Michael Grieves

²NASA

³Rosen et al.

⁴System Lifecycle

۱-۵ دوکلوبی دیجیتال در صحنه‌های شهری و ترافیکی

دوکلوبی دیجیتال ابزاری قدرتمند و محبوب در حیطه توسعه سیستم‌های حمل و نقل هوشمند برای مدیریت صحنه‌های ترافیکی هستند. دوکلوبی دیجیتال با پیاده‌سازی بسیار دقیق و جامع از جاده‌ها و زیرساخت آن‌ها از جمله ماشین‌ها و عابرین پیاده، می‌توانند داده‌های آماری مهمی را از جاده‌ها بدست بیاورند. آنها همچنین فضایی تحت کنترل و امن را برای آزمودن مشابه واقعیت سیستم‌های طراحی شده برای کنترل جریان و تراکم ترافیک را مهیا می‌کنند. به طور مثال می‌توان به پروژه عظیم کشور چین برای ساخت دوکلوبی دیجیتال شهر هوشمند بیجینگ اشاره کرد که بخش بزرگی از آن ساخت دوکلوبی دیجیتال شبکه جاده‌های شهری است [۱].



شکل ۱-۱ تصویری از دوکلوبی دیجیتال یک اتوبان در بیجینگ که به صورت زنده در حال شبیه‌سازی وضعیت کنونی اتوبان است. [۱]

در شکل ۱-۱، تصویری از دوکلوبی دیجیتال اتوبان بیجینگ غربی سوم^۱ را مشاهده می‌کنیم. در این تصویر، دوکلوبی دیجیتال صحنه ترافیکی علاوه بر مدل سه بعدی فضای شهری شامل خیابان‌ها و فضای سبز و دیگر المان‌های ثابت در صحنه، اطلاعات پویا مانند خودروهای در حال تردد را نیز شامل می‌شود. حال با استفاده از این دوکلوبی دیجیتال، می‌توان تراکم ترافیک در ساعات مختلف در این اتوبان دیجیتالی را به دست آورد. این اطلاعات کاربردهای متعددی می‌تواند داشته باشد. برای مثال، می‌توان از این اطلاعات بعنوان دادگانی برای آموزش مدل‌های یادگیری عمیق^۲ استفاده کرد و تراکم ترافیک را در ساعات مختلف روز پیش‌بینی کرد. همان‌طور که قبل‌اً نیز ذکر شد، دوکلوبی دیجیتال می‌تواند دارای جریان دوطرفه اطلاعات باشد. به طور مثال در این اتوبان، داده‌های گرفته شده از حسگرهای مختلف

¹Beijing West 3rd Ring Road

²Deep Learning

همانند لایدارها^۱ و دوربین‌های مستقر (یا به عبارت دیگر یک آر.اس.یو.^۲) به عنوان ورودی دریافت می‌شوند (جريان داده دنیای واقعی به دوقلوی دیجیتال). پس از پردازش داده‌ها در دوقلوی دیجیتال و تشخیص خودروها و موقعیت مکانی آن در زمان‌های مختلف روز، مدل یادگیری عمیقی را توسعه دهیم که بتواند در راستای هوشمندسازی مدیریت ترافیک، زمان‌بندی چراغ‌های راهنمایی را بگونه‌ای کنترل کند که جريان ترافیکی بهینه شود (جريان داده دوقلوی دیجیتال به دنیای واقعی).

۶-۱ ضرورت استفاده از دوقلوی دیجیتال

برخی از برتری‌های استفاده از دوقلوی دیجیتال به عنوان بستر شبیه‌سازی عبارتند از:

۱. نزدیکی به واقعیت: همانطور که ذکر شد، دوقلوی دیجیتال از مدل‌های پویا نیز بهره می‌برد و هدف آن نزدیکی به واقعیت تا حد امکان است. پس با مدل کردن یک صحنه ترافیکی براساس داده‌های بلاذرنگ، می‌توان بستری برای سیستم‌های مدیریت ترافیک و خودروهای خودران مهیا کرد. نتایج آزمایش سیستم‌های طراحی شده در دوقلوی دیجیتال را می‌توان به دنیای واقعی نسبت داد زیرا عملای آزمایش در دوقلوی دیجیتال، ما در بستری بسیار نزدیک به واقعیت سیستم خود را آزمایش کرده‌ایم.
۲. افزایش سرعت تحقیقات: با بکارگیری از دوقلوی دیجیتال صحنه‌های ترافیکی، نیازی به تولید سناریوهای مختلف ترافیکی وجود ندارد زیرا هر لحظه از دنیای واقعی در دوقلوی دیجیتال نیز در حال تشکیل است و به صورت خودکار سناریوهای گوناگون رخ می‌دهند، در نتیجه سرعت تحقیقات و آزمایش سیستم‌های مدیریت ترافیک گوناگون به مراتب بیشتر است.
۳. امنیت بالا: در دوقلوی دیجیتال از مدل‌های پویا استفاده می‌شود، یعنی عابر پیاده و ماشین‌های دیگر نیز شبیه‌سازی می‌شوند و سیستم‌هایی هوشمند همچون ماشین‌های خودران و مدیریت ترافیک می‌توانند با وجود این مدل‌ها آزموده شوند. اما در دنیای واقعی آزمودن این در سطح شهر امری بسیار خطروناک است و ممکن است خسارات جبران ناپذیری بر سطح شهر وارد شود.
۴. قابلیت پیش‌بینی بهتر: با استفاده از دوقلوی دیجیتال، ما می‌توانیم ترافیک را پیش‌بینی کنیم و نسبت به آن سیستم مدیریت ترافیک خود را تعلیم دهیم. به علت متصل بودن داده‌های دوقلوی دیجیتال به داده‌های دنیای واقعی، سناریوهای گوناگون رخ می‌دهند و سیستم‌های مدیریت ترافیک داده‌های تمرینی بهتر و متنوع‌تر خواهند داشت [۲۶].

¹LiDAR Sensor

²Road-Side Unit (RSU)

۱-۷ تعریف مسئله

مشاهده کردیم که کشور چین، دو قلوبی دیجیتال یکی از اتوبان‌های خود را در سال ۲۰۲۱ با موفقیت پیاده‌سازی کرد و در حال تحقیق بر روی این موضوع است. با الهام گرفتن از این پروژه، اهداف این پژوهش را در دو دسته‌ی هدف اصلی و اهداف فرعی بخش‌بندی می‌کنیم.

۱-۷-۱ هدف اصلی

هدف اصلی این پروژه، پیاده‌سازی دو قلوبی دیجیتالی از صحنه ترافیکی خیابان رشت، از دید درب رشت دانشگاه صنعتی امیرکبیر است. این دو قلوبی دیجیتال به عنوان یک بازنمایی دقیق و پویا از خیابان رشت ایجاد می‌شود تا امکان شبیه‌سازی و مطالعه وضعیت‌های ترافیکی مختلف در این منطقه را فراهم کند. (مشابه دو قلوبی دیجیتال اتوبان بیجینگ با جزئیات به مراتب ساده‌تر)

۱-۷-۲ اهداف فرعی

برای دستیابی به هدف اصلی پروژه، اهداف فرعی زیر تعیین شده‌اند:

۱. مدل‌سازی ایستای سه‌بعدی خیابان رشت با استفاده از نرم‌افزارهای طراحی سه‌بعدی مبتنی بر برداشت داده‌های حسگر لایدار (به صورت دستی با کمک نرم‌افزارهای طراحی سه‌بعدی)
۲. مستقر کردن حسگرهای لایدار و دوربین در محل نگهبانی درب رشت دانشگاه صنعتی امیرکبیر به منظور برداشت داده‌های پویا.
۳. پردازش داده‌های تصویر و ابر نقاط^۱ به منظور شناسایی و مکانیابی پدیده‌های پویا با استفاده از مدل‌های آماده یادگیری ماشین در زمینه تشخیص اجسام سه‌بعدی.
۴. بصری‌سازی مدل سه‌بعدی ایستا و داده‌های پویای شناسایی شده در بستر سه‌بعدی با استفاده از شبیه‌ساز AWSIM
۵. بصری‌سازی اطلاعات آماری بدست آمده از دو قلوبی دیجیتال به منظور تجزیه و تحلیل نتایج پروژه.

این اهداف فرعی به منظور دستیابی به هدف اصلی پروژه ارائه شده‌اند و با استفاده از آنها، پروژه به طور جامع توصیف و اجرا می‌شود.

^۱Point Cloud (PCL)

فصل دوم

مفاهیم پایه و نگاهی بر کارهای پیشین

۱-۲ مفاهیم پایه

برای آنکه درک مناسبی از مسئله پیاده‌سازی دوکلوبی دیجیتال و تشخیص اجسام پویا ایجاد شود، نیاز است که ابزارها و الگوریتم‌هایی که در این پژوهش از آنها استفاده شده است به صورت اجمالی تعریف شوند.

۱-۱-۲ حسگرها

حسگرها به عنوان اجزای ضروری در عرصه فناوری و جمع‌آوری داده‌ها، به عنوان چشم‌ها و گوش‌های سیستم‌های مختلف عمل می‌کنند. این دستگاه‌های الکترونیکی برای تشخیص و پاسخ به تغییرات فیزیکی یا محیطی طراحی شده‌اند و ظواهر جهان واقعی را به داده‌های قابل اندازه‌گیری تبدیل می‌کنند. حسگرها انواع گوناگونی دارند؛ از حسگرهای دما که تغییرات حرارتی را نظارت می‌کنند تا حسگرهای حرکت که حرکت را تشخیص می‌دهند. آنها در بسیاری از کاربردها مورد استفاده قرار می‌گیرند، از سیستم‌های خودرویی که اینمی وسیله نقلیه را تضمین می‌کنند تا دستگاه‌های خانه هوش مصنوعی که راحتی و کارایی را افزایش می‌دهند. با توانایی در تجزیه و تحلیل داده‌ها به صورت بلادرنگ، حسگرها نقش اساسی در زمینه‌هایی مانند بهداشت، تولید، نظارت بر محیط زیست و موارد دیگر ایفا می‌کنند و تصمیم‌گیری آگاهانه و خودکارسازی فرآیندهای مختلف را در دنبای متصل ما امکان‌پذیر می‌سازند.

۱-۱-۱-۲ حسگر لایدار

لایدار یک اختصار برای تشخیص و فاصله‌سنجی با نور^۱ است. در لایدار، نور لیزر از یک منبع (فرستنده) ارسال می‌شود و از اجسام حاضر در صحنه بازتاب می‌یابد. نور بازتاب شده توسط گیرنده سیستم تشخیص داده می‌شود و از زمان پرواز^۲ این نور، برای توسعه نقشه فاصله^۳ اجسام استفاده می‌شود [۲۷]. به طور اصولی، لایدار یک دستگاه اندازه‌گیری فاصله به هدف است که با ارسال یک پالس کوتاه نوری و ثبت زمان گذشته بین پالس نوری خروجی و تشخیص پالس نوری بازتابی (برگشتی)، فاصله را اندازه‌گیری می‌کند.

طبق شکل ۱-۲، با دانستن سرعت نور که با c نمایش داده می‌شود داریم:

$$d = c \cdot t / 2 \quad (1-2)$$

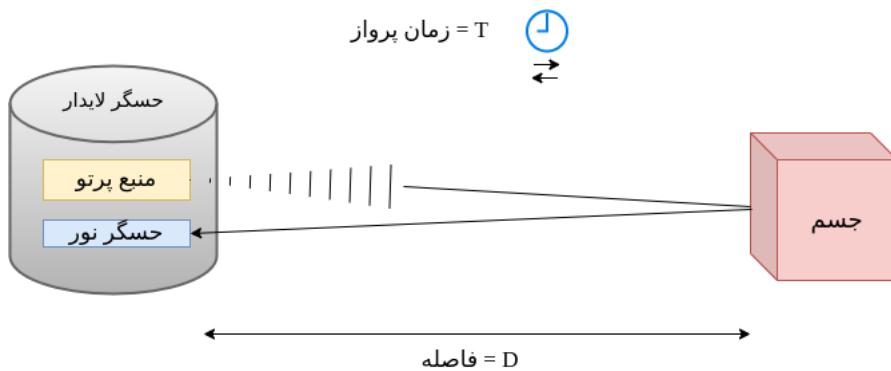
یک سیستم لایدار ممکن است از آینه اسکن^۴، چند پرتو لیزر یا ابزارهای دیگر برای اسکن فضای شیء استفاده کند. با توانایی ارائه اندازه‌گیری دقیق فواصل، لایدار می‌تواند برای حل مسائل متنوعی مورد

¹LiDAR = Light Detection and Ranging

²TOF: Time of Flight

³Distance Map

⁴Scan Mirror



شکل ۱-۲ تصویری از نحوه محاسبه فاصله اجسام توسط حسگر لیدار

استفاده قرار گیرد.

در حوزه حسگری از دور^۱، سیستم‌های لایدار برای اندازه‌گیری پراکندگی، جذب یا بازتاب از ذرات یا مولکول‌های موجود در جو استفاده می‌شوند. برای این اهداف، سیستم‌ها ممکن است نیازهای خاصی در خصوص طول موج پرتوهای لیزری داشته باشند. می‌توان غلظت یک گونه مولکولی خاص در جو، به عنوان مثال متان و بار آئروسل^۲ را اندازه‌گیری کرد. همچنین قطرات باران در جو می‌توانند اندازه‌گیری شوند تا فاصله یک توفان و نرخ بارندگی را تخمین بزنند.

سیستم‌های دیگر لایدار محور، مشخصات سطوح سه‌بعدی در فضای جسمی^۳ را ارائه می‌دهند. در این سیستم‌ها، پرتوهای لیزری طیف خاص و مشخصی ندارند. به جای آن، ممکن است طول موج پرتوهای لیزری برای اطمینان از ایمنی چشمی یا جلوگیری از ویژگی‌های طیفی جو انتخاب شود. پرتوی مورد ارزیابی برخورد کرده و توسط یک "هدف سفت" به سمت گیرنده لایدار بازتاب می‌شود.



شکل ۲-۲ حسگر لایدار OS1:64 از شرکت [۲] Ouster

لایدار همچنین می‌تواند برای تعیین سرعت یک جسم هدف، مورد استفاده قرار گیرد. این کار می‌تواند به وسیله تکنیک داپلر^۴ یا اندازه‌گیری فاصله تا یک هدف در مدت زمان کوتاهی انجام شود. به عنوان مثال، سرعت باد جوی و سرعت یک خودرو می‌تواند توسط یک سیستم لایدار اندازه‌گیری شود.

¹Remote Sensing

²Aerosol Loading

³Object Space

⁴Doppler Technique

علاوه بر این، سیستم‌های لایدار می‌توانند برای ایجاد مدل سه‌بعدی از یک صحنه پویا مورد استفاده قرار گیرند، همانند صحنه‌هایی که یک خودروی خودران با آن مواجه می‌شود. این کار در روش‌های مختلفی قابل انجام است اما معمولاً با استفاده از تکنیک‌های اسکن کردن ۳۶۰ درجه و محاسبه زمان پرواز صورت می‌گیرد.

در حال حاضر، به طور متداول برای ایجاد مدل سه‌بعدی از دنیای اطراف حسگر لایدار مورد استفاده قرار می‌گیرد. سیستم‌های مسیریابی خودکار از سیستم‌هایی است که از ابر نقاط ایجاد شده توسط حسگر لایدار استفاده می‌کند. برخلاف تصاویر، ابر نقاط پراکنده هستند: نمونه‌ها به طور بکواخت در فضا توزیع نشده‌اند. لایدارها به عنوان حسگرهای فعال، نیازی به نور محیطی ندارند و از این رو می‌توان با توجه به شرایط آب و هوایی نامساعد، تشخیص قابل اعتمادتری انجام داد [۲۸]. سیستم‌های کوچکتر لایدار حتی در دستگاه‌هایی با اندازه‌های کوچک مانند تلفن‌های همراه نیز یافت می‌شوند.

۲-۱-۱-۲ دستگاه RSU

یک گیرنده و فرستنده است که عموماً در کنار خیابان‌ها و خطوط عابرپیاده مستقر می‌شود. این دستگاه‌ها می‌توانند بر روی یک خوردو نیز مستقر شوند و یا با دست حمل شوند اما تنها زمانی شروع به کار کردن می‌کند که ثابت باشد و در حال حرکت نباشد. RSUs را تنها در منطقه‌ای که مجوز آن را داشته باشند می‌توانند عمل کنند. اما آنها باید توسط دست نیز حمل می‌شوند، می‌توانند بدون مجوز در منطقه‌هایی که تداخلی با بقیه RSUs دارای مجوز و دولتی ندارند، عمل کنند. این دستگاه‌ها داده‌های ترافیکی و آب و هوایی را به سیستم‌های مدیریت کلان شهری می‌فرستند و همچنین از آنها اطلاعات می‌گیرند. همچنین دستورهایی را می‌توانند به ماشین‌ها بدeneند تا از خطر احتمالی و یا ترافیک احتمالی جلوگیری شود. در این پژوهش منظور از RSU، حسگر لایداری است که در کنار خیابان مستقر شده است.

۲-۱-۲ شبکه‌های عصبی

یادگیری ماشین^۲ یک روش موثر برای رسیدن به اهداف هوش مصنوعی است و روش‌های متعددی برای آموزش ماشین جهت دسته‌بندی و پیش‌بینی ارائه شده است. در میان این روش‌ها، یادگیری عمیق با استفاده از شبکه‌ی عصبی به نتایج فوق العاده‌ای در انواع کارها مانند پردازش تصویر، تشخیص چهره و غیره دست یافته است. شبکه‌ی عصبی مجموعه‌ای از الگوریتم‌ها است که تلاش می‌کند تا روابط زیربنایی را در مجموعه‌ای از داده‌ها از طریق فرایندی که نحوه‌ی عملکرد مغز انسان را تقلید می‌کند، تشخیص دهد [۳].

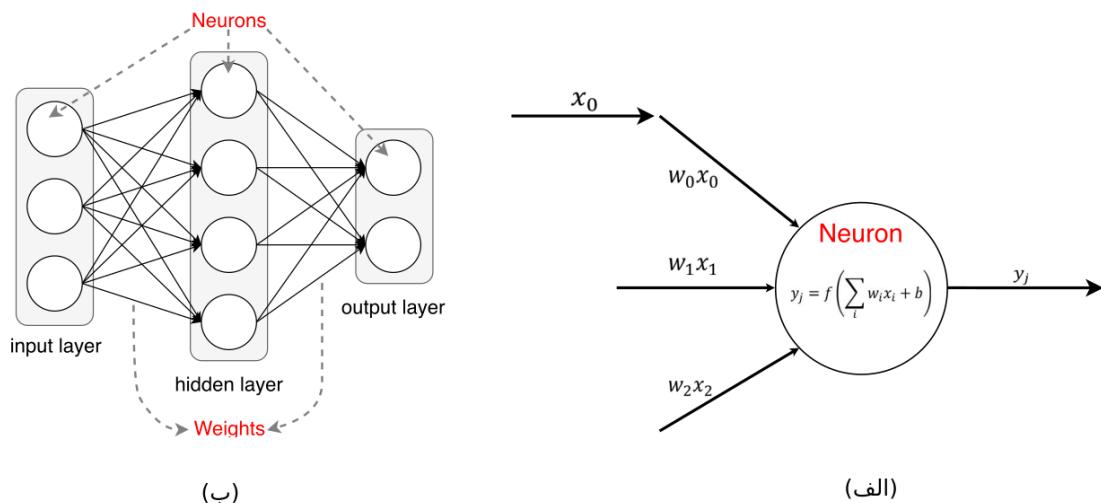
از آن جا که شبکه‌ی عصبی^۳ در یادگیری عمیق، شامل چندین لایه است، به آن شبکه‌ی عصبی

¹Roadside Unit

²Machine Learning (ML)

³Neural Networks

عمیق اطلاق می‌شود. همان‌طور که در شکل ۳-۲ مشاهده می‌شود هر لایه در شبکه‌ی عصبی عمیق از نورون‌هایی تشکیل شده است که قادرند بر اساس داده‌ی ورودی، خروجی غیرخطی تولید نمایند.



شکل ۳-۲ مدل استاندارد شبکه‌ی عصبی عمیق [۳]
الف) معماری یک نورون (ب) لایه‌های شبکه‌ی عصبی عمیق

شبکه‌های عصبی پیچشی^۱ یا به اختصار شبکه‌های پیچشی، دسته‌ای از شبکه‌های عصبی مصنوعی هستند که از دیگر شبکه‌های عصبی در کاربردهایی با ورودی‌های سیگنال تصویر، گفتار یا صوتی عملکرد بهتری نشان می‌دهند و معماری آن‌ها از سازمان‌دهی قشر بینایی حیوانات الهام گرفته شده است. در این مدل، لایه‌ها با استفاده از تابع پیچش^۲، ویژگی‌های ساده ورودی را استخراج می‌کنند. فیلترهای پیچشی در این مدل سهم زیادی در نمایش سطح بالای داده‌های ورودی داشته‌اند و در این معماری شبکه مستقیماً از داده‌ها یاد می‌گیرد و نیاز به استخراج دستی ویژگی را از بین می‌برد. یک نمونه از این شبکه‌ها که برای طبقه‌بندی عکس از آن استفاده شده است در شکل ۴-۲ قابل مشاهده است. این نوع شبکه‌ی عصبی معمولاً دارای سه لایه‌ی اصلی است که عبارت‌اند از:

۱. لایه‌ی پیچش

۲. لایه‌ی ادغام^۳

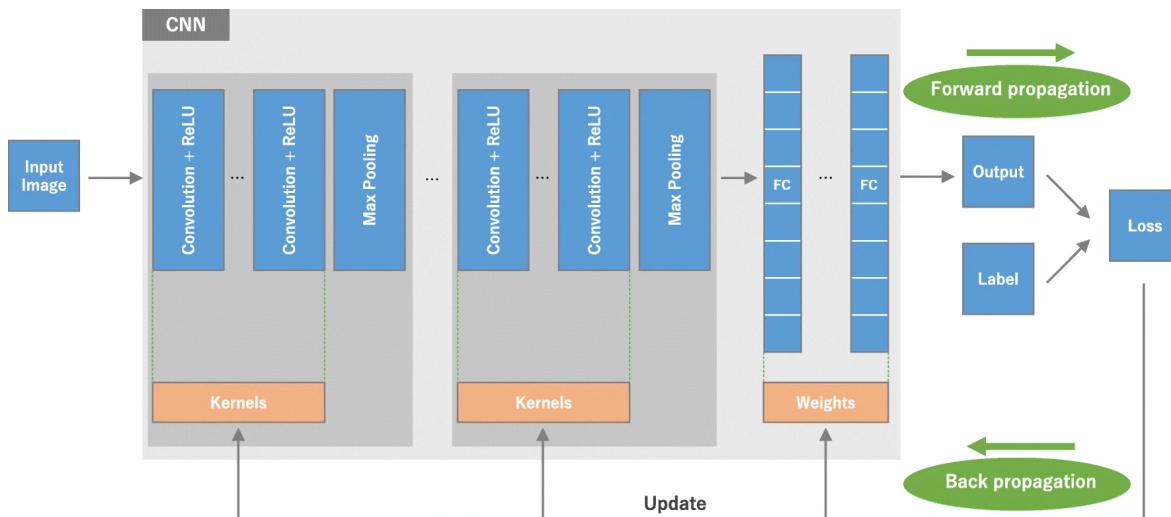
۳. لایه‌ی تماماً متصل^۴

¹Convolutional Neural Network (CNN)

²Convolution

³Pooling Layer

⁴Fully-connected Layer



شکل ۴-۴ نگاهی کلی به معماری استاندارد شبکه‌های پیچشی [۴]

دو لایه‌ی اول و دوم که لایه‌های پیچش و ادغام هستند، استخراج ویژگی را انجام می‌دهند. در حالی که لایه‌ی سوم، ویژگی‌های استخراج شده را در خروجی نهایی، نگاشت می‌کند یا به عبارتی دیگر از ویژگی‌های استخراج شده، استنتاج می‌کند. لایه‌ی پیچشی نقش کلیدی را در شبکه‌های پیچشی ایفا می‌کند، که از مجموعه‌ای از عملیات ریاضی، نظیر پیچش تشکیل شده است [۴].

۲-۱-۲ لایه پیچش

لایه پیچش از اجزای اساسی معماری شبکه‌های عصبی پیچشی است که عملیات استخراج ویژگی را انجام می‌دهد. این عملیات به طور معمول شامل ترکیبی از عملیات خطی و غیرخطی، یعنی عملیات پیچش و تابع فعال‌سازی می‌شود.

پیچش یک عملیات ویژه خطی است که برای استخراج ویژگی‌ها استفاده می‌شود. در این عملیات، یک ماتریس کوچک از اعداد به نام هسته یا فیلتر^۱، بر روی ورودی اعمال می‌شود. ورودی نیز یک ماتریس از اعداد به نام تنسور^۲ است. با قرار گرفتن فیلتر بر روی هر بخش از تنسور، اعداد فیلتر درایه به درایه در مولفه‌های تنسور متناظر ضرب می‌شوند و در نهایت همه اعداد با هم جمع می‌شوند تا یک نقشه ویژگی^۳ شکل بگیرد. اگر یک تنسور دو بعدی از یک تصویر به نام I و یک فیلتر به نام k داشته باشیم، عملیت پیچش طبق فرمول زیر انجام می‌شود:

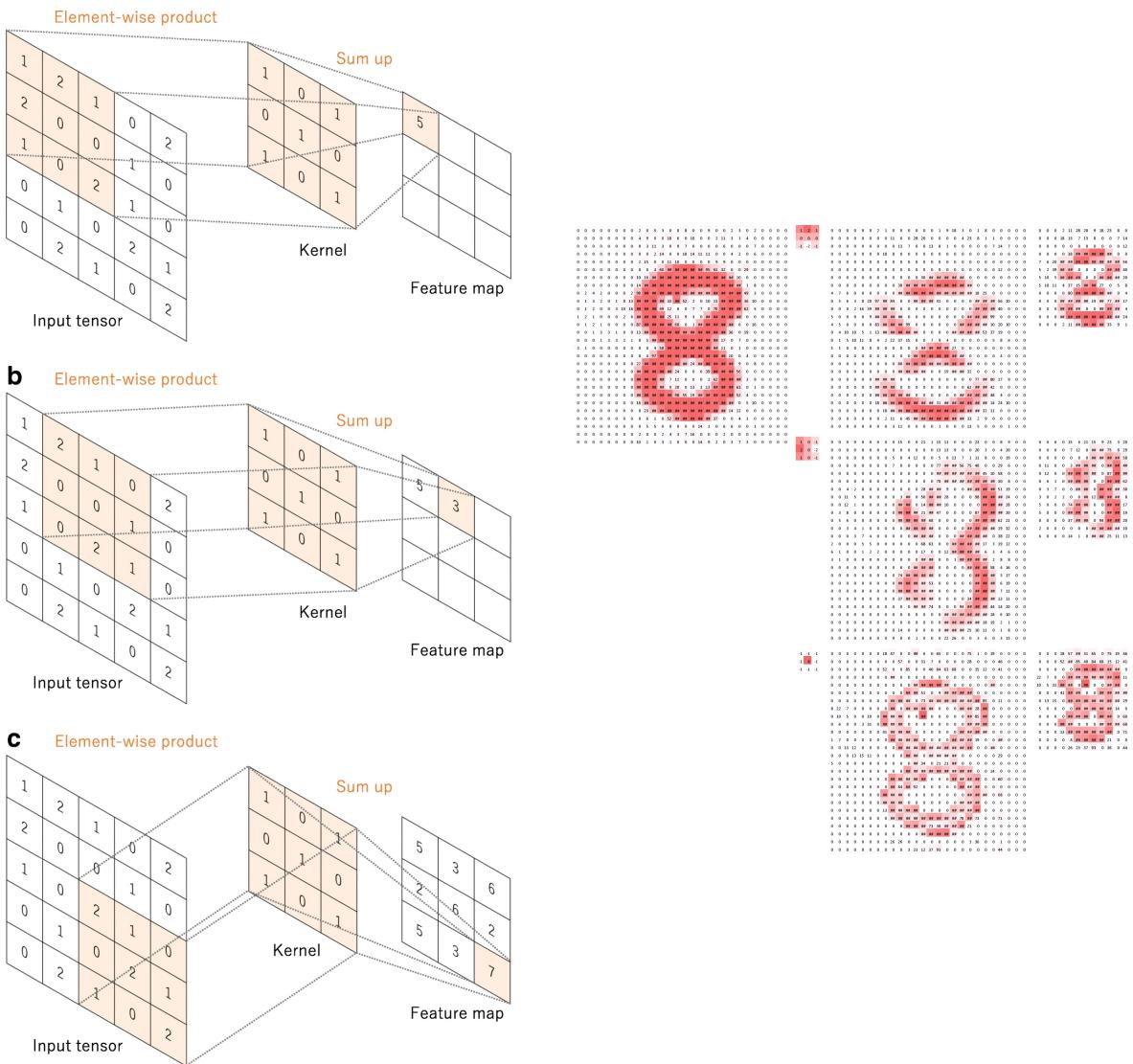
$$\sum_m \sum_n i(m, n)k(i - m, j - n) \quad (۲-۲)$$

¹Kernel (Filter)

²Tensor

³Feature Map

این پروسه با فیلترهای مختلف تکرار می‌شود که هر کدام ویژگی‌های مختلفی از تنسور را استخراج می‌کنند. ابرپارامترهایی^۱ که نتیجه عملیات پیچش را تحت تاثیر قرار می‌دهند، ابعاد و تعداد ماتریس‌های



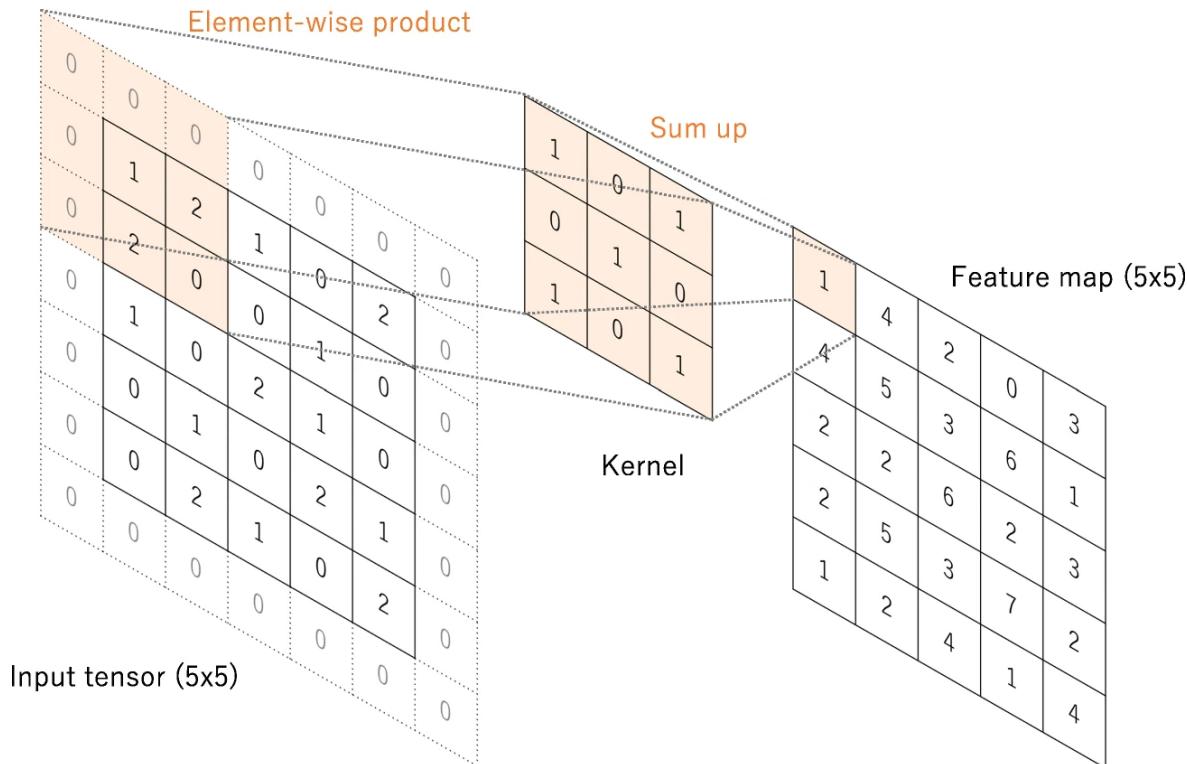
[۴] شکل ۵-۲ عملیت پیچش با فیلترهای 3×3 و گام ۱ در سمت راست شکل، نقشه‌های ویژگی فیلترهای مختلف را مشاهده می‌کنیم.

فیلتر است. عموماً ابعاد فیلترها 3×3 است، اما بعضی اوقات از ابعاد 5×5 و 7×7 نیز استفاده می‌شود. تعداد فیلترها عموماً دلخواه است و عمق نقشه‌های ویژگی را تحت تاثیر قرار می‌دهد. عملیات پیچشی که در بخش ۲-۱-۲ دیده می‌شود، نمی‌گذارد که ماتریس فیلتر از ماتریس تنسور خارج شود. در واقع نقطه مرکز فیلتر هیچ‌گاه بر روی درایه‌های گوشه‌ای تنسور قرار نمی‌گیرد. این باعث می‌شود که ابعاد نقشه‌ی ویژگی حاصل، کوچک‌تر از ابعاد تنسور باشد. لایه‌گذاری^۲، که معمولاً لایه‌گذاری

¹Hyperparameter

²Padding

صفر^۱ است، یک روش برای حل این مشکل است. در این روش سطرها و ستون‌های تماماً صفر به دور ماتریس تنسور اضافه می‌شوند. با این کار، مرکز فیلتر بر روی مولفه‌های گوشه ماتریس نیز جای می‌گیرد و نقشه ویژگی حاصل، هماندازه تنسور خواهد شد. عموم مدل‌های عمیق شبکه‌های پیچشی، از لایه‌های پیچش زیادی استفاده می‌کنند و اگر از تکنیک لایه‌گذاری صفر استفاده نکنند، نقشه ویژگی آنها کوچکتر و کوچکتر می‌شود و عملابدرد نخواهد خورد.



شکل ۶-۲ یک عملیات پیچش با لایه‌گذاری صفر [۴]

طبق شکل ۶-۶، مشاهده می‌کنیم که با لایه‌گذاری صفر، ابعاد نقشه ویژگی هم اندازه تنسور ورودی است. اما بایستی دقت کرد که برای فیلترهای بزرگ‌تر و گام‌های^۲ بیشتر، نیاز به لایه‌گذاری صفر بیشتری خواهیم داشت. به فاصله بین دو عملیات اعمال فیلتر در یک سطر را گام می‌گویند. اندازه گام عموماً ۱ است، اما گام‌های بیشتر از ۱ نیز در مواقعی که نیاز به نمونه‌کاهی^۳ نقشه ویژگی باشد، استفاده می‌شود. عملیات پیچش سه‌بعدی نیز به همین شکل است با این تفاوت که فیلترها سه‌بعدی هستند و در راستای عمق تنسور سه‌بعدی ورودی نیز حرکت می‌کنند.

شبکه‌های عصبی پیچشی، دارای دامنه‌ی گسترده‌ای از کاربردها هستند و به خارج از تشخیص تصویر^۴ گسترش یافته‌اند. از یادگیری سلسه‌وار ویژگی‌های آنها نیز در پردازش زبان طبیعی^۵ و تجزیه و تحلیل

¹Zero Padding

²Stride

³Downsampling

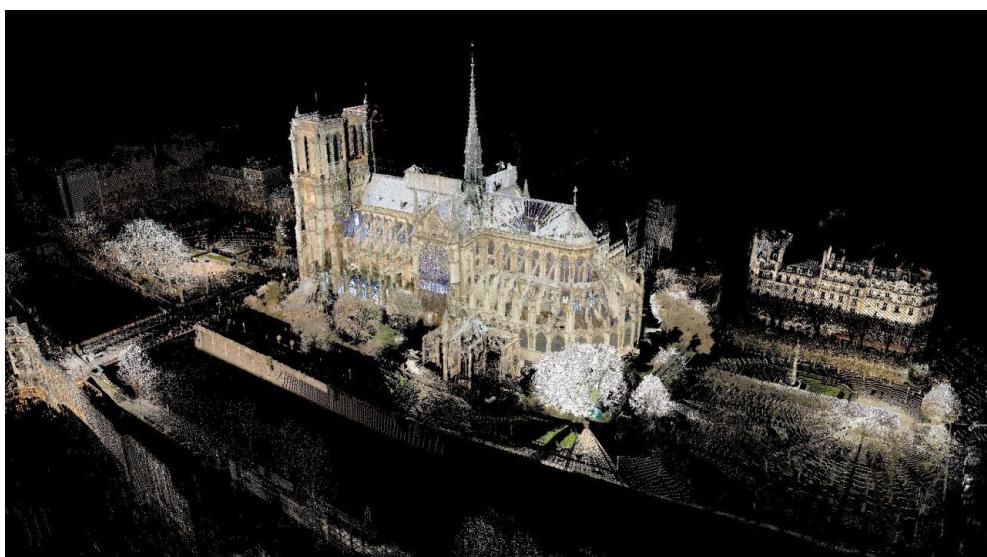
⁴Image Recognition

⁵Natrual Language Processing (NLP)

گفتار^۱ بهره‌برداری می‌شود. علاوه بر این، شبکه‌های عصبی پیچشی جلوه‌دار تازه‌ترین پیشرفت‌ها در خودروهای خودران بوده‌اند، جایی که داده‌ها از حسگرها‌یی مانند لایدار و دوربین‌ها پردازش می‌شود، تا تصمیم‌گیری بلادرنگ در ستاریوهای مختلف رانندگی را ممکن کند. با توسعه مدام معماری‌های شبکه‌های عصبی پیچشی و تکنیک‌های بهینه‌سازی، قابلیت‌های آنها برای مقابله با وظایف به تدریج پیچیده‌تر، گسترش یافته است. با پیشرفت تکنولوژی، شبکه‌های عصبی پیچشی همچنان در خط مقدم تحقیقات هوش مصنوعی قرار دارند و توانایی بی‌نظیری را ارائه می‌دهند تا جنبه‌های مختلفی از زندگی روزمره ما را بهبود دهند و اتوماسیون بخش‌های مختلفی از زندگی را امکان‌پذیر کنند.

۳-۱-۲ ابر نقاط و تشخیص اجسام سه‌بعدی

در مدل‌سازی سه‌بعدی، ابر نقاط مجموعه‌ای از داده‌های نقطه‌ای در یک سیستم مختصات سه‌بعدی است که به طور معمول به عنوان دستگاه مختصات کارتزین^۲ یا محورهای XYZ شناخته می‌شود. هر نقطه نمایانگر یک اندازه‌گیری فضایی واحد بر روی سطح شیء است. با تجمع این نقاط، ابر نقاط کل سطح



شکل ۷-۲ اسکن رنگی لایدار از ساختمان معروف نوتردام پاریس که با استفاده از آن این ساختمان را بازسازی کردند.

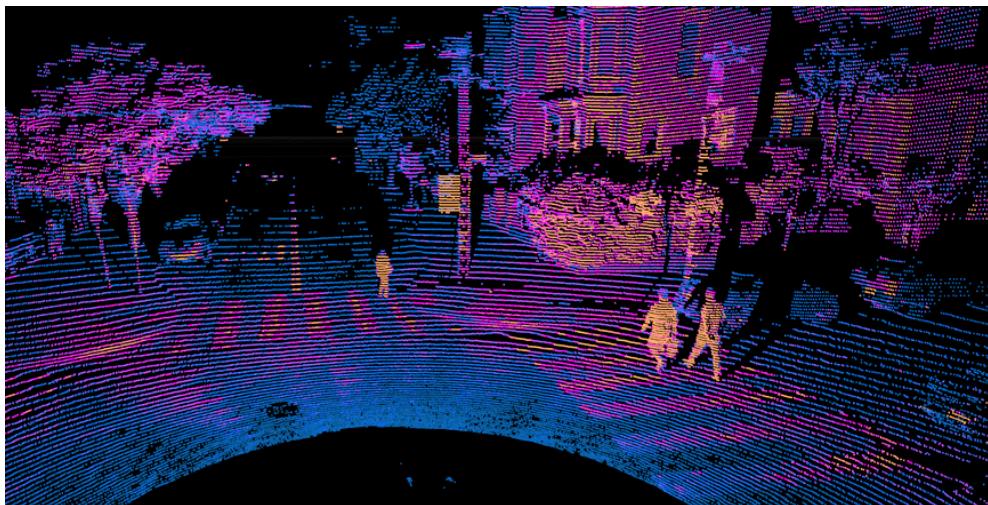
خارجی یک شیء را می‌تواند نمایان کند. اگر مقادیر رنگ هر نقطه نیز اندازه‌گیری شود، اطلاعات رنگی نیز می‌تواند به ابر نقاط افزوده شود. ابر نقاط با استفاده از یک اسکنر سه‌بعدی، لایدار یا نرم‌افزار فتوگرامتری^۳ ایجاد می‌شود.

در حوزه‌ی خودروهای خودران و سیستم‌های مدیریت هوشمند ترافیک نیز از لایدارها استفاده می‌شود تا ابر نقاط حاصل از فضای دور ماشین یا جاده‌ها به عنوان ورودی به مدل‌های هوش مصنوعی داده شود

¹Speech Recognition

²Cartesian Coordinate System

³Photogrammetry



شکل ۲-۸ اسکن لایدار OS1:64 از شرکت Ouster که بر روی یک ماشین نصب شده است.

و اجسام داخل این فضا شناسایی شوند.

۱-۳-۱-۲ تشخیص اجسام سه بعدی

ادراک فضای سه بعدی یک پیش‌نیاز در حوزه رانندگی خودکار و دوچلوبی دیجیتال است. درک کامل از آنچه که در حال حاضر در جلوی حسگر رخ می‌دهد، تسهیل دهنده‌ای برای اجزای پایین-دست^۱ است تا به موقع واکنش نشان دهند. این همان چیزی است که تشخیص اجسام سه بعدی^۲ را هدف قرار می‌دهد. تشخیص اجسام سه بعدی، فضای اطراف ما را از طریق اختصاص یک برچسب^۳، تعیین شکل اجسام با کشیدن کادر محصور کننده^۴، و اعلام فاصله اجسام از حسگر توصیف می‌کند. علاوه بر این، تشخیص سه بعدی حتی زاویه‌ی جهت‌گیری^۵ اجسام را به ما می‌دهد. این اطلاعات بدست آمده از پشتی ادراک^۶، این امکان را فراهم می‌کند که مدل‌های برنامه‌ریزی پایین-دست تصمیم‌گیری کنند [۵].

شکل ۹-۲ یک بررسی اجمالی از تشخیص اجسام سه بعدی با استفاده از ابر نقاط و تصویر را نشان می‌دهد. مشاهده می‌کنیم که چالش‌هایی در تشخیص اجسام سه بعدی داریم:

- a) عدم دریافت نقطه: هنگامی که سیگنال‌های لایدار نتوانند از سطح شیء بازتاب شوند و به حسگر برگردند.

- b) مسدودی از بیرون: هنگامی که سیگنال‌های لایدار توسط موائع اطراف مسدود شوند.

¹Downstream

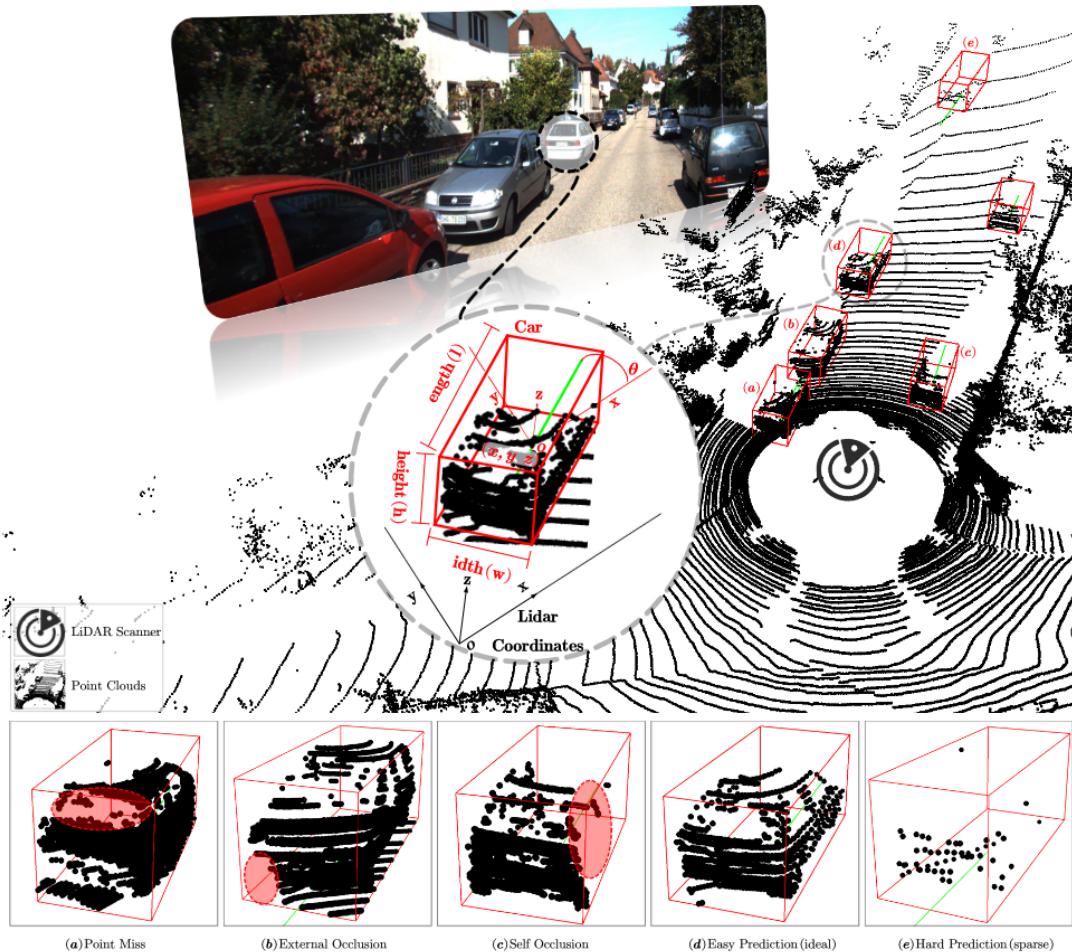
²3D Object Detection

³Label

⁴Bounding Box

⁵Orientation

⁶Detection Stack



شکل ۹-۲ بررسی اجمالی وظیفه تشخیص اجسام سه بعدی از تصاویر و ابر نقاط [۵]

- ۳) مسدودی از درون: هنگامی که طرف نزدیک شی طرف دیگر آنرا مسدود کند، که ابر نقطه جسم را در عمل به ابر نقطه‌ای ۲.۵ بعدی تبدیل می‌کند.

مشاهده می‌کنیم که پیش‌بینی کادر محصور‌کننده در مورد (d) به مراتب آسان‌تر از مورد (e) است؛ زیرا اجسامی که در فواصل دورتر از لایدار هستند، نقاط پراکنده و به اصطلاح تنک‌تری^۱ دارند. هدف اصلی تشخیص اجسام سه‌بعدی، تشخیص اجسام با ایجاد کادر محصور‌کننده سه‌بعدی و اختصاص برچسب به آنها است. برای حل این مسئله دو مدل متداول تشخیص اجسام سه‌بعدی، یعنی تصاویر و ابر نقاط، با چالش‌های مختلفی روبرو می‌شوند. با استفاده تنها از تصاویر، چالش اصلی عدم وجود اطلاعات عمق اجسام است؛ زیرا با وجود پیشرفت‌های زیاد در زمینه بازیابی عمق از تصاویر، این مسئله هنوز یک مسئله با پیچیدگی بالا و حل نشده محسوب می‌شود. همچنین، یک شی در وضعیت‌های سه‌بعدی مختلف ممکن است به نمایش‌های تصویری مختلفی منجر شود، که برای یادگیری نمایش کادرها سودمند نیست. علاوه بر این، از آنجا که دوربین حسگری غیرفعال^۲ است، تصاویر طبیعتاً نسبت

¹Sparse

²Passive Sensor

به روشنایی (به عنوان مثال در شب) یا شرایط جوی (هوای بارانی) تاثیرپذیرند. از سوی دیگر، با استفاده تنها از ابر نقاط، مواجهه با چالش‌های مربوط به یادگیری بازنمایی^۱ کادر محصورکننده‌ی اجسام رخ می‌دهد. طبیعت ابر نقاط، بی‌نظمی و تنک بودن است، به همین دلیل اعمال پیچش بر روی آن، منجر به "از دست رفتن اطلاعات شکل و واریانس در ترتیب نقاط" می‌شود. به علاوه، ابر نقاط در عمل دو و نیم بعدی هستند که مشکلات آن در شکل ۹-۲ قابل مشاهده است. با حذف المان استفاده از شبکه‌های عصبی پیچشی، یک راه برای نمایش ابر نقاط، نمایش آنها به عنوان شبکه‌های از واکسل‌ها^۲ می‌باشد. معصل این روش این است که افزایش اندازه مکعب واکسل، باعث از دست دادن وضوح و به تبع آن کاهش دقت مکان‌یابی می‌شود؛ در حالی که کاهش اندازه آن، پیچیدگی و استفاده از حافظه را افزایش می‌دهد. راه دیگر نمایش ابر نقاط به عنوان مجموعه‌هایی از نقاط^۳ می‌باشد. با این حال، حدود ۸۰ درصد زمان اجرا توسط بازیابی نقاط اشغال می‌شود که سرعت را به شدت کند می‌کند. با داشتن همزمان هم تصاویر و هم ابر نقاط، مشکل دشوار اغلب اوقات مطابقت معنایی^۴ بین تصویر و ابر نقاط است. تصاویر و ابر نقاط دو منبع داده غیرهمگن^۵ هستند که توسط زاویه دید دوربین و زاویه دید سه‌بعدی واقعی لایدار نمایش داده می‌شوند. نتیجه تطابق نقطه‌ای بین نقاط لایدار و پیکسل‌های تصویر منجر به محو شدن ویژگی^۶ می‌شود [۵].

از آنجایی که در این پژوهش از دوربین برای تشخیص اجسام استفاده نشده است، به روش‌های تشخیص اجسام سه‌بعدی مبتنی بر ابر نقاط روی آورده شده است.

۲-۳-۱-۲ روش‌های مبتنی بر ابر نقاط

شبکه‌های عصبی پیچشی همواره به دلیل قابلیت بهره‌برداری از همبستگی‌های^۷ محلی در فضای همسایگی پیکسل‌های موجود در شبکه‌های منظم متراکم (مانند تصاویر)، در علم بینایی ماشین به کار گرفته شده‌اند. اما نقاط داخل ابر نقاط به صورت پخش شده و تنک، با ساختاری نامنظم و بدون ترتیب می‌باشند. به عبارت دیگر، استفاده مستقیم از عملگر پیچش بر روی ابر نقاط، به اعوجاج بازنمایی آنها منجر خواهد شد. برای رفع مشکلات ذکر شده، برخی از مقالات ورودی‌های خود را به شکل واکسلی تجزیه و تحلیل می‌کنند تا با عملگر پیچش قابل تطبیق باشند، در حالی که کارهای دیگر به آخرین پیشرفت‌ها در زمینه یادگیری ویژگی‌ها و معنای بین نقاط در مجموعه نقاط، رجوع می‌کنند. بر اساس راه کارهای یادگیری بازنمایی ابر نقاط، این مطالعات به سه گروه زیر تقسیم می‌شوند:

۱. روش‌های مبتنی بر واکسل^۸

¹Representation Learning

²Voxel Grid

³Point Sets

⁴Semantic Alignment

⁵Heterogeneous

⁶Feature Blurring

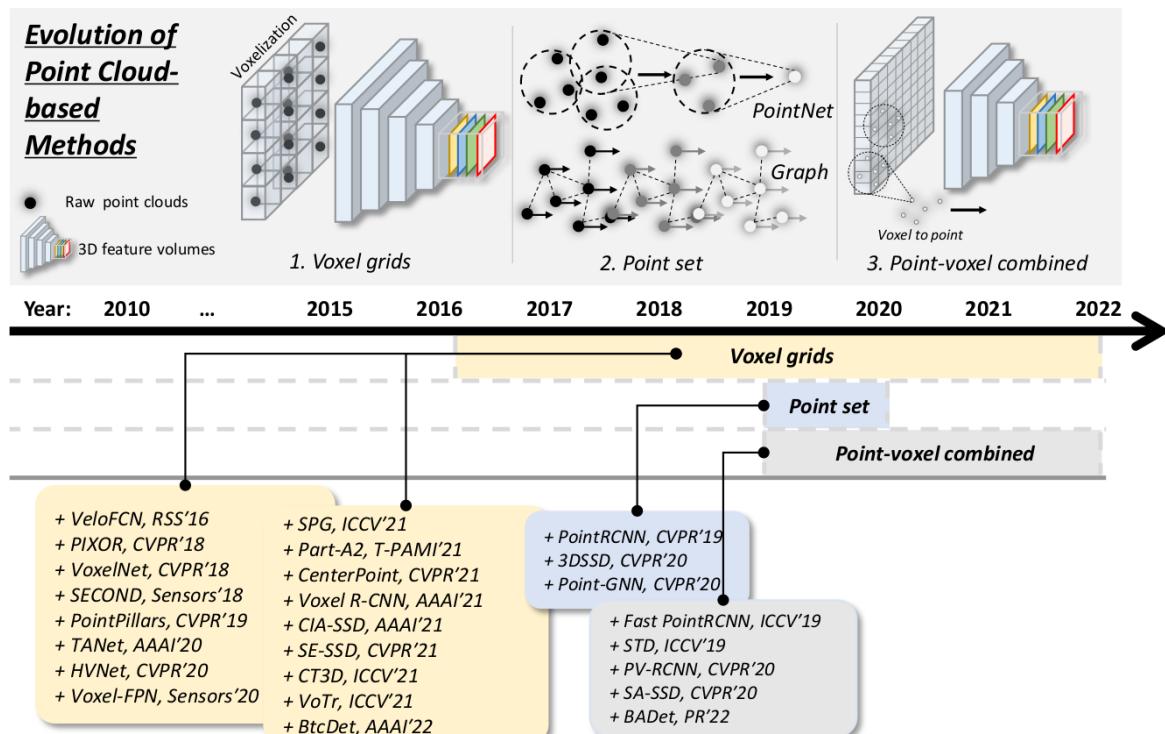
⁷Correlation

⁸Voxel Based Methods

۲. روش‌های مبتنی بر نقطه^۱

۳. روش‌های مبتنی بر واکسل-نقطه^۲

همانطور که در شکل ۱۰-۲ مشاهده می‌کنیم، سیر تکامل روش‌های مختلف تشخیص مبتنی بر ابر نقاط قابل مشاهده هستند. روش‌های مبتنی بر نقطه و مبتنی بر واکسل-نقطه نسبتاً جدید هستند و به نظر می‌آید که در آینده مدل‌های بیشتری را در این گروه‌ها مشاهده خواهیم کرد [۵].



شکل ۱۰-۲ سیر زمانی پیشرفت روش‌های مبتنی بر ابر نقاط [۵]

۳-۳-۱-۲ روش‌های تشخیص مبتنی بر واکسل

این گروه از روش‌ها، ابر نقاط نامنظم را به شبکه‌های مرتب و فشرده دو بعدی و سه بعدی تبدیل می‌کنند. سپس آن‌ها را به شکل نمای دید پرندۀ^۳ دو بعدی تبدیل می‌کنند تا شبکه‌های پیچشی همانند تصاویر با آن برخورد کنند و توابع پیچش موثر و سریع تری داشته باشند. به عنوان مثال، مدل VoxelNet ابر نقاط را به شبکه‌های واکسلی متراکم تقسیم‌بندی می‌کند و از شبکه‌های عصبی پیچشی سه بعدی برای پیچش در امتداد هر بعد استفاده می‌کند، اگرچه این رویکرد با هزینه محاسباتی و حافظه زیادی همراه است. از سوی دیگر، مدل SECOND از عملیات پیچش پراکنده^۴ استفاده می‌کند تا محاسبات غیرضروری ناشی

¹Point Based Methods

²Voxel-Point Based Methods

³Bird's Eye View (BEV)

⁴Sparse Convolution

از لایه‌گذاری صفر^۱ در واکسل‌ها را کاهش دهد. مدلی معروف به نام PointPillars واکسل‌ها را به شکل ستون‌های دربرگیرنده کل نقاط در راستای عمودی، در نمای دید پرندۀ امتداد می‌دهد. البته این روش به قیمت کاهش وضوح عمودی تمام می‌شود که بر کیفیت یادگیری بازنمایی تاثیر می‌گذارد، اما سرعت چشم‌گیری دارد [۵].

برای مهار کردن این محدودیت PointPillars، برخی از روش‌ها به شبکه‌های واکسلی بازگشتند. مدل Voxel R-CNN ویژگی‌های واکسلی را با استفاده از پیچش‌های سه‌بعدی برای بهبود پیشنهاد^۲، جمع‌آوری می‌کند، در حالی که مدل SE-SSD همزمان یک شبکه عصبی دانش‌آموز را تحت راهنمایی دانش تصفیه‌شده شبکه عصبی استاد، نظارت می‌کند. روش‌های دیگر، استراتژی‌های چندمقیاسه واکسلیزه کردن را به جمع‌آوری ویژگی^۳ اضافه می‌کنند، از وابستگی‌های زمینه‌ای بین واکسل‌هایی با فواصل زیاد^۴ بهره‌مند یا یادگیری شکل^۵ را برای مقابله با مسائل مربوط به کیفیت ابر نقاط تحت تأثیر عواملی مانند شرایط جوی نامساعد، انسداد^۶ و انتهای افق دید لایدار به کار می‌گیرند. لازم به ذکر است که روش‌های مبتنی بر شبکه‌های واکسلی، از بازنمایی نمای دید پرندۀ مرتبط بهره‌مند یا مانند که ابهام در مقیاس را کاهش می‌دهد و حداقل انسدادها را دارد [۵].

۴-۳-۱-۲ روش‌های تشخیص مبتنی بر نقطه

این دسته از روش‌ها از عامل‌هایی استفاده می‌کنند که وابسته به ترتیب نیستند و در نتیجه امکان شناسایی ساختارهای محلی و الگوهای پیچیده در ابر نقاط را بدون نیاز به کوانتش^۷ ممکن می‌سازند. این رویکرد، خصوصیات هندسی و ذاتی ابر نقاط خام را حفظ می‌کند. به عنوان مثال، مدل PointRCNN از یک تکنیک مشابه مدل معروف PointNet برای یادگیری نشانه‌های معنادار^۸ نقاط گوشه‌ای جسم تشخیص داده شده استفاده می‌کند، که امکان تولید پیشنهادات سه‌بعدی^۹ برای شکل جسم را به صورت گام‌به‌گام ممکن می‌سازد. لازم به ذکر است که روش‌های مجموعه‌ی PointNet، به طور گسترده‌ای از ابزارهای نمونه‌افزایی^{۱۰} و پخش اطلاعات مفهومی در نقاط کلیدی بهره می‌برند.

از طرف دیگر، رویکرد مدل 3DSSD برای افزایش بهره‌وری سیاست نمونه‌برداری نقاط، نگرشی تازه به فاصله ویژگی‌های^{۱۱} نقاط به همراه فاصله اقلیدسی^{۱۲} آن‌ها می‌اندازد. مدل Point-GNN با الهام گرفتن

¹Zero Padding

²Proposal Refinement

³Feature Extraction

⁴Long Range Contextual Dependancies

⁵Shape Learning

⁶Occlusion

⁷Quantization

⁸Semantic Cues

⁹3D Proposals

¹⁰Upsampling

¹¹Feature Distance

¹²Euclidean Distance

از عملکرد موفق ابر نقاط در وظایفی مانند طبقه‌بندی^۱ و بخش‌بندی معنایی^۲، تصمیم‌گیری را بر اساس گراف‌های همسایه محلی نقاط ساخته شده‌اند، انجام می‌دهد. در این گراف‌ها، هر گره به صورت تدریجی اطلاعات مفهومی را از نواحی مجاور خود جمع‌آوری می‌کند. با این حال، لازم به ذکر است که روش‌های مبتنی بر نقاط، ممکن است مصرف محاسباتی بسیار بالایی داشته باشند و با پیچیدگی جستجوی توپی^۳ ($O(k|\chi|)$) مواجه باشند. برخی رویکردها این چالش را با پیاده‌سازی گروه‌بندی چند مقیاسی و چند رزولوشنی حل می‌کنند؛ اما این روش با افزایش تعداد نقاط در مجموعه χ ، سربار پردازشی خواهد داشت [۵].

۱-۳-۵ روش‌های تشخیص مبتنی بر واکسل- نقطه

روش‌های موجود در این دسته، یک رویکرد نوآورانه معرفی می‌کنند که نقاط قوت دو روش قبلی را ترکیب می‌کند. روش‌های مبتنی بر واکسل از لحاظ محاسباتی کارآمد هستند، اما ممکن است الگوهای ریز همانند انسان‌ها را نادیده بگیرند و بر روی بهبود دقت در هر تکرار^۴ یادگیری، تاثیر منفی بگذارند. مثقالاً، روش‌های مبتنی بر نقاط معمولاً تاخیر نسبی بیشتری دارند اما در حفظ ناهنجاری و محلی بودن ابر نقاط بهتر عمل می‌کنند. به عنوان مثال، مدل STD از PointNet++ برای متراکم‌سازی اطلاعات معنادار بین نقاط پراکنده استفاده می‌کند، سپس آنها را واکسلیزه می‌کند و یک بازنمایی متراکم‌تر برای بهبود دقت ایجاد می‌کند. مدل PV-RCNN، با یکپارچه کردن بهره‌وری اطلاعات مفید با استفاده از عملیات پیچش پراکنده سه‌بعدی^۵، و استخراج داده‌های حائز اهمیت و معنادار با استفاده از مجموعه‌های PointNet شبیه، فرآیند یادگیری معانی متمازیتر را تقویت می‌کند. مدل SA-SSD، با استفاده از یک شبکه عصبی کمکی، ویژگی‌های واکسل‌هایی که با استفاده از پیچش پراکنده سه‌بعدی بدست آمده است را نسبت به ساختار نقاط خود آگاه‌تر می‌کند. یعنی این شبکه عصبی کمکی، ساختار اشکال داخل واکسل‌ها را حدس می‌زند و سعی می‌کند اشکالی که نقاط آنها ناقص هستند را تکمیل کند [۵].

۱-۳-۶ تشخیص اجسام سه‌بعدی در دو قلوهای دیجیتال

در حال حاضر، به نظر می‌آید که روش‌های مبتنی بر واکسل انتخاب اصلی محققین هستند، همانطور که در شکل ۱۰-۲ نیز مشاهده می‌شود. روش‌های مبتنی بر واکسل، قابلیت انعطاف بالا برای پیاده‌سازی در سخت‌افزار را دارند و دقتی قابل توجه و زمان پاسخ نسبتاً کمتری دارند. از سوی دیگر، روش‌های مبتنی بر نقطه توانایی حفظ ویژگی‌های محلی فضایی^۶ ابرنقطاً را دارند، اما نسبت به روش‌های مبتنی بر

^۱Classification

^۲Semantic Segmentation

^۳Ball Query Complexity

^۴Iteration

^۵3D Sparse Convolutions

^۶Spatially-Local Features

واکسل زمان بیشتری برای پیشروی رو به جلو^۱ دارند. با مقایسه سه روش تشخیص مبتنی بر ابر نقاط، آشکار می‌شود که روش‌های مبتنی بر واکسل، همچنان به عنوان جهتی بسیار امیدبخش‌تر در کاربردهای بلاذرنگ، به عنوان مثال در حوزه رانندگی خودروهای خودران و دوقولوهای دیجیتال، مورد مطالعه قرار گرفته شده‌اند.

۴-۱-۲ مجموعه داده مدل‌های تشخیص سه‌بعدی

دسترسی به مجموعه داده‌هایی^۲ با مقیاس‌های عظیم، نقش مهمی در پیشرفت حوزه‌های مختلف هوش مصنوعی ایفا کرده است. در زمینه تشخیص اجسام سه‌بعدی، چندین مجموعه داده عمومی به تحقیقات کمک کرده‌اند. مجموعه داده‌های معروفی از جمله KITTI [۲۹]، nuScenes [۳۰]، و Waymo Open [۳۱] وجود دارند که به عنوان منابع گسترده برای کاربردهای مرتبط با خودروهای خودران به کار می‌روند. این مجموعه داده‌ها در ابعاد مختلف از جمله اندازه^۳، تنوع^۴، مزايا و معایب با یکدیگر تفاوت دارند.

۱-۴-۱-۲ اندازه

مجموعه داده KITTI شامل ۲۰۰،۰۰۰ کادر محصور کننده حاشیه‌نویسی شده^۵ و توزیع شده در ۱۵،۰۰۰ فریم می‌شود. این مجموعه داده برای آموزش و آزمون، نمونه‌هایی تقریباً مشابه دارد. در مقابل، مجموعه داده nuScenes شامل ۴.۱ میلیون کادر محصور کننده توزیع شده در ۴۰،۰۰۰ فریم است و مجموعه‌هایی جدا و متمایز برای آموزش، اعتبارسنجی و آزمون دارد. مجموعه داده Waymo Open با بیش از ۱۱۲ میلیون کادر محصور کننده در ۲۰۰،۰۰۰ فریم و داده‌هایی با مقیاس بزرگ برای آموزش، اعتبارسنجی و آزمون، از سایرین برجسته‌تر است. لازم به ذکر است که در حالی که حاشیه‌نویسی‌ها برای آموزش و اعتبارسنجی در دسترس هستند، هیچ یک از این مجموعه‌های داده حاشیه‌نویسی‌ای برای مجموعه آزمون ارائه نمی‌دهند و شرکت‌کنندگان باید پیش‌بینی‌های خود را در وبسایت آنلاین مجموعه داده‌ها ارسال کرده و ارزیابی^۶ کنند [۵].

۲-۴-۱-۲ تنوع

مجموعه داده KITTI، داده‌ها را از ۵۰ صحنه ترافیکی مختلف با ۸ کلاس اجسام فراهم می‌کند و ارزیابی آنلاین آن بر روی کلاس‌های خودرو، پیاده‌رو و دوچرخه‌سوار تمرکز دارد. این مجموعه داده صحنه‌ها را بر اساس میزان ارتفاع کادرهای محصور کننده دو بعدی، میزان انسداد در ابرنقاط و میزان برش ابرنقاط به

¹Feedforward

²Datasets

³Size

⁴Diversity

⁵Annotated Bounding Boxes

⁶Benchmark

علت رسیدن به انتهای افق دید لایدار، طبقه‌بندی می‌کند. از طرفی، مجموعه داده nuScenes از ۱،۰۰۰ صحنه ترافیکی در ۲۳ کلاس داده ارائه می‌دهد و در ارزیابی آنلاین، تمرکز خود را بر روی ۱۰ کلاس تشخیص اجسام سه‌بعدی می‌گذارد. مجموعه داده Waymo Open شامل ۱،۱۵۰ صحنه ترافیکی در ۴ کلاس است و ارزیابی آن بر روی سه کلاس مشابه با KITTI تمرکز دارد. لازم به ذکر است که مجموعه داده‌های Waymo Open و nuScenes شامل باران، مه، برف، روز و شب جمع‌آوری کرده‌اند و تنوع بیشتری در سناریوها نسبت به KITTI ارائه می‌دهند، که تنها به روزهای آفتایی محدود است [۵].

جدول ۱-۲ توزیع داده در داده‌های آموزش KITTI. کلاس خودرو ۸۲.۹۹٪ سه کلاس اصلی (خودرو، عابرپیاده، دوچرخه‌سوار) را تشکیل می‌دهد [۵].

کلاس	انسان نشسته	قطار غیره	کامیون	دوچرخه‌سوار	ون	عابرپیاده	خودرو
تعداد	۱۴،۳۵۷	۲۲۴	۳۳۷	۷۳۴	۴۴۸	۱۲۹۷	۲،۰۷

جدول ۲-۱ توزیع داده در داده‌های آموزش nuScenes [۵].

کلاس	اتوبوس	تریلی	کامیون	مخروط ترافیکی	مانع ترافیکی	عابرپیاده	خودرو
تعداد	۴۱۳،۳۱۸	۱۳،۱۶۳	۱۸۵،۸۴۷	۸۲،۳۶۲	۷۲،۸۱۵	۲۰،۷۰۱	۱۲۵،۰۹۵

کلاس	دوچرخه	موتورسیکلت	جرثقیل
تعداد	۱۱،۹۹۳	۹،۴۷۸	۱۰،۱۰۹

۳-۴-۱-۲ مزایا و معایب

این مجموعه داده‌ها تأثیر قابل توجهی بر جامعه علمی داشته‌اند. به ویژه KITTI تأثیر عمیقی در زمینه‌های جمع‌آوری داده‌ها، پروتکل‌ها و ارزیابی داشته است. با این حال، باستی توجه داشت که جمع‌آوری داده‌های KITTI، تنها در روزهای آفتایی انجام شده است. یعنی شرایط روشنایی و جوی که می‌تواند تأثیر زیادی بر سناریوهای واقعی داشته باشد، در نظر گرفته نشده است. این مشکل توسط مجموعه داده‌های Waymo Open و nuScenes حل شده است. این دو مجموعه داده، داده‌های خود را در شرایط مختلف جوی و روشنایی مانند باران، مه، برف، روز و شب جمع‌آوری می‌کنند. علاوه بر این، مجموعه داده‌های واقعی معمولاً از عدم توازن کلاس‌ها^۱ رنج می‌برند، که اکثر کلاس‌های آنها به درصد کوچکی از دسته‌ها تعلق دارند. با توجه به جدول ۱-۲ و جدول ۲-۲، می‌توان این مشکل را در مجموعه داده‌های nuScenes و KITTI مشاهده کرد [۵].

^۱Data Imbalance

۵-۱-۲ معیارهای ارزیابی

در بخش قبلی مشاهده کردیم که مجموعه دادگانی وجود دارند که برای آموزش، آزمون و ارزیابی مدل‌های تشخیص اجسام سه‌بعدی، طراحی شده‌اند. حال به روش‌هایی که این مجموعه‌دادگان با استفاده از آن مدل را ارزیابی می‌کنند، می‌پردازیم. در ابتدا لازم است که مفاهیم را توضیح دهیم:

۱-۵-۱-۲ ماتریس درهم ریختگی

ماتریس درهم ریختگی^۱، ابزاری حیاتی در یادگیری ماشین و آمار است. این ماتریس، خلاصه‌ای از مقایسه

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

V7 Labs

شکل ۱۱-۲ ماتریس در هم ریختگی [۶]

پیش‌بینی‌های مدل با مقادیر واقعی^۲ ارائه می‌دهد. این ماتریس از چهار بخش تشکیل شده است:

۱. مثبت‌های واقعی^۳ (نمونه‌های مثبتی که به درستی تشخیص داده شده‌اند)
۲. منفی‌های واقعی^۴ (نمونه‌های منفی که درستی تشخیص داده شده‌اند)
۳. مثبت‌های کاذب^۵ (نمونه‌های منفی که به اشتباه به عنوان مثبت تشخیص داده شده‌اند)

¹Confusion Matrix

²Ground Truth

³True Positive (TP)

⁴True Negative (TN)

⁵False Positives (FP)

۴. منفی‌های کاذب^۱ (نمونه‌های مثبت که به اشتباه به عنوان منفی تشخیص داده شده‌اند)

دقت ۲-۵-۱-۲

دقت^۲، نشان‌دهنده این است که چه تعداد از نمونه‌هایی که مثبت پیش‌بینی شده‌اند، درست پیش‌بینی شده‌اند و رابطه‌ی آن از روش زیر قابل محاسبه است:

$$Precision = \frac{TP}{TP + FP} \quad (3-2)$$

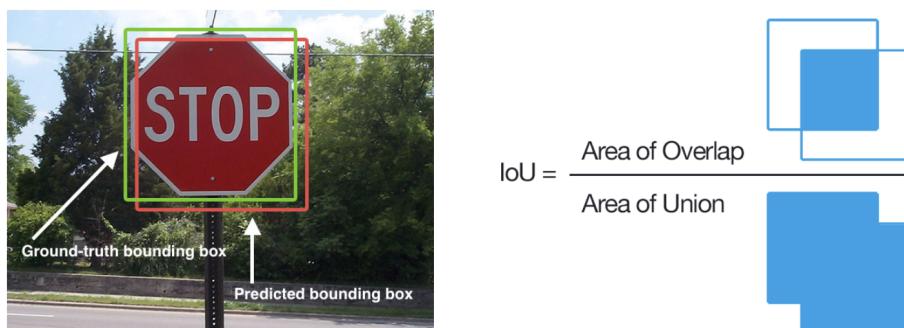
پوشش ۳-۵-۱-۲

پوشش^۳، مشخص می‌کند که چه تعداد از نمونه‌هایی که مثبت بوده‌اند، درست پیش‌بینی شده‌اند. رابطه‌ی آن به شکل زیر نوشته می‌شود:

$$Recall = \frac{TP}{TP + FN} \quad (4-2)$$

نسبت اشتراک بر روی اجتماع ۴-۵-۱-۲

معیار نسبت اشتراک بر روی اجتماع^۴، نشان‌دهنده انطباق مختصات کادر محصور‌کننده تخمین زده شده با کادر واقعی (یا داده‌های واقعی) است.



شکل ۱۲-۲ نسبت اشتراک بر روی اجتماع [۶]

همانطور که در شکل ۱۲-۲ مشاهده می‌کنیم، نحوه محاسبه معیار نسبت اشتراک بر روی اجتماع نشان داده شده است. هدف از این کار، سنجیدن واقعی یا کاذب بودن تشخیص مدل است.

¹False Negatives (FN)

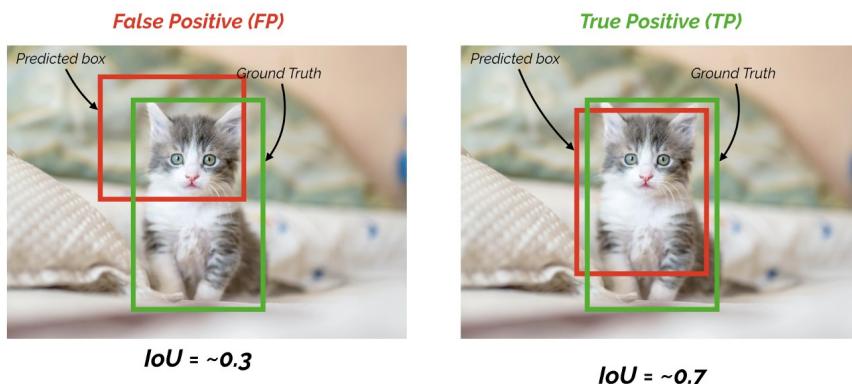
²Precision

³Recall

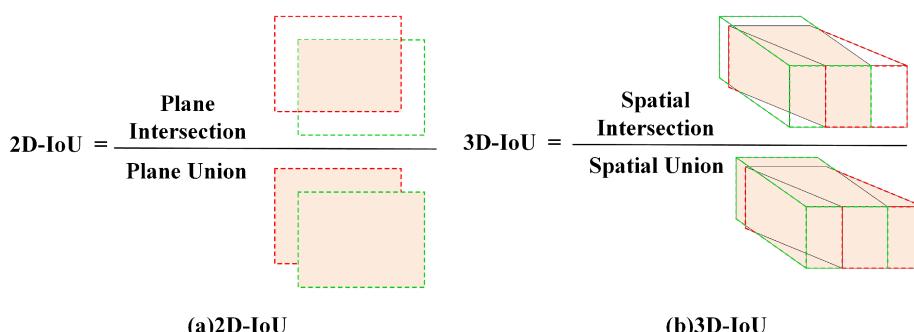
⁴Intersection over Union (IoU)

برای سنجش واقعی یا کاذب بودن، عموماً حد آستانه‌ای^۱ را برای نسبت اشتراک بر روی اجتماع می‌گذارند. به طور مثال در شکل ۱۳-۲، حد آستانه برابر با ۰.۵ است. پس کادر تخمینی‌ای که نسبت اشتراک بر روی اجتماع آن کمتر از این مقدار باشد، مثبت کاذب است و اگر بیشتر از حد آستانه باشد، مثبت واقعی است.

If IoU threshold = 0.5



شکل ۱۳-۲ سنجش واقعی یا کاذب بودن براساس حد آستانه [۶]



شکل ۱۴-۲ نسبت اشتراک بر روی اجتماع در فضای سه‌بعدی و دو بعدی [۷]

این معیار فقط برای تصاویر دوبعدی نیست و در تشخیص اجسام سه‌بعدی نیز استفاده می‌شود. اما محاسبات این نسبت در اشکال سه‌بعدی بسیار دشوارتر است و عموماً از تخمین برای محاسبه حجم مشترک استفاده می‌شود.^۲.

¹Threshold

²<https://pytorch3d.org/docs/iou3d>

۵-۵-۱-۲ میانگین دقت و میانگین دقت درونیابی شده

در تشخیص اجسام سه بعدی و دو بعدی، امتیاز اطمینان^۱ برای هر کادر محصور کننده تخمین زده شده محاسبه می شود. فرض کنیم زیرمجموعه نزولی پیش بینی $\{y_1, \dots, y_n\}$ وجود دارد که به ترتیب امتیاز اطمینان s_i مرتب شده است، پیش بینی y_i مثبت واقعی است اگر نسبت اشتراک بر روی اجتماع کادر محصور کننده i ، از حد آستانه مشخص شده بیشتر باشد. اگر خلاف این باشد، مثبت کاذب است [۵]. نمودار زیگ-زاگی دقت-پوشش این زیرمجموعه، به راحتی قابل رسم است و مقدار میانگین دقت^۲ آن برابر با مساحت زیر نمودار است. اما در عمل، محاسبه مساحت زیر نمودار دشوار است. به همین دلیل محققانی در مسابقه PASCAL VOC، روش هایی برای تسهیل محاسبه میانگین دقت کشف کردند که از میانگین دقت درونیابی شده^۳ استفاده می کند.

برای درک بهتر محاسبه میانگین دقت درونیابی شده، مثالی آورده شده است. فرض کنیم تشخیص دهنده اجسام، خروجی زیر را داده است و آن را براساس امتیاز اطمینان هر تصویر، مرتب کرده است.

input	score	TP or FP
image #1	0.8	FP
image #1	0.35	TP
image #1	0.85	TP
image #2	0.7	TP
image #2	0.55	FP
image #2	0.95	TP
image #2	0.6	FP

Sort based on the scores

score	TP or FP
0.95	TP
0.85	TP
0.8	FP
0.7	TP
0.6	FP
0.55	FP
0.35	TP

شکل ۲ ۱۵-۲ مرتب سازی براساس امتیاز اطمینان کادرهای پیش بینی شده

حال مقادیر دقت و پوشش هر تخمین را به صورت تجمعی^۴ محاسبه می کنیم:
فرمول محاسبه پوشش و دقت به شکل زیر تعریف می شود:

$$P = \frac{acc.TP}{acc.TP + acc.FP} \quad R = \frac{acc.TP}{N} \quad (۵-۲)$$

که N تعداد کادر محصور کننده های واقعی^۵ است. شکل ۲ ۱۶-۲ جدول پوشش و دقت را نشان می دهد که با استفاده از فرمول (۵-۲) محاسبه شده اند.

¹Confidence Score

²Average Precision (AP)

³Interpolated AP

⁴Accumulated

⁵Ground-Truth Bounding Boxes

score	TP or FP	acc. TP	acc. FP	P	R
0.95	TP	1	0	1.00	0.17
0.85	TP	2	0	1.00	0.33
0.8	FP	2	1	0.67	0.33
0.7	TP	3	1	0.75	0.50
0.6	FP	3	2	0.60	0.50
0.55	FP	3	3	0.50	0.50
0.35	TP	4	3	0.57	0.67

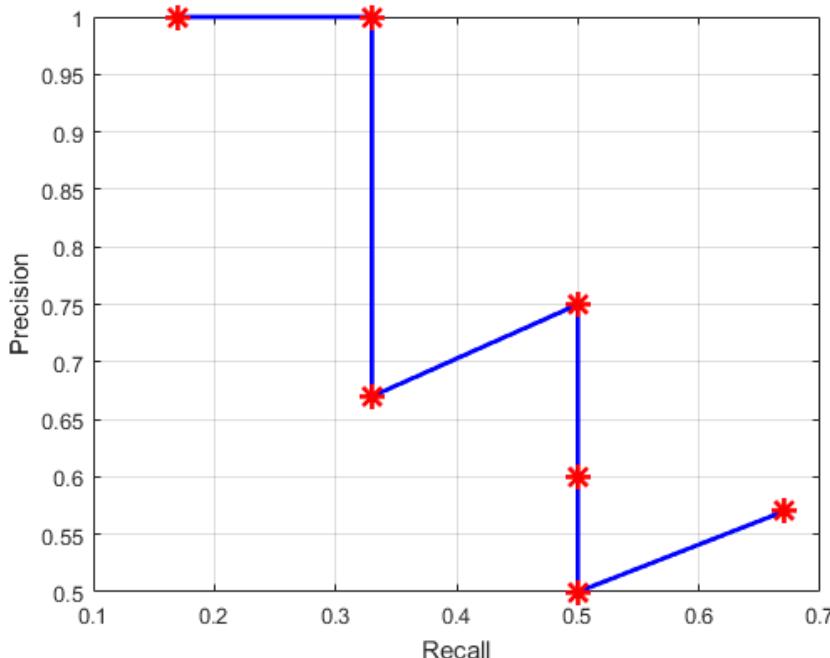
Number of ground truth bounding boxes: 6

acc. TP: accumulated True Positive

acc. FP: accumulated False Positive

شكل ۱۶-۲ جدول محاسبه دقت و پوشش هر تخمین به صورت تجمعی

حال نمودار دقت-پوشش را رسم می‌کنیم. مشاهده می‌کنیم که نمودار به شکل زیگ-زاگی در آمده است. این اتفاق به علت مرتبسازی جدول براساس امتیاز اطمینان رخ داده است.



شكل ۱۷-۲ نمودار دقت-پوشش

برگزارکنندگان مسابقه PASCAL VOC به دنبال راهی بودند که مساحت زیر نمودار دقت-پوشش را بتوان محاسبه کرد. از این رو به روش درونیابی میانگین دقت روی آوردند.

تعریف ۱-۲. معیار پوشش درونیابی شده $AP|_{RN}$ در زیرمجموعه پوشش R محاسبه شده است که

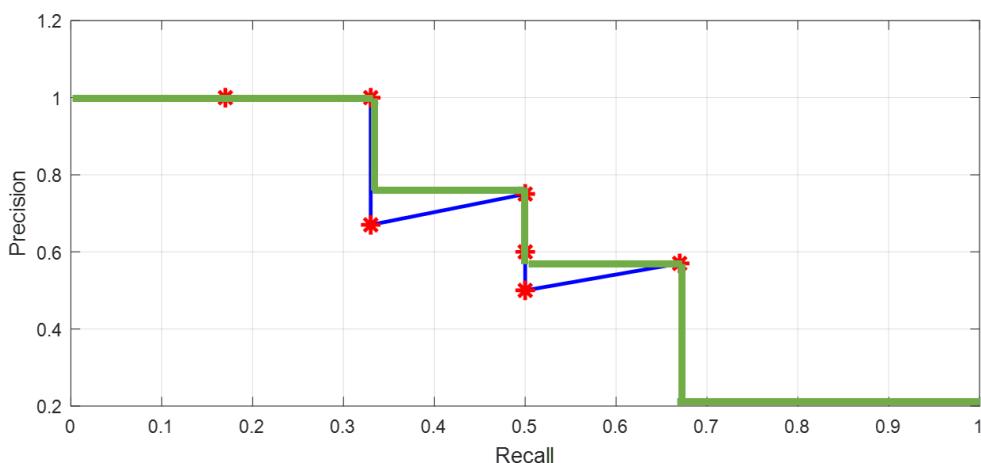
از N سطح پوشش به شکل زیر تشکیل شده است [۵]:

$$AP|_{R_N} = \frac{1}{N} \sum_{r \in R} P_{interpolate}(r) \quad (6-2)$$

که ازای هر سطح پوشش r ، دقت متناظر آن با $R = \left[r_0, r_0 + \frac{r_1 - r_0}{N-1}, r_0 + \frac{2(r_1 - r_0)}{N-1}, \dots, r_1 \right]$ بیشینه سطح پوشش‌های بیشتر یا مساوی r درون‌یابی می‌شود، یا به عبارتی

$$P_{interpolate}(r) = \max_{\tilde{r}: \tilde{r} \geq r} P(\tilde{r}) \quad (7-2)$$

حاصل درون‌یابی مثال زده شده، نمودار زیر است:



شکل ۱۸-۲ نمودار دقت-پوشش درون‌یابی شده

حال با فرض اینکه ۱۱ سطح پوشش داریم (طبق قرارداد VOC که البته در حال حاضر به ۴۰ سطح رسیده است)، میانگین دقت را طبق فرمول (۶-۲) محاسبه می‌کنیم که خواهیم داشت:

$$AP = \frac{1}{11} (P_{interp}(0) + P_{interp}(0.1) + \dots + P_{interp}(0.9) + P_{interp}(1)) \quad (8-2)$$

۶-۵-۱-۲ معیار mAP

معیار mAP، عموماً برای ارزیابی مدل‌های تشخیص اجسام در تصاویر دو بعدی (مانند Fast R-CNN و غیره) استفاده می‌شود. این مقدار با میانگین گرفتن از میانگین دقت^۱ کلاس‌ها محاسبه می‌شود و مقداری بین ۰ تا ۱ دارد.

^۱Average Precision

فرمول محاسبه mAP به شکل زیر است:

$$mAP = \frac{1}{K} \sum_{C_i \in C} AP_{Ci} \quad (9-2)$$

که AP همان میانگین دقت است، C مجموعه کلاس‌ها است و K تعداد کلاس‌ها است. مقدار میانگین دقت می‌تواند به طور عادی با بدست آوردن سطح زیر نمودار محاسبه شود یا از حالت درون‌یابی شده آن یا $AP_{interpolate}$ استفاده شود. عموماً از حالت دوم استفاده می‌شود زیرا نسبت به تغییرات پوشش حساس نیست و نتیجه بهتری می‌دهد.

7-۵-۱-۲ معیار ارزیابی KITTI

مجموعه داده KITTI، از میانگین دقت درون‌یابی شده و استاندارد $AP|_{R_{11}}$ ، به عنوان معیار اصلی ارزیابی تشخیص دهنده $\mathcal{F}(\mathcal{X}; \theta)$ استفاده می‌کند. مجموعه داده KITTI، عموماً از دو جدول رده‌بندی برای سنجش مدل‌ها استفاده می‌کند:

۱. تشخیص سه‌بعدی

۲. تشخیص از نمای دید پرند (BEV)

تشخیص سه‌بعدی، مقدار میانگین دقت درون‌یابی شده $AP_{3D}|_{R_{11}}$ را با آستانه‌های ۰.۷، ۰.۵ و ۰.۳ برای IoU_{3D} ، به ترتیب در کلاس‌های خودرو، عابر پیاده و دوچرخه محاسبه می‌کند. بیشتر سیاست‌های استفاده شده در تشخیص سه‌بعدی، در تشخیص از نمای دید پرند نیز استفاده شده است؛ بجز محاسبه IoU_{BEV} ، که با تصویر کردن کادر محصور کننده سه بعدی β_{3D} از فضای سه‌بعدی به صفحه دو بعدی زمین، محاسبه می‌شود. توجه شود که مجموعه داده KITTI، از سال ۲۰۱۹ به بعد سطوح پوشش خود را از ۱۱ سطح $[0, 1/10, 2/10, \dots, 1]$ به 40 سطح $[1/40, 2/40, 3/40, \dots, 1]$ تغییر داده است. همانطور که دیده می‌شود، سطح پوشش در نسخه جدید، از صفر شروع نمی‌شود [۵].

8-۵-۱-۲ معیار ارزیابی nuScenes

مجموعه داده nuScenes از امتیاز NDS^۱، به عنوان معیار اصلی خود برای ارزیابی تشخیص دهنده $\mathcal{F}(\mathcal{X}; \theta)$ استفاده می‌کند. فرض کنیم زیرمجموعه‌ای از میانگین خطاهای به نام ϵ داریم، که متشکل از خطای تبدیل^۲، ابعاد^۳، جهت‌گیری^۴، ویژگی^۵ و سرعت^۶ است. توجه شود که این خطاهای مربوط به

¹NuScenes Detection Score

²Translation

³Size

⁴Orientation

⁵Attribute

⁶Velocity

کادر محصور کننده تخمین زده شده توسط مدل است، که با قادر واقعی تفاوت هایی دارد. با نگارش زیرمجموعه فوق به صورت $\{\text{mATE}, \text{mASE}, \text{mAOE}, \text{mAAE}, \text{mAVE}\} = \varepsilon$ داریم:

$$NDS = \frac{1}{10} \left[5mAP + \sum_{err \in \varepsilon} (1 - \min(1, err)) \right] \quad (10-2)$$

معیار NDS، اشتراکی از میانگین وزن دار mAP و میانگین خطاهای مجموعه ε در ۱۰ کلاس است. لازم به ذکر است که معیار mAP، با استفاده از فاصله مرکز کادر تخمین زده شده و قادر واقعی، در نمای دید پرندۀ است و از حد آستانه های $\{0.5m, 1m, 2m, 4m\}$ استفاده می کند. این برخلاف حالت استاندارد محاسبه نسبت اشتراک بر روی اجتماع، که در معیارهای دیگر استفاده می شود، است [۵].

۹-۵-۱-۲ معیار ارزیابی Waymo

مجموعه داده Waymo Open از معیار ارزیابی میانگین دقت درون یابی شده $|AP|_{R_{21}}$ و میانگین دقیقی که وزن آن، جهت قرار گیری کادر تخمین زده است (APH^1)، برای ارزیابی مدل تشخیص دهنده ($\mathcal{F}(\mathcal{X}; \theta)$) استفاده می کند. برای محاسبه میانگین دقت، Waymo از ۲۰ سطح بوشش $[1, 0, 1/20, 2/20, \dots]$ و حد آستانه های 0.5° و 0.7° برای کلاس های خودرو و عابر پیاده، استفاده می کند. برای محاسبه APH دقت های جهت گیری کادر تخمین زده شده، مثبت واقعی محسوب می شوند و توسط فرمول زیر وزن می گیرند:

$$\min(|\theta - \theta^*|, 2\pi - |\theta - \theta^*|)/\pi \quad (11-2)$$

که θ و θ^* زوایای آزمیوت^۲ تخمین زده شده و واقعی متناظر آن هستند، که در بازه $[-\pi, \pi]$ قرار می گیرند. مجموعه داده Waymo، از دو سطح سختی متشكل شده اند:

- مرحله اول^۳: که در آن همه کادرهای محصور کننده از نقاط زیادی تشکیل شده اند یا به قول خودشان، حاوی حداقل پنج سیگنال لایدار تشکیل هستند.
- مرحله دوم^۴: که در آن کادر محصور کننده هایی داریم که از نقاط کمی تشکیل شده اند و تشخیص شکلشان دشوار است [۵].

¹Average Precision weighted by Heading

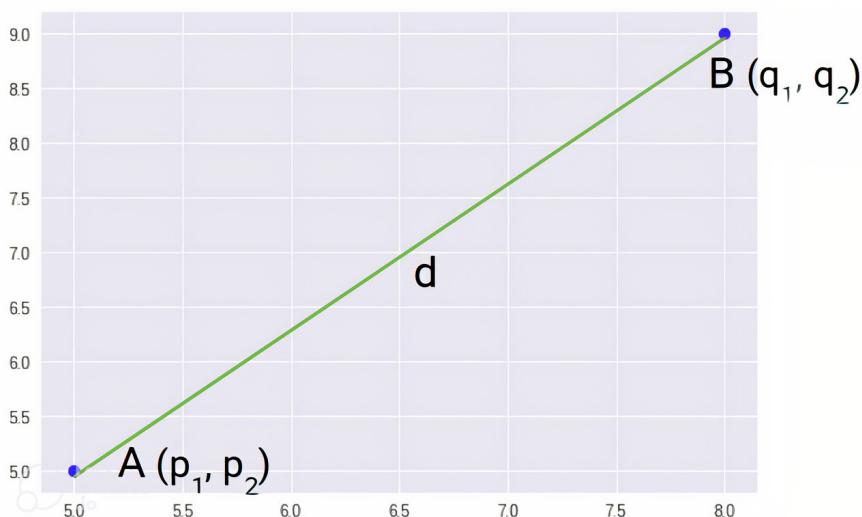
²Azimuth

³LEVEL_1

⁴LEVEL_2

۱۰-۵-۱-۲ معیار فاصله اقلیدسی

فاصله‌ی اقلیدسی^۲، معمول‌ترین روش برای محاسبه‌ی فاصله‌ی بین دو نقطه (داده) است. بهترین تعریف برای این معیار، طول قطعه‌ای است که دو نقطه را به هم متصل می‌کند. اگرچه این یک اندازه‌گیری فاصله رایج است، فاصله‌ی اقلیدسی در مقیاس متغیر نیست، به این معنی که فاصله‌های محاسبه شده، به واحدهای ویژگی‌ها حساس هستند. به طور معمول، قبل از استفاده از این معیار، داده‌ها باید نرمال شوند. علاوه بر این، با افزایش ابعاد داده‌ها، فاصله‌ی اقلیدسی، مناسب‌ترین گزینه نیست و دلیل آن به مشکل ابعاد مربوط می‌شود. با افزایش ابعاد داده‌ها، فاصله بین دو نقطه، دیگر مانند دو بعد و سه بعد به طور شهودی قابل پیش‌بینی نخواهد بود؛ بنابراین، این معیار مناسب مجموعه داده‌های نرمال با ابعاد کم است. در شکل ۱۹-۲ فاصله‌ی بین دو نقطه بر اساس این معیار^۳ و نحوه‌ی محاسبه‌ی فاصله‌ی اقلیدسی در رابطه‌ی ۱۲-۲ آمده است.



شکل ۱۹-۲ فاصله‌ی دو نقطه بر اساس فاصله‌ی اقلیدسی در دو بعد

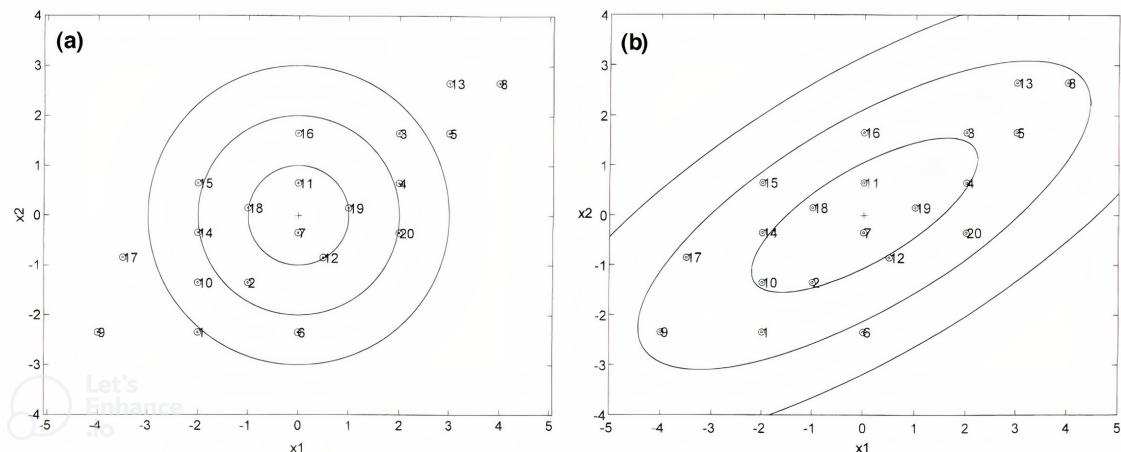
$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (12-2)$$

²Euclidean Distance

³<https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning>

۶-۱-۲ معیار فاصله‌ی ماهالانوبیس

فاصله‌ی ماهالانوبیس^۱، فاصله‌ی بین دو نقطه در فضای چندمتغیره^۲ است. در یک فضای اقلیدسی منظم، متغیرها با محورهایی که در زوایای قائم با یکدیگر ترسیم شده‌اند، نشان داده می‌شوند. فاصله‌ی بین هر دو نقطه را می‌توان با خطکش اندازه‌گیری کرد. برای متغیرهای غیر همبسته، فاصله‌ی اقلیدسی برابر با فاصله‌ی ماهالانوبیس است. با این حال، اگر دو یا چند متغیر، همبسته باشند، محورها دیگر در زوایای قائم نیستند و اندازه‌گیری با یک خطکش غیرممکن می‌شود. علاوه بر این، اگر تعداد ابعاد بیشتر از سه باشد، دیگر در فضای سه بعدی معمولی قابل ترسیم نخواهند بود. فاصله‌ی ماهالانوبیس، این مشکل اندازه‌گیری را حل می‌کند، زیرا فواصل بین نقاط را نسبت به یک مرکز اندازه‌گیری می‌کند. یک نقطه پایه یا مرکزی که می‌تواند به عنوان یک میانگین کلی برای داده‌های چند متغیره در نظر گرفته شود. هر چه فاصله‌ی ماهالانوبی بزرگ‌تر باشد، داده از مرکز دورتر است [۸]. نحوه‌ی محاسبه‌ی فاصله‌ی ماهالانوبیس در رابطه‌ی ۱۳-۲ آمده است. در شکل ۲۰-۲، دو نمونه



شکل ۲۰-۲ فاصله‌ی ماهالانوبیس حول یک نقطه مرکزی [۸]

از محاسبه‌ی فاصله‌ی ماهالانوبیس برای تعدادی داده با دو متغیر x_1 و x_2 را مشاهده می‌کنید. همان‌طور که مشخص است فواصل تا یک نقطه‌ی مرکزی محاسبه شده است.

$$\Delta^2 = (x - m)^\top \Sigma^{-1} (x - m) \quad (13-2)$$

¹Mahalanobis Distance

²Multivariate Space

۷-۱-۲ خوشبندی اقلیدسی

خوشبندی اقلیدسی^۱، یک روش محبوب برای افزایش یک ابر نقاط نامنظم به خوشها^۲ یکی کوچکتر است، که زمان لازم برای پردازش را به مراتب کاهش می‌دهد. این بازنمایی‌های خوشها، به فضای ابر نقاط نظم می‌دهند و برای زمانی که یک بازنمایی حجمی از فضای مسدود شده نیاز است، استفاده می‌شود. این روش برای زمانی که می‌خواهیم شکل اجسام تشخیص داده شده در کادر محصور کننده را داشته باشیم نیز بسیار ارزشمند است.

در عمل، برای پیدا کردن و بخش‌بندی^۳ خوش اجسام متمایز، معمولاً از یک ساختار درخت کی دی^۴ استفاده می‌شود تا نزدیک‌ترین همسایه‌ها^۵ تشخیص داده شوند. الگوریتم این خوشبندی به شکل زیر است:

۱. یک بازنمایی درخت کی دی برای ابر نقاط ورودی ساخته می‌شود؛ P

۲. یک فهرست^۶ از خوش‌های خالی C و یک صفت^۷ از نقاطی که بایستی بررسی شوند Q ، ایجاد می‌شود؛

۳. به ازای هر نقطه $p_i \in P$ ، گام‌های زیر اجرا می‌شود:

(آ) نقطه (p_i) را به صفت اضافه کن؛

(ب) به ازای هر نقطه $q_i \in Q$ ، گام‌های زیر را انجام بده:

- در مجموعه نقاط همسایگی P_i^k ، به دنبال نقاطی که در فاصله $rd_t h$ هستند، جستجو کن؛

- به ازای هر همسایگی $p_i^k \in P_i^k$ ، بررسی کن که آیا نقطه قبل از پردازش شده است یا نه؛ اگر نه، آن را به صفت Q اضافه کن؛

(ج) زمانی که تمام نقاط داخل صفت Q پردازش شدند، Q را به فهرست خوش‌ها C اضافه کن و Q را خالی کن.

۴. الگوریتم زمانی خاتمه می‌یابد که تمام نقاط $p_i \in P$ ، پردازش شده باشند و عضو فهرست خوش‌های شده باشند.^۸

¹Euclidean Clustering

²Segmentation

³Kd-Tree

⁴Nearest Neighbours

⁵List

⁶Queue

⁷https://pcl.readthedocs.io/projects/tutorials/en/master/cluster_extraction.html

۸-۲ مدل سازی سه بعدی

مدل سازی سه بعدی هنر و علمی است که به ایجاد نمایش‌های دیجیتالی سه بعدی اجسام، صحنه‌ها یا محیط‌ها می‌پردازد. این تکنیک حیاتی در زمینه‌های مختلفی از جمله گرافیک کامپیوتری، انیمیشن، معماری، طراحی محصول و واقعیت مجازی به کار گرفته می‌شود. در دنیای مدل سازی سه بعدی، هنرمندان و طراحان از ابزارهای نرم‌افزاری ویژه برای ساخت مدل‌های سه بعدی با دقت و واقع‌گرایی بالا استفاده می‌کنند که قابلیت تغییر و مشاهده از زوایای مختلف را دارند. این مدل‌ها می‌توانند از اشکال هندسی ساده تا شخصیت‌های پیچیده، ساختارهای معماری یا جهان‌های مجازی کامل متغیر باشند. مدل سازی سه بعدی به عنوان پایه‌ای برای بسیاری از کاربردها عمل می‌کند و امکان ایجاد شخصیت‌های بازی ویدئویی واقع‌گرایانه، تصویرسازی طراحی‌های معماری، شبیه‌سازی پدیده‌های فیزیکی و بسیاری دیگر امکان‌پذیر می‌کند. با تنوع کاربردهای خود، مدل سازی سه بعدی نیرویی پیشرانه در عصر دیجیتال است که مرزهای خلاقیت و نوآوری را گسترش می‌دهد.

یکی از جنبه‌های کلیدی مدل سازی سه بعدی نمایش اجسام در فضای سه بعدی است، که معمولاً با استفاده از مختصات محورهای کارتزین تعریف می‌شود. این نمایش مکانی امکان ایجاد تجربیات جاذبه‌ای و تعاملی را فراهم می‌کند، زیرا مدل‌های سه بعدی می‌توانند برای شبیه‌سازی وضعیت‌های واقعی چهان، کاوش محیط‌های مجازی و انتقال اطلاعات پیچیده مورد استفاده قرار گیرند. برای مثال، توسعه شخصیت‌های واقع‌گرایانه برای فیلم‌های انیمیشنی، تجسم طراحی‌های معماری یا تجزیه و تحلیل داده‌های علمی، مدل سازی سه بعدی نقش اساسی در احیای ایده‌ها و مفاهیم در دنیای دیجیتال ایفا می‌کند. با پیشرفت فناوری، تکنیک‌های مدل سازی سه بعدی نیز تکامل می‌کنند و فرصت‌های جدیدی برای خلاقیت و حل مسائل در صنایع مختلف ارائه می‌دهند.



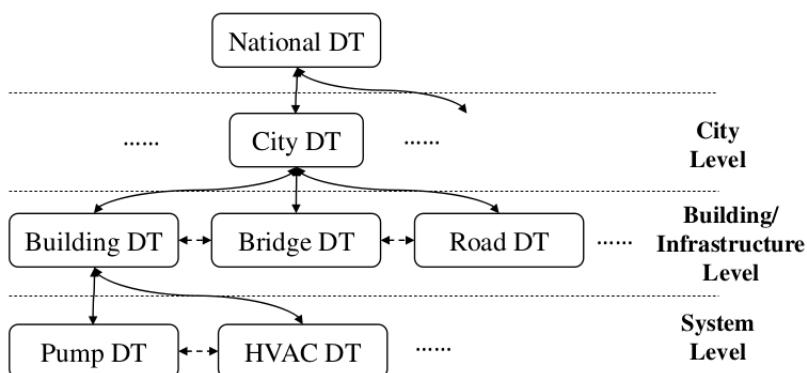
شکل ۲۱-۲ مدل سه بعدی یک ساختمان

۲-۲ نگاهی بر کارهای پیشین

در این بخش نگاهی مختصر به پژوهش‌هایی می‌اندازیم که علم دوکلوهای دیجیتال و نحوه ساختن آن را به چالش می‌کشند.

۱-۲-۲ طراحی سیستماتیک دوکلوبی دیجیتال

تحقیقات گوناگونی در زمینه طراحی سیستماتیک دوکلوبی دیجیتال وجود دارند. در سال ۲۰۱۹، گروهی از محققین دانشگاه کمبریج^۱ اقدام به ساخت دوکلوبی دیجیتالی از پر迪س غربی کمبریج کردند [۹]. در این مقاله، تمرکز اصلی بر روی عقب بودن عمران، معماری و مدیریت سازه^۲ از تکنولوژی و عدم بکارگیری دوکلوهای دیجیتالی برای پیاده‌سازی پروژه‌های ساختمانی است. در این مقاله آمده است که در سال ۲۰۱۷، دولت بریتانیا پیشنهادی مبتنی بر ساختن دوکلوبی دیجیتالی از کشور مطرح کرد و طبق گزارش‌های منتشر شده، این اقدام نیازمند آن است که شرکت‌های عمرانی و معماری، دوکلوبی دیجیتال سازه‌های خود را تهیه نمایند. برای پیاده‌سازی دوکلوبی دیجیتال از یک کشور، بایستی از هزاران دوکلوبی دیجیتالی کوچک‌تر مانند دوکلوبی دیجیتالی شهری و ساختمانی و جاده‌ای بهره برد. در ادامه شکل ۲-۱ از نحوه شکل‌گیری دوکلوبی دیجیتالی برای یک کشور آورده شده است:



شکل ۲۲-۲ معماری دوکلوهای دیجیتالی و سلسله مراتب آنها [۹]

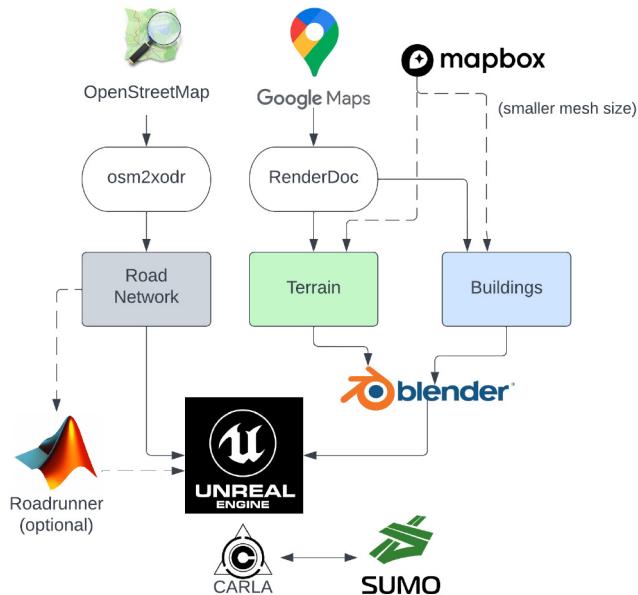
با توجه به شکل ۲۲-۲، می‌توان دریافت کرد که دوکلوبی دیجیتال یک مجموعه بزرگ مانند کشور یا شهر، از دوکلوهای دیجیتال کوچک‌تری شکل گرفته است. این مقاله به طراحی دوکلوبی دیجیتال در ابعاد کوچک می‌پردازد که راه و روش‌های ایجاد دوکلوبی دیجیتال پر迪س دانشگاه را بررسی می‌کند. این مقاله، اولین مقاله‌ای است که به طور سیستماتیک ساختن یک مدل از پر迪س دانشگاه را انجام داده است و مراحل طراحی و معماری دوکلوبی دیجیتال خود را به ساختن یک شهر یا کشور دیجیتالی براساس مقاله‌های گوناگون تعمیم داده است و تمامی محققان را در بکارگیری از این معماری تشویق کرده است.

¹University of Cambridge

²AEC/FM

۲-۲-۲ نحوه‌ی بهینه مدل‌سازی سه‌بعدی دوکلوهای دیجیتال

در مقاله‌ای دیگر که در سال ۲۰۲۲ توسط محققین دانشگاه تگزاس ال پاسو^۱ منتشر شده است، یک روش عملی و کارآمد برای ایجاد دوکلوبی دیجیتال هر مکان جغرافیایی، بخصوص دوکلوبی دیجیتال دانشگاه‌ها و شبکه‌ی جاده‌ها معرفی شده است [۱۰]. نویسنندگان این مقاله، به چالش‌هایی که محققین گوناگون در ایجاد مدل‌های سه‌بعدی دقیق روبرو می‌شوند را مورد تاکید کرده است و اهمیت تصویرسازی و شبیه‌سازی در تحقیقات دوکلوهای دیجیتال را بیان می‌کنند. آن‌ها یک جریان کاری ارائه می‌دهند که از داده سرویس OSM^۲ به عنوان پایه برای مدل‌سازی شبکه‌های جاده استفاده می‌کند و روشی برای ادغام اطلاعات زمینه‌ای مانند ساختمان‌ها و پستی بلندی^۳ ارائه می‌دهد. این مقاله در مورد کاربردهای این روش در حوزه‌های مختلف همانند طراحی شبکه‌های جاده، شبیه‌سازی خودروهای خودران با ابزارهایی مانند سومو^۴ و کارلا^۵ و بخصوص در مورد تحقیقات بینایی کامپیوترا^۶ در محیط‌های دیجیتال و مصنوعی صبحت می‌کند. سپس این مقاله با برگسته کردن قابلیت چندمنظوره بودن، مقیاس پذیری و انعطاف پذیری روش پیشنهادی برای تحقیقات مختلف به پایان می‌رسد. این مقاله تاکید می‌کند که این رویکرد برای مدل‌سازی، شبیه‌سازی و تصویرسازی سه‌بعدی، بهره‌وری و کارآمدی دارد در حیطه‌های مختلف مانند حمل و نقل، زیرساخت و بینایی کامپیوترا مفید و مقرر به صرفه است.



شکل ۲۳-۲ خلاصه‌ای از روش پیشنهاد شده برای ایجاد دوکلوبی دیجیتال یک منطقه (در فصل چهارم به تحلیل دقیق این شکل می‌پردازیم) [۱۰]

^۱University of Texas El Paso

^۲OpenStreetMap

^۳Terrain

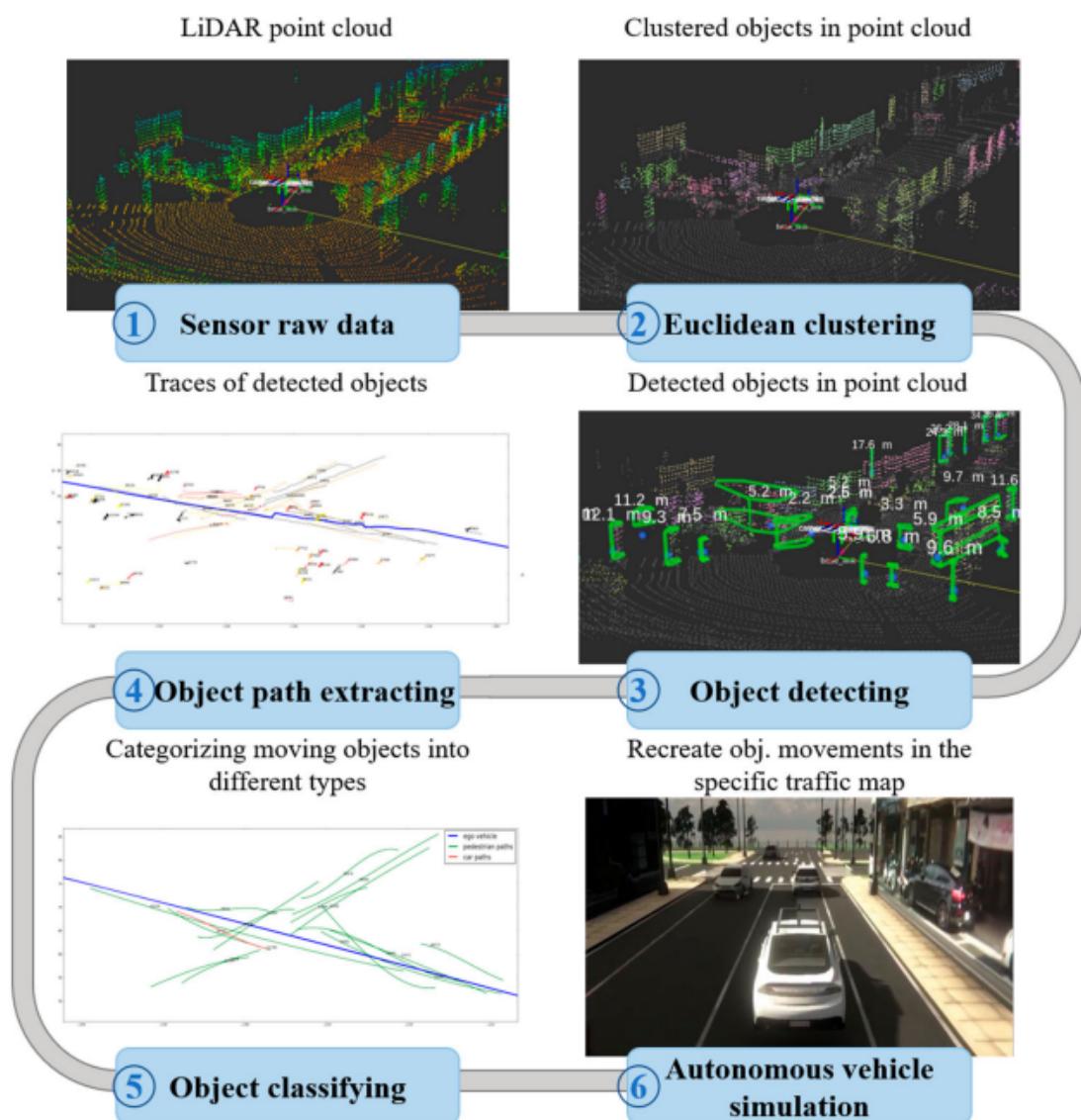
^۴SUMO

^۵CARLA

^۶Computer Vision

۳-۲-۲ چرخهٔ شبیه‌سازی خودکار اجسام پویا در دوکلوهای دیجیتال

مقاله‌های دیگری نیز در زمینه تولید خودکار دوکلوب دیجیتال از یک جاده وجود دارند. مقاله‌ای در سال ۲۰۲۲ منتشر شده است که در زمینه ساخته شدن خودکار مدل‌های دوکلوب دیجیتالی صحنه‌های ترافیکی تحقیق می‌کند [۱۱]. در این مقاله با استفاده از حسگرهای لایدار و ابر نقاط، محیط اطراف با استفاده از خوش بندی با مقیاس فاصله اقلیدسی^۱ دسته‌بندی می‌شود و براساس داده‌های پردازش شده از خوش‌ها، دوکلوب دیجیتال ساخته می‌شود. شکل ۲۴-۲ ۶ مرحله برای شکل‌گیری خودکار دوکلوب دیجیتال را نشان می‌دهد.



شکل ۲۴-۲ چرخهٔ پردازش داده و تشخیص پارامترهای پویا و ساختن دوکلوب دیجیتال به صورت خودکار [۱۱]

¹Euclidean Clustering

حال به شرح هر کدام از بخش‌های شکل ۲۴-۲ می‌پردازیم:

۱. برداشت اطلاعات سه بعدی از حسگر لایدار و ارسال اطلاعات در فرمت ابر نقاط به سرور
 ۲. از الگوریتم خوشبندی با مقیاس فاصله اقلیدسی بر روی ابر نقاط برای تشخیص اجسام استفاده می‌شود. همچنین شناسه‌ای برای هر خوش تشخیص داده شده در نظر گرفته می‌شود تا در مراحل بعدی جهت حرکت اجسام تشخیص داده شود.
 ۳. با استفاده از الگوریتم‌های تشخیص اجسام براساس کادر محصور کننده خوش‌ها^۱، اجسام خوشبندی شده برچسب گذاری می‌شوند و اجمامی که کادر محصور کننده آن‌ها بیشتر از کمینه تعیین شده توسط الگوریتم نباشد فیلتر می‌شوند (به علت‌های گوناگون ممکن است خوش‌هایی وجود داشته باشد که شئ واقعی نباشند). همچنین جهت قرارگیری جسم، ثابت یا متحرک بودن جسم و جهت حرکت جسم تشخیص داده می‌شود.
 ۴. محاسبه مسیر حرکتی اجسام تشخیص داده شده با تعیین موقعیت آن جسم در طول زمان
 ۵. دسته‌بندی اجسام براساس اندازه کادر محصور کننده به دسته‌های خودرو، موتورسیکلت و عابر پیاده
 ۶. پارامترهای گوناگون تشخیص داده شده در مراحل قبل بایستی پردازش شوند و مختصات و مسیرهای حرکت اجسام تشخیص داده شده، به مختصات مدل سه بعدی شبیه‌سازی، تبدیل بشوند. در نهایت نیز مدل‌های سه بعدی اجسام براساس نوع آن جسم در نقشه مدل سه بعدی شبیه‌سازی با توجه به مختصات‌شان قرار داده می‌شوند و براساس مسیر حرکت خود، شروع به حرکت می‌کنند. حال می‌توان سیستم‌های ماشین خودران را در این بستر ایجاد شده آزمود.
- مشاهده کردیم که هر کدام از مقالات، به دنبال مطرح کردن روشی سیستماتیک برای پیاده‌سازی دوکلوهای دیجیتال هستند. همچنین مشاهده کردیم که در سال ۲۰۲۲ تحقیقاتی در زمینه خودکار کردن این پروسه صورت گرفته است. این پایان‌نامه نیز با الهام گیری از مقالات فوق، اقدام به پیاده‌سازی دوکلوبی دیجیتال کرده است.

¹Bounding-Box Based Detection Algorithms

فصل سوم

ابزارها و کتابخانه‌ها

در این فصل، نگاهی به تمام ابزارها و کتابخانه‌های مورد استفاده پژوهش می‌اندازیم. دانستن نحوه معماری و کارکرد این ابزارها، برای درک فصل طراحی و پیاده‌سازی پژوهش، ضروری است.

ROS ۱-۳

ROS^۱، مجموعه‌ای از کتابخانه‌های نرم‌افزاری و ابزارها است که در زمینه پیاده‌سازی ربات‌ها فعالیت می‌کند. ROS یک بسته توسعه نرم‌افزار^۲ ربات متن باز^۳ است. این نرم‌افزار، یک سکوی نرم‌افزاری را در دسترس توسعه‌گران صنعت رباتیک گذاشته است تا به تحقیقات، نمونه‌سازی و گسترش علم رباتیک سرعت بخشد. ROS بیش از ۱۰ سال است که در صنعت رباتیک، توسط میلیون‌ها توسعه‌گر در حال استفاده است. این نرم‌افزار، در اکثر سیستم‌عامل‌ها و سیستم‌های نهفته^۴ قابل توسعه است، و در حوزه‌های پردازش بلادرنگ نیز نقشی اساسی دارد [۱۳].

۱-۳ اکوسیستم

برخلاف اسم آن، ROS یک سیستم عامل نیست و همانطور که ذکر شد، یک بسته توسعه نرم‌افزار ربات است. اکوسیستم این بسته نرم‌افزاری به شکل زیر است:



شکل ۱-۳ تصویری از اجزای ROS [۱۲]

همانطور که مشاهده می‌کنیم، اکوسیستم نرم‌افزار ROS از چهار بخش اساسی تشکیل شده است:

۱. لوله‌کشی^۵: در اصل، ROS یک سیستم انتقال پیام فراهم می‌کند که اغلب به عنوان "میان‌افزار"^۶ یا "لوله‌کشی" شناخته می‌شود. ارتباط اولین نیازی است که هنگام پیاده‌سازی یک برنامه رباتیک جدید یا به طور کلی هر سیستم نرم‌افزاری که با سخت‌افزار تعامل خواهد داشت، به وجود می‌آید. سیستم پیام‌دهی محبوب و تست شده ROS، باعث صرفه جویی در زمانی می‌شود که برای مدیریت جزئیات مربوط به ارتباط بین گره‌های^۷ توزیع شده، صرف شده است (از طریق یک

¹Robot Operating System

²Software Development Kit (SDK)

³Open Source

⁴Embedded Systems

⁵Plumbing

⁶Middleware

⁷Nodes

الگوی انتشار/اشتراک ناشناس^۱). این رویکرد به شما کمک می‌کند تا شیوه‌های خوبی را در توسعه نرم‌افزارهای خود رعایت کنید، که شامل جدا کردن ایرادها، جداسازی نیازمندی‌ها و رابطه‌ای^۲ واضح می‌شود. استفاده از ROS منجر به ایجاد سیستم‌هایی می‌شود که آنها را آسان‌تر برای نگهداری، مشارکت در توسعه، و استفاده مجدد می‌کند.

در طی این مسیر، می‌توان از تجربیات جامعه گسترده‌ای که به ایجاد استانداردهای پیام ROS منجر شده است، بهره برد. این استانداردها برای تعامل با همه چیز از سنجش فاصله لیزری و دوربین‌ها تا الگوریتم‌های موقعیت‌یابی و رابطه‌ای کاربری استفاده می‌شوند [۱۲].

۲. ابزار: پیاده‌سازی برنامه‌های رباتیک چالش‌برانگیز است. شما با تمام مشکلات هر تلاش توسعه نرم‌افزاری مواجه می‌شوید و علاوه بر آن، نیاز به ارتباط ناهمگام^۳ با دنیای فیزیکی از طریق حسگرها و اعمال کننده‌ها هم به لیست مشکلات اضافه می‌شود. برای ساختن برنامه‌ها به صورت کارآمد، به ابزارهای توسعه‌دهنده خوبی نیاز است. ROS این ابزارها را دارد که شامل ابزارهایی مانند راهاندازی^۴، خودکاوی^۵، اشکال‌زدایی^۶، بصری‌سازی^۷، کشیدن نمودار، ثبت اتفاقات^۸ و پخش^۹ می‌شوند. این ابزارها به پیشرفت تیم‌های توسعه، شتاب می‌دهند و می‌توانند همراه محصول عرضه شده قرار گیرند [۱۲].

۳. قابلیت‌ها: جامعه ROS یک مجموعه‌ی نرم‌افزارهای رباتیک است. اگر نیاز به یک درایور برای دستگاه موقعیت‌یاب دارید، یا به یک کنترل کننده‌ی حرکت و تعادل برای ربات چهارپا، یا یک سیستم نقشه‌برداری برای ربات متحرکتان، ROS ابزاری برای شما دارد. از درایورها تا الگوریتم‌ها و رابطه‌ای کاربری، ROS اجزای اصلی را فراهم می‌کند که به شما اجازه می‌دهد تا بر روی برنامه‌ی خود تمرکز کنید. هدف پروژه‌ی ROS، ارتقاء همیشگی استانداردهای رایج و از این راه کاهش مانع در توسعه برنامه‌های رباتیک است. هر کسی که یک ایده خوب برای یک ربات مفید (یا سرگرم‌کننده، یا جالب) دارد، باید بتواند آن ایده را بدون نیاز به درک تمامی اجزای نرم‌افزاری و سخت‌افزاری مرتبط آن، به واقعیت تبدیل کند [۱۲].

۴. جامعه: جامعه ROS وسیع، متنوع و جهانی است. از آدم‌های علاقه‌مند عادی تا دانشجویان، حتی شرکت‌های بین‌المللی و دولت‌ها نیز از ROS برای توسعه تکنولوژی رباتیک استفاده می‌کنند.

¹Anonymous Publish/Subscribe Patterns

²Interface

³Asynchronous Interaction

⁴Launch

⁵Introspection

⁶Debugging

⁷Visualization

⁸Logging

⁹Playback

۲-۱-۳ نسخه‌ها

نرم‌افزار ROS، دو نسخه اصلی دارد که ROS1 و ROS2 نام دارند. امروزه بیشتر شرکت‌ها از نسخه ROS استفاده می‌کنند زیرا از نظر زیرساخت شبکه‌ای، امنیت و قابلیت‌های توسعه با استانداردهای امروزه، پیشرفته‌تر است.

Category	ROS 1	ROS 2
Network Transport	Bespoke protocol built on TCP/UDP	Existing standard (DDS), with abstraction supporting addition of others
Network Architecture	Central name server (<code>roscore</code>)	Peer-to-peer discovery
Platform Support	Linux	Linux, Windows, macOS
Client Libraries	Written independently in each language	Sharing a common underlying C library (<code>rcl</code>)
Node vs. Process	Single node per process	Multiple nodes per process
Threading Model	Callback queues and handlers	Swappable executor
Node State Management	None	Lifecycle nodes
Embedded Systems	Minimal experimental support (<code>rosserial</code>)	Commercially supported implementation (micro-ROS)
Parameter Access	Auxilliary protocol built on XMLRPC	Implemented using service calls
Parameter Types	Type inferred when assigned	Type declared and enforced

[۱۳] شکل ۲-۳ تفاوت قابلیت‌های ROS2 نسبت به ROS1

هر کدام از نسخه‌های ROS، توزیع‌هایی^۱ را در اختیار توسعه‌دهندگان قرار داده‌اند. از آنجایی که در این پژوهش نیز از نسخه جدیدتر ROS2 استفاده شده است. در شکل ۳-۳ می‌توانیم چهار توزیع اصلی مورد استفاده توسعه‌دهندگان رباتیک را مشاهده کنیم. در این پژوهش، از توزیع ROS2 Humble استفاده شده است که نسخه پایدار و بروز سال ۲۰۲۳ به حساب می‌آید.

^۱ROS Distributions

Distro	Release date	Logo	EOL date
Iron Irwini	May 23rd, 2023		November 2024
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		December 9th, 2022
Foxy Fitzroy	June 5th, 2020		June 20th, 2023

شکل ۳-۳ چهار توزیع اصلی ROS2

۳-۱-۳ معماری بسته توسعه نرم افزار ROS2

هر نرم افزار توسعه شده توسط ROS2، از چند اجزای اصلی تشکیل شده است که در ادامه با آنها آشنا خواهیم شد.

۱-۳-۱-۳ گراف ROS2

گراف ROS2^۱، شبکه‌ای از عناصر ROS2 است که در همزمان در کنار یکدیگر داده‌ها را پردازش می‌کنند. این گراف شامل تمامی اتصالات شبکه‌ی داده‌ای و عناصر قابل اجرا است.

۲-۳-۱-۳ گره ROS2

گره ROS2^۲، یک شرکت‌کننده در گراف ROS2 که از یک کتابخانه کلاینت^۳، برای ارتباط با سایر گره‌ها استفاده می‌کند. گره‌ها می‌توانند با سایر گره‌ها در داخل همان فرایند، در یک فرایند متفاوت، یا در ماشین متفاوتی ارتباط برقرار می‌کنند. معمولاً، گره‌ها واحد محاسباتی در یک گراف ROS هستند؛ هر گره باید یک کار منطقی انجام دهد.

گره‌ها می‌توانند داده‌های خود را به مبحثی^۴ انتشار کنند^۵ تا بقیه گره‌ها استفاده کنند، یا در موضوعی

¹ROS2 Graph

²ROS2 Node

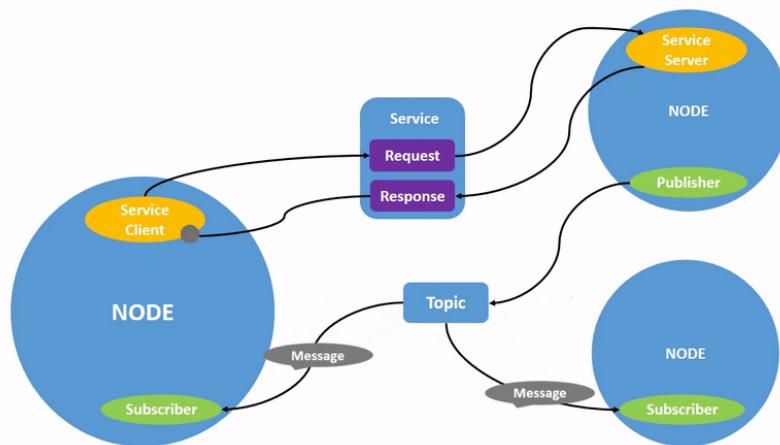
³Client Library

⁴Topic

⁵Publish

اشتراك داشته باشند^۱ و اطلاعات دریافت کنند. آن‌ها می‌توانند به عنوان یک سرویس کلاینت^۲ عمل کنند و محاسبات خود را بر عهده گره دیگری قرار دهند، یا خودشان یک سرویس سرور^۳ باشند و کار محاسبات بقیه گره‌ها را بر عهده بگیرند. برای محاسباتی که طولانی هستند، این گره‌ها می‌توانند یک کلاینت عمل^۴ باشند یا یک سرور عمل^۵ باشند. گره‌ها می‌توانند پارامترهای^۶ قابل تنظیمی داشته باشند که در حین اجرا توسط بقیه گره‌ها تنظیم شوند.

تعريف ۳-۱-۱. گره یک واحد سازمان‌دهنده مهم است که به کاربر امکان تفکر درباره یک سیستم پیچیده را می‌دهد [۱۳].



شکل ۳-۴ یک گراف ROS2 ساده [۱۴]

۳-۳-۱-۳ رابطه‌ها

تمامی برنامه‌ها و گره‌های توسعه شده تحت ROS، از سه نوع الگوی اصلی برای ارتباط استفاده می‌کنند:

- مبحث‌ها: الگوی رایج‌تری که کاربران با آن تعامل می‌کنند، مبحث‌ها هستند که یک چارچوب انتقال پیام ناهمگام هستند. کاربران برای ارسال یا دریافت پیام از مبحث‌ها، بایستی از روش انتشار/اشتراك استفاده کنند. ROS2 مرکز خود را بر روی استفاده از پیام‌رسانی ناهمگام برای سازماندهی یک سیستم با رابطه‌های قوی متناظر گذاشته است و این کار را با سازماندهی نقاط

¹Subscribe

²Service Client

³Service Server

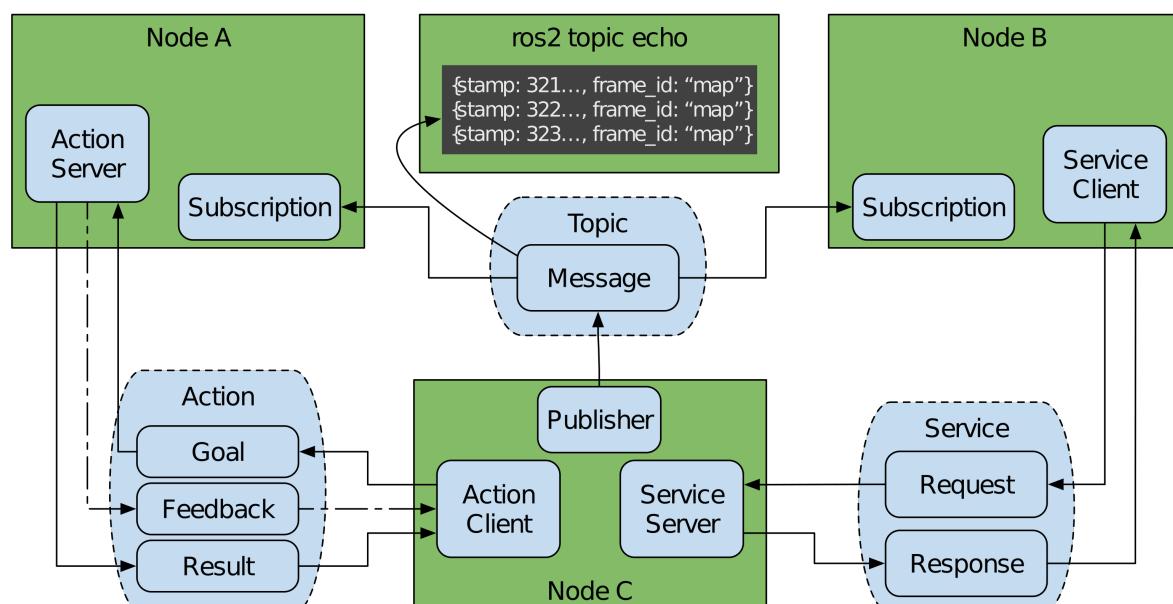
⁴Action Client

⁵Action Server

⁶Parameters

پایانی^۱ در یک گراف محاسباتی تحت مفهوم گره انجام می‌دهد. معماری انتشار-اشتراك ناشناس، ارتباطات چند به چند^۲ را امکان‌پذیر می‌کند. یک توسعه‌دهنده می‌تواند با ایجاد یک اشتراك به یک مبحث، تمام پیام‌های عبوری از این مبحث را بدون اعمال تغییری در عملکرد آن، مشاهده کند [۱۳].

- سرویس‌ها: ارتباط ناهمگان همیشه بهترین ابزار نیست. ROS2 یک الگوی ارتباطی درخواست-پاسخ^۳، به نام سرویس‌ها را نیز فراهم می‌کند. ارتباط درخواست-پاسخ تخصیص داده بین یک جفت درخواست و پاسخ را آسان می‌کند. به طور منحصر به فرد، ROS2 به یک پردازش گر سرویس اجازه می‌دهد که در طول یک فرآخوانی به آن، مسدود^۴ نشود که یعنی درخواست‌ها و پاسخ‌های دیگر هم بتوانند پردازش شوند و منتظر نمانند. سرویس‌ها نیز تحت گره‌ها سازماندهی می‌شوند [۱۳].
- اعمال: اعمال، الگوی ارتباط منحصر به فرد ROS2 است. اعمال هدف محور هستند و ارتباط‌هایی ناهمگام با الگوی درخواست، پاسخ، و بازخورد دوره‌ای با قابلیت لغو هستند. این الگوی ارتباطی، در وظایف طولانی مانند مسیریابی خودکار یا مدیریت خودکار استفاده می‌شود، اگرچه کاربردهای دیگری هم دارد. مشابه سرویس‌ها، عمل‌ها بدون مسدود کردن هستند و تحت گره سازماندهی می‌شوند [۱۳].



شکل ۳-۵ رابطه‌ای یک گره ROS2: مبحث‌ها، سرویس‌ها، اعمال [۱۳]

¹Endpoints

²Many to Many

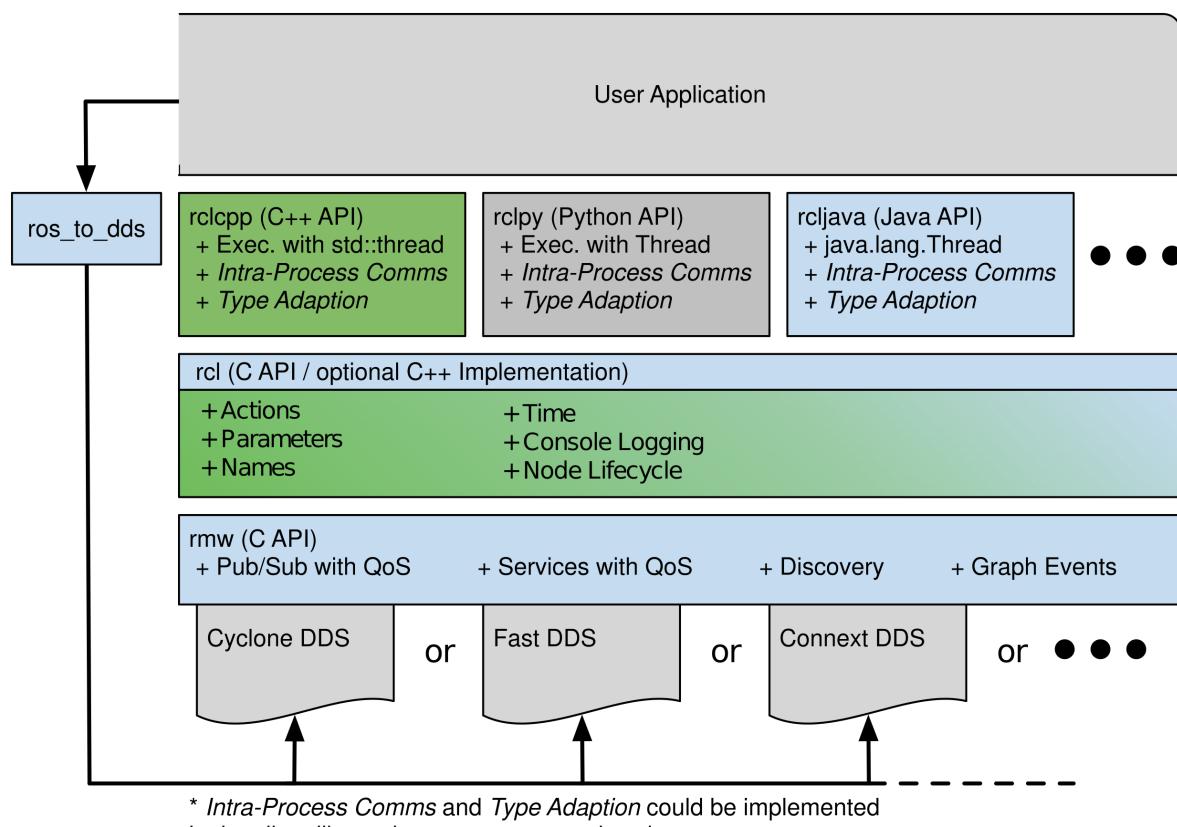
³Request-Response

⁴Blocking

۴-۳-۱-۳ کتابخانه‌های کلاینت

کتابخانه‌های کلاینت، واسطه‌های برنامه‌نویسی هستند که به کاربران امکان پیاده‌سازی کد ROS2 را می‌دهند. با استفاده از کتابخانه‌های کلاینت، کاربران به مفاهیم مختلف مانند گره‌ها، مبحث‌ها، سرویس‌ها و اعمال دسترسی پیدا می‌کنند. کتابخانه‌های کلاینت، به انواع زبان‌های برنامه‌نویسی ارائه می‌شوند تا کاربران بتوانند کد ROS2 خود را به زبانی که برای برنامه‌شان مناسب‌تر است، بنویسند. به عنوان مثال، ممکن است کاربری بخواهد ابزارهای تصویرسازی خود را با زبان پایتون^۱ نویسد چرا که سریع‌ترین تکرار نمونه‌سازی‌ها را داشته باشد؛ در حالی که برای بخش‌هایی از سیستم خود که به راندمان بیشتری نیاز دارد، از زبان برنامه‌نویسی C++ استفاده کند [۱۴].

گره‌های نوشته شده با کتابخانه‌های کلاینت مختلف، قادر به اشتراک‌گذاری پیام‌ها با یکدیگر هستند، زیرا تمام کتابخانه‌های کلاینت ROS2، مولدهای کدنویسی‌ای پیاده‌سازی کرده‌اند که به کاربران امکان تعامل با فایل‌های رابط ROS2، به زبان مربوطه را می‌دهد (در فصل چهارم نیز اشاره‌ای به این موضوع می‌شود زیرا نیاز به استفاده از یک مولد است که بتوان یک سری از رابطه‌های Autoware را برای بسته [۱۴]، ترجمه کرد) ROS2-For-Unity.



[۱۳] کتابخانه‌های کلاینت ROS2

علاوه بر ابزارهای ارتباطی مختص به زبان، کتابخانه‌های کلاینت به کاربران توانایی دسترسی به

^۱Python Programming Language

عملکرد اصلی‌ای که ROS را به "ROS" تبدیل می‌کند را ارائه می‌دهد. به عنوان مثال، در ادامه لیستی از عملکردهایی آمده است که معمولاً از طریق یک کتابخانه کلایینت می‌توان به آنها دسترسی داشت [۱۴]:

- نام‌ها و فضای نام‌ها^۱

- زمان (واقعی یا شبیه‌سازی)

- پارامترها

- ثبت وقایع کنسول

- مدل موازی‌سازی نخ‌ها^۲

- ارتباط داخلی فرایندها

در شکل ۳-۶ نیز می‌توان کلیات نحوه کارکرد کتابخانه‌های کلایینت را مشاهده کرد.

Autoware ۲-۳

نرم‌افزار Autoware از شرکت AWF^۳، یک پشتی نرم‌افزاری متن‌باز برای خودروهای خودران است که بر پایه ROS ساخته شده است. این مجموعه شامل تمام عملکردهای لازم برای رانندگی خودروی خودران اعم از موقعیت‌یابی، تشخیص اجسام، برنامه‌زیری مسیر و کنترل می‌شود. هدف این مجموعه نرم‌افزار، فراهم کردن بستری است که امکان مشارکت افراد و سازمان‌های مختلف در نوآوری‌های متن‌باز مربوط به فناوری‌های رانندگی خودران را می‌دهد.

این نرم‌افزار، هم برای توسعه‌دهندگان فناوری‌های رانندگی خودران و هم برای اپراتورهای سرویس‌های این فناوری، ارزشمند است. توسعه‌دهندگان می‌توانند مولفه‌های^۴ جدیدی را براساس Autoware، برای خودروهای خودران خود پیاده‌سازی کنند. از سوی دیگر، اپراتورهای سرویس‌های خودروی خودران نیز می‌توانند از مولفه‌های مختلف Autoware، برای فناوری خود استفاده کنند. وجود این امکانات، به علت استفاده از معماری خودمختاری در ابعاد میکرو^۵ است، یعنی هر مولفه کوچک از این نرم‌افزار، مستقل از دیگر مولفه‌ها می‌تواند اجرا شود و اعمال تغییر در هر کدام تاثیری بر دیگر مولفه‌ها نمی‌گذارد [۱۵].

¹Namespaces

²Threading Model

³Autoware Foundation

⁴Component

⁵Microautonomy

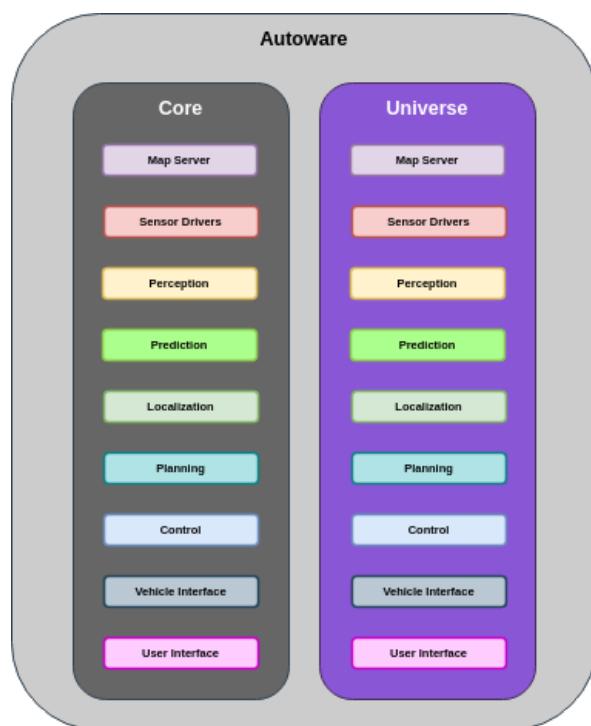
۱-۲-۳ معماری خودمختار در ابعاد میکرو

نرم‌افزار Autoware، از معماری خط لوله^۱، برای توسعه سیستم‌های خودروی خودران استفاده می‌کند. معماری خط لوله مورد استفاده در Autoware، مولفه‌هایی دارد که مشابه معماری سه لایه‌ای^۲ [۳۲] عمل می‌کنند و بصورت موازی اجرا می‌شوند. دو مدول اساسی در معماری Autoware وجود دارند:

۱. مدول Core

۲. مدول Universe

مولفه‌های این مدول‌ها جوهری طراحی شده‌اند که انعطاف پذیر و قابل استفاده مجدد باشند [۱۵].



شکل ۷-۳ دو مدول اصلی [۱۵]

۲-۲-۳ مدول Core

مدول Core یا هسته، شامل برنامه‌های اجرایی اساسی و مولفه‌هایی است که امکان حس کردن، محاسبه و اجرای مورد نیاز برای خودروهای خودران را فراهم می‌سازد. شرکت AWF، مدول Core را از طریق گروههای کاری خود و با معماران و اعضای برجسته‌ی خود توسعه داده و بهروزرسانی می‌کند. هر کسی

¹Pipeline Architecture: <https://www.cs.sjsu.edu/~pearce/modules/patterns/distArch/pipeLine.htm>

²Three-Layer-Architecture

می‌تواند به مدول Core مولفه یا کدی اضافه کند؛ اما معیارهای پذیرش این مشارکت، نسبت به مدول Universe سخت‌گیرانه‌تر و دقیق‌تر است [۱۵].

۳-۲-۳ مدول Universe

مدول‌های Universe یا دنیا، افزونه‌هایی برای مدول Core هستند که توسط توسعه‌دهندگان فناوری ارائه می‌شوند، تا عملکرد و توانایی حس کردن، محاسبه و اجرا را بهبود دهند. شرکت AWF، مدول پایه Universe را برای گسترش ارائه می‌دهد. یک ویژگی کلیدی از معماری خودمختار در ابعاد میکرو، این است که مدول‌های Universe می‌توانند توسط مشارکت هر سازمان و فردی ساخته شوند. به عبارت دیگر، شما می‌توانید حتی مدول‌های خودتان را ایجاد کرده و برای جامعه و اکوسیستم Autoware، در دسترس قرار دهید. شرکت AWF، مسئول کنترل کیفیت مدول‌های Universe در طول فرآیند توسعه آنها هستند. در نتیجه، مدول‌های Universe دارای انواع مختلفی هستند - برخی توسط AWF تأیید و اعتبارسنجی شده‌اند و برخی دیگر نه. انتخاب و ادغام مدول‌های Universe برای ایجاد برنامه‌های نهایی Universe به عهده کاربران Autoware است. کاربران می‌توانند مولفه‌هایی از مدول Core را با مولفه‌های Universe که عموماً بروزتر و نوین‌تر است، تعویض کنند [۱۵].

در ادامه، نگاهی اجمالی به مولفه‌های اصلی تشکیل‌دهنده این مدول‌ها می‌اندازیم.

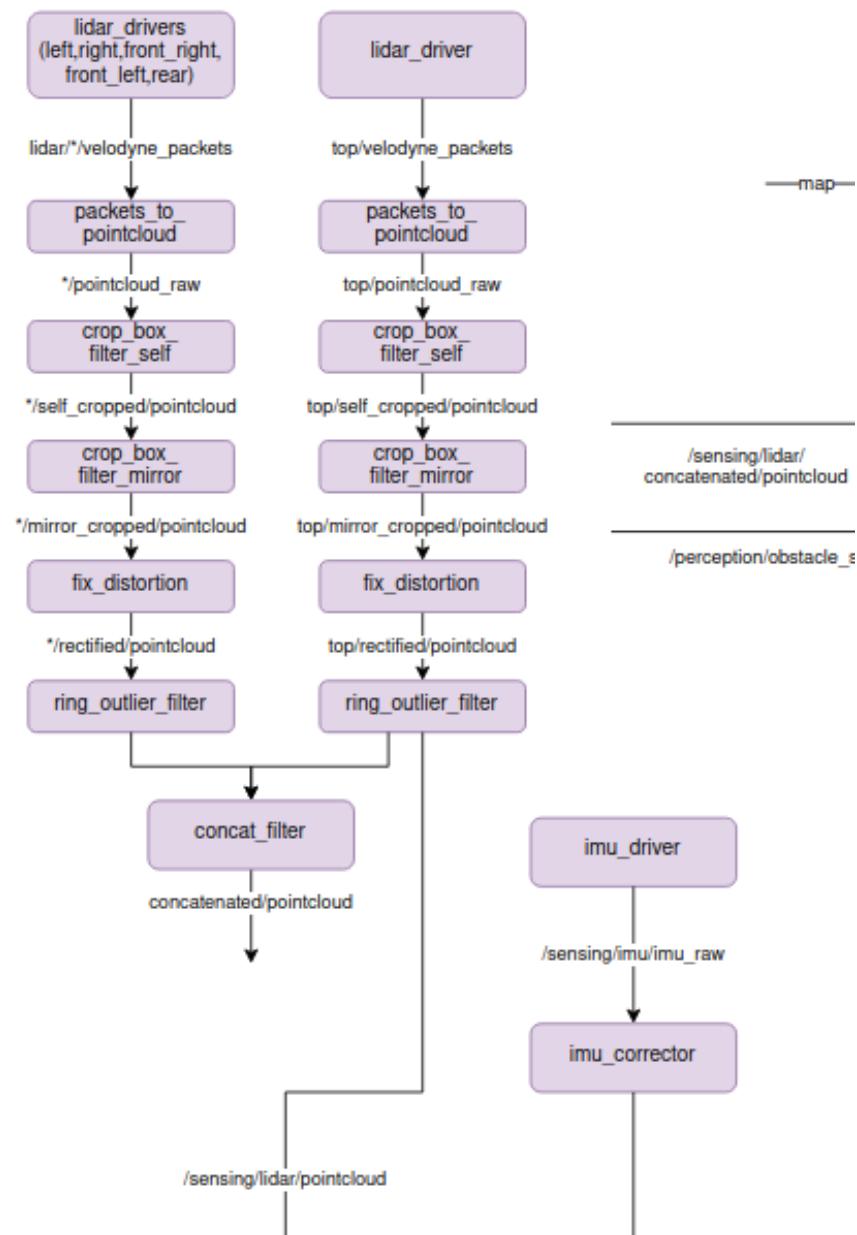
۴-۲-۳ مولفه Sensing

مولفه Sensing یا احساس، مجموعه‌ای از مدول‌ها است که پیش پردازش‌هایی اولیه را بر روی داده‌ی ورودی از سمت حسگرهای اعمال می‌کنند. ورودی‌های این مولفه در شکل ۴-۲-۳ آمده است [۱۵].

Input types

Sensor Data	Message Type
Point cloud (Lidars, depth cameras, etc.)	sensor_msgs/msg/PointCloud2.msg
Image (RGB, monochrome, depth, etc. cameras)	sensor_msgs/msg/Image.msg
Radar scan	radar_msgs/msg/RadarScan.msg
Radar tracks	radar_msgs/msg/RadarTracks.msg
GNSS-INS position	sensor_msgs/msg/NavSatFix.msg
GNSS-INS orientation	autoware_sensing_msgs/GnssInsOrientationStamped.msg
GNSS-INS velocity	geometry_msgs/msg/TwistWithCovarianceStamped.msg
GNSS-INS acceleration	geometry_msgs/msg/AccelWithCovarianceStamped.msg
Ultrasonics	sensor_msgs/msg/Range.msg

شکل ۴-۲-۳ انواع ورودی‌های قابل پذیرش توسط مولفه Sensing [۱۵]



[۱۵] شکل ۹-۳ گراف گرهی مولفه Sensing

همانطور که در شکل ۹-۳ مشاهده می‌شود، مباحثه‌ای ROS که هر کدام از فیلترهای اعمال شده بر ابر نقاط ورودی، به آن داده منتشر می‌کنند، نوشته شده است.^۱

¹ <https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-re-architecture/sensing/>

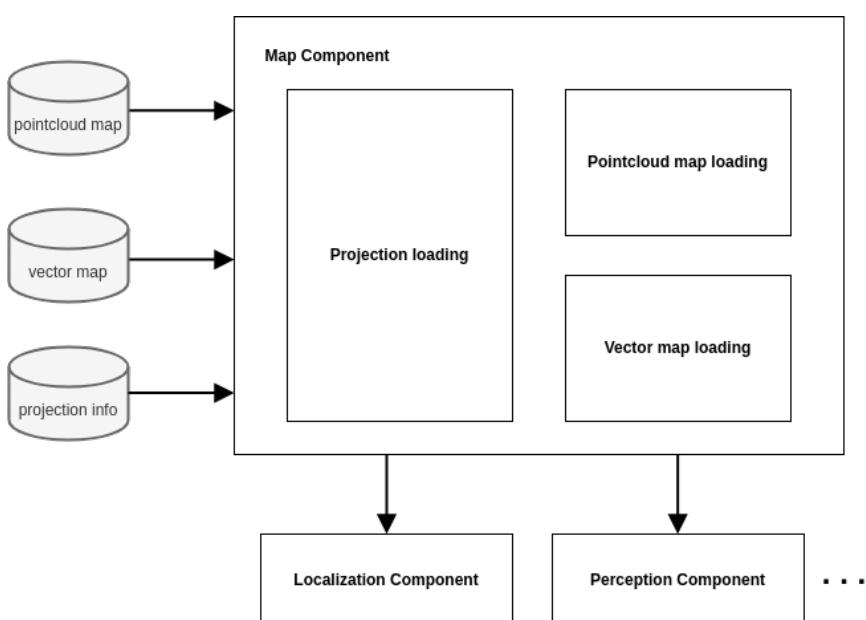
۵-۲-۳ مولفه Map

نرم‌افزار Autoware، برای انجام وظایف مختلفی نظیر موقعیت‌یابی^۱، برنامه‌ریزی مسیر^۲، تشخیص چراغ‌های راهنمایی و رانندگی، و پیش‌بینی مسیر حرکت عابرپیاده‌ها و دیگر وسایل نقلیه، از نقشه‌های ابر نقاط^۳ با وضوح بالا و نقشه‌های برداری^۴ از محیط رانندگی، استفاده می‌کند.

مولفه Map، بایستی دو مدل اطلاعات را برای دیگر پشت‌های فراهم کند:

۱. اطلاعات معنایی^۵ مربوط به جاده‌ها در نقشه بُرداری

۲. اطلاعات ژئومتریک^۶ مربوط به محیط در نقشه ابر نقاط



شکل ۳-۱۰ [۱۵] معماری مولفه Map

یک نقشه بُرداری، شامل اطلاعات دقیق در مورد شبکه جاده، هندسه مسیرهای عبوری و چراغ‌های راهنمایی و رانندگی می‌شود. این اطلاعات برای برنامه‌ریزی مسیر، تشخیص چراغ‌های راهنمایی و پیش‌بینی مسیر حرکت دیگر وسایل نقلیه و عابرپیاده‌ها ضروری هستند.

یک نقشه ابر نقاط سه بعدی، در درجه اول برای موقعیت‌یابی لایدار-محور و بخشی از مولفه Perception در Autoware استفاده می‌شود. به منظور تعیین مکان و جهت فعلی وسیله نقلیه، اسکن زنده‌ای که از یک یا چند واحد لایدار ثبت شده است، با یک نقشه ابر نقاط سه بعدی پیش‌تولید شده

¹Localization

²Route Planning

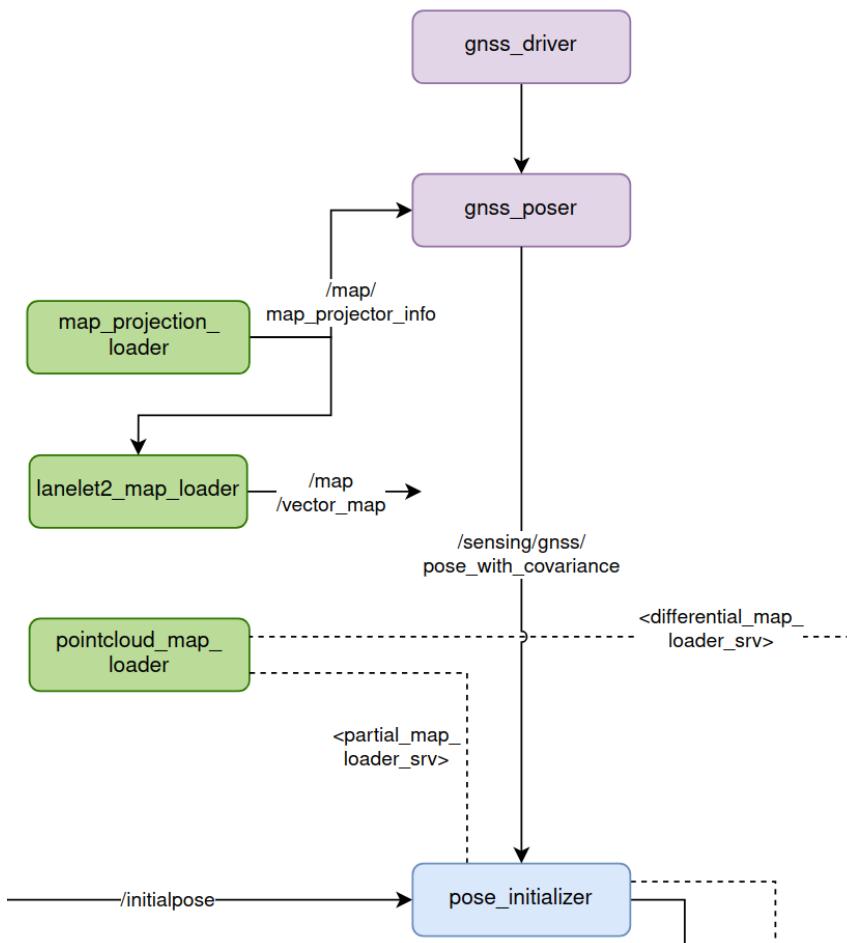
³Pointcloud Map

⁴Vector Map

⁵Semantic Information

⁶Geometric Information

همخوانی داده می‌شود. بنابراین، یک نقشه دقیق ابر نقاط برای نتایج خوب موقعیت‌یابی بسیار حیاتی است. با این حال، اگر وسیله نقلیه دارای یک روش موقعیت‌یابی جایگزین با دقت کافی باشد، به عنوان مثال با استفاده از موقعیت‌یابی بر اساس دوربین، احتمالاً نیازی به استفاده از نقشه ابر نقاط برای وجود نخواهد داشت [۱۵]. در شکل ۳، معماری سطح بالایی از مولفه Map را مشاهده می‌کنیم^۱.



شکل ۳ گراف گرهی مولفه Map (مستطیلهای سبز) [۱۵]

۶-۲-۳ مولفه Localization

مولفه Localization یا موقعیت‌یابی، سعی در تخمین موقعیت، سرعت و شتاب خودروی خودران دارد. همچنین سعی شده است تا سیستمی طراحی شود که عملیات موقعیت‌یابی را با حسگرهای مختلف انجام دهد و در صورتی که تخمین با خطای زیادی مواجه باشد، پیام هشداری را به سیستم نظارتی خودروی خودران ارسال کند. این مولفه، با پیکربندی حسگرهای مختلف زیر کار می‌کند:

^۱ <https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-re-architecture/map/>

۱. لایدار سه‌بعدی + نقشه ابر نقاط

۲. لایدار سه‌بعدی یا دوربین + نقشه بُرداری

۳. دوربین (ادومتری بصری، محلی‌سازی و نقشه‌برداری همزمان^۱ بصری)

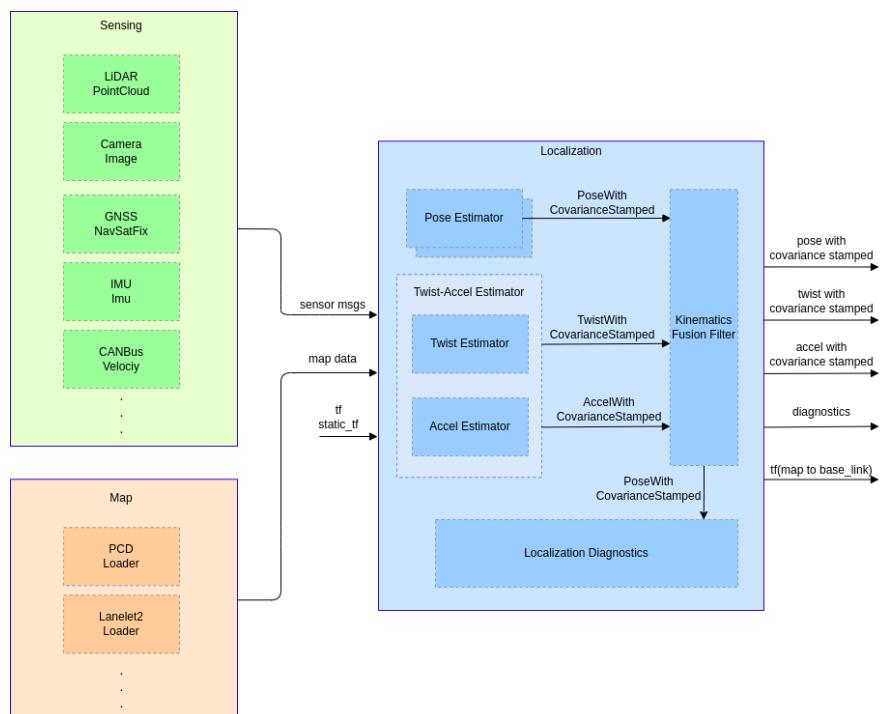
۴. حسگر سرعت چرخ

۵. واحد اندازه‌گیری لختی^۲

۶. حسگر ژئو-مغناطیسی^۳

۷. نشانگرهای مغناطیسی^۴

در شکل ۱۲-۳، معماری پیشنهادی، که توسط توسعه‌دهندگان Autoware برای توسعه‌دهندگان دیگر در اختیار گذاشته‌اند، مشاهده می‌شود. برای دقت بالا در موقعیت یابی، پیشنهاد شده است که واحد اندازه‌گیری لختی و حسگر لایدار حتماً استفاده شوند؛ همچنین نقشه بُرداری و نقشه ابر نقاط محیط نیز به عنوان ورودی به مولفه Localization داده شوند [۱۵].



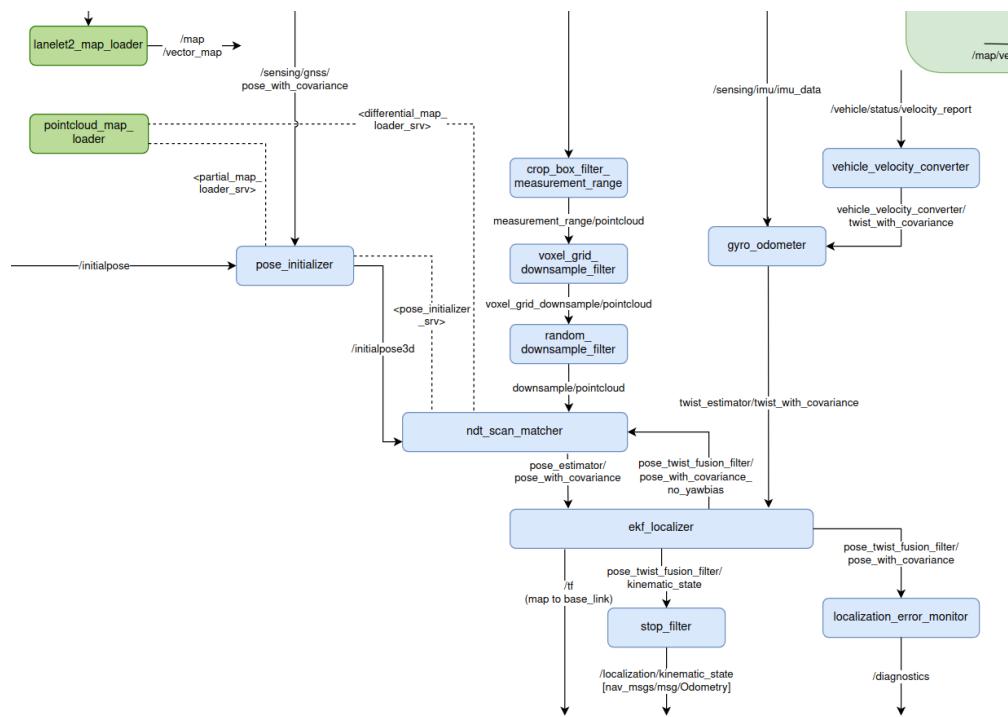
شکل ۱۲-۳ معماری پیشنهادی Autoware برای مولفه Localization [۱۵]

^۱SLAM: Simultaneous Localization And Mapping

^۲IMU: Inertial Measurement Unit

^۳Geomagnetic Sensor

^۴Magnetic Markers



شکل ۳-۱۵ گراف گرهی مولفه Localization

۷-۲-۳ مولفه Perception

مولفه Perception یا ادراک، وظیفه‌ی تشخیص انواع وسایل نقیه مانند خودرو، کامیون، موتور، اتوبوس، و همچنین عابرین پیاده را بر عهده دارد. این مولفه، علاوه بر تشخیص اجسام سه‌بعدی ذکر شده، شناسه‌ی یکتاپی را به آنها نسبت می‌دهد و آنها را ردیابی^۱ می‌کند تا موقعیت، شتاب، سرعت، مسیر حرکت و جهت قرارگیری آنها را محاسبه کند. این مولفه در مدول Universe دارای الگوریتم‌های گوناگونی برای ادراک محیط است و ورودی‌های آن همانند مولفه Sensing می‌باشد. این مولفه از چند گره اصلی تشکیل شده است:

۱. گره پیش‌پردازش

۲. گره تشخیص و ردیابی اجسام سه‌بعدی

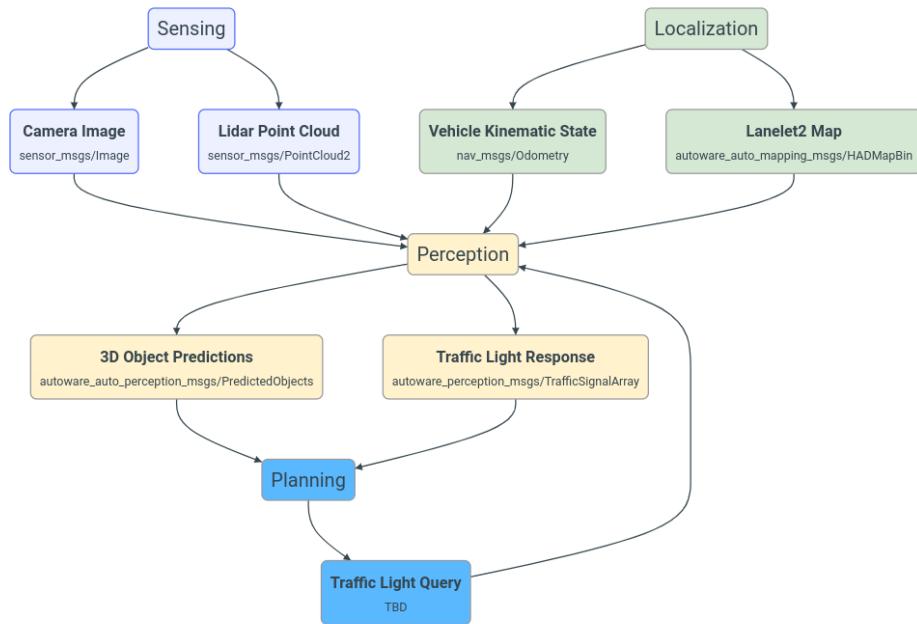
۳. گره تشخیص علایم راهنمایی و رانندگی

۴. گره پیش‌بینی مسیر حرکت اجسام

۵. گره پس‌پردازش

بایستی توجه داشت که ورودی‌های ابر نقاط لایدار، برای این مولفه ضروری است و نداشتن آن، دقت تشخیص را به مراتب کاهش می‌دهد. در معماری رابطه‌های مولفه Perception، مشاهده می‌شود. همانطور که ذکر شد، پیشنهاد می‌شود که حتماً از داده‌های ابر نقاط برای تشخیص اجسام استفاده کرد.

¹Track

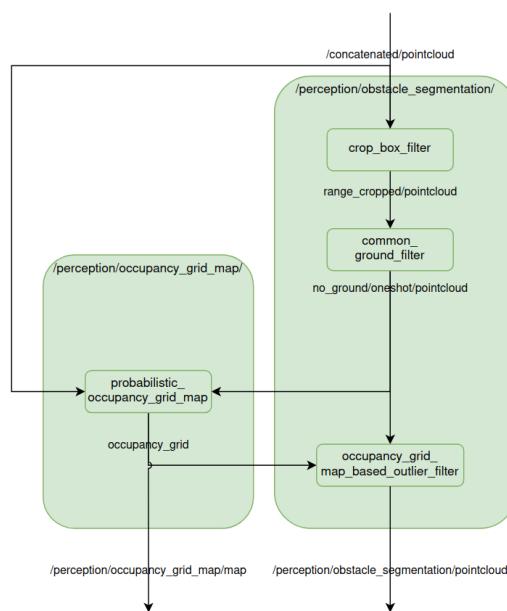


شکل ۳-۱۴ [۱۵] معماری رابطه‌ای مولفه Perception

از آنجایی که در این پژوهش، از این مولفه استفاده شده است، لازم است تا برخی از گره‌های مهم آن توضیح داده شوند.

۱-۷-۲-۳ گره پیش‌پردازش

این گره وظیفه اعمال فیلترهایی اولیه به ابر نقاط ورودی را بر عهده دارد.



شکل ۳-۱۵ [۱۵] گراف گرهی پیش‌پردازش مولفه Perception

طبق شکل ۱۵-۳ مشاهده می‌کنیم که گره پیش‌پردازش، خود از گره‌هایی کوچک‌تر ساخته شده‌اند که عبارتند از:

۱. فیلتر crop_box: این گره با استفاده از کتابخانه PCL^۱ تمام نقاطی که در یک منطقه مکعبی مشخص شده باشد را از ابر نقاط حذف می‌کند. از این گره برای فیلتر کردن نقاطی که مربوط به خودروی خودران است، استفاده می‌شود.
۲. فیلتر common_ground: این گره برای حذف نقاط زمین از ابر نقاط استفاده می‌کند.^۲
۳. نقشه شبکه اشغال احتمالاتی^۳: از این گره برای ساختن نقشه‌ای از احتمال وجود موانع در اطراف ماشین خودران، استفاده می‌شود.^۴

۲-۷-۲-۳ گره تشخیص و ردیابی اجسام سه‌بعدی

در این گره که همانند گره پیش‌پردازش از چندین گره کوچک‌تر ساخته شده است، ابر نقاط فیلتر شده به عنوان ورودی به مدل تشخیص‌دهنده اجسام سه‌بعدی داده می‌شود و سپس بر روی کادر محصور کننده خروجی این مدل، الگوریتم‌های ردیابی زده می‌شود. در شکل ۱۶-۳ معماری گره تشخیص مشاهده می‌شود. همانطور که از شکل معلوم است، گره Perception از گره‌های کوچک‌تری تشکیل شده است که عبارتند از:

۱. گره تشخیص دهنده سه‌بعدی (lidar_centerpoint): این گره، یک مدل هوش مصنوعی تشخیص سه‌بعدی ابر نقاط-محور مبتنی بر واکسل، به نام Centerpoint است [۲۰]. ابر نقاط فیلتر شده به عنوان ورودی به این تشخیص دهنده داده می‌شود و کادرهای محصور کننده اجسام تشخیص داده شده، به عنوان خروجی دریافت می‌شوند. این مدل هوش مصنوعی، از یک شبکه عصبی بر مبنای PointPillars [۳۳] استفاده می‌کند. این شبکه عصبی با بهره‌وری از کتابخانه TensorRT^۵، که برای سرعت‌های بسیار بالا (تشخیص بلادرنگ) و استفاده بهینه از کارت گرافیکی ساخته شده است، پیاده‌سازی شده است.

۲. گره اعتبار سنجی موانع براساس ابر نقاط (obstacle_pointcloud_based_validator): این گره، خروجی مدل هوش مصنوعی را بررسی می‌کند و اجسامی که از مجموعه نقاط کمی تشکیل شده باشند را به عنوان مثبت‌های کاذب تشخیص داده و حذف می‌کند.

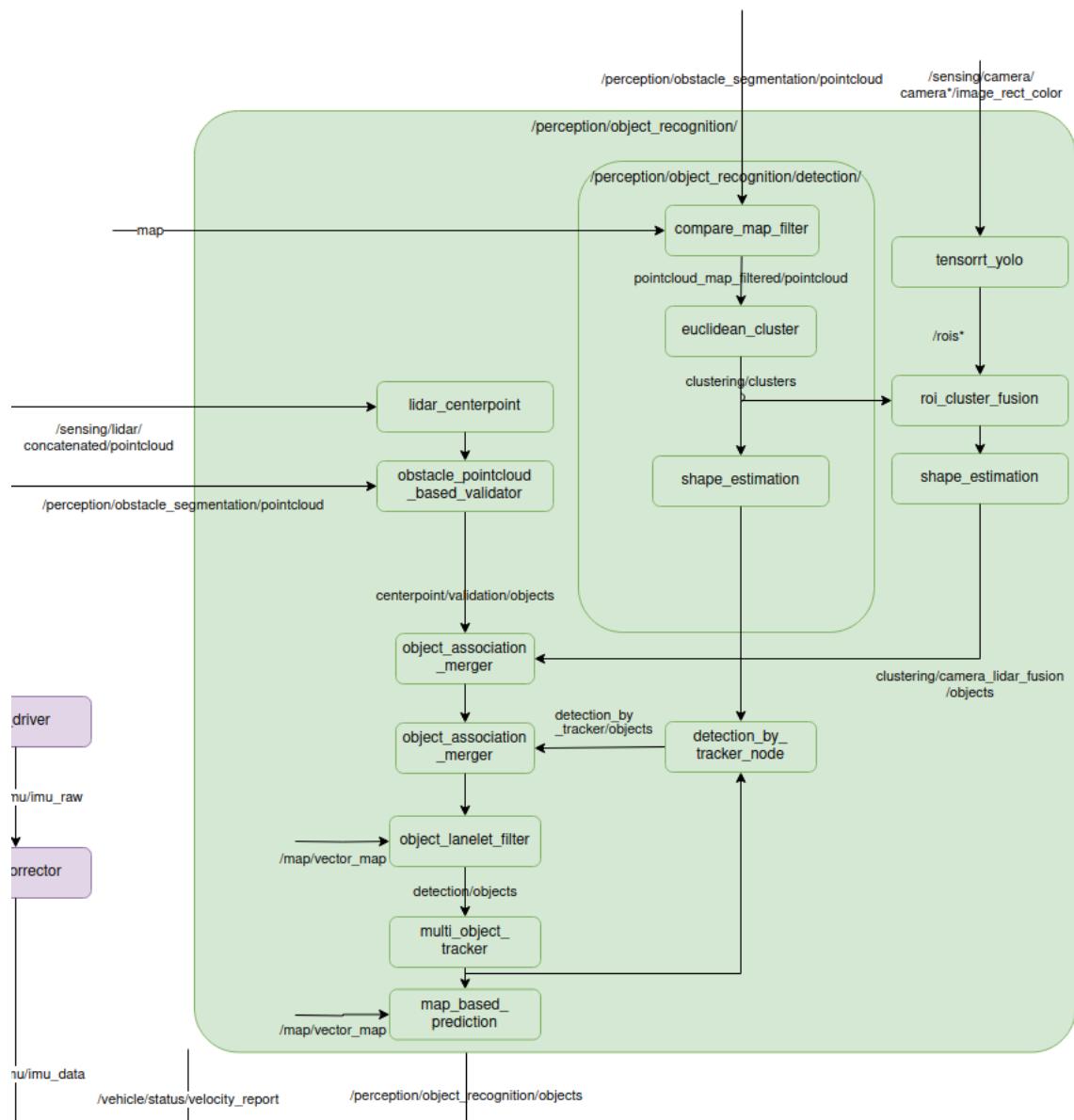
¹ Point Cloud Library : <https://pcl.readthedocs.io/projects/tutorials/en/master/#>

² https://autowarefoundation.github.io/autoware.universe/main/perception/ground_segmentation/

³ Probabilistic Occupancy Grid Map

⁴ https://autowarefoundation.github.io/autoware.universe/main/perception/probabilistic_occupancy_grid_map/

⁵ <https://developer.nvidia.com/tensorrt>



شکل ۳-۱۶ [۱۵] Perception گراف گرهی تشخیص اجسام و ردیابی مولفه

۳. گرۀ فیلتر مقایسه نقشه (compare_map_filter): این گرۀ برای تشخیص نقاط مربوط به زمین و حذف آن‌ها با استفاده از نقشه ابر نقاط است.^۱

۴. گرۀ خوشه‌بندی اقلیدسی (euclidean_cluster): این گرۀ، با استفاده از خوشه‌بندی اقلیدسی کتابخانه PCL، اقدام به خوشه‌بندی ابر نقاط می‌کند که از آن برای تشخیص اجسام و همچنین تشخیص ابعاد اجسام استفاده می‌شود.^۲

¹https://autowarefoundation.github.io/autoware.universe/main/perception/compare_map_segmentation/

²<https://autowarefoundation.github.io/autoware.universe/main/perception/euclidean/>

۵. گره تخمین شکل (shape_estimation): این گره، شکل دقیق‌تری از جسم (کادر محصور‌کننده استوانه، پوشش محدب^۱) را که خوش‌های ابر نقاط در ارتباط با یک برچسب، در آن جا می‌شود را محاسبه می‌کند.
۶. گره ادغام کننده اجسام (object_association_merger): این گره اجسام تشخیص داده شده از دو روش تشخیص اجسام مختلف را ادغام می‌کند.^۲
۷. گره فیلتر لینلت (object_lanelet_filter): این گره، اجسام تشخیص داده شده را براساس نقشه بُرداری فیلتر می‌کند. این گره تنها اجسامی که داخل خطوط جاده‌ها و پیاده‌روها باشند را از فیلتر عبور می‌دهد.^۳
۸. گره ردیاب همزمان چند جسم (multi_object_tracker): نتایج تشخیص اجسام، توسط یک سری زمانی پردازش می‌شوند. هدف اصلی، دادن شناسه یکتا به هر یک از اجسام تشخیص داده شده و تخمین سرعت آن‌ها است.^۴
۹. گره تشخیص براساس ردیاب (detection_by_tracker): این گره، اجسام ردیابی شده را به گره تشخیص باز می‌گرداند تا پایدار بماند و به شناسایی اجسام ادامه دهد. این گره یک جسم ناشناخته حاوی یک خوش‌های نقاط و یک ردیاب را به عنوان ورودی، دریافت می‌کند. جسم ناشناخته تا مرحله متناسب شدن با ابعاد ردیاب، بهینه می‌شود تا بتواند به شناسایی شدن ادامه دهد. گرهی تشخیص با استفاده از ردیاب، یک شیء ناشناخته شامل یک ابر نقطه و یک ردیاب دریافت می‌کند، سپس شکل شیء ناشناخته با استفاده از خوش‌بندی اقلیدسی، متناسب می‌شود. متناسب‌سازی شکل با استفاده از خوش‌بندی اقلیدسی و روش‌های دیگر مشکلی به نام کم‌ تقسیم‌بندی^۵ و بیش‌ تقسیم‌بندی^۶ دارد، که با اعمال سیاست‌هایی بهبود می‌یابد^۷.

an_cluster/

¹Convex Hull

²https://autowarefoundation.github.io/autoware.universe/main/perception/object_merger/

³<https://autowarefoundation.github.io/autoware.universe/main/perception/detecte>

d_object_validation/object-lanelet-filter/

⁴https://autowarefoundation.github.io/autoware.universe/main/perception/multi_o

bject_tracker/

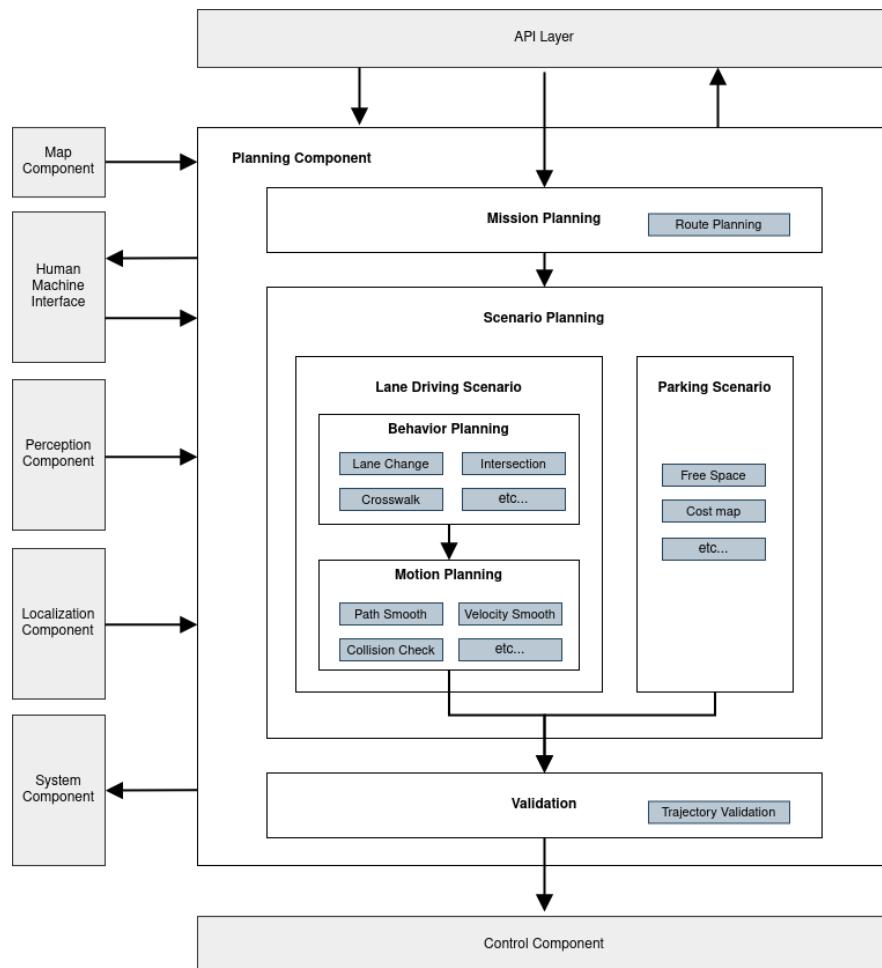
⁵Under-Segmentation

⁶Over-Segmentation

⁷<https://autowarefoundation.github.io/autoware.universe/main/perception/detecti>
on_by_tracker/

Planning مولفه ۸-۲-۳

مولفه Planning یا برنامه‌ریزی مسیر، پیام مسیر پیشنهادی را براساس وضعیت محیط که از مولفه‌های ادراک و موقعیت‌یابی بدست می‌آورد، منتشر می‌کند و مولفه Control از این پیام استفاده می‌کند.



[۱۵] شکل ۱۷-۳ معماری سطح بالای مولفه Planning

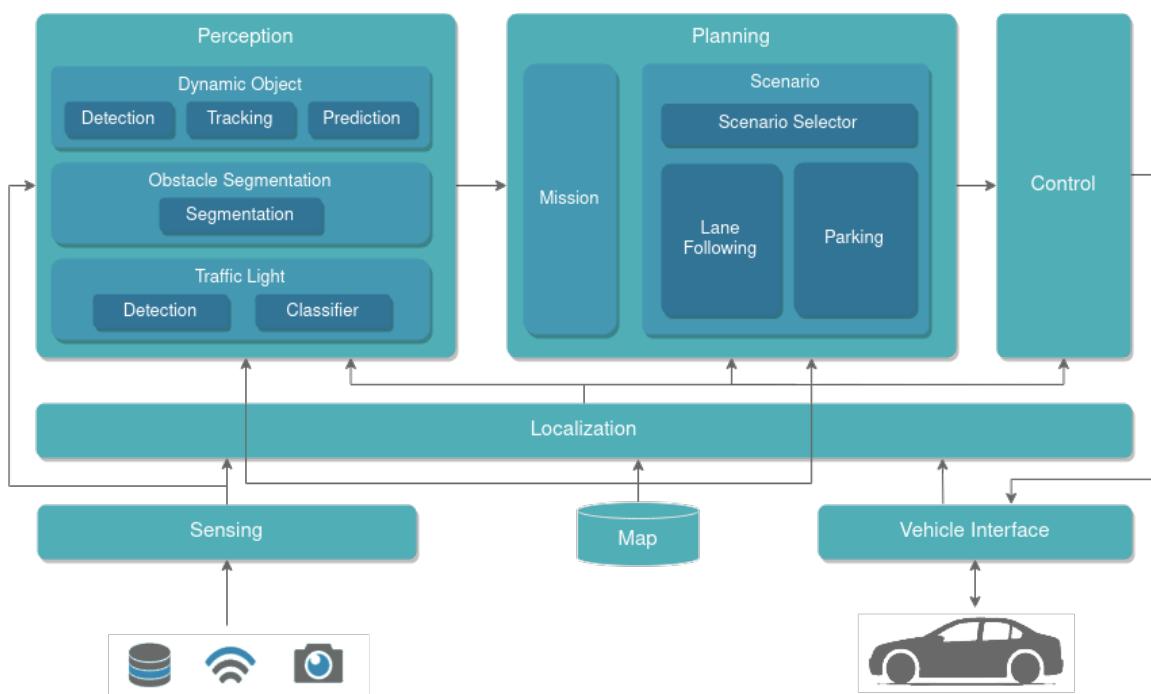
همانطور که طبق شکل ۱۷-۳ مشاهده می‌شود، مولفه Planning از چند مولفه کوچک‌تر ساخته شده است:

۱. مولفه برنامه‌ریزی ماموریت (Mission Planning): مسیر را براساس هدف و اطلاعات نقشه داده شده، محاسبه می‌کند.
۲. مولفه برنامه‌ریزی سناریو (Scenario Planning): مسیر را براساس سناریو حال حاضر تعیین می‌کند (به طور مثال سناریو رانندگی در خطوط جاده یا سناریو پارک کردن).
- مولفه رانندگی در خطوط (Lane Driving): مسیر را برای رانندگی در بین خطوط جاده را محاسبه می‌کند.

- مولفه برنامه‌ریزی رفتار (Behavior Planner): مسیر مطلوب را براساس ملاحظات ایمنی و قوانین راهنمایی رانندگی، محاسبه می‌کند.
- مولفه برنامه‌ریزی حرکت (Motion Planner): مسیر مطلوب خودرو را براساس عامل‌های ایمنی، ملاحظات سرعت خودرو و دستورات مولفه برنامه‌ریزی رفتار، محاسبه می‌کند.
- مولفه پارک کردن (Parking): مسیر را در سناریوهای پارک کردن در جای ناشناخته، محاسبه می‌کند.

۳. مولفه اعتبارسنجی (Validation): ایمن بودن مسیر محاسبه شده را می‌سنجد.

هر کدام از مولفه‌ها می‌توانند براساس وضعیت، به صورت پویا اضافه و حذف شوند. این معماری، برگرفته از معماری خودمختار در ابعاد میکرو نرمافزار Autoware است، که این اجازه را می‌دهد که هر کدام از مولفه‌ها نسبت به شرایط، به خط لوله اجرایی اضافه یا از آن حذف شوند [۱۵]. برای مشاهده گراف گره این مولفه، می‌توان به صفحه گراف گره کلی Autoware مراجعه کرد.^۱. همچنین دو مولفه پایانی Control و Vehicle، به علت مربوط نبودن به پژوهش، توضیح داده نمی‌شوند. شکل ۱۸-۳، معماری کلی نرمافزار Autoware را در یک نگاه نشان می‌دهد.



شکل ۱۸-۳ معماری سطح بالای نرمافزار Autoware [۱۵]

^۱<https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-re-architecture/node-diagram/>

۳-۳ شبیه‌ساز AWSIM



شکل ۱۹-۳ شبیه‌ساز AWSIM [۱۶]

AWSIM یک شبیه‌ساز متن باز ساخته شده با استفاده از موتور بازی‌سازی Unity است که برای تحقیق و توسعه در حوزه رانندگی خودکار است. این شبیه‌ساز برای نرم‌افزارهای خودروی خودران همانند Autoware، ساخته شده است. هدف این شبیه‌ساز، ایجاد ارتباط بین دنیای مجازی و واقعی است، به طوری که به کاربران اجازه می‌دهد تا سیستم‌های خودران خود را قبل از اینکه بر روی خودروهای واقعی مستقر کنند؛ در یک محیط ایمن و کنترل شده، آموزش دهنده و ارزیابی کنند. این شبیه‌ساز محیط مجازی واقع‌گرا، برای آموزش، آزمایش و ارزیابی جوانب مختلفی از سیستم‌های رانندگی خودکار ارائه می‌دهد.

AWSIM سناریوهای واقعی متنوعی را با مدل‌های دقیق فیزیک و حسگر، شبیه‌سازی می‌کند. این شبیه‌ساز دارای مجموعه گسترده‌ای از حسگرهای مانند دوربین‌ها، لایدارها، واحد اندازه‌گیری لختی و سیستم ناوبری ماهواره مبنای^۱ می‌باشد که به توسعه‌دهندگان اجازه می‌دهد تا تعاملات وسیله نقلیه خودران خود را با محیط به صورت دقیق شبیه‌سازی کنند. این شبیه‌ساز همچنین اشیاء پویا مانند عابرین پیاده، دیگر وسایل نقلیه و چراغ‌های راهنمایی رانندگی را نیز مدل‌سازی می‌کند، که امکان مطالعه تعاملات و تصمیم‌گیری در سناریوهای پیچیده ترافیکی را فراهم می‌کند. این موضوع، امکان آزمون و ارزیابی الگوریتم‌های ادراک، برنامه‌ریزی و کنترل در پیکربندی‌های مختلف حسگرها و سناریوها را می‌دهد.

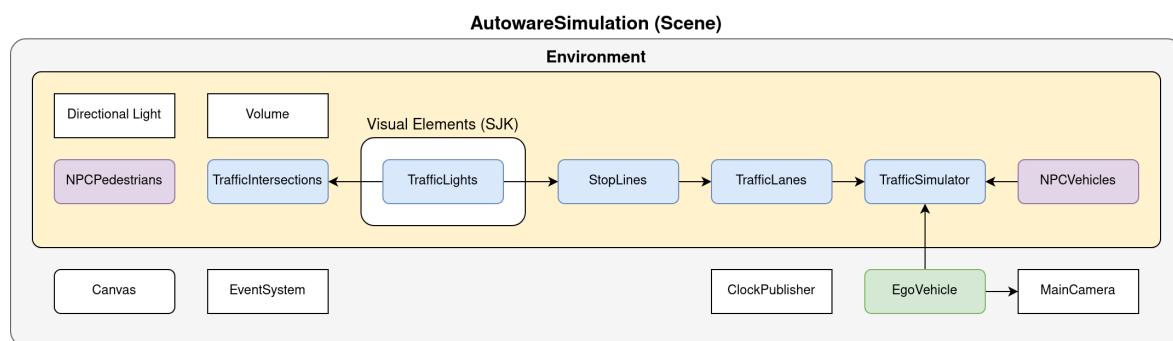
AWSIM، از یک معماری انعطاف پذیر و مدولار^۲ پشتیبانی می‌کند، که امکان تغییر و گسترش

¹Global Navigation Satellite System (GNSS)

²Modular

قابلیت‌های آن را آسان می‌کند. کاربران می‌توانند محیط^۱ فعلی شبیه‌ساز را تغییر دهند یا محیط جدیدی با عناصر^۲ و قوانین ترافیک خود اضافه کنند، تا سناریوهای خاص خود را برای تحقیقاتشان ایجاد کنند [۱۶].

۱-۳-۳ معماری



شکل ۲۰-۳ معماری سطح بالای شبیه‌ساز AWSIM [۱۶]

برای توصیف معماری AWSIM، ابتدا باید به صحنه^۳ بازی اشاره کرد. این صحنه، تمام اشیاء موجود در شبیه‌سازی یک سناریو خاص و تنظیمات آن‌ها را شامل می‌شود. صحنه پیش‌فرض AWSIM که برای کار با AWSIM توسعه یافته است، به نام AutowareSimulation شناخته می‌شود. در این صحنه می‌توان اجزاء اساسی مانند دوربین اصلی (MainCamera)، انتشار کننده زمان (ClockPublisher)، سیستم رویداد (EventSystem) و کانواس (Canvas) را مشاهده کرد. توضیحات دقیق در مورد صحنه و اجزای آن را می‌توان در صفحه توضیحات AWSIM خواند^۴. علاوه بر موارد ذکر شده، این صحنه دارای دو مؤلفه دیگر بسیار مهم و پیچیده به نام‌های محیط (Environment) و خودروی خودران (EgoVehicle) است. حال به توضیح هر کدام می‌پردازیم [۱۵].

۲-۳-۳ مولفه محیط

محیط، یک مؤلفه مهم است که تمام عناصر بصری^۵ را که محیط را در صحنه شبیه‌سازی می‌کنند و همچنین کنترل بر روی آن‌ها فراهم می‌کنند را شامل می‌شود. این مؤلفه شامل دو زیر مؤلفه‌ی Light و Volume نیز می‌شود که به تأمین نور مناسب برای عناصر بصری و شبیه‌سازی شرایط آب و هوا

¹Environment

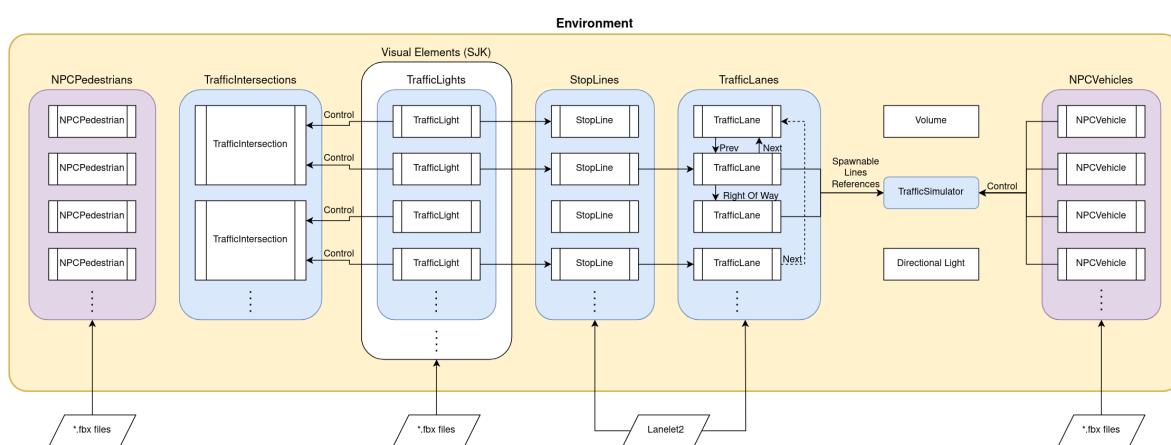
²Assets

³Scene

⁴<https://tier4.github.io/AWSIM/ProjectGuide/Scenes/>

⁵Visual Elements

می‌پردازند^۱. علاوه بر عناصر بصری مانند ساختمان‌ها و برگ‌ها، این مؤلفه شامل کلیه ساختار مربوط به ترافیک نیز می‌شود. ترافیک شامل وسایل نقلیه غیرقابل کنترل (NPCVehicles) می‌شود که توسط مؤلفه شبیه‌ساز ترافیک (TrafficSimulator) در شبیه‌سازی ایجاد می‌شوند و از مؤلفه‌های مرتبط با ترافیک استفاده می‌کنند^۲. عناصر NPCPedestrians نیز جزو مؤلفه‌های محیط به حساب می‌آیند، اما توسط مؤلفه شبیه‌ساز ترافیک کنترل نمی‌شوند. برای حرکت آن‌ها از کدهایی مخصوص استفاده می‌شود^۳.



شکل ۲۱-۳ معماری مؤلفه محیط شبیه‌ساز AWSIM [۱۶]

۱-۲-۳-۳ مولفه‌های ترافیک (Traffic)

مولفه‌های خطوط ترافیک (TrafficLanes) و خطوط ایست (StopLines)، عناصری هستند که از Lanelet2 (نرم‌افزار ساختن نقشه بُرداری) به محیط اضافه می‌شوند. خطوط ترافیک به گونه‌ای مراجعه متقابل^۴ دارند تا مسیرهایی را در محیط Unity بر روی خطوط ترافیک ایجاد کنند. همچنین، هر خط ترافیک موجود در تقاطع^۵، شرایط خاصی برای اعطای حق تقدم دارد. مؤلفه شبیه‌ساز ترافیک، از مولفه خطوط ترافیک برای ایجاد NPCVehicles و تضمین حرکت آن‌ها در امتداد این خطوط استفاده می‌کند. اگر برخی از خطوط ترافیک همچنان پیش از تقاطع به پایان برسند، دارای ارجاعی به خط ایست روبروی خود می‌باشند. هر خط ایست در تقاطع، دارای ارجاع به نزدیک‌ترین چراغ راهنمایی رانندگی (TrafficLight) می‌باشد. چراغ‌های راهنمایی رانندگی به یکی از گروه‌های عناصر بصری تعلق دارند و رابطی برای کنترل عناصر بصری که منابع چراغ‌های ترافیکی (لامپ‌ها) را شبیه‌سازی می‌کنند، فراهم می‌کند. یک مؤلفه

¹ <https://tier4.github.io/AWSIM/Components/Environment/AddNewEnvironment/AddEnvironment/#create-an-environment-prefab>

² <https://tier4.github.io/AWSIM/Components/Traffic/TrafficComponents/>

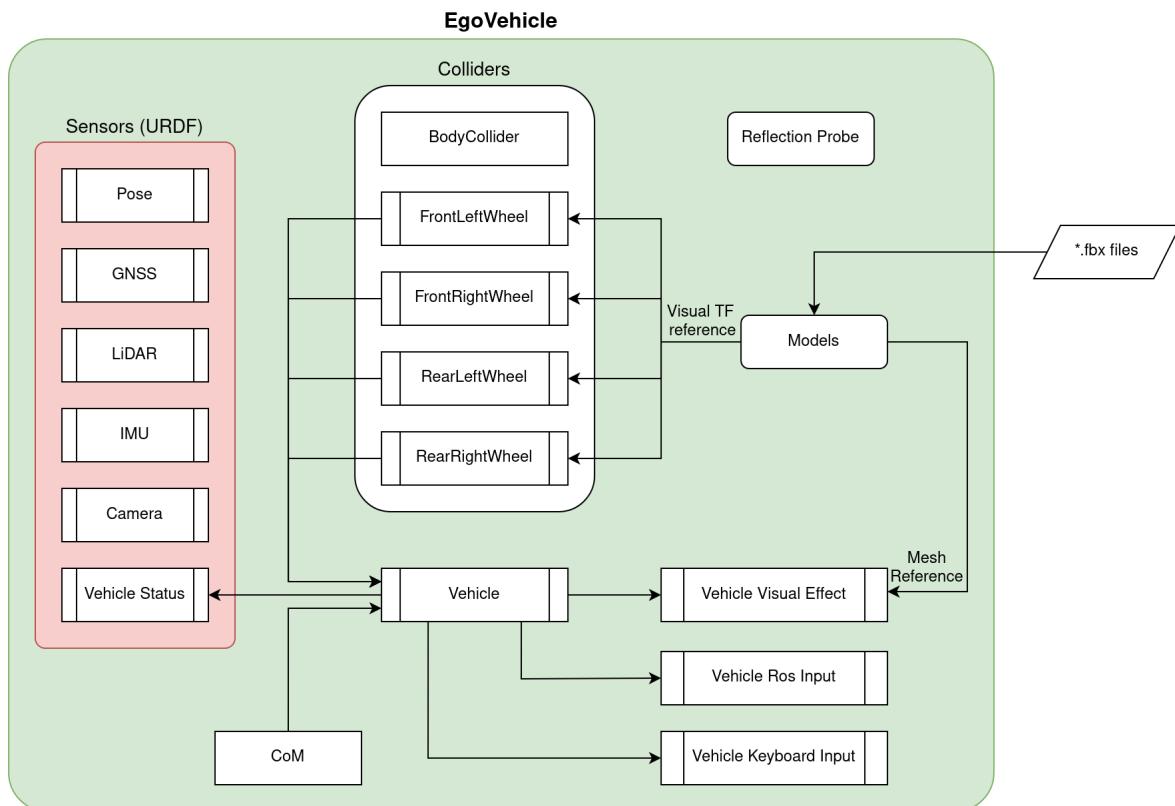
³ <https://tier4.github.io/AWSIM/Components/Traffic/NPCs/Pedestrian/>

⁴ Cross-Reference

⁵ Intersection

تقاطع ترافیکی (TrafficIntersection)، به تنها‌ی مسئول کنترل تمام چراغ‌های راهنمایی رانندگی در یک تقاطع است^۱. [۱۶]

۳-۳-۳ مولفه خودروی خودران



[۱۶] شکل ۳-۲۲-۳ معماری مولفه خودروی خودران شبیه‌ساز AWSIM

مولفه خودروی خودران، یک مؤلفه مسئول برای شبیه‌سازی یک خودروی خودران، حین حرکت در صحنه است. این مؤلفه شامل موارد زیر می‌شود:

- مولفه مدل‌ها (Models) و Reflection Probe که برای ظاهر دیداری ماشین هستند.
- مولفه برخورد دهنده (Collider)، که برخوردها و قابلیت حرکت ماشین بر روی جاده‌ها را فراهم می‌کند.
- مولفه حسگرها که داده‌های مرتبط با وضعیت خودرو در محیط، مانند موقعیت و سرعت آن را فراهم می‌کنند.
- مولفه وسیله نقلیه (Vehicle)، که دینامیک ماشین را شبیه‌سازی می‌کند و حرکت مولفه‌های چرخ (Wheel) را کنترل می‌کند.

^۱<https://tier4.github.io/AWSIM/Components/Traffic/TrafficComponents/>

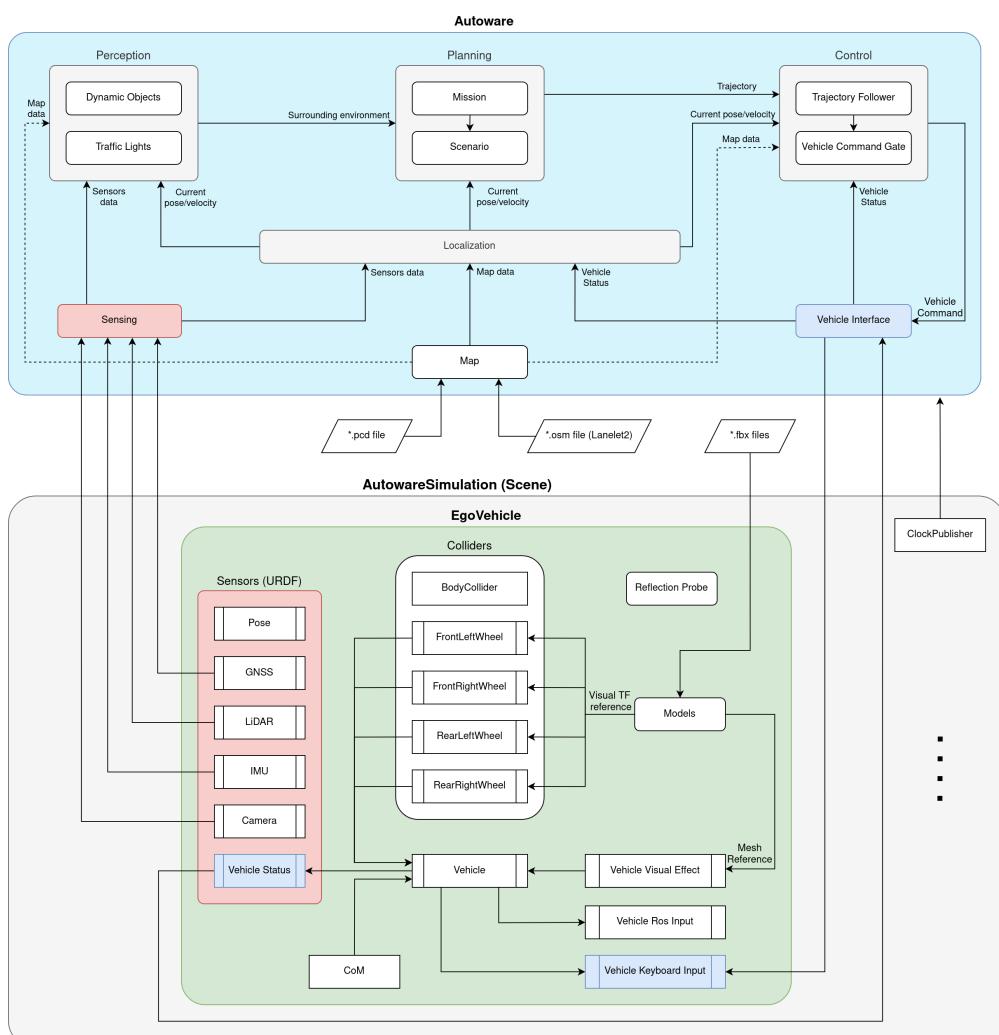
- مولفه‌های Vehicle Keyboard Input و Vehicle Ros Input مرجعی به مولفه وسیله نقلیه دارند و دستورهای کنترلی را به آن می‌دهند.

- مولفه جلوه‌های بصری وسیله نقلیه (Vehicle Visual Effects)، که یک بستر را برای کنترل نورپردازی مولفه وسیله نقلیه، فراهم می‌کند.

توضیحات کامل این مولفه، در صفحه توضیحات شبیه‌ساز AWSIM، آمده است.^۱

۴-۳-۳ شبیه‌ساز AWSIM و ترکیب آن با Autoware

ترکیب Autoware و AWSIM، امکان بررسی درستی رفتار خودروی خودران را در سناریوهای مختلف ترافیکی، فراهم می‌کند. ترکیب AWSIM با Autoware به واسطه مولفه‌های بستر خودرو و احساس



شکل ۲۳-۳ [۱۶] معماری ترکیب Autoware و AWSIM

^۱<https://tier4.github.io/AWSIM/Components/Vehicle/EgoVehicle/>

در معماری Autoware امکان‌پذیر است. مولفه مسئول برای اطمینان از ارتباط با این مدول‌ها از سوی AWSIM، مولفه EgoVehicle است. این مولفه، به معماری Autoware تطبیق یافته و ارتباط بر اساس مباحث ROS2 را فراهم می‌کند. با این حال، یک اجزا مهم دیگر به نام انتشارکننده زمان وجود دارد که زمان شبیه‌سازی را برای Autoware فراهم می‌کند و همچنین بر روی یک مبحث ROS2 منتشر می‌شود.^۱

مولفه EgoVehicle اطلاعات حال حاضر خودرو را از طریق یک اسکریپت درون مولفه وضعیت خودرو (Vehicle Status) منتشر می‌کند. این اطلاعات به صورت بلادرنگ ارائه می‌شود و شامل مواردی نظیر: سرعت جاری، جهت گیری چرخ‌ها یا وضعیت جاری چراغ‌های خودرو است، که از سوی AWSIM به دست می‌آیند و خروجی‌های آن هستند.

از سوی دیگر، مولفه Vehicle Ros Input، مسئول ارائه مقادیر خروجی از Autoware است. این مولفه در دستورات جاری مرتبط با شتاب دادن، دندنه‌های گیربکس یا کنترل نورهای مشخص مشترک می‌شود.

اجرای دستورات دریافتی از طریق مولفه وسیله امکان‌پذیر است، که تنظیم شتاب مناسب بر روی چرخ و کنترل عناصر بصری خودرو را انجام می‌دهد.

سایر داده‌های ارسالی از سوی AWSIM به Autoware، اطلاعات حسگرها هستند که اطلاعاتی در مورد وضعیت فعلی محیط اطراف و اطلاعات مورد نیاز برای تخمین دقیق موقعیت EgoVehicle را فراهم می‌کند^۲. [۱۶]



شکل ۳-۲۴ تصویری از همکاری شبیه‌ساز AWSIM (سمت چپ) و نرم‌افزار Autoware (سمت راست) [۱۶]

¹<https://tier4.github.io/AWSIM/Components/ROS2/ROS2ForUnity/#extension-scripts>

²<https://tier4.github.io/AWSIM/Introduction/CombinationWithAutoware/>

۵-۳-۳ ROS2ForUnity افزونه

مدول (R2FU) یک راه حل ارتباطی است که به طور موثر، اتصالی بین موتور بازی سازی Unity و بستر ROS2 ایجاد می‌کند و یک ادغام قوی را تشکیل می‌دهد. این مدول، برخلاف راه حل‌های دیگر، از ارتباط مستقیم استفاده می‌کند و به جای استفاده از پل ارتباطی، از پشتیه میان‌افزاری ROS2 (به ویژه کتابخانه کلاینت rcl و لایه‌های پایین‌تر آن) استفاده می‌کند، که امکان اضافه کردن گره‌های ROS2 را به شبیه‌سازی‌های Unity را فراهم می‌کند.

R2FU در AWSIM، به دلایل متعددی استفاده می‌شود. اول از همه چیز، به علت ارائه ادغامی با عملکرد بالا بین Unity و ROS2، با بهبود توانایی انتقال و کاهش تأخیر نسبت به راه حل‌های پل ارتباطی، این مدول قابلیت‌های واقعی ROS2 را برای موجودیت‌های شبیه‌سازی در Unity فراهم می‌کند، پیام‌های استاندارد و سفارشی^۱ را پشتیبانی می‌کند و انتزاعات^۲ و ابزارهای مفیدی را ارائه می‌دهد که همگی به عنوان یک عنصر Unity تعییه شده‌اند.

افرونه R2FU، برخی از پیام‌های نرم‌افزارهای ROS2 و Autoware را به صورت پیش‌فرض، پشتیبانی می‌کند که در جدول ۲-۴ مشاهده می‌شوند: برای اینکه بسته پیامی سفارشی را بتوان در Unity استفاده

جدول ۱-۳ جدول پیام‌های پشتیبانی شده توسط R2FU [۱۶]

پیام	بسته
std_msgs geometry_msgs sensor_msgs nav_msgs diagnostic_msgs	common_interfaces
builtin_interfaces action_msgs rosgraph_msgs test_msgs	rcl_interfaces
autoware_auto_control_msgs autoware_auto_geometry_msgs autoware_auto_planning_msgs autoware_auto_mapping_msgs autoware_auto_vehicle_msgs	autoware_auto_msgs
tier4_control_msgs tier4_vehicle_msgs	tier4_autoware_msgs
tf2_msgs unique_identifier_msgs	others

کرد، با استفاده از فایل‌های *.dll و *.so. آن توسط R2FU ساخته شوند^۳.

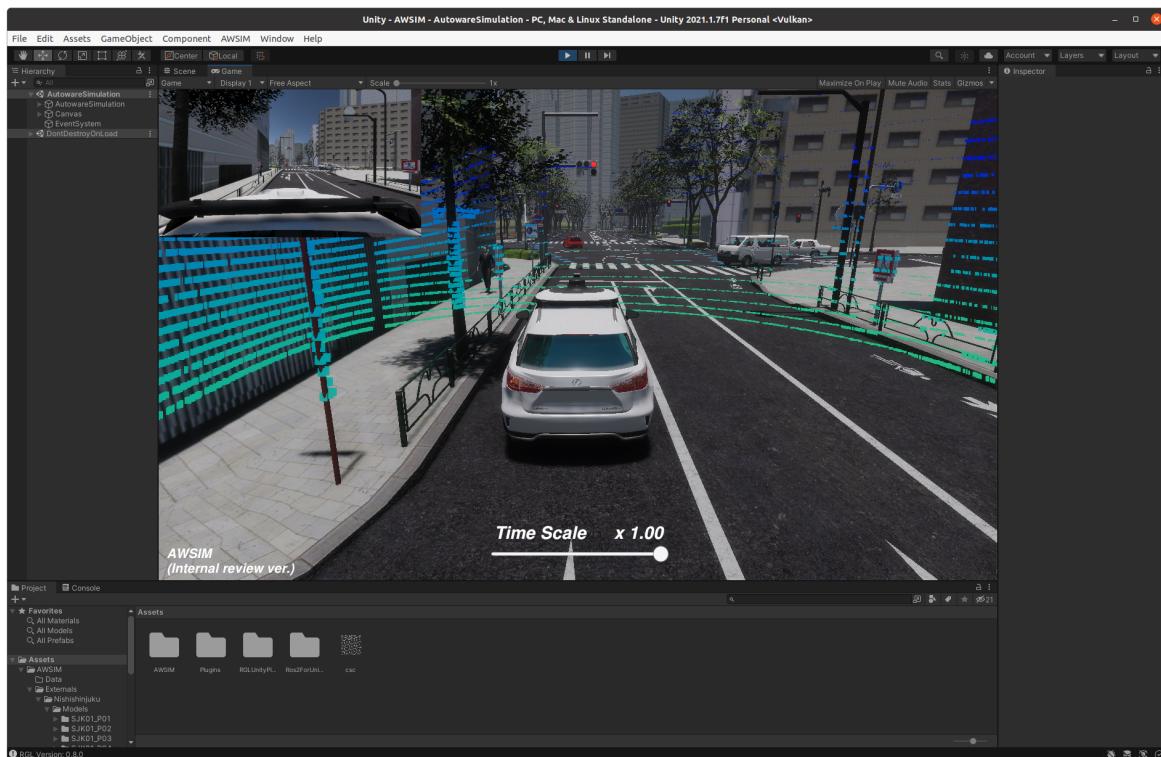
¹Custom

²Abstractions

³<https://tier4.github.io/AWSIM/Components/ROS2/ROS2ForUnity/>

۴-۳ موتور بازی Unity

Unity، یک موتور بازی^۱ و سکو توسعه‌ی گسترده است که در صنایع بازی و رسانه‌های تعاملی بسیار محبوب است. این موتور بازی، به دلیل چند منظوره گرایی و دسترسی آسان شناخته شده است و به موتور انتخابی برای توسعه‌دهندگان تبدیل شده است، که قصد دارند محتوای گسترده‌ای از بازی‌های کامپیوتری تا برنامه‌های واقعیت افزوده و شبیه‌سازی‌ها را ایجاد کنند. Unity سکوهای مختلف را پشتیبانی می‌کند و این امکان را به توسعه‌دهندگان می‌دهد که پروژه‌های خود را بر روی دستگاه‌های مختلف مانند کامپیوتر شخصی، تلفن همراه، کنسول و حتی هدست‌های واقعیت مجازی/افزوده اجرا کنند. فروشگاه عناصر یونیتی دارای مجموعه‌ای بزرگ از منابع پیش‌ساخته، اسکریپت‌ها و ابزارها است که توسعه‌ی پروژه‌ها را ساده‌تر و سریع‌تر می‌کند، در حالی که سیستم اسکریپت‌نویسی بصری‌اش به نام "Bolt" به افرادی که برنامه‌نویس نیستند اجازه می‌دهد تا رفتارها و تعامل‌های پیچیده‌ای را ایجاد کنند. اسکریپت‌های این موتور، با زبان برنامه‌نویسی C# نوشته می‌شوند.



شکل ۳-۲۵ شبهیه‌ساز AWSIM در بستر موتور بازی Unity [۱۶]

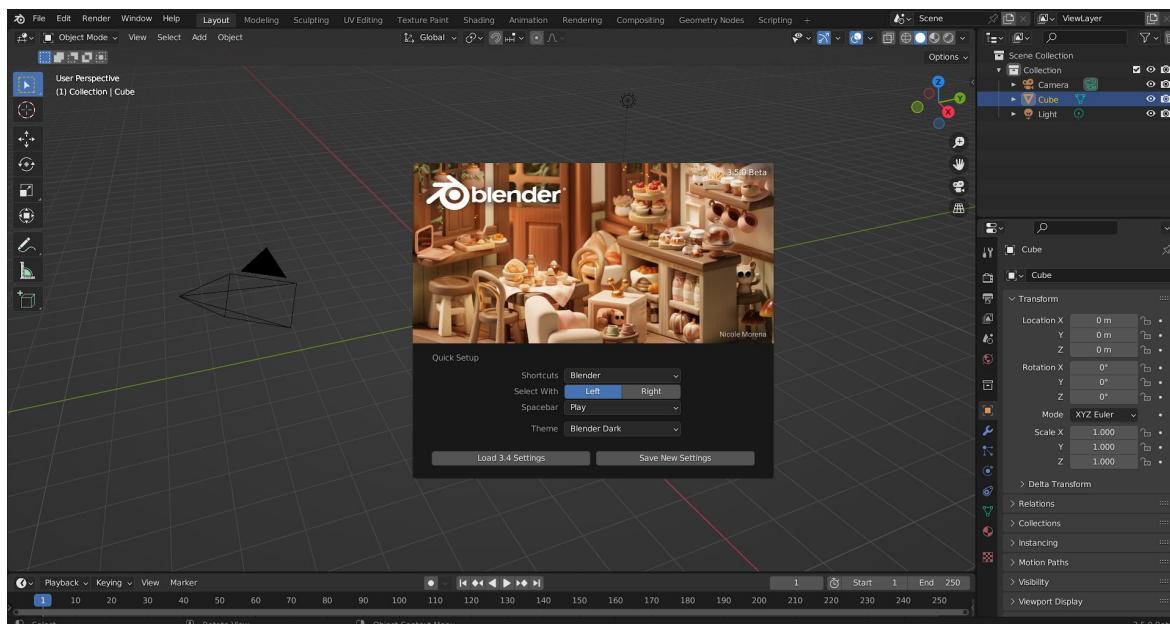
یکی از ویژگی‌های برجسته‌ی Unity، توانایی پردازش بلادرنگ آن است، که تجربیاتی با کیفیت بالای بصری و تعاملی را ارائه می‌دهد. این موتور امکان نورپردازی و سایه‌دهی پویا، شبیه‌سازی فیزیکی و پشتیبانی از گرافیک‌های دو بعدی و سه بعدی را فراهم می‌کند که همگی اساسی برای ایجاد جهان‌های غنی و شبیه‌سازی‌های واقع‌گرایانه هستند. به علاوه، رابط کاربری کاربر پسند این موتور، به جفت

¹Game Engine

توسعه‌دهندگان مبتدی و حرفه‌ای این امکان را می‌دهد که پروتوتایپ‌سازی و تکرار سریع ایده‌های خود را به واقعیت تبدیل کنند. با جامعه‌ی پر رونق توسعه‌دهندگان و مستندات جامع، یونیتی همچنان در حال تکامل است و خالقان را به اجرای ایده‌های خلاقانه خود ترغیب می‌کند و موقعیت برتری را به عنوان یکی از موتورهای اصلی صنعت بازی، برای خود ثبت می‌کند.^۱

۵-۳ Blender ابزار

Blender یک ابزار قدرتمند و چندکاره در دنیای مدل‌سازی سه‌بعدی، انیمیشن و رندرینگ^۲ است که به عنوان یک نرم‌افزار متن‌باز معرفی شده و در دنیای مدل‌سازی سه‌بعدی، انیمیشن و رندرینگ جایگاه ویژه‌ای دارد. این نرم‌افزار به خاطر داشتن مجموعه‌ای گسترده از امکانات معروف است، که از مدل‌سازی سه‌بعدی و نقاشی تا ریگینگ^۳، انیمیشن و حتی ویرایش ویدیویی می‌پردازد. رابط کاربری دوستانه Blender، به همراه مجموعه گسترده ابزار و امکانات، آن را مناسب برای جفت مبتدیان و هنرمندان ماهر می‌کند. این نرم‌افزار به دلیل عملکرد قوی خود و همچنین به عنوان یک جایگزین مقرر بوده برای راه حل‌های نرم‌افزاری سه‌بعدی متعلق به شرکت‌های مختلف، شناخته شده است. این ابزار، برای کار روی انیمیشن‌های پیچیده شخصیتی، تصویرسازی‌های معماري یا اشیاء بازی، Blender مجموعه‌ای جامع از ابزارها را فراهم می‌کند.



شکل ۳ محیط ابزار Blender

یکی از ویژگی‌های برجسته Blender، ادغام آسان آن با Unity است. یونیتی به طور پیش‌فرض،

¹<https://unity.com/>

²Rendering

³Rigging

امکان وارد کردن فایل‌های Blender را پشتیبانی می‌کند، که امکان بارگذاری مدل‌ها و انیمیشن‌های ایجاد شده در Blender به طور مستقیم در پروژه‌های یونیتی را فراهم می‌کند. این سازگاری جربان کار را تسهیل می‌کند، زمان طی شده در فرآیند کار را کاهش می‌دهد و به هنرمندان اطمینان می‌دهد که بتوانند محتوای خود را در Blender ایجاد کرده و آن را به سرعت در محیط Unity مشاهده کنند. ترکیب قدرتمند مدل‌سازی سه‌بعدی و انیمیشن این ابزار، با ابزارهای توسعه بازی همانند Unity، دنیایی از امکانات خلاقانه را ارائه می‌دهد و به توسعه‌دهندگان این امکان را می‌دهد تا محیط‌ها، شخصیت‌ها و اشیاء بازی را به راحتی طراحی و نمونه‌سازی کنند.^۱

۱-۵-۳ Blosm افزونه

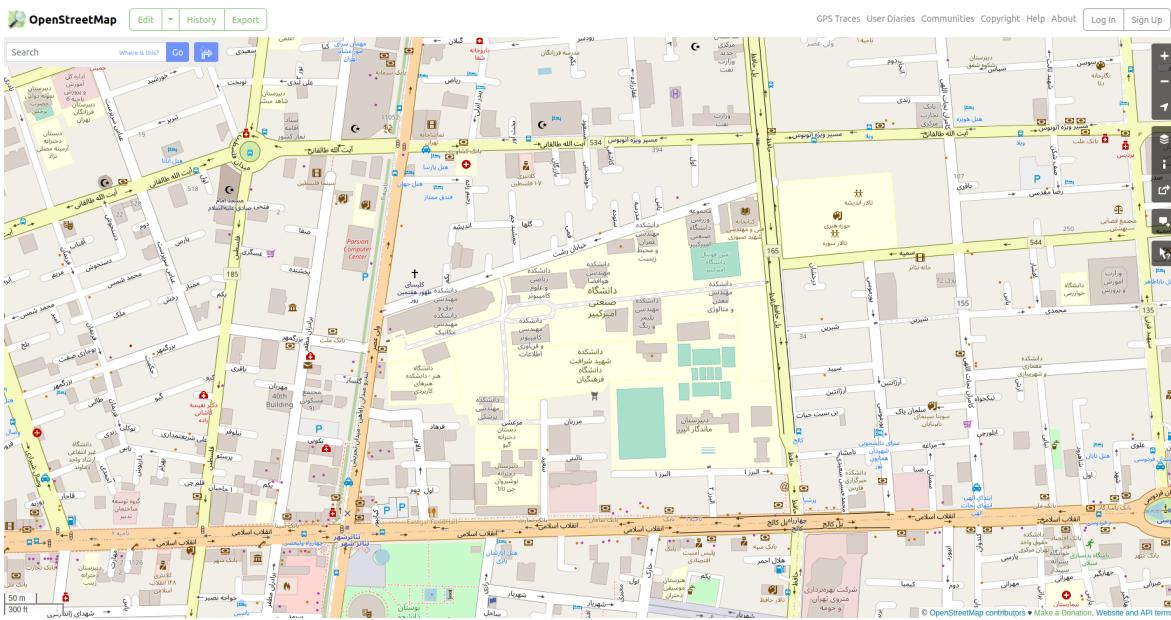
که قبلاً به عنوان Blender-OSM شناخته می‌شد، یک افزونه قدرتمند برای Blender است که طراحی شده تا فرآیند وارد کردن داده‌ها و نقشه‌های جغرافیایی را ساده‌تر کند. با تنها چند کلیک، این افزونه دسترسی به مجموعه‌ای گسترده از منابع داده‌ای فراهم می‌کند، از جمله شهرهای سه‌بعدی گوگل، اطلاعات OpenStreetMap و داده‌های پستی بلندی جهانی. این ابزار چند منظوره به کاربران این امکان را می‌دهد که به آسانی جزئیات جغرافیایی واقعی را به پروژه‌های Blender خود وارد کنند. Blosm، یک پروژه متن‌باز است. قابلیت‌های این افزونه بسیار قدرتمند است و به کاربران این امکان را می‌دهد تا انواع عناصر از جمله ساختمان‌ها، اشیاء آبی مانند رودخانه‌ها و دریاچه‌ها، مناطق گیاهی، جاده‌ها، مسیرها، راه‌آهن‌ها و داده‌های دقیق زمین با وضوح پوشش جهانی تقریباً ۳۰ متر را وارد کنند.^۲

۶-۳ ابزار OpenStreetMap

ابزار طراحی نقشه‌برداری (OSM)، یک پروژه مشترک و متن‌باز است که یک مخزن گسترده از داده‌های مکانی ارائه می‌دهد، شامل نقشه‌های دقیق و اطلاعات جغرافیایی که توسط داوطلبان از سراسر جهان تهیه می‌شود. این مشارکت‌ها مجموعه‌ای وسیع از ویژگی‌های جغرافیایی از جمله جاده‌ها، نمادها، شهرها و موارد دیگر را در بر می‌گیرد، که از مهم‌ترین منابع برای برنامه‌هایی مانند ناوبری، برنامه‌ریزی شهری، مدیریت بحران و خدمات مکانی مختلف هستند. مدل داده باز OSM، کاربران را تشویق به ویرایش و بهبود نقشه‌ها می‌کند، تضمین می‌کند که داده‌های این پلتفرم به روز می‌شود و برای مجموعه‌ای گسترده از موارد کاربردی معتبر باقی می‌ماند.

¹<https://www.blender.org/>

²<https://github.com/vvoovv/blosm/wiki/Documentation>



شکل ۲۷-۳ نقشه دانشگاه صنعتی امیرکبیر در ابزار OSM

۷-۳ کتابخانه OusterSDK

کتابخانه توسعه نرم افزار حسگرهای لایدار شرکت Ouster، واسطه‌هایی را برای تعامل با سخت افزار حسگر و داده‌های حسگر ضبط شده که برای انجام آزمون‌ها، ارزیابی، و برنامه‌های غیر مرتبط با اینمی، مناسب هستند را در زبان‌های پایتون و C++ فراهم می‌کند. کدهای مثال و مرجع برای عملیات معمول بر روی داده‌های حسگر در هر دو زبان نیز ارائه شده است. این کتابخانه، شامل واسطه‌های برنامه‌نویسی برای:

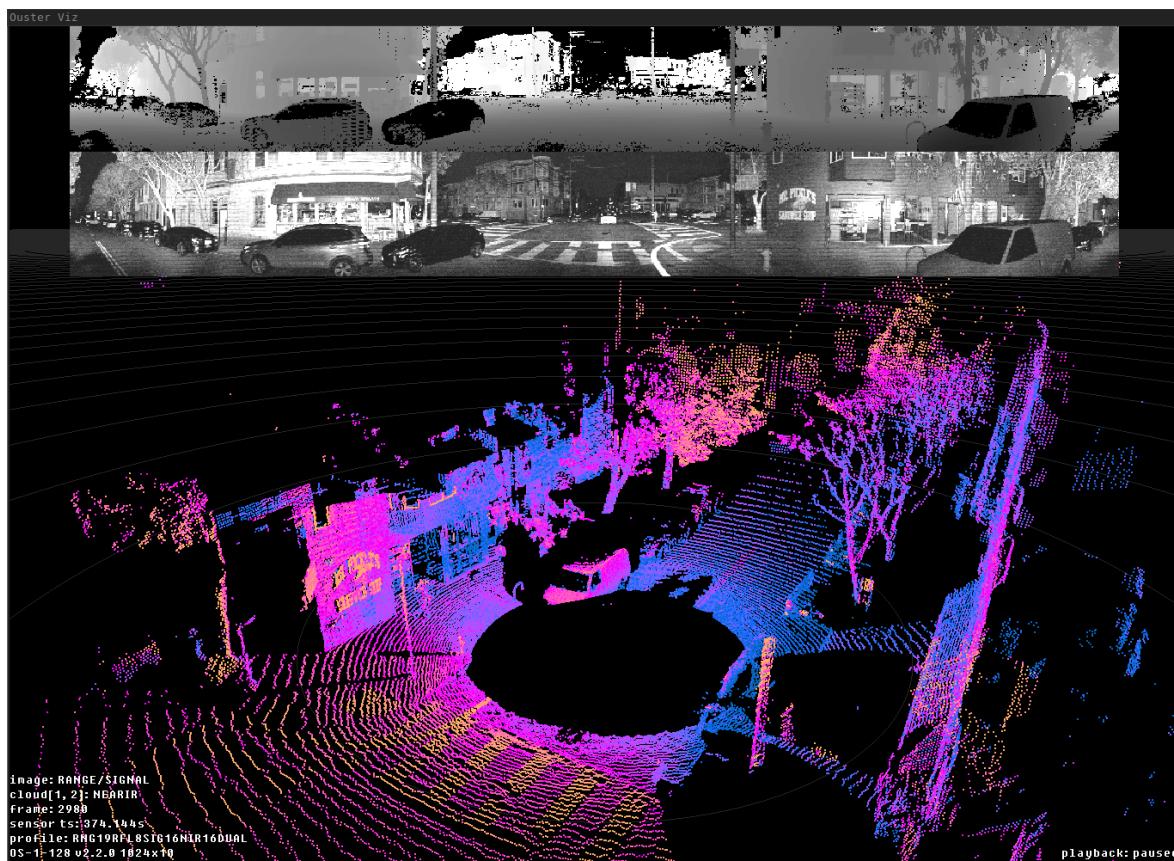
۱. پرس و جوی تنظیمات حسگر و تنظیمات آنها
 ۲. ضبط و خواندن داده‌ها به فرمت .pcap
 ۳. خواندن و بافرینگ داده‌های UDP به صورت قابل اطمینان
 ۴. تبدیل داده‌های خام به تصاویر فاصله^۱، سیگنال^۲، نور مادون قرمز نزدیک^۳، انعکاس پذیری^۴
 ۵. تبدیل اندازه‌گیری‌های کانال فاصله به مختصات کارتزین (x, y, z)
- علاوه بر این، این کتابخانه در زبان پایتون قابلیت‌های زیر را نیز دارد:
۱. دسترسی به فریم‌های داده‌های لایدار به عنوان داده‌های numpy
 ۲. یک ابزار نمایشگر برای داده‌های ضبط شده‌ی .pcap و حسگر لایدار [۱۷] (شکل ۲۸-۳)

¹Range

²Signal

³Near-IR

⁴Reflectivity



شکل ۲۸-۳ ابزار نمایشگر کتابخانه OusterSDK برای ابر نقاط [۱۷]

۸-۳ ابزار Ouster Studio

ابزار Ouster Studio یک نرمافزار قدرتمند و کاربرپسند است که برای پردازش و نمایش داده‌های لایدار توسعه یافته است. با Ouster Studio، شما می‌توانید به سرعت و سادگی داده‌های لایدار Ouster را وارد کرده، تجزیه و تحلیل کرده و نمایش دهید. این نرمافزار برای ساده‌تر کردن فرآیند تنظیم و نمایش حسگرها ساخته شده و با کشف خودکار حسگر، امکان تشخیص و شناسایی حسگرهای متصل به یک شبکه یا سیستم را فراهم می‌کند. علاوه بر این، امکان پخش زنده، به کاربران این امکان را می‌دهد که داده‌های حسگر را به صورت بلاذرنگ دریافت و پردازش کنند، که می‌تواند برای به دست آوردن بینش‌های فوری و امکان تصمیم‌گیری به موقع، مفید باشد. این نرمافزار به شما این امکان را می‌دهد که داده‌ها را به سرعت خود مرور و مطالعه کنید؛ همچنین امکان به اشتراک گذاری ضبط داده‌های لایدار ضبط شده و شناسایی الگوها و روندهای آن را فراهم می‌کند، که ممکن است در داده‌های بلاذرنگ به سرعت مشخص نشود.

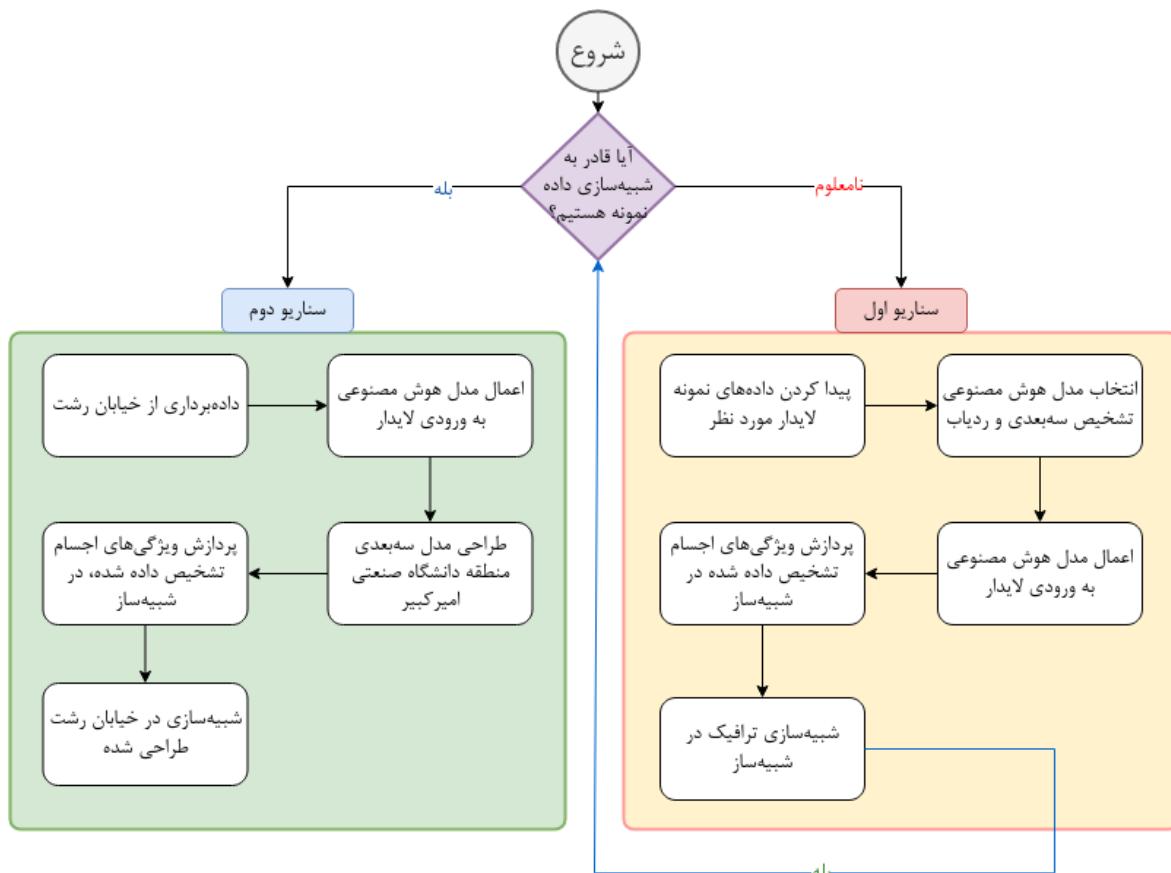
فصل چهارم

پیاده‌سازی و طراحی

در این فصل، به چگونگی پیاده‌سازی پروژه و طریقه شکل‌گیری دو قلوی دیجیتال خیابان رشت می‌پردازیم.

۱-۴ نگاهی به مسیر طی شده

از آنجایی که این پروژه، پژوهشی بزرگ محسوب می‌شود، نیاز است تا فلوچارتی^۱ برای درک آن کشیده شود.



شکل ۱-۴ فلوچارت پروژه

همانطور که طبق شکل ۱-۴ مشاهده می‌شود، اولین سوالی که پیش می‌آید، امکان‌پذیر بودن این پژوهش است.

۲-۴ سناریو اول: استفاده از داده‌های نمونه

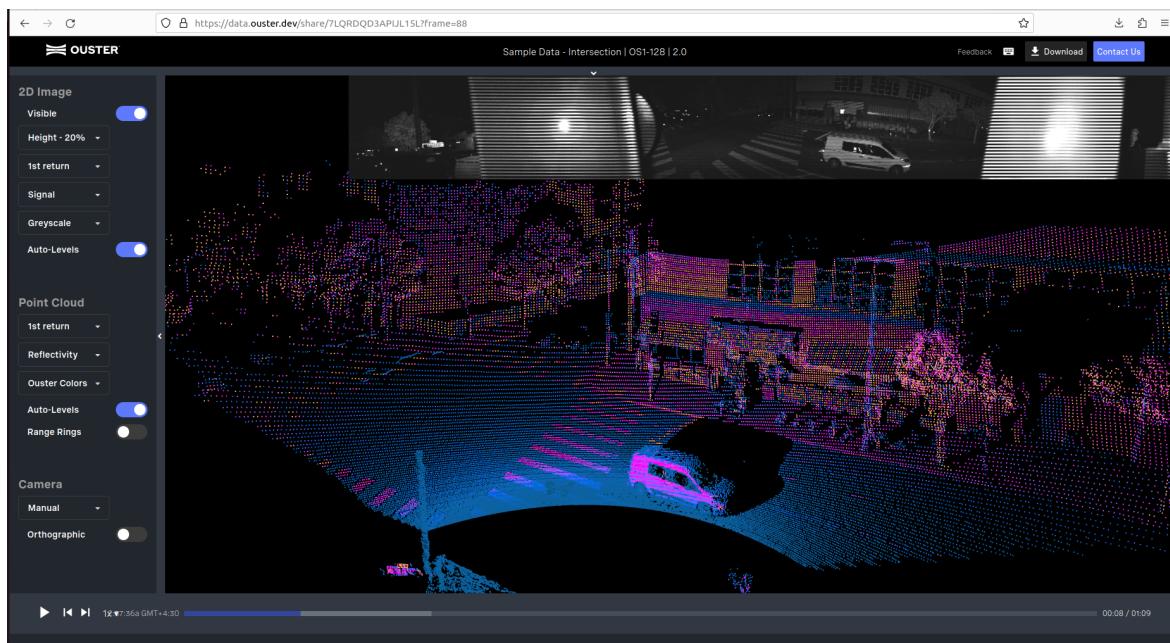
گرفتن مجوز برای داده‌برداری از خیابان رشت و هماهنگی‌های لازم آن، هزینه‌بر است و تا زمانی که نتوان از امکان‌پذیر بودن این پژوهش اطمینان حاصل یافت، نمی‌توان آن را انجام داد. به همین دلیل، در ابتدا

¹ Flowchart

سناریو اول پیش گرفته شد که در آن، اقدام به شبیه‌سازی داده‌های نمونه موجود در تارنمای^۱ شرکت Ouster شده است.

۱-۲-۴ پیدا کردن داده‌های نمونه لایدارهای شرکت Ouster

تارنمای شرکت Ouster، داده‌های نمونه‌ای را در اختیار توسعه‌دهندگان گذاشته است. یکی از این داده‌های نمونه، حسگر لایداری است که بر روی یک چراغ راهنمایی رانندگی، در یک چهارراه نصب شده است.^۲



شکل ۲-۴ داده‌ی نمونه که با لایدار OS1:128 ضبط شده است.

با توجه به شکل ۲-۴، مشاهده می‌شود که می‌توان داده نمونه را بر روی کامپیوتر شخصی نیز دریافت کرد. داده‌های ابرنقاط عموماً حجمی هستند و به طور مثال، داده فوق حدود ۲ گیگابایت^۳ حجم دارد.

۴-۳ انتخاب مدل هوش مصنوعی تشخیص سه‌بعدی و ردیاب

حال زمان آن رسیده است که مدل هوش مصنوعی تشخیص دهنده‌ای را برای این پژوهش انتخاب کرد. در این پژوهش، تشخیص اجسام تنها با استفاده از ابر نقاط صورت می‌گیرد. از فصل مفاهیم پایه به یاد داریم که سه روش مختلف برای تشخیص اجسام سه‌بعدی استفاده می‌شود. عموماً سرعت مدل‌های مبتنی بر واکسل، از دیگر روش‌ها بیشتر است و از آنجایی که در این پژوهش، داده‌های ورودی بلادرنگ

¹ Website

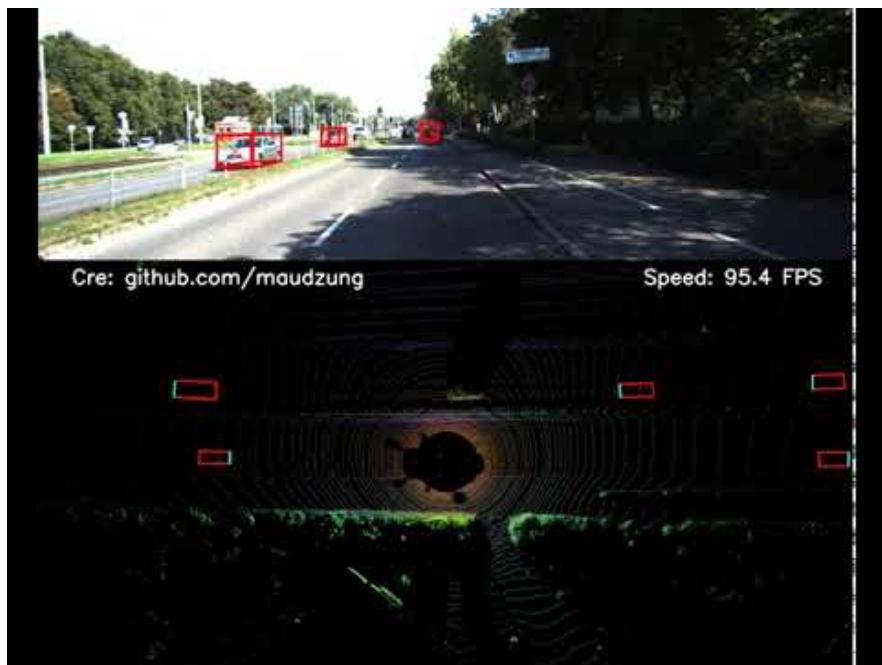
² <https://data.ouster.dev/#/share/7LQRDQD3APIJL15L>

³ Gigabyte

هستند، بایستی سرعت مدل جدی گرفته شود. با جستجو و تحقیقات در مورد مدل‌های تشخیص اجسام سه بعدی، به چند مخزن^۱ و مقاله دست یافته شده است.

۱-۳-۴ مدل SFA3D

این مخزن معروف، مدلی بسیار سریع است که از مجموعه داده KITTI برای تمرین و آزمون استفاده می‌کند. در ابتدا این مدل ایده‌آل به نظر می‌رسید، اما مشکلاتی دارد که عبارتند از:



شکل ۳-۴ تشخیص بلاذرنگ مدل SFA3D، با سرعت تشخیص بیشتر از ۹۰ فریم در ثانیه [۱۸]

۱. این مدل از مجموعه داده KITTI استفاده می‌کند که نسبت به دو مجموعه داده Waymo Open و nuScenes، بسیار ناقص‌تر است. این مدل بر روی مجموعه داده KITTI خوب عمل می‌کند، اما در عمل با افت دقت روبرو خواهد شد.

۲. این مدل، سیستمی برای ردیابی و انتساب شناسه به اجسام ندارد.

۳. این مدل نسبتاً قدیمی است و مدل‌های بهتر و قوی‌تری وجود دارند.

۲-۳-۴ مخزن MMDetection3D

مخزن MMDetection3D، یک جعبه‌ابزار متن‌باز برای تشخیص اجسام سه‌بعدی است. این مخزن توسط شرکت OpenMMLab^۲ از کشور چین ساخته شده است، که جعبه‌ابزارهایی را در انواع زمینه‌های هوش

¹Repository

²<https://openmmlab.com/>

Methods	MMDetection3D	OpenPCDet	votenet	Det3D
VoteNet	358	✗	77	✗
PointPillars-car	141	✗	✗	140
PointPillars-3class	107	44	✗	✗
SECOND	40	30	✗	✗
Part-A2	17	14	✗	✗

[۱۹] تعداد مدل‌های پیاده‌سازی شده در مخزن MMDetection3D

مصنوعی، طراحی کرده است. کاربران با توجه به نیازمندی خود، می‌توانند به مخزن هر کدام از این جعبه‌ابزارها مراجعه کنند. این جعبه‌ابزارها بسترهایی هستند که با استفاده از آنها، کاربران می‌توانند مدل‌های دلخواه خود را انتخاب کنند و تنها با دریافت کردن مجموعه‌داده مورد نیاز، مدل انتخاب شده را آموخت دهند. به طور مثال، در مخزن MMDetection3D، بیش از ۳۰۰ مدل مختلف پیاده‌سازی شده‌اند و مدل‌های جدید همواره در حال پیاده‌سازی و اضافه شدن به این مخزن هستند.

طبق شکل ۴-۴، می‌توان مشاهده کرد که این مخزن در حال حاضر ۱۰۷ عدد مدل ابرنقطاط محور مبتنی بر واکسل دارد و از معماری PointPillars استفاده می‌کنند. یکی از مدل‌های معروف این مخزن، که در سال ۲۰۲۱ ساخته شد و همچنان از آن استفاده می‌شود، مدل CenterPoint [۲۰] است.

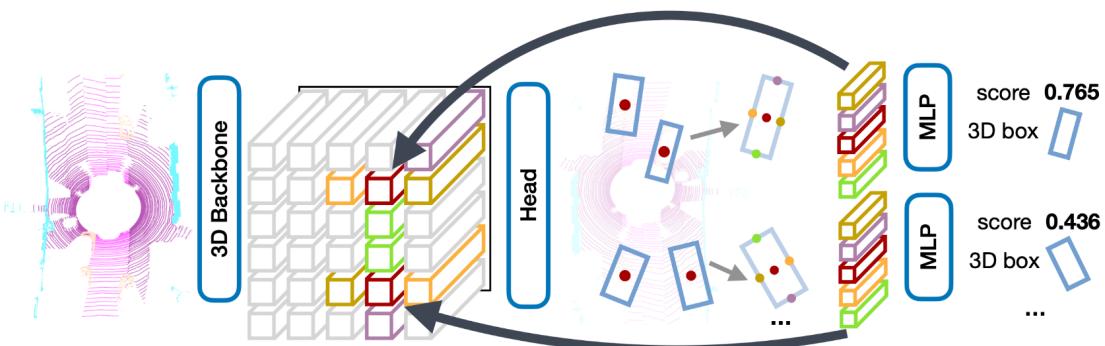
۱-۲-۳-۴ مدل تشخیص اجسام سه‌بعدی CenterPoint

اشیاء سه‌بعدی معمولاً به عنوان کادرهای محصور کننده سه‌بعدی در ابرنقطاط نمایش داده می‌شوند. این نمایش، شبیه کشف کادرهای محصور کننده دو بعدی بر پایه تصویر است، اما در فضای سه‌بعدی با مشکلاتی مواجه می‌شود. اشیاء در جهان سه‌بعدی هیچ جهت خاصی را دنبال نمی‌کنند و مدل‌های مبتنی بر این کادرها، مشکلاتی در شمارش تمام جهت‌ها یا تلفیق یک کادر مرزی تطبیقی به اشیاء چرخان دارند. اما در این مقاله، اشیاء سه‌بعدی به عنوان یک سری نقاط، نمایش، تشخیص و ردیابی می‌شوند. چهارچوب این مدل، CenterPoint، ابتدا مراکز اشیاء را با استفاده از یک تشخیص‌دهنده نقاط کلیدی^۱ یافته و سپس به ویژگی‌های دیگر از جمله اندازه سه‌بعدی، جهت سه‌بعدی و سرعت بازمی‌گرداند. در مرحله دوم، این تخمین‌ها را با استفاده از ویژگی‌های نقطه‌ای دیگر استنتاج شده از اجسام، بهبود می‌دهد. در مرحله دوم، این تخمین‌ها را با تطبیق حریصانه به نزدیک‌ترین نقطه^۲ ساده‌سازی می‌شود. CenterPoint، ردیابی اجسام سه‌بعدی با تطبیق حریصانه به نزدیک‌ترین نقطه^۲ ساده‌سازی می‌شود زیرا اجسام به نقاط معارف آن جسم تصویر شده‌اند. الگوریتم تشخیص و ردیابی حاصل، ساده، کارآمد و موثر است. CenterPoint عملکرد برجسته‌ای را در آزمایش‌های آنلاین مجموعه داده nuScenes در رده‌بندی‌های تشخیص و ردیابی سه‌بعدی، به ثبت رساند. این مدل با ۶۵.۵ امتیاز NDS و ۶۳.۸ امتیاز AMOTA برای یک مدل مبتنی بر ابر نقاط، به بهترین عملکرد ممکن در زمان خود رسید. این مدل در

¹Keypoint Detector

²Greedy Closest-Point Matching

مجموعه داده Waymo Open، تمام روش‌های مدل تک مرحله‌ای قبلی را با اختلاف زیادی پیش‌می‌گیرد و در میان تمام مدل‌های مبتنی بر ابر نقاط، جایگاه اول را در سال ۲۰۲۱ گرفت [۲۰].



[۲۰] شکل ۴-۵ معماری مدل CenterPoint

جدول ۱-۴ جدول مقایسه مدل‌های سه بعدی براساس ارزیابی مجموعه داده Waymo Open [۲۰]

سختی	مدل	خودرو	خودرو	عابر پیاده	عابر پیاده
		mAPH	mAP	mAPH	mAP
	StarNet	۶۱.۰	۶۱.۵	۶۷.۸	۵۹.۹
	PointPillars	۶۲.۸	۶۳.۳	۶۲.۱	۵۰.۲
Level 1	PPBA	۶۷.۰	۶۷.۵	۶۹.۷	۶۱.۷
	RCD	۷۱.۶	۷۲.۰	-	-
	CenterPoint	۷۹.۷	۸۰.۲	۷۸.۳	۷۲.۱
	StarNet	۵۴.۵	۵۴.۹	۶۱.۱	۵۴.۰
	PointPillars	۵۵.۱	۵۵.۶	۵۵.۹	۴۵.۱
Level 2	PPBA	۵۹.۱	۵۹.۶	۶۳.۰	۵۵.۸
	RCD	۶۴.۷	۶۵.۱	-	-
	CenterPoint	۷۱.۸	۷۲.۲	۷۲.۲	۶۴.۴

جدول ۴-۵ جدول مقایسه مدل‌های سه بعدی براساس ارزیابی مجموعه داده nuScenes [۲۰]

روش	mAP↑	NDS↑	PKL↓
WYSIWYG	۳۵.۰	۴۱.۹	۱.۱۴
PointPillars	۴۰.۱	۵۵.۰	۱.۰۰
CVCNet	۵۵.۳	۶۴.۴	۰.۹۲
PointPainting	۴۶.۴	۵۸.۱	۰.۸۹
PMPNet	۵۴.۴	۵۳.۱	۰.۸۱
SSN	۴۶.۳	۵۶.۹	۰.۷۷
CBGS	۵۲.۹	۶۳.۳	۰.۷۷
CenterPoint	۵۸.۰	۶۵.۵	۰.۶۹

این مدل، در مخزن MMDetection3D نیز پیاده‌سازی شده است و برای این پژوهش، ایده‌آل است. در ابتدا سعی در نصب این مخزن شد و اقدام به آموزش این مدل شد. اما موقع آموزش مدل Centerpoint

با چالش‌هایی روبرو شدیم:

- مجموعه داده nuScenes، دارای حجمی حدود ۱ ترابایت^۱ است و محدودیت حجمی کامپیوتر، مانع فرایند آموزش می‌شود. مجموعه داده آموزشی Waymo Open نیز بسیار حجمی است.
- به علت داشتن مجموعه داده‌ای عظیم برای آموزش، آموزش این مدل مدت زمان بسیار زیادی را نیاز خواهد داشت.

با وجود این دشواری‌ها، به نظر می‌آمد که نمی‌توان مدل تشخیص دهنده سه‌بعدی‌ای را آموزش داد. پس تنها راه باقی مانده، استفاده از مدل‌های از پیش آموزش دیده است.

۳-۳-۴ ایده: نگاه انداختن به معماری Autoware

در بخش قبل نتیجه گرفته شده که به یک مدل تشخیص اجسام سه‌بعدی آماده و از پیش آموزش دیده نیاز است. پس باستی به دنبال چنین مدلی گشت. حال سوالی پیش می‌آید: مگر خودروهای خودران نیاز به تشخیص بلادرنگ اجسام در محدودیت‌های زمانی چند دهم ثانیه‌ای نیستند؟ این خودروها نیز قطعاً در داخل معماری خود، از یک مدل تشخیص اجسام استفاده می‌کنند. همچنین از آنجایی که مسیر حرکت وسایل نقلیه اطراف و سرعت آنها نیز برای این خودروها مهم است، پس قطعاً مجهز به یک ردیاب نیز هستند. پس گام بعدی پژوهش، نگاه انداختن به معماری محبوب‌ترین ابزار توسعه خودروهای خودران یا همان Autoware بود. در فصل قبل، معماری مولفه‌های مهم برای پژوهش این ابزار، به صورت دقیق بررسی شدند. این معماری‌ها، در اصل برای پیدا کردن مدل هوش مصنوعی و ردیاب مورد استفاده آن‌ها مطالعه شده‌اند.

۴-۳-۴ مولفه ادارک نرم‌افزار Autoware

طبق شکل ۱۶-۳، در معماری مولفه ادارک Autoware، گره‌ای به نام گره lidar_centerpoint وجود دارد. این گره، همان مدل هوش مصنوعی انتخابی برای آموزش در این پژوهش است. توسعه‌دهندگان ابزار Autoware، با استفاده از همان مخزن MMDetection3D و با استفاده از مجموعه داده آموزشی PointPillars، مدل nuScenes را آموزش داده‌اند. آنها در معماری مدل خود، از مدل هسته CenterPoint، که توسط محققین NVIDIA ساخته شده است، استفاده کرده‌اند. این مدل هسته، سرعت چشم‌گیری دارد زیرا از شبکه‌های عصبی TensorRT استفاده می‌کند [۲۳].

همچنین با نگاهی دقیق‌تر به معماری مولفه ادارک، متوجه گره multi_object_tracker، یا ردیاب همزمان چند جسم، می‌شویم. تمامی ردیاب‌ها از الگوریتمی به نام ارتباط داده^۲ استفاده می‌کنند. ارتباط داده‌ها، به فرآیند ارتباط دادن یا متناظر کردن مشاهدات (نقاط داده^۳) از حسگرها، مانند داده‌هایی که

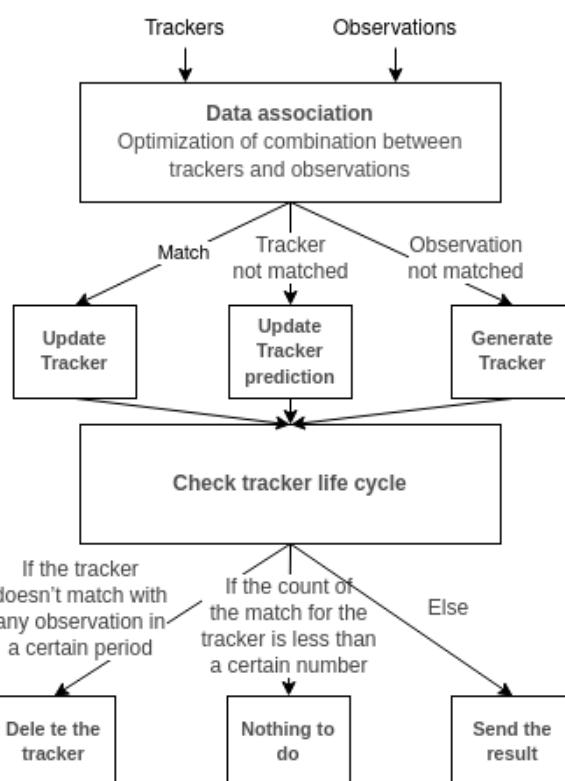
¹Terabyte

²Data Association

³Data Points

از حسگرهای ادراکی مثل لایدار یا رادار جمع‌آوری شده‌اند، به اشیائی خاص یا ردهایی در محیط اشاره دارد. هدف این است که تعیین شود کدام مشاهدات هوش مصنوعی، به کدام اشیاء در فریم قبلی به صورت دقیق متناظر می‌شوند.

الگوریتم‌های ارتباط داده، مسئله‌ای به نام تطبیق امتیاز بیشینه^۱ دارند، که اشاره به یک روش دارد که به دنبال یافتن بهترین ارتباطات بین مشاهدات و اشیاء یا ردها، براساس یک امتیاز یا هزینه می‌باشد. این فرآیند گاهی به عنوان "مسئله حداقل هزینه حداکثر جریان"^۲ نیز شناخته می‌شود و عموماً با استفاده از الگوریتم‌هایی مانند حل کننده muSSP حل می‌شود. این حل کننده، اتصال بهینه مشاهدات به اشیاء را با کمینه کردن هزینه کل ارتباطات بهینه می‌کند، که ممکن است شامل فاکتورهایی مانند مساحت شیء از نمای دید پرند، فاصله ماهالانوبیس و حداقل فاصله باشد. این معیارها به تعیین کیفیت ارتباط بین یک مشاهده و یک شیء کمک می‌کنند.



شکل ۶-۴ نحوه کارکرد گره ردیاب همزمان چند جسم

در گره ردیاب همزمان چند جسم، از فیلترهای کالمن تعمیم‌یافته شده^۳ استفاده می‌شود تا وضعیت انواع اجسام مختلف حاضر در صحنه را تخمین بزند. این گره، مدل‌های ویژه‌ای را برای عابرین پیاده، دوچرخه‌ها، خودروها و وسائل نقلیه بزرگ تعریف کرده است، زیرا ابعاد آن‌ها بسیار متفاوت است و یک فیلتر کالمن جوابگوی تمام حالات نیست. این مدل‌ها به صورت موازی اجرا می‌شوند و بدین شکل، یک

¹Maximum Score Matching

²Minimum Cost Maximum Flow

³EKF-Filter

سیستم ردیاب پایدار و دقیق پیاده‌سازی می‌شود [۱۶].

۵-۳-۴ نتیجه‌گیری

در این بخش، به این نتیجه رسیده شد که آموزش مدل‌های تشخیص اجسام سه‌بعدی، نیاز به مجموعه داده‌های عظیم دارد که خارج از توانایی این پژوهش است. راه حل پیشنهادی، استفاده از مدل‌های از پیش آموزش داده شده است. ایده‌ای که در این زمینه زده شد، استفاده از مدل هوش مصنوعی مورد استفاده ابزار Autoware، برای تشخیص و ردیابی اجسام است.

۴-۴ اعمال مدل هوش مصنوعی به ورودی لایدار

در بخش قبلی، به این نتیجه رسیدیم که می‌توان از مدل CenterPoint مورد استفاده در ابزار Autoware به عنوان تشخیص دهنده اجسام استفاده کرد. همچنین می‌توان از گره ردیاب همزمان چند جسم آن، برای ردیابی اجسام تشخیص داده شده استفاده کرد. با نگاهی دیگر به شکل ۱۶-۳ که گراف گره مولفه ادراک است، در می‌یابیم که گرهی lidar_centerpoint، دارای اشتراکی به یک مبحث^۱ است. این مبحث، خروجی پیش‌پردازش صورت گرفته بر روی ابر نقاط خام لایدارهای خودروی خودران است. پس گمان می‌رود که با انتشار ابر نقاط خود به این مبحث، بتوانیم از لایه‌ی ادراک استفاده کنیم، که یعنی عملیات تشخیص و ردیابی بر روی ابر نقاط داده‌ی نمونه ما انجام شود.

جنس داده‌ی این مبحث، از نوع پیام‌های sensor_msgs/PointCloud2 است. اما فرمت داده‌های ضبط شده‌ی لایدارهای شرکت Ouster، فرمت pcap.* است که در داخل آن بسته‌های^۲ شبکه‌ای لایدار ذخیره شده‌اند. پس نیاز است که این بسته‌ها در درجه اول پردازش شوند، و سپس اطلاعات ابر نقاط داخل هر بسته آن به پیامی از جنس PointCloud2، که از پیام‌های پیش‌فرض برای ابر نقاط در بستر ROS2 است، تبدیل بشوند. به همین منظور، کدی در بستر ROS2 نوشته شد، که داده‌های بلادرنگ یا ROS2 ضبط شده‌ی لایدارهای Ouster را به ابر نقاط مورد استفاده در ROS2، تبدیل کند. این بسته³ که "pcap_to_pointcloud2_publisher_py" نام دارد، با استفاده از کتابخانه کلاینت rclpy نوشته شده است که با استفاده از آن می‌توان از کتابخانه‌های ROS2 در زبان برنامه‌نویسی پایتون استفاده کرد. در این بسته، با استفاده از کتابخانه OusterSDK، داده‌های pcap. خوانده می‌شوند و سپس با استفاده از کانال فاصله لایدار، مختصات کارتزین ابر نقاط هر فریم بدست می‌آید. همچنین شدت سیگنال بازتابی هر کدام از نقاط با استفاده از کانال انعکاس‌پذیری به دست می‌آید. سپس با استفاده از کتابخانه رابط PointCloud2 که برای بسته ROS2 است، این نقاط را به همراه سرایند^۴ که در آن زمان انتشار پیام به

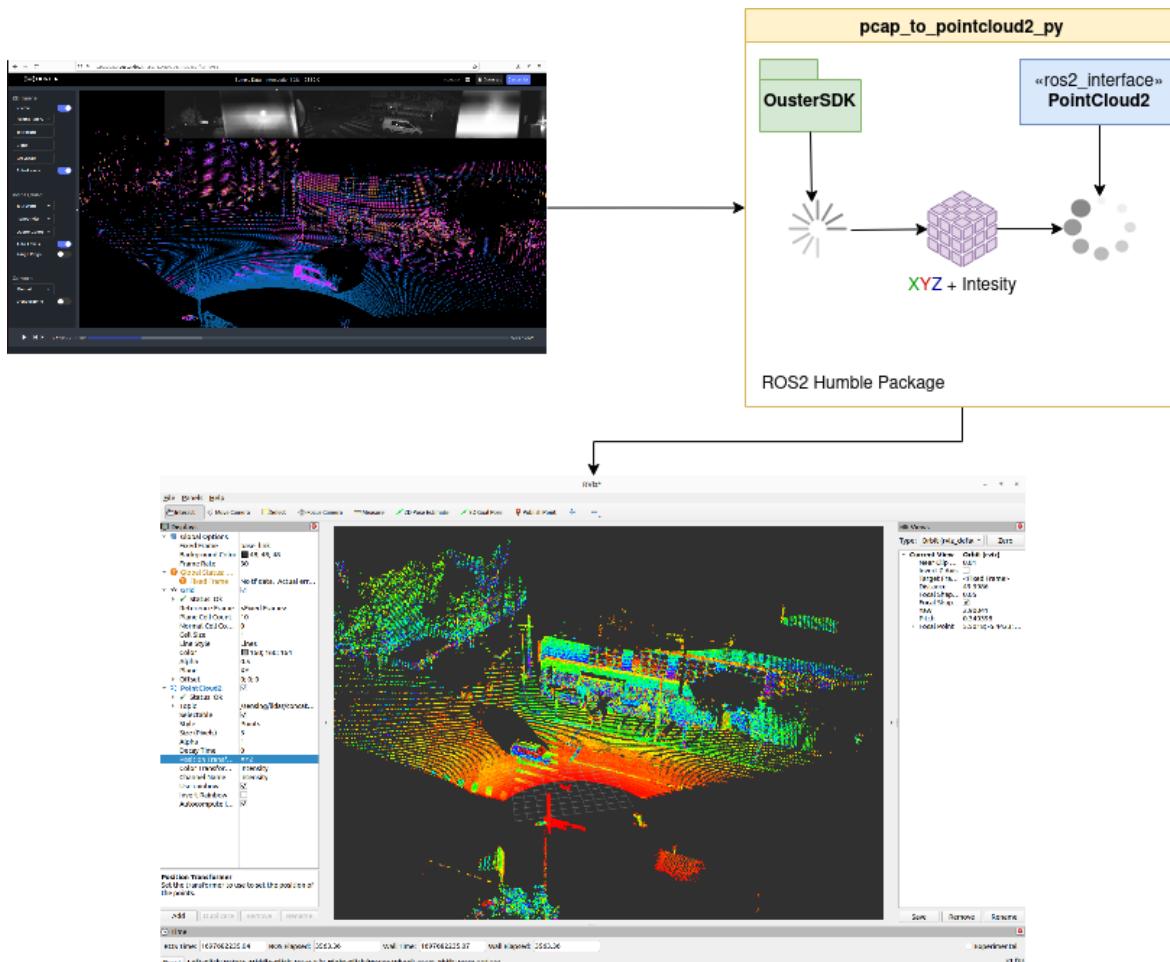
¹/sensing/lidar/concatenated/pointcloud

²Packet

³Python Programming Language

⁴Header

همراه دستگاه مختصاتی مربوط به آن است، منتشر می‌کند.



شکل ۷-۴ روند تبدیل داده‌های ضبط شده لایدار به پیام PointCloud2 و انتشار آن.

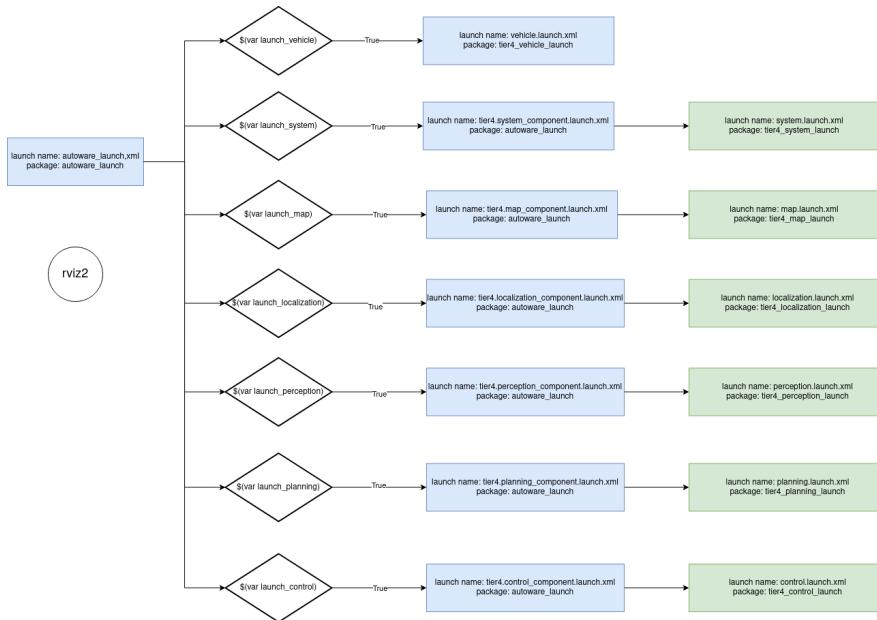
در شکل ۷-۴، شماتیکی از روند تبدیل داده‌ی pcap. ضبط شده توسط Ouster Studio، به عنوان ورودی به بسته pcap_to_pointcloud2 داده شده است و نتیجه آن در محیط RViz2، که محیطی برای مشاهده مباحثه‌ای ROS2 است، مشاهده می‌شود.

۱-۴-۴ دستکاری فایل‌های اجرایی Autoware

پس با موفقیت توانستیم این تبدیل داده را پیاده‌سازی کنیم. این ابر نقاط تبدیل شده، با فرکانس ۱۰ هرتز بر روی مبحث مورد استفاده توسط مولفه ادراک، انتشار می‌یابد. حال برای اینکه مولفه ادراک به درستی بتواند عملیات تشخیص را بر روی ابر نقاط ورودی انجام دهد، نیاز است تا فایل‌های اجرایی اش^۱ دستکاری شوند و برخی از فیلترهای آن غیرفعال شوند. پس لازم است تا معماری فایل‌های اجرایی Autoware را درک کنیم و آنها را دستکاری کنیم. فایل‌های اجرایی این ابزار از چند لایه تشکیل شده‌اند که آنها را گام

¹Launch Files

به گام شرح می‌دهیم. در ابتدا نگاهی به معماری فایل اجرایی اصلی Autoware می‌اندازیم. در این فایل اجرایی، همه‌ی مولفه‌های اصلی و مهم Autoware بصورت موازی اجرا می‌شوند. شکل ۸-۴ نشان‌دهنده معماری این فایل اجرایی است. هر کدام از مولفه‌ها با مثبت بودن متغیر متناظرشان، می‌توانند اجرا شوند. یعنی کاربر این قابلیت اجرا نکردن مولفه‌هایی که نیاز ندارد را دارد.



شکل ۸-۴ معماری اجرایی اصلی autoware_launch

از آنجایی که ما فقط به مولفه ادراک Autoware نیاز داریم، متغیرهای متناظر مولفه‌های دیگر را همانند شکل ۹-۴ منفی می‌گذاریم.

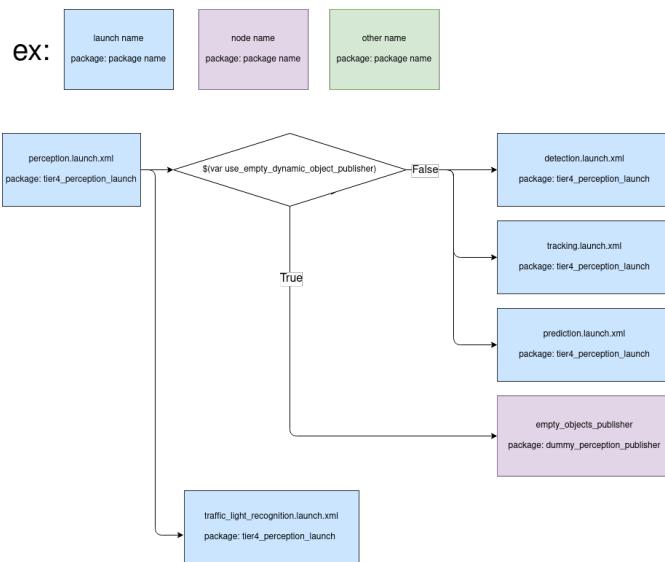
```

<!-- Optional parameters -->
<!-- Modules to be launched -->
<arg name="launch_vehicle" default="false" description="launch vehicle"/>
<arg name="launch_system" default="false" description="launch system"/>
<arg name="launch_map" default="false" description="launch map"/>
<arg name="launch_sensing" default="false" description="launch sensing"/>
<arg name="launch_sensing_driver" default="false" description="launch sensing driver"/>
<arg name="launch_localization" default="false" description="launch localization"/>
<arg name="launch_perception" default="true" description="launch perception"/>
<arg name="launch_planning" default="false" description="launch planning"/>
<arg name="launch_control" default="false" description="launch control"/>

```

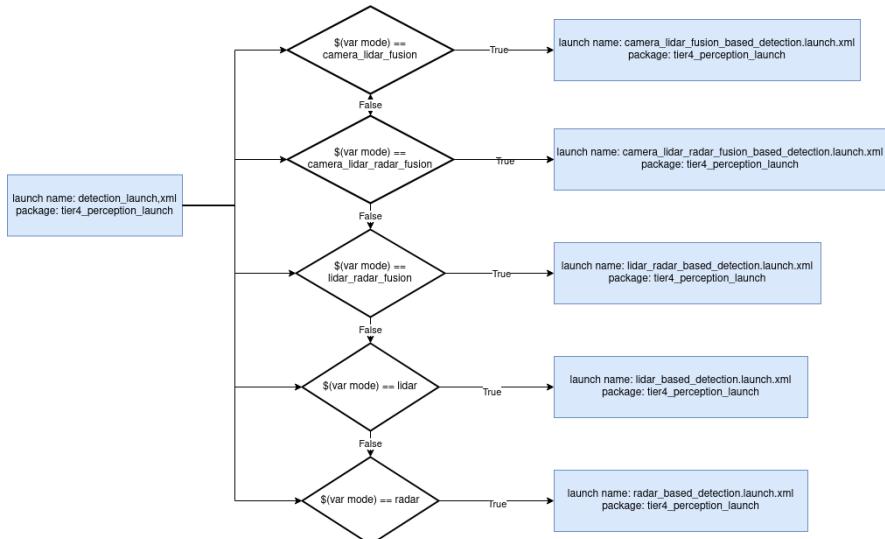
شکل ۹-۴ منفی گذاشتن تمامی مولفه‌ها بجز مولفه ادراک

گام بعدی، نگاه انداختن به فایل اجرایی مولفه ادراک است. در شکل ۱۰-۴ مشاهده می‌کنیم که این مولفه، از سه فایل اجرایی اصلی به نام‌های prediction.launch، tracking.launch و detection.launch استفاده می‌کند. کار اصلی ما با دو فایل اجرایی اول است. فایل اجرایی سوم به نقشه‌ی برداری از محل مورد نظر نیاز دارد، که در حال حاضر آن را نداریم، پس آن را غیرفعال می‌کنیم. دو فایل اجرایی دیگر نیز در فایل اجرایی مولفه ادراک هستند که در شکل کشیده نشده‌اند. این فایل‌های اجرایی مربوط به گره‌های پیش‌پردازشی بخش‌بندی اجسام و نقشه شبکه اشغال احتمالاتی هستند. از آنجایی که این لایه‌های پیش‌پردازش مفید هستند، آن‌ها را دستکاری نمی‌کنیم.



[۲۱] معماری فایل‌های اجرایی perception.launch

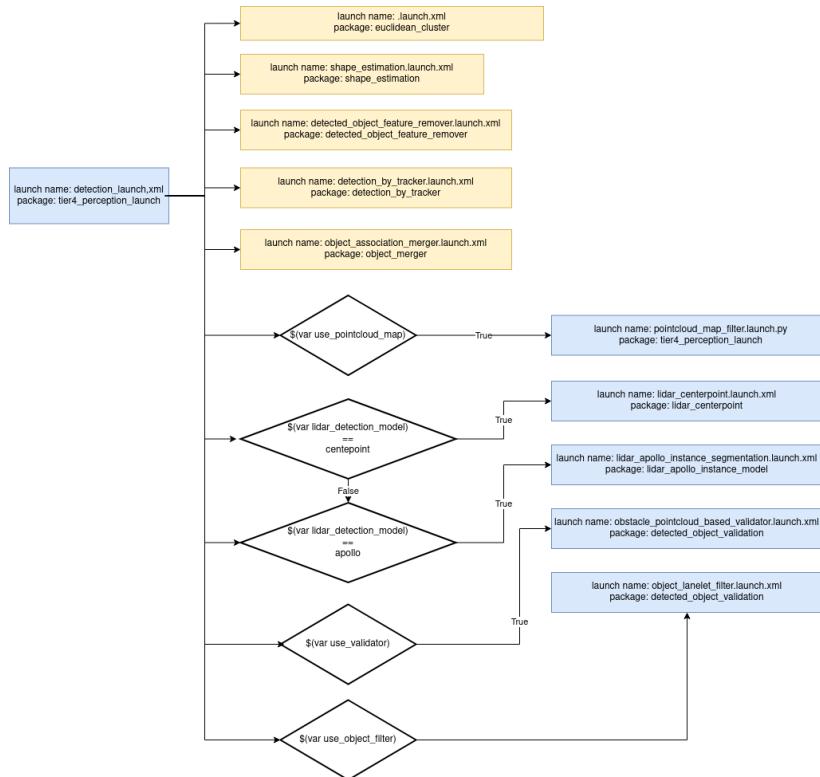
گام بعدی، نگاه کردن به معماری فایل اجرایی detection.launch است. در این فایل اجرایی، انواع فایل‌های اجرایی برای تشخیص‌دهنده‌های مختلف نوشته شده‌اند. همچنین مُد پیش‌فرض بر روی حالت گذاشته شده است و نیازی به اعمال تغییر نیست. شکل ۱۱-۴، معماری این فایل اجرایی را نشان می‌دهد.



شکل ۱۱-۴ معماری فایل اجرایی detection.launch

حال نگاهی به فایل اجرایی lidar_based_detection.launch می‌اندازیم. این فایل برای اجرای تشخیص‌دهنده‌ی مبتنی بر ابر نقاط است. شکل ۱۲-۴، معماری این فایل اجرایی را نشان می‌دهد. در این فایل اجرایی، به صورت پیش‌فرض، از مدل تشخیص سه‌بعدی CenterPoint استفاده می‌شود. همچنین به صورت پیش‌فرض، برخی از گره‌های فیلتر همانند گره فیلتر لینلت، فیلتر اعتبارسنجی موانع براساس ابر نقاط و فیلتر براساس نقشه ابر نقاط نیز اجرا می‌شوند. گره‌های دیگر همانند خوشبندی

اقلیدسی، تخمین شکل جسم، ادغام‌کننده اجسام و تشخیص براساس ردیاب نیز مشاهده می‌شوند. از بین همه این گره‌ها، دو گره فیلتر لینلت و فیلتر براساس نقشه ابرنقاط بایستی خاموش شوند، زیرا نقشه بُرداری و نقشه ابرنقاط نداریم.



شکل ۱۲-۴ معماری فایل اجرایی lidar_based_detection.launch

۲-۴-۴ نتیجه‌گیری

در بخش قبلی معماری فایل‌های اجرایی مولفه ادراک Autoware را مشاهده کردیم. تغییراتی که بایستی اعمال شوند عبارتند از:

۱. خاموش کردن بقیه مولفه‌ها بجز مولفه ادراک، در فایل autoware_launch
۲. خاموش کردن فایل اجرایی prediction.launch، در فایل اجرایی perception.launch
۳. خاموش کردن فایل‌های اجرایی مربوط به گره‌های فیلتر لینلت و فیلتر موانع براساس نقشه ابر نقاط، در فایل اجرایی lidar_based_detection.launch

برای اینکه خود فایل‌های اجرایی Autoware خراب نشوند، فایل‌های اجرایی جدا از سفارشی‌سازی شده نسبت به نیازمندی‌هایمان نوشته شده است، که با پیشوند *_digital_twin_ شروع می‌شوند. این فایل‌ها در ساختار پوشه‌ای فایل‌های اجرایی Autoware جایگذاری شده‌اند. حال وقت تست کردن تشخیص

دهنده سه‌بعدی رسیده است. شکل ۱۳-۴ دستورهای لازم برای اجرای تشخیص‌دهنده سفارشی‌سازی شده نسبت به نیازهایمان را نشان می‌دهد. لازم به ذکر است که در دو ترمینال پایین، ماتریس‌های

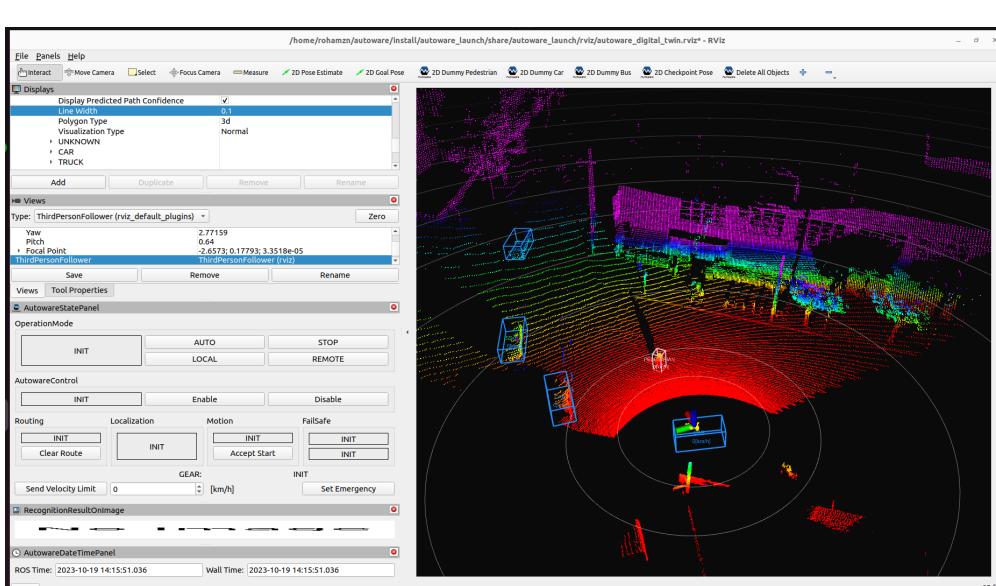
```

rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~/autoware
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~/autoware$ ros2 launch autoware_launch digital_twin_stm_vehicle_model:=sample_vehicle sensor_model:=awsim_sensor_kit map_path:=$HOME/rohamzn/autowar_e_map/installs/nlu0ku_noj[]
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~/autoware$ source install/setup.bash
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~/autoware$ ros2 launch autoware_launch digital_twin_stm_vehicle_model:=sample_vehicle sensor_model:=awsim_sensor_kit map_path:=$HOME/rohamzn/autowar_e_map/installs/nlu0ku_noj[]
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~/autoware$ cd ...
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~$ cd Digital-Twin-of-a-Traffic-Scene-Using-RSU-and-AWSIMSTM/
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~$ source install/setup.bash
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~$ Digital-Twin-of-a-Traffic-Scene-Using-RSU-and-AWSIMSTM/autoware_launch$ ros2 launch pcap_to_pointcloud2_publisher_py pcap_to_ros.launch.xml
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~$ ros2 launch tf2_ros static_transform_publisher 0 0 0 0 map viewer
(base) rohamzn@rohamzn-ROG-Strix-G533ZX-G533ZX: ~$ ros2 run tf2_ros static_transform_publisher 0 0 0 0 map base_link

```

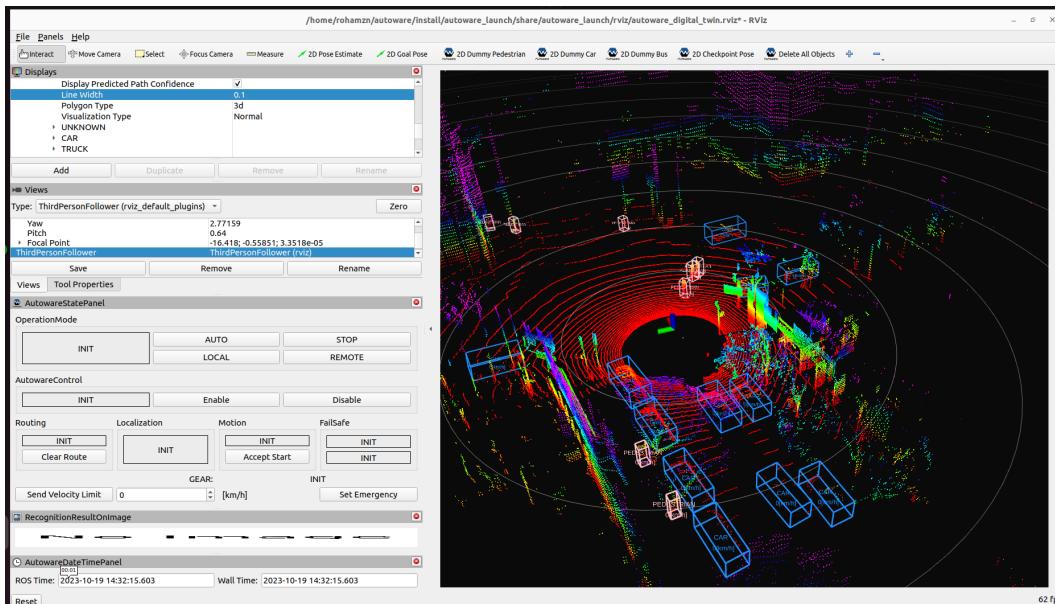
شکل ۱۳-۴ دستورهای اجرایی تشخیص‌دهنده سه‌بعدی

تبديل و دوران لایدار نسبت به سطح زمین و همچنین زمین نسبت به تماشاگر نیز بایستی به صورت دستی وارد شوند. دلیل اینکار به هم راستا نبودن لایدارهای نصب شده بر روی خودروی خودران، نسبت به محورهای مختصاتی ROS2 است. در ادامه به توضیح بیشتر این مشکل، می‌پردازیم. حال طبق شکل ۱۳-۴ دستورات را اجرا می‌کنیم و نتیجه زیر را مشاهده می‌کنیم:

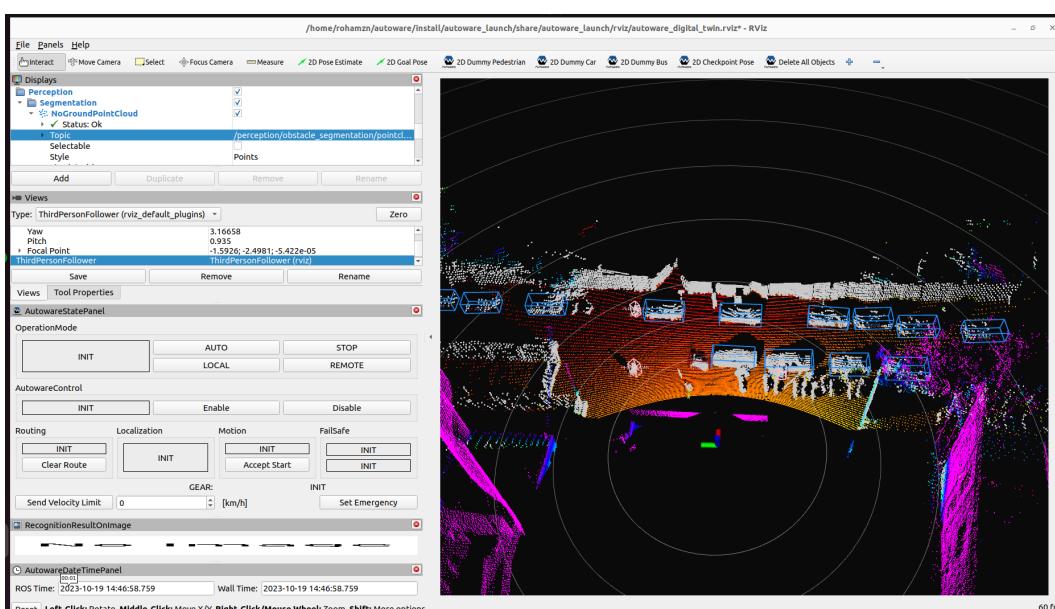


شکل ۱۴-۴ اجرایی از تشخیص اجسام سه‌بعدی توسط مدل CenterPoint در داده‌ی نمونه یک چهارراه

تشخیص دهنده استفاده شده، قابلیت تشخیص عابرین پیاده، دوچرخه سوار، موتور، خودرو، اتوبوس و کامیون را دارد. شکل ۱۵-۴ نیز نشان می‌دهد که تنها با انتشار داده نمونه دیگری که از شرکت Ouster گرفته شده است (یا هر ضبط لایدار با فرمت pcap)، می‌توان عملیات تشخیص بالادرنگ را بر روی آن اعمال کرد.



شکل ۱۵-۴ اجرایی از تشخیص اجسام سه بعدی توسط مدل CenterPoint در داده‌ی نمونه یک ماشین در حال رانندگی در سطح شهر



شکل ۱۶-۴ مشاهده خروجی بخش‌بندی مواعن، که نقاط زمین و نویزها را از ابر نقاط فیلتر می‌کند.

با توجه به شکل ۱۶-۳، می‌توان مشاهده کرد که هر کدام از گره‌ها بر روی چه مبحثی، خروجی‌های خود را انتشار می‌کنند. در محیط RViz2 می‌توان در این مباحث اشتراک گرفت و خروجی‌های آن را

مشاهده کرد. به طور مثال در شکل‌های فوق، اشتراک در مبحثی^۱ گرفته شده است که در آن مختصات، ابعاد کادر محصور کننده و برچسب اجسام فیلتر شده براساس گره اعتبارسنجی اجسام بر اساس ابر نقاط، منتشر می‌شوند. به طور مثال می‌توانیم در مبحث بخش‌بندی موانع^۲، که خروجی گره پیش‌پردازش مولفه ادراک است، اشتراکی بگیریم. شکل ۱۶-۴ نتیجه‌ی گرفتن اشتراک در این مبحث، با استفاده از نمایشگر RViz2 است.

۵-۴ پردازش ویژگی‌های اجسام تشخیص داده شده در شبیه‌ساز

در بخش قبلی با موفقیت توانستیم مدلی را پیدا کنیم و با استفاده از آن، تشخیص بلاذرنگ اجسام سه بعدی را در داده های نمونه لایدار Ouster انجام دهیم. طبق شکل ۳-۱۶، می توان مشاهده کرد که ورودی گرهی ردیاب همزمان چند جسم یا همان multi_object_tracker، خروجی گرهی ادغام کننده اجسام است. همچنین با مشاهده فایل های اجرایی و کد این ردیاب، نتیجه گرفته شد که خروجی این گره، بر روی میبختی ۳ منتشر می شود.

شکل ۱۷-۴ مشخصات مبحث خروجی گره ردیاب همزمان چند جسم

خروجی این مبحث برای این پژوهش بسیار ارزشمند است، زیرا تمام ویژگی‌های اجسام تشخیص داده شده اعم از سرعت، شتاب، جهت، شناسه یکتا و برچسب در آن موجود است. شکل ۱۷-۴ ۱۷ مشخصات این مبحث را نشان می‌دهد. مشاهده می‌شود که جنس پیام‌های انتشار یافته در این مبحث، از نوع autoware auto perception msgs/msg/TrackedObjects است.

¹/perception/object_recognition/detection/centerpoint/validation/objects

²/perception/obstacle segmentation/pointcloud

³/perception/object recognition/tracking/objects

```

1 #include "autoware_auto_perception_msgs/msg/TrackedObject.idl"
2 #include "std_msgs/msg/Header.idl"
3
4 module autoware_auto_perception_msgs {
5     module msg {
6         @verbatim (language="comment", text=
7             " This is the output of object tracking and the input to
8                 prediction.")
9         struct TrackedObjects {
10             std_msgs::msg::Header header;
11             sequence<autoware_auto_perception_msgs::msg::TrackedObject>
12                 objects;
13         };
14     };
15 }
```

شکل ۱۸-۴ کد مربوط به ساختار پیام خروجی ردیاب همزمان چند جسم

```

1 #include "autoware_auto_perception_msgs/msg/ObjectClassification.
2     idl"
3 #include "autoware_auto_perception_msgs/msg/Shape.idl"
4 #include "autoware_auto_perception_msgs/msg/TrackedObjectKinematics
5     .idl"
6 #include "unique_identifier_msgs/msg/UUID.idl"
7
8 module autoware_auto_perception_msgs {
9     module msg {
10         struct TrackedObject {
11             unique_identifier_msgs::msg::UUID object_id;
12
13             @range (min=0.0, max=1.0)
14             float existence_probability;
15
16             sequence<autoware_auto_perception_msgs::msg::
17                 ObjectClassification> classification;
18             autoware_auto_perception_msgs::msg::TrackedObjectKinematics
19                 kinematics;
20
21             autoware_auto_perception_msgs::msg::Shape shape;
22         };
23     };
24 }
```

شکل ۱۹-۴ کد مربوط به ساختار پیام جسم در حال ردیابی

شکل ۱۸-۴، نشان می‌دهد که این پیام در داخل خود آرایه‌ای از اجسام در حال ردیابی دارد که از جنس autoware_auto_perception_msgs/msg/TrackedObject هستند. شکل ۱۹-۴ نیز، ساختار پیام هر جسم در حال ردیابی را نشان می‌دهد. همانطور که مشاهده می‌کنیم، به هر جسم شناسه‌ای

یکتا^۱ داده می‌شود. همچنین درصد احتمال وجود^۲ آنها با استفاده از روش‌های مختلف محاسبه شده است. در آخر نیز سه داده‌ی مهم داریم که به ترتیب طبقه‌بندی^۳، سینماتیک^۴ و شکل جسم^۵ را در خود جای می‌دهند.

جفت پیام‌های بررسی شده، زیرمجموعه بسته پیام‌های autoware_auto_perception_msgs^۶ می‌باشد و با بهره‌بری از این بسته، می‌توان پیام‌های فوق را خواند و پردازش کرد. بخاطر داریم که در شبیه‌ساز AWSIM، از افزونه R2FU استفاده می‌شود و با استفاده از آن می‌توان در مباحث ROS2 اشتراک گرفت و یا به آنها داده منتشر کرد. در جدول ۲-۴، تمامی پیام‌هایی که توسط این افزونه به صورت پیش‌فرض پشتیبانی شده‌اند نوشته شده است. مشاهده می‌کنیم که پیام‌های بسته autoware_auto_perception_msgs توسط این افزونه پشتیبانی نمی‌شوند و در نتیجه نمی‌توان پیام‌های گرهی ردیاب هم‌زمان چند جسم را به خواند و پردازش کرد. ابزار AWSIM در مستندات خود، روشی را برای اضافه کردن بسته‌های پیام دلخواه

```

1 [SerializeField] string detectedObjectsTopic = "/perception/
  object_recognition/tracking/objects";
2 [SerializeField] string detectedObjectsPublisherTopic = "/unity/
  perception/object_recognition/tracking/objects";
3
4 ISubscription<autoware_auto_perception_msgs.msg.TrackedObjects>
  trackedObjectsSubscriber;
5 IPublisher<autoware_auto_perception_msgs.msg.TrackedObjects>
  trackedObjectsPublisher;
6
7 trackedObjectsPublisher = SimulatorROS2Node.CreatePublisher<
  autoware_auto_perception_msgs.msg.TrackedObjects>(
  detectedObjectsPublisherTopic, qos);
8 trackedObjectsSubscriber = SimulatorROS2Node.CreateSubscription<
  autoware_auto_perception_msgs.msg.TrackedObjects>(
  detectedObjectsTopic,
  msg => {trackedObjectsPublisher.Publish(msg);}, qos);
9

```

شکل ۲۰-۴ کد مربوط گرفتن اشتراک برای مبحث ردیاب و انتشار به یک مبحث جدید

به افزونه R2FU، مطرح کرده است.^۷ با پیش‌روی طبق دستورات مستندات AWSIM، پیام‌های بسته ادراک نیز به این افزونه اضافه شدند. حال موتور Unity می‌تواند با استفاده از کد شکل ۲۰-۴، در مباحث مورد نظر اشتراک بگیرد. طبق کد، اگر بتوانیم به درستی پیام‌های ردیابی را در بستر Unity بخوانیم، بایستی بتوانیم آن‌ها را به مبحثی جدید نیز منتشر کنیم. در شکل ۲۱-۴ مشاهده می‌کنیم که پیام‌ها در

¹Universally Unique Identifier: UUID

²Existence Probability

³Classification

⁴Kinematics

⁵Shape

⁶autoware_auto_perception_msgs Package

⁷<https://tier4.github.io/AWSIM/Components/ROS2/AddACustomROS2Message/>

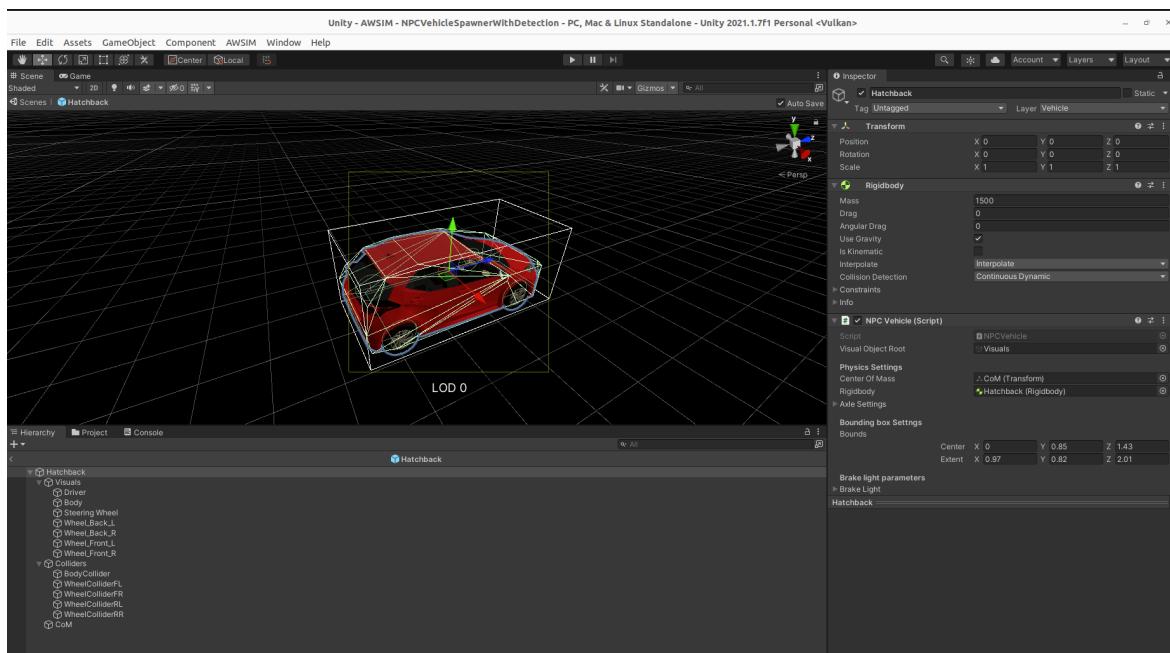
```
/unity/perception/object_recognition/tracking/objects
/vehicle/status/gear_status
/vehicle/status/velocity_status
(base) rohman@rohman-R0G-Strix-G533ZX-G533ZX:~/autoware$ ros2 topic info /unity/perception/object_recognition/tracking/objects -v
topic: /unity/perception/object_recognition/tracking/objects
Publisher count: 1
Node name: AWSIM
Node namespace: /
Topic type: autoware_auto_perception_msgs/msg/TrackedObjects
Endpoint type: PUBLISHER
QoS Policy: best_effort
Reliability: RELIABLE
History (Depth): KEEP_LAST (5)
Durability: VOLATILE
Lifespan: Infinite
Deadline: Infinite
Liveliness: AUTOMATIC
Liveliness lease duration: Infinite
Subscription count: 0
```

شکل ۲۱-۴ مبحث منتشر شده پیام‌های ردیاب، توسط افرونه R2FU در بستر Unity

حال انتشار شدن هستند و جنس آن‌ها نیز همان autoware_auto_perception_msgs/msg/TrackedObjects است.

۶-۴ شبیه‌سازی ترافیک در شبیه‌ساز

پس با موفقیت توانستیم پیام‌های گره ردیاب هم‌زمان چند جسم را با استفاده از Unity بخوانیم. حال بایستی بتوانیم با پردازش این اطلاعات، ماشین‌هایی را در یک صحنه جدید در پروژه AWSIM، پدیدار کنیم و سپس آن‌ها را حرکت دهیم.

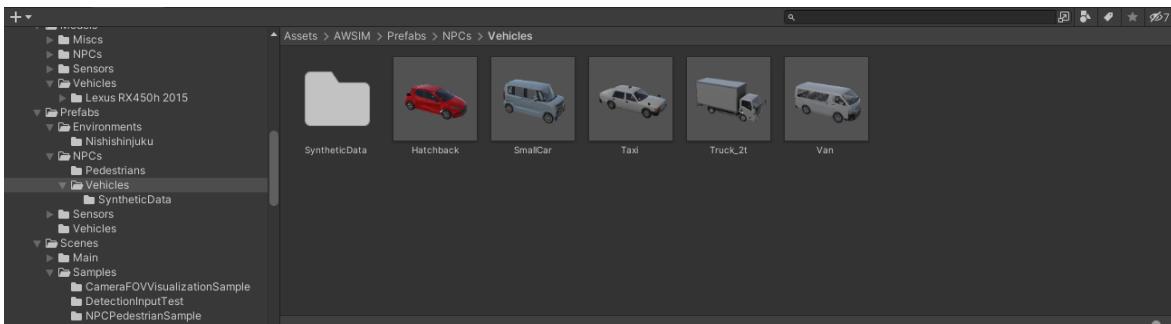


شکل ۲۲-۴ یک خودروی نمونه پیش‌ساخته در پروژه AWSIM

بخاطر داریم که در این شبیه‌ساز، مولفه‌ای به نام مولفه شبیه‌ساز ترافیک^۱ وجود دارد. در این مولفه، ماشین‌هایی با روش‌های سنتز ریاضی پدیدار می‌شوند که نام آن‌ها NPCVehicle است. این ماشین‌ها

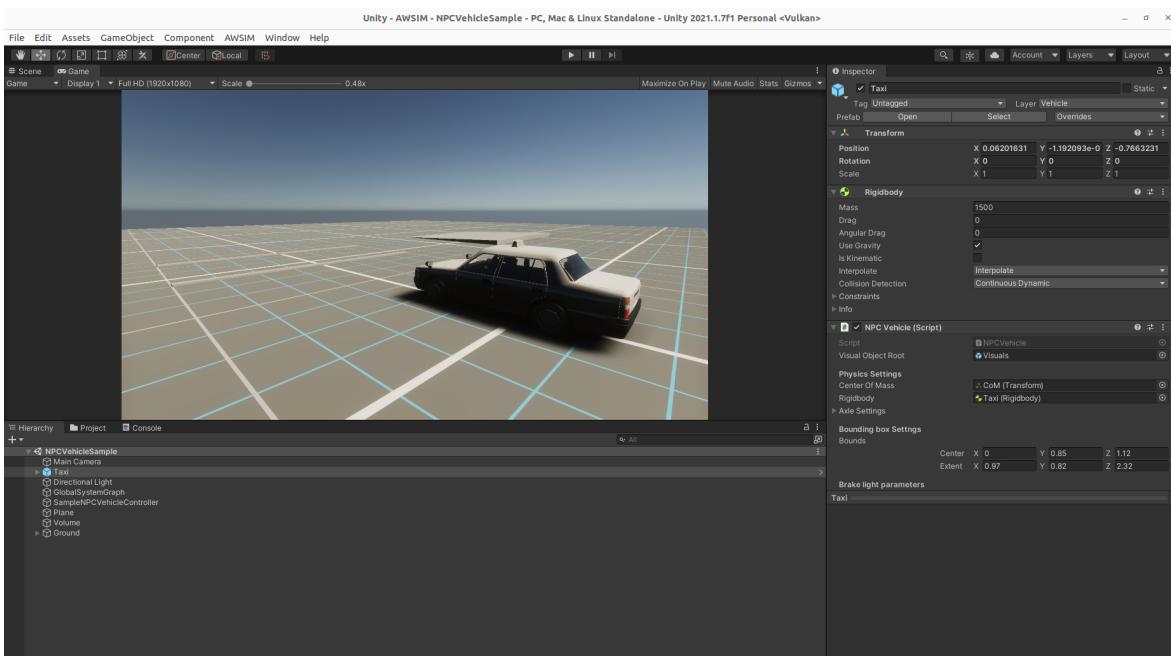
¹Traffic Simulator

جزء اشیاء بازی^۱ این شبیه‌ساز هستند و به همه آن‌ها به صورت پیش‌فرض، اسکریپتی مجهز می‌شود که وظیفه پیاده‌سازی رفتارها و کنترل‌های این ماشین‌ها را بر عهده دارد. در شکل ۴-۲۲، یکی از ماشین‌های پیش‌ساخته شده شبیه‌ساز AWSIM را مشاهده می‌کنیم. ملاحظه می‌شود که در بخش اسکریپت‌ها، اسکریپتی به نام NPCVehicle.cs اضافه شده است.



شکل ۴-۲۳ شبیه‌ساز AWSIM، چندین مدل خودرو برای ایجاد تنوع دارد.

شبیه‌ساز AWSIM، در ساختار پوشه‌ای خود یک صحنه نمونه^۲ دارد که نام آن NPCVehicalSample است. شکل ۴-۲۴ این صحنه را نشان می‌دهد. همانطور که مشاهده می‌کنیم، یک خودروی تاکسی با



شکل ۴-۲۴ صحنۀ نمونه یک خودرو از جنس NPCVehicle

اسکریپت NPCVehicle در صحنه وجود دارد. با اجرای این صحنه نمونه، خودروی تاکسی شروع به حرکت می‌کند و تمام حالات حرکتی آن بررسی می‌شود. این دستورات اعمال شده به خودروی تاکسی، توسط اسکریپتی دیگر به نام SampleNPCVehicleController داده می‌شوند.

¹Unity Game Object

²Sample Scene

```

1 IEnumator Routine()
2 {
3     Debug.Log(" --- Start control NPCVehicle --- ");
4     Debug.Log("Straight");
5     yield return UpdatePosAndRot(5f, 3f, 0f);
6     Debug.Log("Turn right");
7     yield return UpdatePosAndRot(13.4f, 3f, 20f);
8     Debug.Log("Straight");
9     yield return UpdatePosAndRot(2f, 3f, 0f);
10    Debug.Log("Left turn signal");
11
12    npcVehicle.SetTurnSignalState(NPCVehicle.TurnSignalState.LEFT);
13    yield return new WaitForSeconds(2f);
14    Debug.Log("Right turn signal");
15
16    npcVehicle.SetTurnSignalState(NPCVehicle.TurnSignalState.RIGHT)
17    ;
18    yield return new WaitForSeconds(2f);
19    Debug.Log("Hazard signal");
20
21    npcVehicle.SetTurnSignalState(NPCVehicle.TurnSignalState.HAZARD
22    );
23    Debug.Log("Back Right");
24    yield return UpdatePosAndRot(3f, -2f, -30f);
25    Debug.Log("Back");
26    yield return UpdatePosAndRot(2f, -2f, 0f);
27    Debug.Log(" --- End control NPC Vehicle --- ");
28}

```

شکل ۴-۲۵ بخشی از کد NPCVehicleController که به خودروهای NPCVehicle دستورهای حرکتی متنوع می‌دهد.

شکل ۴-۲۵، کد مربوط به این حرکات خودروی تاکسی را نشان می‌دهد. مشاهده می‌کنیم که این دستورات توسط کوروتین‌ها^۱ اجرا می‌شوند که باعث اجرای موازی دستورات می‌شود، تا باعث انسداد در خط لوله اجرایی موتور Unity نشود. حال به بررسی تابع UpdatePosAndRot می‌پردازیم. شکل ۴-۲۶ کد را نشان می‌دهد. مشاهده می‌کنیم که برای حرکت خودرو، مدت زمان حرکت، سرعت خطی، و سرعت زاویه‌ای به عنوان ورودی گرفته می‌شود. سپس با اعمال فرمول‌های ساده سینماتیک، مختصات جدید خودرو را در هر واحد زمانی از شبیه‌ساز، محاسبه می‌کند. توابع SetPosition و SetRotation از دستورات اسکریپت NPCVehicle هستند که با توجه به فیزیک خودرو، آن را به زاویه و مختصات خواسته شده حرکت می‌دهد.

¹ Coroutine

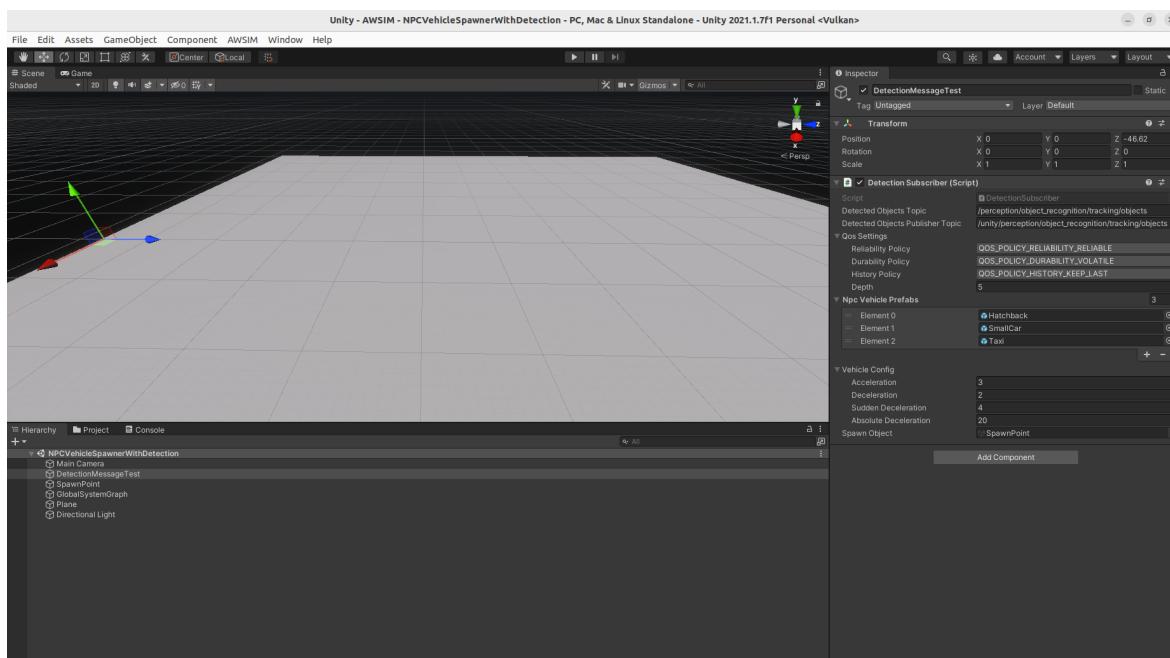
```

1 IEnumerator UpdatePosAndRot(float duration, float speed, float
2   yawSpeed, bool validatePose = true)
3 {
4   var startTime = Time.fixedTime;
5   yield return new WaitForFixedUpdate();
6   while (Time.fixedTime - startTime < duration)
7   {
8     var euler = currentRotation.eulerAngles;
9     currentRotation = Quaternion.Euler(euler.x, euler.y +
10    yawSpeed * Time.fixedDeltaTime, euler.z);
11    currentPosition += currentRotation * Vector3.forward *
12      speed * Time.fixedDeltaTime;
13    npcVehicle.SetRotation(currentRotation);
14    npcVehicle.SetPosition(currentPosition);
15    yield return new WaitForFixedUpdate();
16  }
17}

```

شکل ۲۶-۴ بخشی از کد NPCVehicleController که در آن نحوه شبیه‌سازی حرکت خودروها نمایش داده شده است.

۱-۶-۴ ایجاد صحنه جدید



شکل ۲۷-۴ صحنه جدید ساخته شده برای شبیه‌سازی ترافیک

قبل از آنکه بتوانیم از اسکریپت حرکت دادن خودروهای NPCVehicle استفاده کنیم، بایستی یک صحنه جدید بسازیم. سپس در این صحنه یک زمین صاف قرار دهیم و شئ بازی‌ای اضافه کنیم که مسئولیت اشتراک‌گیری از مبحث پیام ردیاب را دارد، سپس با توجه به درصد احتمال وجود جسم و

برچسب آن‌ها، خودروها را در صحنه شبیه‌سازی پدیدار کند. شکل ۴-۳۰، این صحنه جدید را نشان می‌دهد. ملاحظه می‌شود که داخل سلسه مراتب^۱ این صحنه، شئ بازی‌ای به نام DetectionMessageTest ساخته شده است. به این شئ، اسکریپتی متصل شده است که اشتراکی در مبحث اجسام رادیابی شده دارد.

```

1 trackedObjectsSubscriber
2 = SimulatorROS2Node.CreateSubscription<
3   autoware_auto_perception_msgs.msg.TrackedObjects>(
4   detectedObjectsTopic, msg =>
5   {
6     trackedObjectsPublisher.Publish(msg);
7     currentFrameVehicles = new List<String>();
8     foreach (autoware_auto_perception_msgs.msg.TrackedObject obj in
9       msg.Objects){
10       if (obj.Classification[0].Label == car && obj.
11           Existence_probability >= 0.3){
12         string uuid = BitConverter.ToString(obj.Object_id.Uuid)
13             .Replace("-", " ");
14         currentFrameVehicles.Add(uuid);
15         if (detectedVehicles.ContainsKey(uuid)){
16           detectedVehicles[uuid] = obj;
17           continue;
18         }
19         Debug.Log("New Object ID: " + uuid);
20       }
21     }
22   },
23   qos);

```

شکل ۴-۲۸ بخشی از کد DetectionSubscriber

در شکل ۴-۲۸، بخشی از کد DetectionSubscriber را مشاهده می‌کنیم، که وظیفه فیلتر کردن اجسام رادیابی شده را دارد. این فیلتر، براساس درصد احتمال وجود تخمین زده اجسام کار می‌کند که توسط رادیاب Autoware، محاسبه شده است. حد آستانه این فیلتر، حد ۳۰٪ در نظر گرفته شده است. شناسه خودروهایی که از فیلتر عبور کنند، به آرایه‌ای به نام detectedVehicles اضافه می‌شوند. سپس کوروتینی به این جسم تعلق می‌گیرد که اقدام به پدیدار کردن خودروی رادیابی شده و حرکت دادن آن می‌کند. برای این کار، مختصات تشخیص داده شده از جسم، به همراه جهت آن را از پیام رادیابی مربوط به آن جسم استخراج می‌کنیم و با تابعی به نام npcVehicleSpawner آن را نسبت به شئ بازی‌ای به نام Spawn Point برای حرکت دادن ماشین‌های حاضر در صحنه، در کوروتین متعلق به جسم، شروع به دریافت

^۱Hierarchy

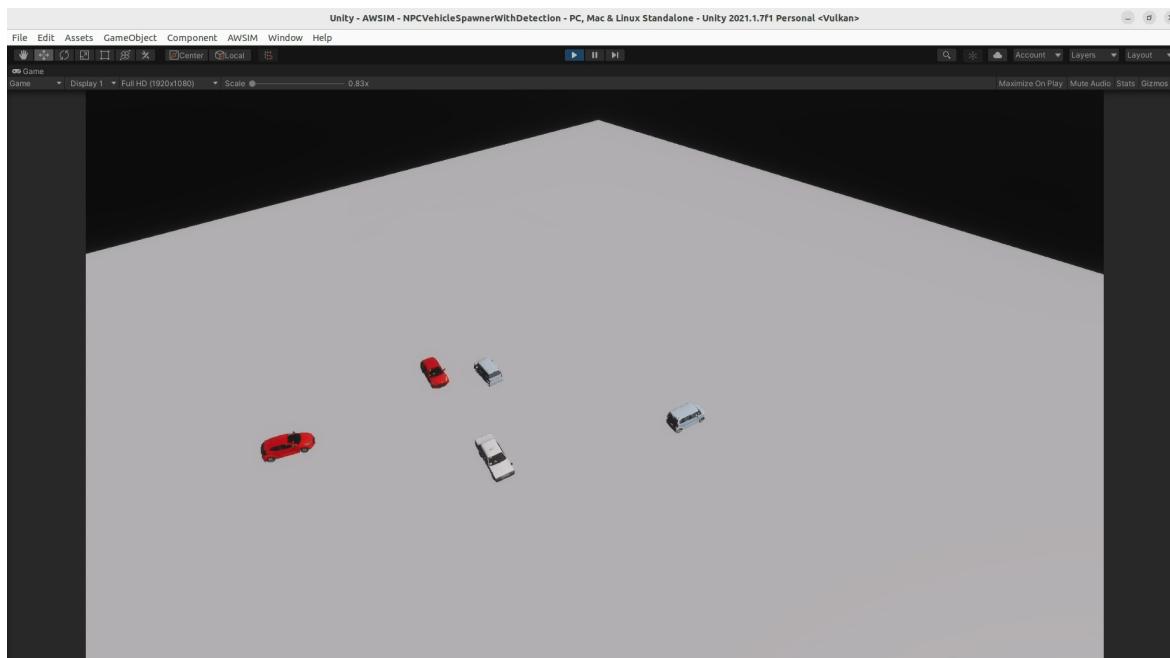
```

1 IEnumarator Routine(string vehicleID)
2 {
3     Vector3 spawnPosition = new Vector3((float)-detectedVehicles[vehicleID].Kinematics.Pose_with_covariance.Pose.Position.Y,
4                                         0f, (float)detectedVehicles[vehicleID].Kinematics.Pose_with_covariance.Pose.Position.X);
5     Quaternion rotation = new Quaternion(0f, (float)-
6                                         detectedVehicles[vehicleID].Kinematics.Pose_with_covariance.Pose.Orientation.Z, 0f, (float)detectedVehicles[vehicleID].Kinematics.Pose_with_covariance.Pose.Orientation.W);
7     NPCVehicle vehicle = npcVehicleSpawner.Spawn(npcVehicleSpawner.
8         GetRandomPrefab(), SpawnIdGenerator.Generate(),
9         spawnPosition, rotation, spawnObject.transform);
10    npcVehicles.Add(vehicleID, vehicle);
11    // ...
12 }

```

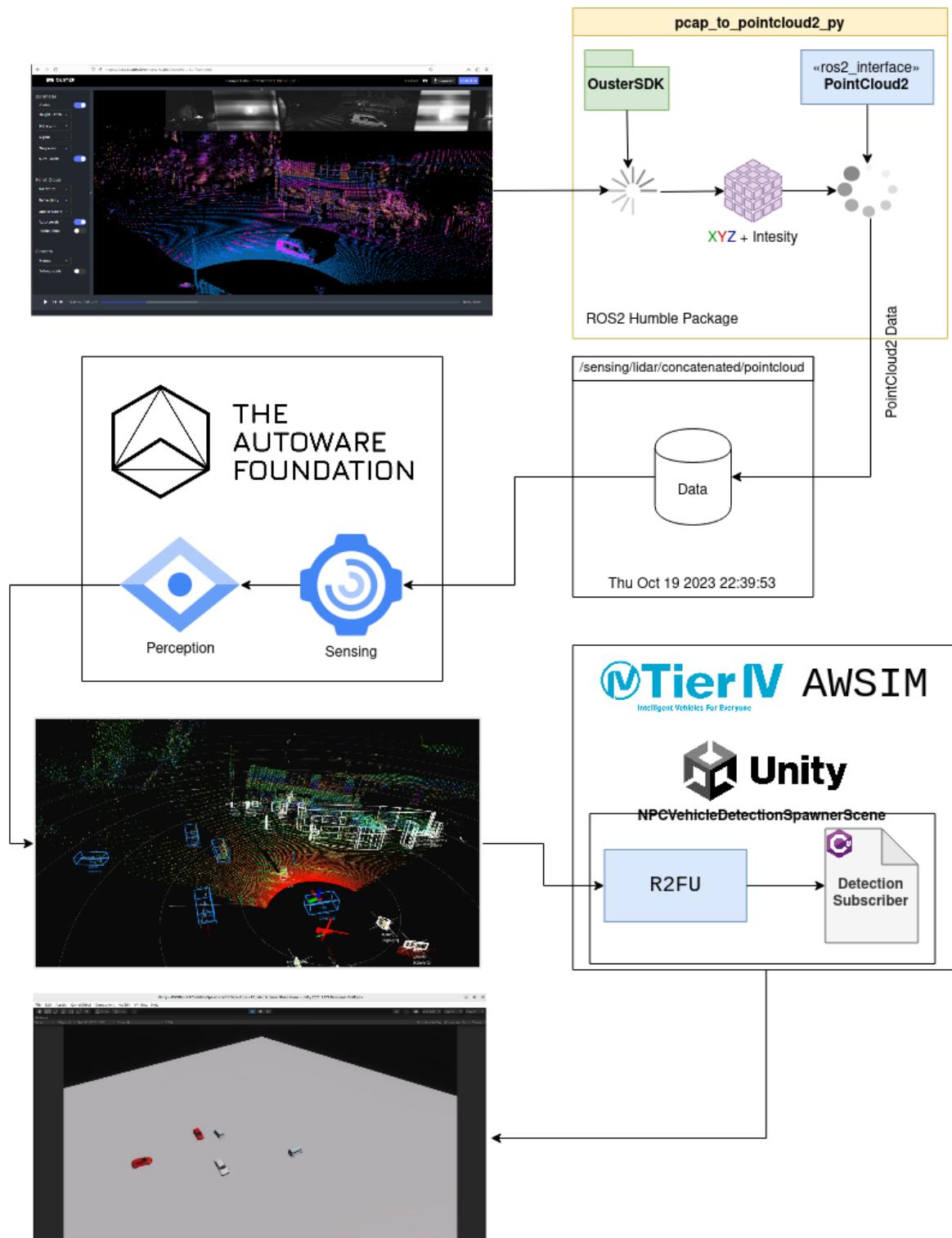
شکل ۲۹-۴ بخشی از کوروتین کد DetectionSubscriber

مختصات و جهت‌های خودرو می‌کنیم تا زمانی که دیگر خودرویی با این شناسه، در آرایه خودروهای تشخیص داده شده جدید وجود نداشته باشد. سپس این جهت‌ها و مختصات جدید را به توابع حرکتی (که در شکل ۲۶-۴ مشاهده کردیم) می‌دهیم تا خودروی NPCVehicle، به سمت مختصات جدید حرکت کند.



شکل ۳۰-۴ شبیه‌سازی موفق از ترافیک مشاهده شده در داده نمونه

همانطور که از شکل ۳۰-۴ معلوم است، مشاهده می‌کنیم که خودروها به درستی پدیدار شده‌اند و حرکت می‌کنند! در شکل ۳۱-۴ نیز نگاهی به پروسه طی شده در این سناریو می‌اندازیم.



شکل ۳۱-۴ پروسه شبیه‌سازی بلاذرنگ سناریو اول با داده نمونه

۲-۶-۴ نتیجه‌گیری سناریو اول

مشاهده کردیم که در این پژوهش، به شبیه‌سازی بلاذرنگ خودروها با استفاده از داده نمونه گرفته شده از Ouster، رسیدیم. پس می‌توان با قاطعیت گفت که سناریو دوم امکان پذیر است. البته لازم به ذکر

است که از آنجایی که تشخیص‌دهنده دقت ۱۰۰٪ ندارد، شبیه‌سازی با عیب و نقص‌هایی روبرو خواهد شد، اما نتیجه گرفته شده امیدمان را نسبت به پیاده‌سازی دوقلویی دیجیتال با دقت بالای ۹۰٪، افزایش می‌دهد.

۷-۴ سناریو دوم: داده‌برداری از منطقه خیابان رشت و طراحی سه‌بعدی

با اطمینان از اینکه شبیه‌ساز با هر داده ابر نقطه‌ای کار می‌کند، عملیات گرفتن مجوز از دانشگاه و داده‌برداری از خیابان رشت، آغاز شد.

۱-۷-۴ داده‌برداری از خیابان رشت

خیابان رشت، خیابانی است که در ضلع شمالی دانشگاه امیرکبیر واقع شده است و حجم زیادی تردد را همیشه به همراه خود دارد. شکل ۳۲-۴، محوطه دانشگاه امیرکبیر را نشان می‌دهد.



شکل ۳۲-۴ نقشه پردیس دانشگاه امیرکبیر

در این داده‌برداری، از سنسور لایدار OS1:64 استفاده می‌شود و دوربین فیلم‌برداری نیز دوربین موبایل است. محلی که سنسورها مستقر شدند، پشت بام خیابان رشت بود. شکل ۳۴-۴، منطقه پیشنهادی را نشان می‌دهد.



شکل ۳۲-۴ تصاویر خیابان رشت از زاویه دید درب رشت



شکل ۳۴-۴ محل پیشنهادی دادهبرداری

برای عملیات داده برداری، از یک سیم رابط ۱۰ متری، سه پایه برای لایدار، لپتاپ برای ضبط داده، لایدار، دوربین موبایل و یک نردنگان استفاده شد. برای ضبط داده نیز از نرم افزار Ouster Studio استفاده شد.



شکل ۳۵-۴ تصاویری از عملیات داده‌برداری



شکل ۳۶-۴ تصویری از خیابان رشت و ابر نقاط متناظر آن

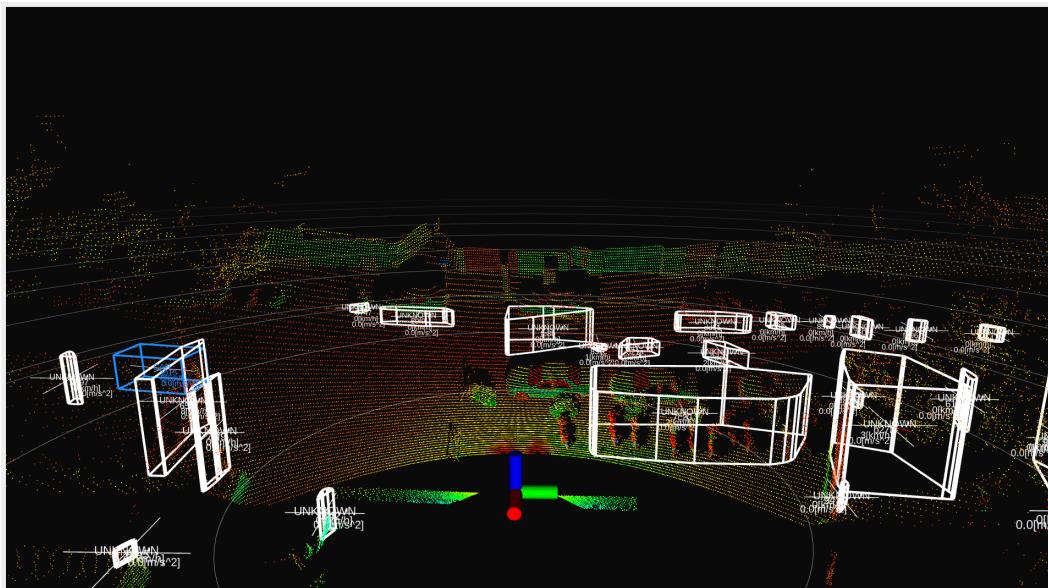


شکل ۳۷-۴ تصویری از لایدار OS1:64 که بر روی سهپایه نصب شده است.

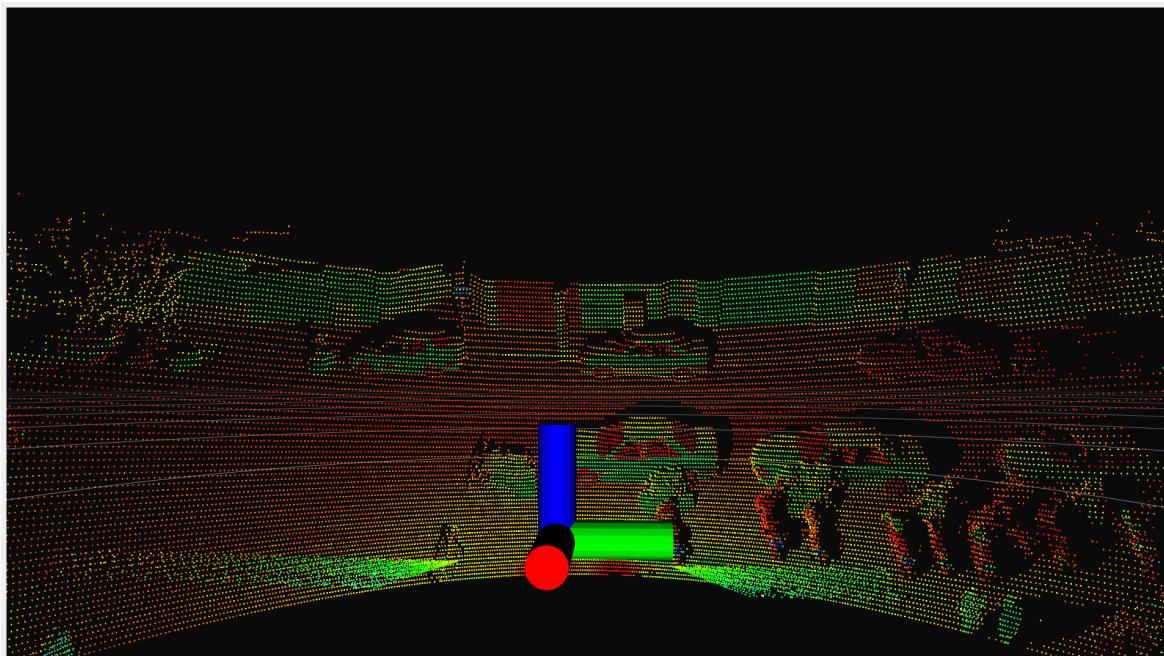
این داده برداری، از ساعت ۹:۳۰ تا ساعت ۱۰:۴۵ ادامه داشت و حسگر لایدار به مدت نیم ساعت داده جمع‌آوری کرد. حجم فایل pcap آن ۱۱.۷ گیگابایت شد. همانطور که طبق شکل ۳۷-۴ مشاهده می‌کنیم، لایدار نصب شده بر روی سهپایه، برای داشتن دید بهتر به خیابان، با سطح افق زاویه دارد.

۲-۷-۴ اعمال مدل هوش مصنوعی به ورودی لایدار

در بخش قبل، توانستیم با موفقیت یک ضبط لایدار نیم ساعته از خیابان رشت ضبط کنیم. این ضبط در فرمت “*.pcap” ذخیره شده است، پس می‌توانیم آن را به خورد تبدیل کننده pcap_to_pointcloud2_py بدهیم. همچنین می‌توانیم این داده تبدیل شده به PointCloud2 را برای مبحث ورودی مولفه ادراک



شکل ۳۸-۴ اقدام ناموفق تشخیص بر روی ابر نقاط خیابان رشت



شکل ۳۹-۴ نمای نزدیک از کالیبره نبودن ابر نقاط خیابان رشت.

Autoware، منتشر کنیم و عملیات تشخیص بلاذرنگ بر روی آن انجام شود. شکل ۳۸-۴ نشان می‌دهد که تشخیص بر روی ابر نقاط رشت، به درستی کار نمی‌کند. دلیل این اتفاق با توجه کردن به همین شکل و شکل ۳۷-۴ قابل فهم است. در بخش قبلی ذکر کردیم که لایدار، با سطح افقی، زاویه دارد. پس ابر نقاطی که ضبط می‌کند نیز نسبت به افق زاویه خواهد داشت. به عبارت دیگر، ابر نقاط این لایدار کالیبره^۱ نشده‌اند. شکل ۳۹-۴، این مشکل را به وضوح نمایان می‌کند. برای حل این مشکل، نیاز است تا زاویه لایدار را نسبت به سطح افق بدست بیاوریم. از آنجایی که سه‌پایه استفاده شده در داده‌برداری، دست نخورده بود؛ توانستیم زاویه لایدار را با افق بدست بیاوریم. زاویه لایدار با سطح افق، مقدار ۳۱.۵۱ درجه یا ۱.۵۵ رادیان در محور Y است. همچنین با کمی دقیق در شکل ۳۸-۴، متوجه می‌شویم که محور مختصات لایدار برعکس است، یعنی پشت لایدار به سمت خیابان رشت بوده است. پس لازم است تا محور مختصات XY را ۱۸۰ درجه حول محور Z بچرخانیم. در نهایت نیز به نتیجه رسیده شد که ارتفاع لایدار نسبت به سطح زمین، ۶ متر است. پس با استی محور مختصات لایدار را ۶ متر به سمت بالا حرکت دهیم تا سطح زمین خیابان رشت، هم سطح نقطه (0, 0, 0) یا همان محور مختصات دنیای ROS2 شود. فرمول ماتریس دوران^۲ در فضای سه‌بعدی به شکل زیر است:

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi s_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi + c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & s_\theta c_\phi \end{bmatrix} \quad (1-4)$$

¹Calibration²Rotation Matrix

و بردار انتقال به شکل زیر است:

$$\mathbf{T} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (2-4)$$

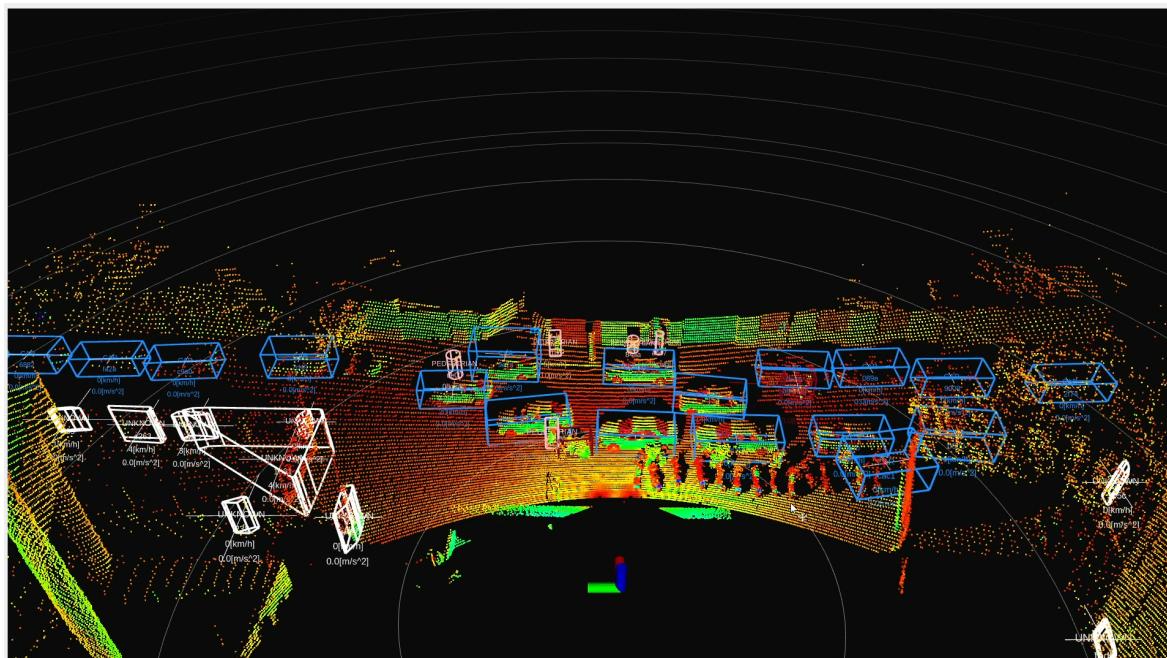
با عدد گذاری در روابط فوق، خواهیم داشت:

$$\mathbf{R} = \begin{bmatrix} 0.8525245 & 0 & -0.5226873 \\ 0 & 1 & 0 \\ 0.5226873 & 0 & 0.8525245 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}$$

و به ازای هر نقطه P داخل ابرنقطه خواهیم داشت:

$$\mathbf{P}_{\text{new}} = \mathbf{RP} + \mathbf{T} \quad (3-4)$$

بهترین جا برای اعمال این تبدیل‌ها، در داخل کد pcap_to_pointcloud_py است، زیرا در هر فریم از ضبط لایدار، مختصات (X, Y, Z) نقطه‌ها بدست می‌آید. پس می‌توانیم ماتریس تبدیل و دوران را بر روی این مختصات‌ها اعمال کنیم. حال، باری دیگر ابر نقاط خیابان رشت را به خورد تشخیص‌دهنده می‌دهیم، اما با این تفاوت که این دفعه ابر نقاط کالیبره شده‌اند.



شکل ۴۰-۴ تشخیص موفق اجسام با ورودی ابر نقاط کالیبره شده

همانطور که از شکل ۴۰-۴ نمایان است، تشخیص اجسام با موفقیت انجام شده است. حال تصویری

از دوربین را در مقابل ابر نقاط متناظر آن قرار می‌دهیم:

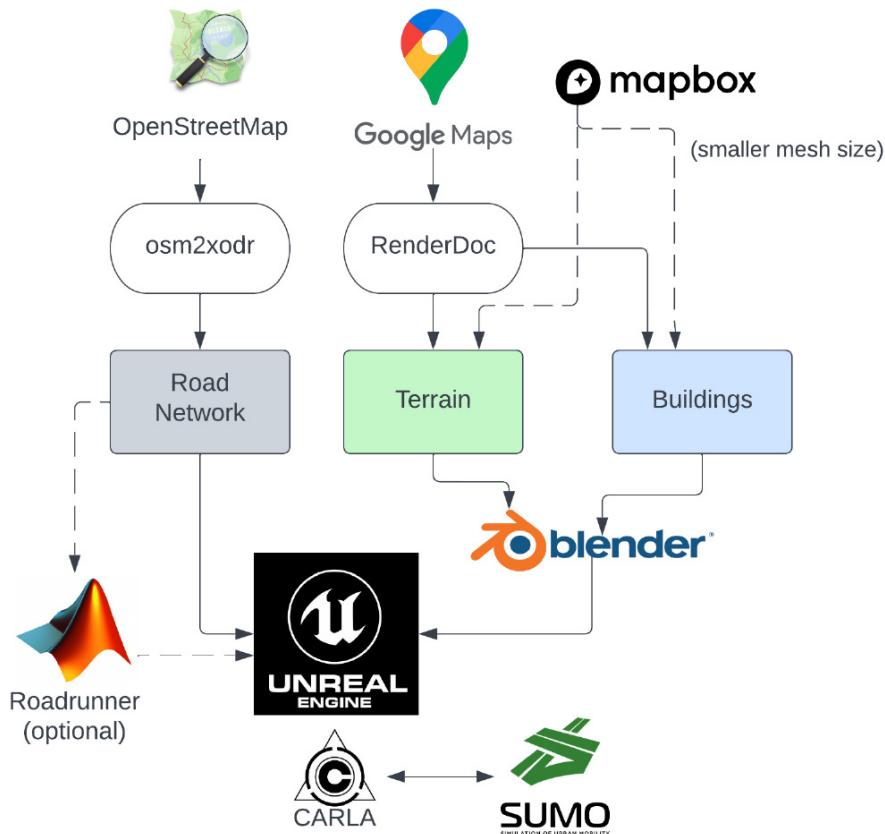


شکل ۴-۴۱ تصویری از خیابان رشت و ابر نقاط پردازش شدهی متناظر آن

۸-۴ طراحی مدل سه بعدی منطقه دانشگاه صنعتی امیرکبیر

آخرین مرحله باقی مانده، طراحی فضای پردیس امیرکبیر است. در فصل دوم، مقاله‌ای را معرفی کردیم که در سال ۲۰۲۲ نوشته شده است و در مورد روش‌های بهینه مدل‌سازی سه‌بعدی دوقلوهای دیجیتال

صحبت کرده است [۱۰]. باری دیگر شکل روند طی شده برای مدل‌سازی سه‌بعدی این مقاله را مرور می‌کنیم:



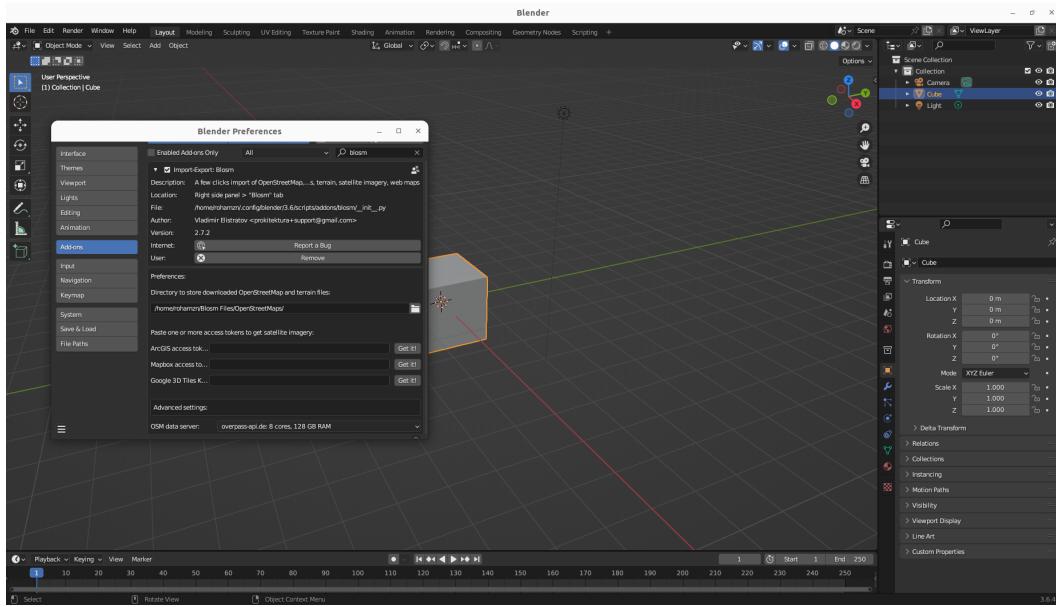
شکل ۴-۴ روش پیشنهاد شده برای ایجاد دوکلوی دیجیتال یک منطقه [۱۰]

این مقاله با استفاده از داده‌های ابزار OpenStreetMap، که در آن خیابان‌ها، ساختمان‌ها و انواع داده‌های دیگر مرتبط با نقشه بُرداری را در خود دارد. از فصل سوم نیز به یاد داریم که افزونه‌ای در ابزار طراحی مدل‌های سه‌بعدی Blender وجود دارد، که Blosm نام دارد. این افزونه با استفاده از داده‌های "osm" گرفته شده از ابزار OpenStreetMap، مدل سه‌بعدی یک منطقه را تنها با چندین کلیک دکمه انجام می‌دهد! این افزونه، از ابزاری به نام MapBox^۱ استفاده می‌کند که تمامی اطلاعات مهم از یک نقشه را دارد. این ابزار، در بسیاری از برنامه‌های مسیریاب، نقشه سه‌بعدی و ناوبری استفاده می‌شود. برای اینکه افزونه Blosm بتواند اطلاعات نقشه سه‌بعدی یک منطقه را با استفاده از واسط برنامه‌نویسی کاربری^۲ MapBox Streets بدست بیاورد، نیاز به یک کلید API است. برای دستیابی به این کلید، نیاز است که در تارنمای این ابزار ثبت‌نام کرد. اما برای آنکه اثبات شود که کاربرهای در حال ثبت‌نام، ربات نیستند؛ این ابزار نیاز به مشخصات کارت اعتباری^۳ بین‌المللی دارد. طبق شکل ۴-۴، مشاهده می‌کنیم که این افزونه یک کلید API از ابزار MapBox می‌خواهد. با پیش‌روی طبق مقاله ذکر شده، منطقه

¹ <https://www.mapbox.com/>

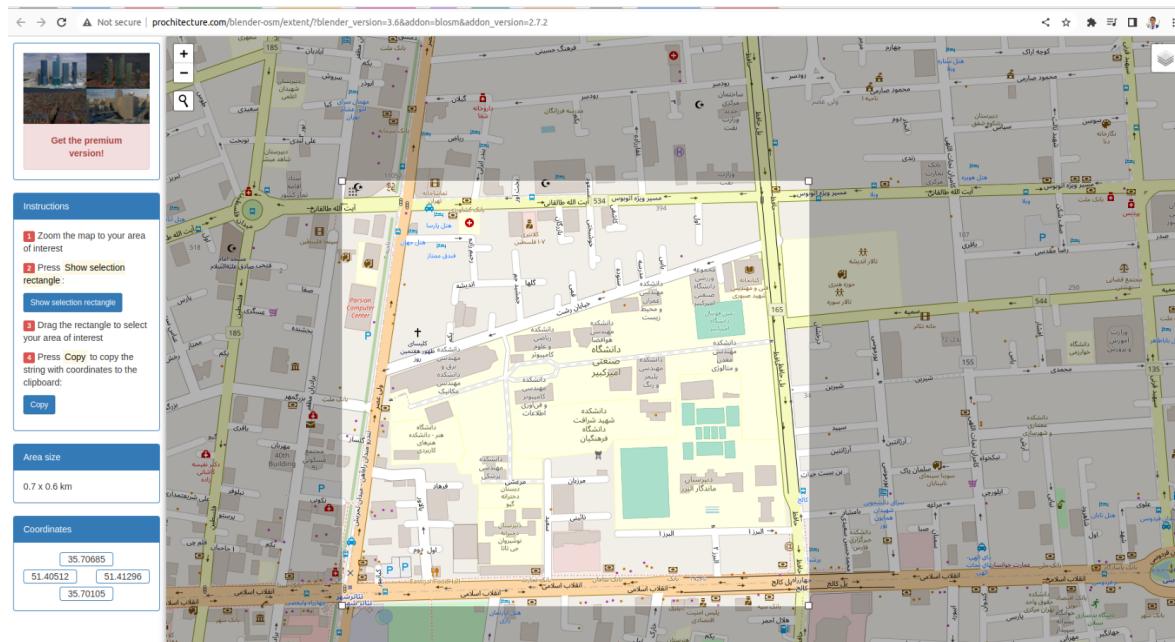
² Application Programming Interface (API)

³ Credit Card



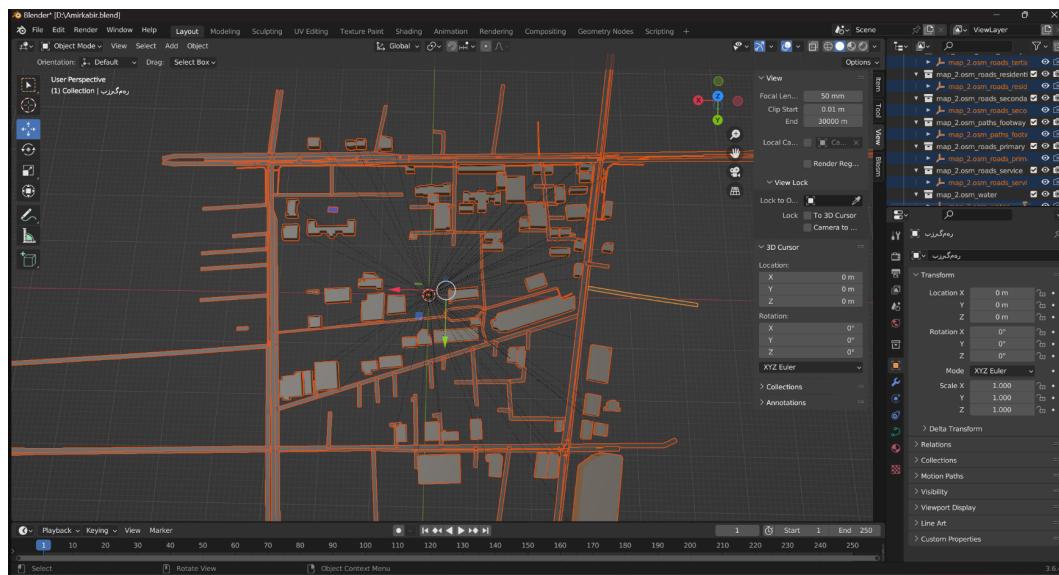
شکل ۴۳-۴ نمایی از افزونه Blosm در ابزار مدل‌سازی سه‌بعدی Blender

امیرکبیر را در این افزونه انتخاب می‌کنیم تا از آن یک مدل سه‌بعدی بسازد. پس از انتخاب محل، دکمه



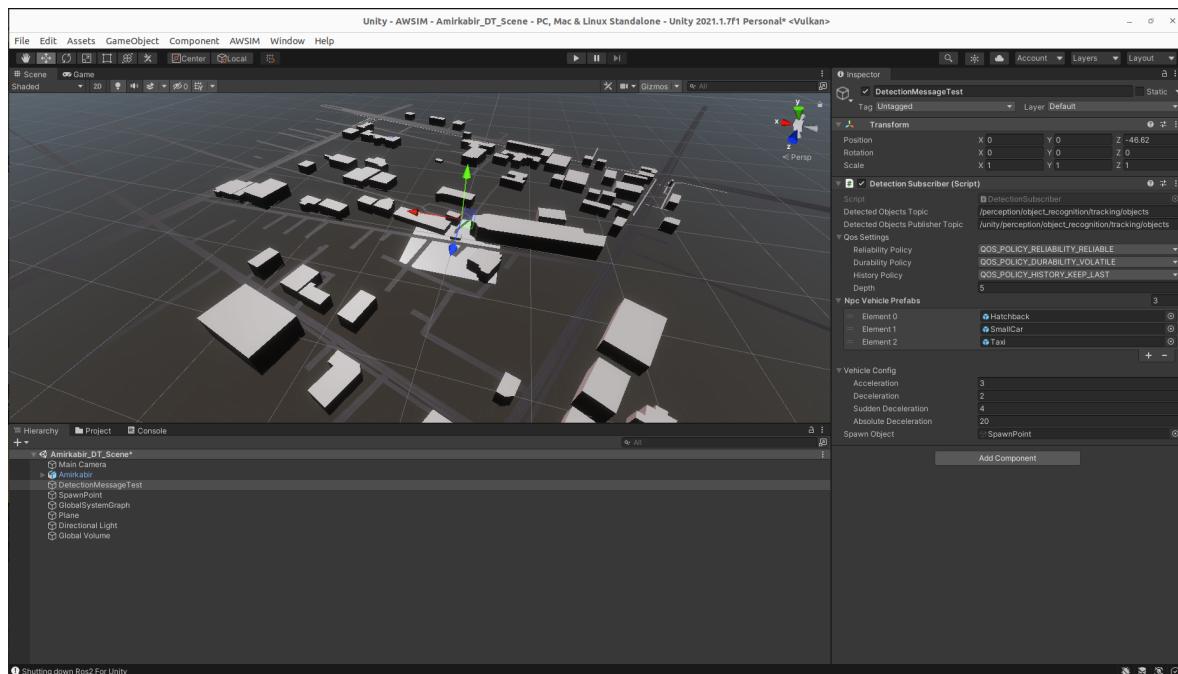
شکل ۴۴-۴ منطقه انتخاب شده در افزونه Blosm

ساختن مدل را می‌زنیم و پس از کمی صبر کردن، مدل سه‌بعدی زیر را مشاهده می‌کنیم. هر چند این مدل سه‌بعدی از منطقه امیرکبیر، ۱۰۰٪ دقیق نیست؛ اما بهینه‌ترین روش برای طراحی منطقه‌ای به این بزرگی است. کار مدل‌سازی سه‌بعدی یک منطقه، کاری بسیار دشوار است و نیاز به متخصص‌هایی در این زمینه دارد. پس در این پژوهش، به همین نقشه بسته می‌کنیم. یکی از ویژگی‌های خوب Blender، سازگاری آن با موتور بازی‌سازی Unity است. این مدل را به راحتی در صحنه جدیدی که نام آن را



شکل ۴-۴۵ مدل سه بعدی منطقه دانشگاه صنعتی امیرکبیر که توسط افزونه Blosm تولید شده است.

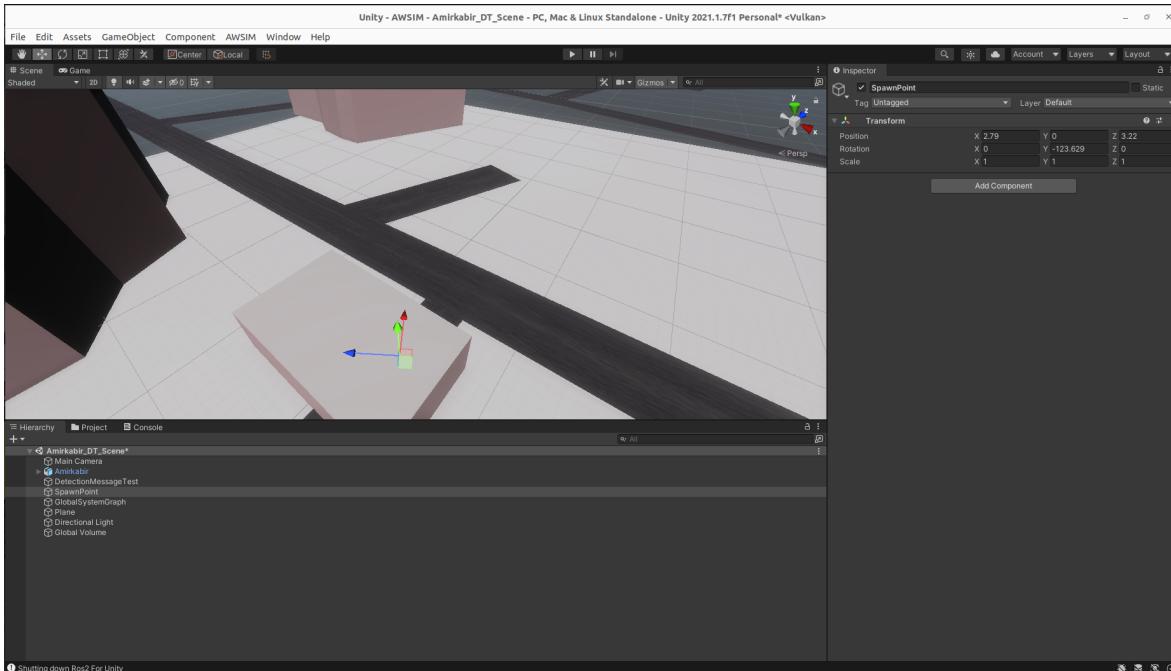
گذاشته‌ایم، بارگذاری می‌کنیم.



شکل ۴-۴۶ نقشه وارد شده در شبیه‌ساز AWSIM

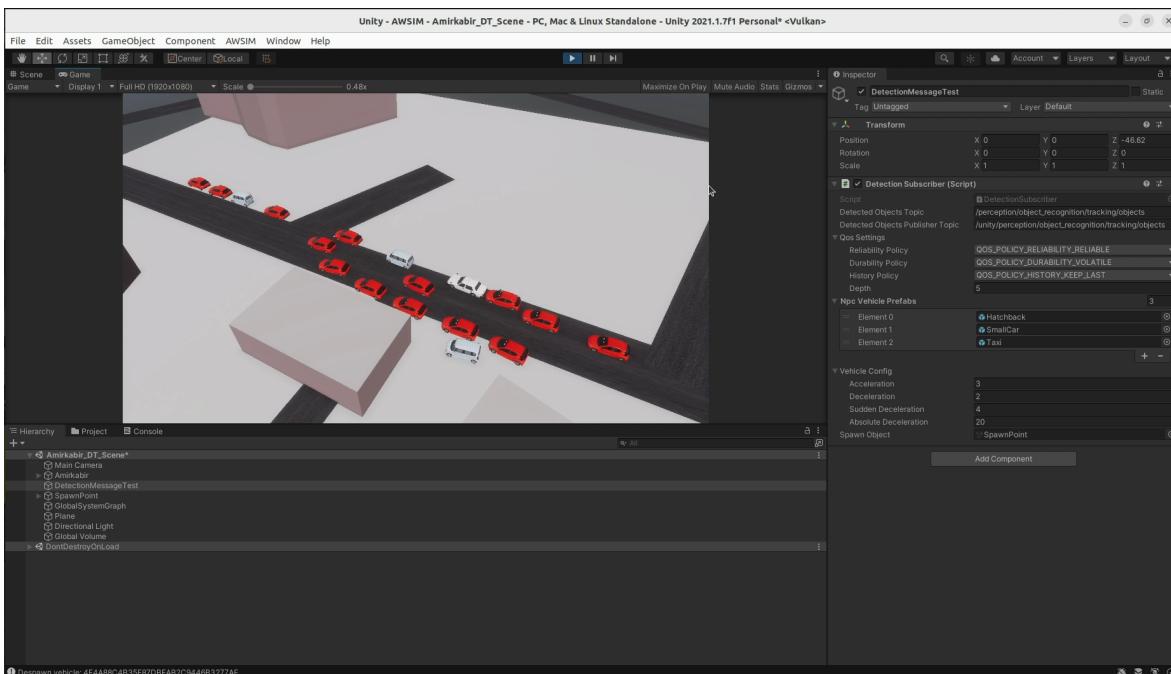
۹-۴ شبیه‌سازی در خیابان رشت طراحی شده

روش پدیدار کردن خودروها همانند قبل است. تنها چیزی که متفاوت است، مختصات شیء SpawnPoint است. مختصات و زاویه این شئ را همانند لایداری که بالای درب رشت بود، جای گذاری می‌کنیم. بایستی

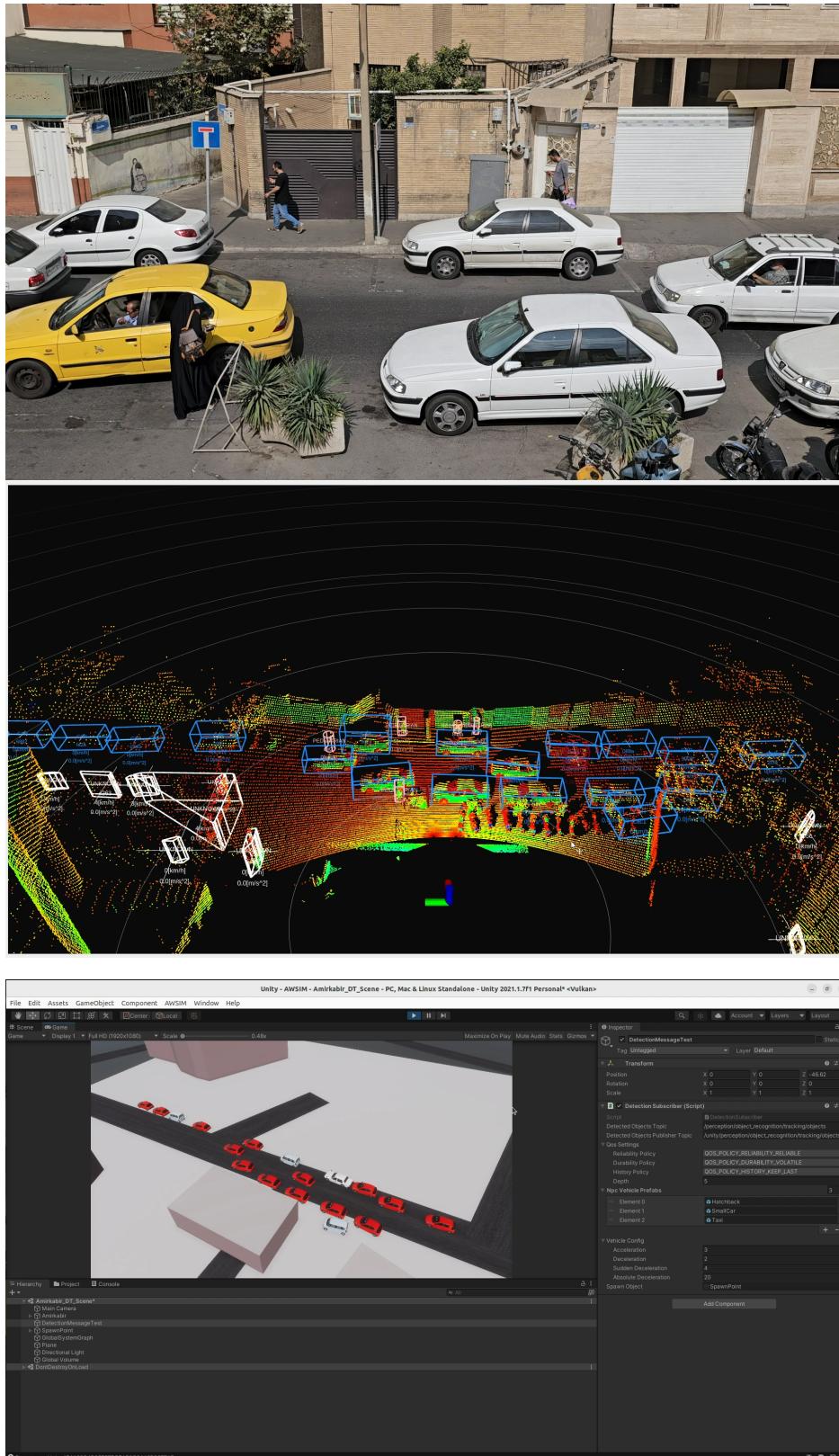


شکل ۴-۴ شیء بازی SpawnPoint که نسبت به موقعیت تبدیل یافته لایدار مستقر شده است.

دقیق شود که مختصات لایدار نیز به علت ضرب شدن در ماتریس تبدیل و ماتریس دوران، تغییر کرده است. حال تبدیل کننده را اجرا می‌کنیم، سپس فایل‌های اجرایی دوکلوبی دیجیتال ابزار Autoware را که خود طراحی کرده‌ایم را اجرا می‌کنیم و در نهایت نیز صحنه Amirkabir_DT_Scene را اجرا می‌کنیم. شکل زیر نتیجه همکاری همه‌ی پروسه‌های بخش‌های قبل، است:



شکل ۴-۵ دوکلوبی تقریبی دیجیتال از خیابان رشت

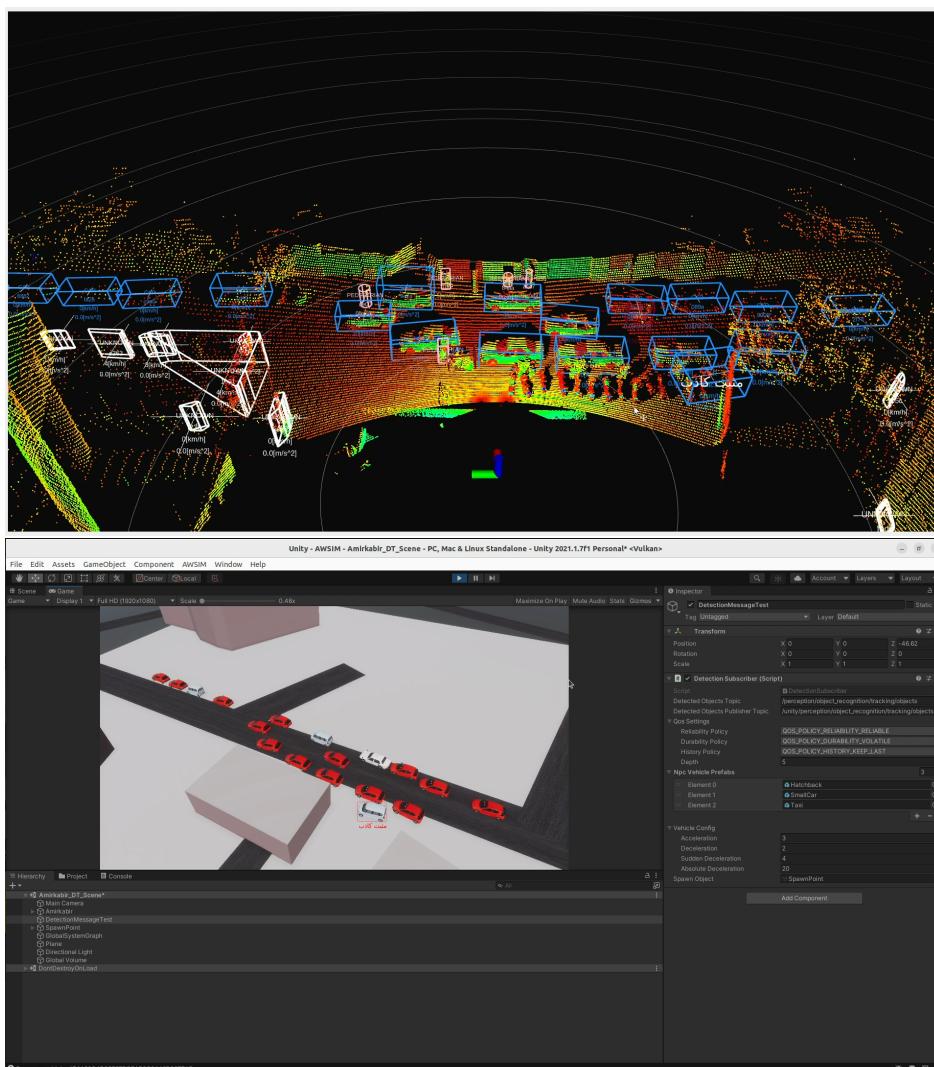


شکل ۴-۴ سه تصویر مختلف از یک لحظه زمانی (تصویر، تشخیص، شبیه‌سازی)

فصل پنجم

ارزیابی و بررسی

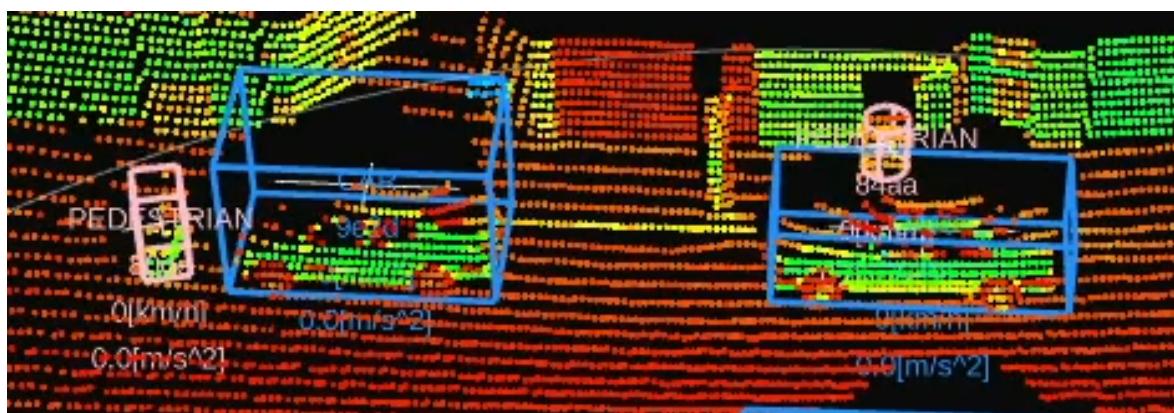
در فصل قبل مشاهده کردیم که شبیه‌سازی ترافیک با موفقیت اجرا شد. مشکل این دو قلوی دیجیتال، دقیق نسبتاً پایین آن است. به طور مثال، بیشتر اوقات می‌توان لرژش و حرکت ریز خودروهای پارک شده را مشاهده کرد که نتیجه یکسان نبودن قادر محصور کننده این جسم، در طول فریم‌های مختلف است. همچنین در شکل ۱-۵ می‌توان مشاهده کرد که در پایین ماشین‌های پارک شده، ماشین دیگری نیز وجود دارد که در واقع متشکل از چهار موتور پارک شده است. این نشان می‌دهد که تشخیص دهنده، خیلی اوقات ممکن است مثبت کاذب داشته باشد و یا به اشتباه طبقه‌بندی کند. پس با این حساب، نمی‌توان به مقدار ماشین‌هایی که در نیم ساعت تشخیص داده شده‌اند، اطمینان کرد.



شکل ۱-۵ تصویری از مثبت کاذب تشخیص داده شده

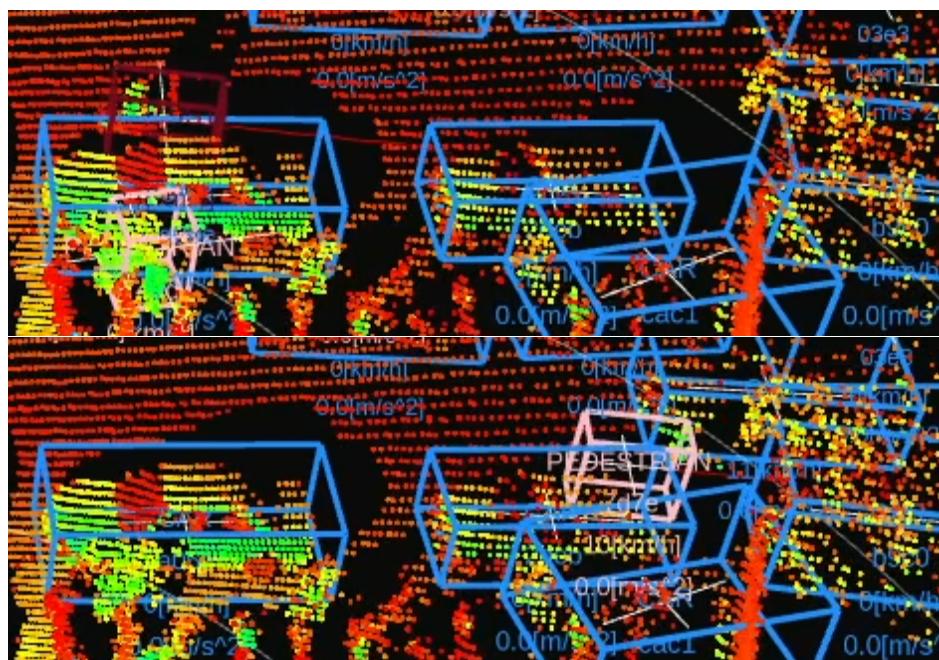
در بخش دستکاری با فایل‌های اجرایی Autoware، دو فیلتر مهم خاموش شده‌اند. این فیلترها باعث می‌شوند که تعداد مثبت‌ها کاذب کاهش یابد. به دلیل نداشتن نقشه ابر نقاط و نقشه برداری دقیق از منطقه دانشگاه صنعتی امیرکبیر، این فیلترها تا اطلاعات ثانویه خاموش خواهند بود. این بدین معنا است که مدل ردمیاب همزمان چند جسم‌ما، با داده‌هایی کاذب بیشتری دارد کار می‌کند و این

موضوع از دقت آن می‌کاهد. به علت وجود نداشتن این فیلترها، برخی از اجسام متعلق به نقشه ابر نقاط نیز، به اشتباه به عنوان اجسام حاضر در صحنه تشخیص داده می‌شوند. همانطور که مشاهده می‌کنیم،



شکل ۲-۵ تصویری از تشخیص اشتباه دو میله به عنوان عابرین پیاده

شکل ۲-۵ این مشکل را به وضوح نمایش می‌دهد. از دیگر مثبت‌های کاذب که می‌توان به آن اشاره کرد، تشخیص اشتباه موتور به عنوان عابرپیاده است. برای حل این مشکل راهی جز افزایش دقت مدل تشخیص‌دهنده و جستجو برای مدل‌های بهتر نداریم. شکل ۳-۵، تشخیص اشتباه موتورسیکلت را نمایش می‌دهد. دلیل این تشخیص اشتباه، به موضوعی



شکل ۳-۵ تصویری از تشخیص اشتباه موتورسیکلت به عنوان عابرپیاده

که در فصل دوم به آن پرداخته بر می‌گردد. این تشخیص اشتباه بخاطر اتفاقی به نام انسداد از بیرون یا External Occlusion است. در این مثال، سیگنال‌های لایدار ماشین‌های پارک شده، مانعی برای تشخیص موتورسیکلت به حساب می‌آیند.

فصل ششم

جمع‌بندی، نتیجه‌گیری و پیشنهادات برای کارهای آتی

۱-۶ جمع‌بندی و نتیجه‌گیری

در این پژوهه، سعی کردیم که یک دوکلوبی دیجیتال از خیابان رشت بسازیم. در این پژوهه، انواع معماری‌های ابزارها با دقت بررسی شدند تا بتوان به یک روش عملی برای پیاده‌سازی این دوکلوبی دیجیتال رسید. مهم‌ترین اجزای این پژوهه، کتابخانه‌های توسعه نرم‌افزار ROS2 و Autoware بودند. بدون دانستن معماری دقیق این دو ابزار، این پژوهه امکان پذیر نبود. فرایند گرفتن الهام و ایده برای استفاده از تشخیص‌دهنده مولفه ادراک Autoware، نشان می‌دهد که برای حل کردن مسائل بزرگ، نیاز است تا آن را به زیرمسئله‌های کوچکتر شکاند. همچنین می‌توان نتیجه گرفت که برخی اوقات جواب داده شده برای یک مسئله بزرگ، ممکن است جوابی برای مسئله‌های بزرگ دیگر نیز باشد؛ به طور مثال، مسئله تشخیص اجسام در خودروهای خودران همانند مسئله تشخیص اجسام در پیاده‌سازی دوکلوبی دیجیتال یک منطقه است. در آخر نیز توجه داشته باشیم که شکل‌گیری این دوکلوبی دیجیتال، به کمک شبیه‌ساز AWSIM امکان‌پذیر شد و بدون افروزه آن که R2FU نام داشت، راهی برای دریافت اطلاعات ردیاب در موتور بازی‌سازی Unity وجود نداشت. در فصل ارزیابی نیز به مشکلات این دوکلوبی دیجیتال پرداخته که حتما نیاز به تحقیق و بررسی دارند. با وجود اینکه نمی‌توان به نتیجه این پژوهش، برچسب دوکلوبی دیجیتال دقیق زد، اما می‌توان آن را گامی مثبت در جهت به واقعیت رسیدن این رویا دانست. با توجه به نتایج گرفته شده، می‌توان برای آینده این فناوری امیدوار بود.

۲-۶ پیشنهادات برای کارهای آتی

این پژوهه، قابلیت تکمیل و توسعه از ابعاد مختلفی را دارد. برخی از آنها نیاز به تحقیقات بیشتر دارند و برخی از آنها، نیازمند کار عملی زیاد است.

۱-۲-۱ طراحی نقشه ابر نقاط

برای اینکه برخی از اجسام مربوط به نقشه، به عنوان اجسام حاضر در صحنه تشخیص داده نشوند، نیاز است که یک نقشه ابر نقاط با ادغام اسکن‌های لایدار در طول خیابان رشت، ساخت. با داشتن نقشه ابر نقاط، می‌توانیم از فیلتر مربوط به آن در ابزار Autoware استفاده کنیم.

۱-۲-۲ تهییه نقشه برداری

با وجود داشتن نقشه برداری از طریق نرم‌افزار OpenStreetMap، نمی‌توان به آن بسنده کرد زیرا ابعاد دقیق لاین‌های خیابان را به درستی در نیاورده است. همچنین اطلاعات مربوط به نقشه ایران در این ابزار، کمی قدیمی و بروزرسانی نشده است. از آنجایی که یکی از فیلترهای ابزار Autoware، مبتنی بر این نقشه است که تا حد خوبی مثبت‌های کاذب را از تشخیص‌ها فیلتر می‌کند. پس حتما به یک نقشه برداری دقیق از منطقه دانشگاه امیرکبیر نیاز خواهیم داشت.

۶-۲-۳ مدل‌سازی کامل منطقه دانشگاه امیرکبیر

مدل سه‌بعدی استفاده شده در این پژوهه، خیلی ساده است و بسیاری از ساختمان‌ها در آن مدل‌سازی نشده‌اند. این کار نیاز به متخصصین مدل‌سازی، مهندسین نقشه‌کشی و معمارها دارد. طراحی مدل سه‌بعدی یک منطقه کاری بسیار دشوار و هزینه‌بر است و در حد این پژوهش نبود.

۶-۲-۴ استفاده از مدل‌های جدید

مدل استفاده شده در گره ادراک ابزار Autoware، مدل هوش مصنوعی CenterPoint بود که در سال ۲۰۲۱ ساخته شده است. هم اکنون سال ۲۰۲۳ است و مدل‌های هوش مصنوعی بسیار قوی‌تری در زمینه تشخیص اجسام سه‌بعدی وجود دارد. می‌توان مدل استفاده شده در Autoware را با این مدل‌ها جایگزین کرد تا نتایجی بهتر گرفت.

كتاب نامه

[1] 51WORLD. Digital twin makes urban travel smoother.

<https://thinkautonomous.medium.com/can-self-driving-car-think-6c9e8d939d60>, 2019. [Accessed Oct. 13, 2023].

[2] Ouster. High resolution os1 sensor.

<https://ouster.com/products/scanning-lidar/os1-sensor/>, 2023.
[Accessed Oct. 13, 2023].

[3] Zhou, Zhi, Chen, Xu, Li, En, Zeng, Liekang, Luo, Ke, and Zhang, Junshan. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 107(8):1738–1762, 2019.

[4] Yamashita, Rikiya, Nishio, Mizuho, Do, Richard Kinh Gian, and Togashi, Kaori. Convolutional neural networks: an overview and application in radiology. Insights into imaging, 9(4):611–629, 2018.

[5] Qian, Rui, Lai, Xin, and Li, Xirong. 3d object detection for autonomous driving: A survey. Pattern Recognition, 130:108796, 2022.

[6] Shah, Deval. map explained. <https://www.v7labs.com/blog/mean-average-precision#:~:text=Average%20Precision%20is%20calculated%20as,mAP%20varies%20in%20different%20contexts.>, 2022. [Accessed Oct. 14, 2023].

- [7] Han, Guangjie, Zhu, Yintian, Liao, Lyuchao, Yao, Huiwen, Zhao, Zhaolin, and Zheng, Qi. Hybrid attention-based 3d object detection with differential point clouds. *Electronics*, 11(23):4010, 2022.
- [8] De Maesschalck, Roy, Jouan-Rimbaud, Delphine, and Massart, Désiré L. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.
- [9] Lu, Qiuchen, Parlikad, Ajith Kumar, Woodall, Philip, Don Ranasinghe, Gishan, Xie, Xiang, Liang, Zhenglin, Konstantinou, Eirini, Heaton, James, and Schooling, Jennifer. Developing a digital twin at building and city levels: Case study of west cambridge campus. *Journal of Management in Engineering*, 36(3):05020004, 2020.
- [10] Azfar, Talha, Weidner, Jeffrey, Raheem, Adeeba, Ke, Ruimin, and Cheu, Ruey Long. Efficient procedure of building university campus models for digital twin simulation. *IEEE Journal of Radio Frequency Identification*, 6:769–773, 2022.
- [11] Wang, Shao-Hua, Tu, Chia-Heng, and Juang, Jyh-Ching. Automatic traffic modelling for creating digital twins to facilitate autonomous vehicle development. *Connection Science*, 34(1):1018–1037, 2022.
- [12] ROS. Ros ecosystem. <https://www.ros.org/blog/ecosystem/>, 2023. [Accessed Oct. 15, 2023].
- [13] Macenski, Steven, Foote, Tully, Gerkey, Brian, Lalancette, Chris, and Woodall, William. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.
- [14] 2, ROS. Ros 2 humble documentation. <https://docs.ros.org/en/humble/index.html>, 2023. [Accessed Oct. 16, 2023].

- [15] AWF. Autoware documentation.
<https://autowarefoundation.github.io/autoware-documentation/main/>,
2023. [Accessed Oct. 17, 2023].
- [16] Tier4. Awsim documentation. <https://tier4.github.io/AWSIM/>, 2023.
[Accessed Oct. 18, 2023].
- [17] Ouster. Oustersdk documentation. <https://static.ouster.dev/sdk-docs/>,
2023. [Accessed Oct. 18, 2023].
- [18] Dung, Nguyen Mau. Super-Fast-Accurate-3D-Object-Detection-PyTorch. <https://github.com/maudzung/Super-Fast-Accurate-3D-Object-Detection>, 2020.
- [19] Contributors, MMDetection3D. MMDetection3D: OpenMMLab next-generation
platform for general 3D object detection.
<https://github.com/open-mmlab/mmdetection3d>, 2020.
- [20] Yin, Tianwei, Zhou, Xingyi, and Krahenbuhl, Philipp. Center-based 3d object
detection and tracking. in Proceedings of the IEEE/CVF conference on computer
vision and pattern recognition, pp. 11784–11793, 2021.
- [21] AWF. Autoware universe documentation.
<https://autowarefoundation.github.io/autoware.universe/main/>, 2023.
[Accessed Oct. 17, 2023].
- [22] Grieves, Michael. Digital twin: manufacturing excellence through virtual factory
replication. White paper, 1(2014):1–7, 2014.
- [23] Glaessgen, Edward and Stargel, David. The digital twin paradigm for future nasa and
us air force vehicles. in 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural
dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures
conference 14th AIAA, p. 1818, 2012.

- [24] Rosen, Roland, Von Wichert, Georg, Lo, George, and Bettenhausen, Kurt D. About the importance of autonomy and digital twins for the future of manufacturing. Ifac-papersonline, 48(3):567–572, 2015.
- [25] Farsi, Maryam, Daneshkhah, Alireza, Hosseinian-Far, Amin, Jahankhani, Hamid, et al. Digital twin technologies and smart cities. Springer, 2020.
- [26] Singh, Maulshree, Fuenmayor, Evert, Hinchy, Eoin P, Qiao, Yuansong, Murray, Niall, and Devine, Declan. Digital twin: Origin to future. Applied System Innovation, 4(2):36, 2021.
- [27] Synopsis. What is lidar?
<https://www.synopsys.com/glossary/what-is-lidar.html>, 2020. [Accessed Oct. 13, 2023].
- [28] Arnold, Eduardo, Al-Jarrah, Omar Y, Dianati, Mehrdad, Fallah, Saber, Oxtoby, David, and Mouzakitis, Alex. A survey on 3d object detection methods for autonomous driving applications. IEEE Transactions on Intelligent Transportation Systems, 20(10):3782–3795, 2019.
- [29] Geiger, Andreas, Lenz, Philip, and Urtasun, Raquel. Are we ready for autonomous driving? the kitti vision benchmark suite. in 2012 IEEE conference on computer vision and pattern recognition, pp. 3354–3361. IEEE, 2012.
- [30] Caesar, Holger, Bankiti, Varun, Lang, Alex H, Vora, Sourabh, Liong, Venice Erin, Xu, Qiang, Krishnan, Anush, Pan, Yu, Baldan, Giancarlo, and Beijbom, Oscar. nuscenes: A multimodal dataset for autonomous driving. in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11621–11631, 2020.
- [31] Sun, Pei, Kretzschmar, Henrik, Dotiwalla, Xerxes, Chouard, Aurelien, Patnaik, Vijaysai, Tsui, Paul, Guo, James, Zhou, Yin, Chai, Yuning, Caine, Benjamin, et al.

Scalability in perception for autonomous driving: Waymo open dataset. in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2446–2454, 2020.

- [32] Gat, Erann, Bonnasso, R Peter, Murphy, Robin, et al. On three-layer architectures. Artificial intelligence and mobile robots, 195:210, 1998.

- [33] Lang, Alex H, Vora, Sourabh, Caesar, Holger, Zhou, Lubing, Yang, Jiong, and Beijbom, Oscar. Pointpillars: Fast encoders for object detection from point clouds. in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12697–12705, 2019.

- [34] Zhang, Xiao, Xu, Wenda, Dong, Chiyu, and Dolan, John M. Efficient l-shape fitting for vehicle detection using laser scanners. in 2017 IEEE Intelligent Vehicles Symposium, June 2017.



Amirkabir University of Technology
(Tehran Polytechnic)

Department of Computer Engineering

Bachelor Thesis

Traffic Scene Digital Twin Based on Camera and Lidar using
ROS and AWSIM

By
Roham Zendehdel Nobari

Supervisor
Dr. Mahdi Javanmardi

September 2023