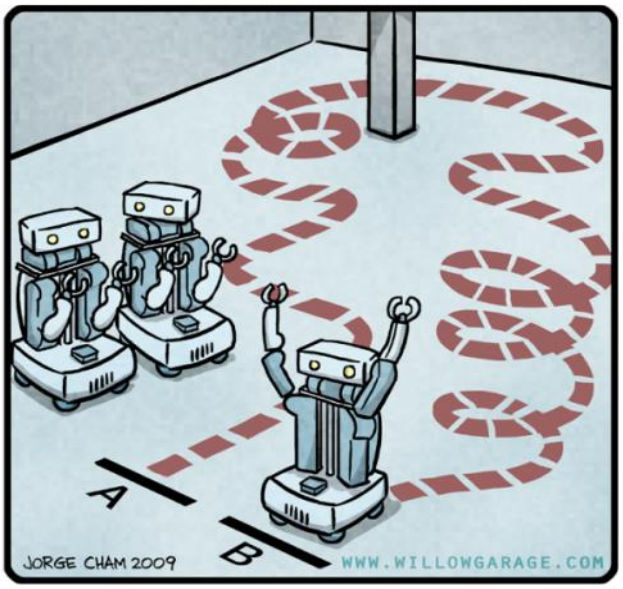
به نـام آنکه خلقت، با نام اوست برپا / مکانی و زمانی، از امر اوست برجـــا
به جلال و شکوهی که از اوست برما / آب گیری، از جوی صنع اوست تنها
مجتبی والی پور

R.O.B.O.T. Comics



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

دانشگاه صنعتی جندی‌شاپور دزفول
دانشکده برق و کامپیوتر
گروه کامپیوتر

پایان‌نامه برای دریافت درجه کارشناسی
در رشته مهندسی کامپیوتر گرایش سخت افزار

طراحی و پیاده سازی سامانه هوشمند ناوبری

استاد راهنما:
دکتر موسی موسی زاده

نویسنده:
مجتبی والی پور

خرداد ۱۳۹۴

تأییدیه‌ی صحت و اصالت نتایج

بسمه تعالی

اینجانب مجتبی والی پور به شماره دانشجویی ۹۰۱۸۴۶۱۲۳ دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی لیسانس تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه/رساله حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر نموده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

مجتبی والی پور

امضا و تاریخ:

دنیا پر از زیبایی ها و زشتی هاست اما قطعا خانواده زیباترین هدیه ای است که خداوند به من داده است. لذتی که در انجام این پروژه داشتم را نمی توانم توصیف و یا شرح دهم اما علم و به خصوص دانش مرتبط با کامپیوتر و از آن بیشتر، دانش هوش مصنوعی برایم بسیار گوارا است که شاید این عشق از عملکرد خانواده ام به وجود آمده است. از پدرم، مادر عزیز تر از جانم و برادران و خواهر مهربانم تشکر می کنم که بیشتر راه را با کمک ایشان پیموده ام. بی شک بدون حضورشان این مسیر ممکن نبود و همواره به یاد خواهم داشت. **تقدیم به آن ها که از صمیم قلب دوستشان دارم.**

تقدیر:

از تمام اساتیدی که در چهار سال کارشناسی و لیسانس در پیمودن این راه به من کمک کردند و راه صحیح را به من نشان دادند تشکر می کنم. واقعا نمی توانم از ایده ها و راهنمایی های جناب آقای حجه الله لطیفی و حمایت های بی دریغ جناب پروفسور علی محمد آخوند علی تشکر نکنم. از راهنمایی و زحمات اساتید محترم دکتر موسی موسی زاده و مهندس مهدی عنایت زارع تشکر ویژه دارم.

فهرست مطالب

عنوان **صفحه**

کلیات **۲**

۱-۱- مقدمه ۳

۱-۲- ساختار پایان نامه ۵

پیشینه مطالعات **٦**

۲-۱- مقدمه ۷

۲-۲- پروژه ها و تلاش های مشابه ۷

۲-۲-۱- گوگل ۷

۲-۲-۲- تیم تحقیقاتی دانشگاه اکسفورد ۱۰

۲-۲-۳- پروژه ی SARTRE و Drive Me ۱۱

۲-۳- ابزار و شبیه سازها ۱۶

۲-۳-۱- سیستم عامل ربات ROS ۱۶

۲-۳-۲- شبیه ساز Rviz ۲۲

۲-۳-۳- شبیه ساز Stage ۲۲

۲-۳-۴- شبیه ساز Gazebo ۲۲

۲-۴- زبان های توصیفی و برنامه نویسی ۲۳

۲-۴-۱- زبان نشانه گذاری URDF ۲۳

۲-۴-۲- زبان نشانه گذاری SDF ۲۴

۲-۵- نتیجه گیری ۲۵

الگوریتم های ناوبری **۲٦**

۳-۱- مقدمه ۲۷

۳-۲- معرفی و بررسی الگوریتم های موجود ۲۷

۳-۲-۱- پیش نیاز ۲۷

۳-۲-۲- فیلتر بیز و هیستوگرام ۲۸

۳-۲-۳- فیلتر کالمن ۲۸

۳-۲-۴- فیلتر ذرات ۳۰

۳-۲-۵- الگوریتم AMCL ۳۱

۳-۲-۶- الگوریتم GMapping ۳۲
۳-۲-۷- الگوریتم DWA ۳۳
۳-۲-۸- نتیجه گیری ۳۵

طراحی و پیاده سازی **۳٦**
۴-۱- طراحی پلتفرم / ربات ۳۷
۴-۱-۱- طراحی روبات فراز سایبرهندس ۳۷
۴-۲- طراحی و پیاده سازی نرم افزار ۴۰
۴-۲-۱- توصیف فراز برای شبیه سازها ۴۰
۴-۲-۲- پیاده سازی، نصب و تنظیم پارامترهای ناوبری ۴۰
۴-۳- نتیجه گیری و ارزیابی نهایی ۴۷
۴-۳-۱- تست و ارزیابی عملکرد نهایی ربات فراز سایبرهندس ۴۷

نتیجه‌گیری و پیشنهادها **٥٦**

مقدمه **۵۷**
۵-۱- نتیجه گیری و پیشنهادهای پژوهشی ۵۷

۵-۱-۱- جمع‌بندی **۵۷**
۵-۱-۲- نوآوری ۵۷
۵-۱-۳- پیشنهادها ۵۸

فهرست منابع **۵۹**

پیوست‌ها **٦۳**

فهرست جداول

عنوان **صفحه**

جدول (۲-۱) مثالی ابتدایی از URDF ۲۳
جدول (۲-۲) مثالی ابتدایی از SDF ۲۴
جدول (۳-۱) شبه کد الگوریتم فیلتر بیز گسسته ۲۸
جدول (۳-۲) شبه کد الگوریتم فیلتر کالمن ۳۰
جدول (۳-۳) شبه کد الگوریتم فیلتر ذرات ۳۰
جدول (۳-۴) شبه کد الگوریتم AMCL [24] ۳۱
جدول (۳-۵) شبه کد الگوریتم DWA [25] ۳۴
جدول (۴-۱) کد انتشار وضعیت فریم ها، اتصالات و لینک های ربات ۴۳
جدول (۴-۲) کد فراخوانی گره ی Slam_Gmapping و تنظیمات آن ۴۳
جدول (۴-۳) کد فراخوانی Map_Server و انتشار نقشه ی ایستا ۴۴
جدول (۴-۴) کد فراخوانی AMCL برای انجام محل یابی ربات ۴۴

فهرست اشکال

عنوان **صفحه**

شکل (۱-۱) ناوبری خودمختار و ماشین های خود هدایت شونده.....۳
شکل (۲-۱) ماشین گوگل در خیابان ها و جاده های عمومی ۸
شکل (۲-۲) ماشین گوگل – تعامل با چندین شی ثابت و متحرک.....۹
شکل (۲-۳) ماشین گروه تحقیقاتی دانشگاه اکسفورد + سیستم نقشه برداری و محل یابی..... ۱۰
شکل (۲-۴) ماشین گروه تحقیقاتی دانشگاه اکسفورد + سیستم نقشه برداری و محل یابی..... ۱۱
شکل (۲-۵) شبیه سازی آیرودینامیکی دینامیک سیالات محاسباتی (CFD) پروژه SARTRE [5]..... ۱۲
شکل (۲-۶) ارزیابی اولیه ی پروژه SARTRE در محیط واقعی و آزمایشی..... ۱۳
شکل (۲-۷) پروژه ی Drive Me شرکت Volvo [7]..... ۱۴
شکل (۲-۸) دید ۳۶۰ درجه ی مجموع سنسورها در پروژه ی Drive Me [8]..... ۱۵
شکل (۲-۹) رادارها در پروژه ی Drvie Me [9]..... ۱۵
شکل (۲-۱۰) دوربین ها در پروژه ی Drive Me [10]..... ۱۵
شکل (۲-۱۱) لیزر چندپرتویی در Drive Me [11]..... ۱۶
شکل (۲-۱۲) سنسورهای UltraSonic در Drive Me [11]..... ۱۶
شکل (۲-۱۳) سیستم عامل رز در یک نگاه..... ۱۷
شکل) ۲-۱۴(مفاهیم پایه در سیستم عامل رز[15] ۲۱
شکل) ۲-۱(محیط Rviz–ت۱..... ۲۲
شکل) ۲-۲(محیط Rviz–ت۲..... ۲۲
شکل (۲-۳) محیط Rviz–ت۳..... ۲۲
شکل) ۲-۴(محیط Stage–ت۱..... ۲۲
شکل) ۲-۵(محیط Stage–ت۲..... ۲۲
شکل (۲-۶) محیط Stage–ت۳..... ۲۲
شکل) ۲-۷(محیط Gazebo–ت۱..... ۲۳
شکل) ۲-۸(محیط Gazebo–ت۲..... ۲۳
شکل (۲-۹) محیط Gazebo–ت۳..... ۲۳
شکل (۳-۱) نمودار توزیع نرمال در دیاگرام انحراف معیار [19]..... ۲۷

شکل (۳-۲) معادلات فیلتر کالمن [21] ۲۹
شکل (۳-۳) نقشه ی فضای باز دانشگاه فرایبورگ ۳۳
شکل (۳-۴) نقشه ی MIT Killian Court ۳۳
شکل (۳-۵) الگوریتم DWA ۳۳
شکل (۴-۱) روبات فراز از نمای عقب + شاسی ۳۸
شکل (۴-۲) روبات فراز از نمای جلو + شاسی ۳۸
شکل (۴-۳) روبات فراز از نمای چپ + شاسی ۳۸
شکل (۴-۴) روبات فراز از نمای بالا + شاسی ۳۸
شکل (۴-۵) روبات فراز از نمای بالا + بدنه ۳۸
شکل (۴-۶) روبات فراز از نمای چپ + بدنه ۳۸
شکل (۴-۷) مدل سه بعدی فراز سایبرهندس – Cycles Render در بلندر ۳۹
شکل (۴-۸) مدل سه بعدی فراز – نمای پشت ۳۹
شکل (۴-۹) مدل سه بعدی فراز – نمای راست ۳۹
شکل (۴-۱۰) مدل سه بعدی فراز – نمای بالا ۴۰
شکل (۴-۱۱) مدل سه بعدی فراز – نمای جلو ۴۰
شکل (۴-۱۲) درخت ارتباط فریم ها در ربات فراز ۴۲
شکل (۴-۱۳) تنظیمات و نصب پشته ی ناوبری در سیستم عامل رز[7] ۴۶
شکل (۴-۱۴) دیاگرام ارتباط بین تمام گره ها در ربات فراز سایبرهندس ۴۶
شکل (۴-۱۵) ماموریت ۱ – ربات فراز – ت ۱ ..... ۴۷
شکل (۴-۱۶) ماموریت ۱ – ربات فراز – ت ۲ ..... ۴۷
شکل (۴-۱۷) ماموریت ۱ – ربات فراز – ت ۳ ..... ۴۷
شکل) ۴-۱۸(ماموریت ۱ – ربات فراز – ت ۴ ..... ۴۸
شکل (۴-۱۹) ماموریت ۱ – ربات فراز – ت ۵ ..... ۴۸
شکل) ۴-۲۰(ماموریت ۱ – ربات فراز – ت ۶ ..... ۴۸
شکل (۴-۲۱) ماموریت ۱ – ربات فراز – ت ۷ ..... ۴۸
شکل (۴-۲۲) ماموریت ۲ – ربات فراز – ت ۱ ..... ۴۹
شکل (۴-۲۳) ماموریت۲ – ربات فراز – ت ۲ ..... ۴۹
شکل (۴-۲۴) ماموریت ۲ – ربات فراز – ت ۳ ..... ۴۹

شکل (۴-۲۵) ماموریت ۲ – ربات فراز – ت ۴ ..... ۴۹
شکل (۴-۲۶) ماموریت ۲ – ربات فراز – ت ۵ ..... ۴۹
شکل (۴-۲۷) ماموریت ۳ – ربات فراز – ت ۱ ..... ۵۰
شکل (۴-۲۸) ماموریت ۳ – ربات فراز – ت ۲ ..... ۵۰
شکل (۴-۲۹) ماموریت ۳ – ربات فراز – ت ۳ ..... ۵۰
شکل (۴-۳۰) ماموریت ۳ – ربات فراز – ت ۴ ..... ۵۰
شکل (۴-۳۱) ماموریت ۳ – ربات فراز – ت ۵ ..... ۵۰
شکل (۴-۳۲) ماموریت ۴ – ربات فراز – ت ۱ ..... ۵۱
شکل (۴-۳۳) ماموریت ۴ – ربات فراز – ت ۲ ..... ۵۱
شکل (۴-۳۴) ماموریت ۴ – ربات فراز – ت ۳ ..... ۵۱
شکل (۴-۳۵) ماموریت ۴ – ربات فراز – ت ۴ ..... ۵۱
شکل (۴-۳۶) ماموریت ۵ – ربات فراز – ت ۱ ..... ۵۱
شکل (۴-۳۷) ماموریت ۵ – ربات فراز – ت ۲ ..... ۵۱
شکل (۴-۳۸) ماموریت ۵ – ربات فراز – ت ۳ ..... ۵۲
شکل (۴-۳۹) ماموریت ۵ – ربات فراز – ت ۴ ..... ۵۲
شکل (۴-۴۰) ماموریت ۵ – ربات فراز – ت ۵ ..... ۵۲
شکل (۴-۴۱) ماموریت ۵ – ربات فراز – ت ۶ ..... ۵۲
شکل (۴-۴۲) فراز در محیط شبیه ساز سه بعدی ۵۳
شکل (۴-۴۳) فراز در محیط شبیه ساز سه بعدی ۵۳
شکل (۴-۴۴) فراز در محیط شبیه ساز سه بعدی ۵۳
شکل (۴-۴۵) فراز در محیط شبیه ساز سه بعدی ۵۳
شکل (۴-۴۶) فراز در محیط شبیه ساز سه بعدی ۵۳
شکل (۴-۴۷) فراز در محیط شبیه ساز سه بعدی ۵۴
شکل (۴-۴۸) فراز در محیط شبیه ساز سه بعدی ۵۴
شکل (۴-۴۹) فراز در محیط شبیه ساز سه بعدی ۵۴

فهرست علائم اختصاری

Robot Operating System ..……...…… ROS

ROS..…….…… سیستم عامل رز

Unified Robot Description Format ..…...…….…… URDF

Open Source Robotics Foundation ..…..…...…..…… OSRF

Extensible Markup Language….....……..…… XML

Simultaneous Localization And Mapping….....…...…….SLAM

Concurrent Mapping and Localization……...……CML

چکیده

امروزه انجام امور تکراری کاری بیهوده و بی ثمر اســـت و باید از ربات ها و فرآیندهای خودکار اســـتفاده کرد. روبات ها به زودی انجام کارهای تکراری و روتین نظیر رانندگی را به عهده خواهند گرفت. در این پروژه فقط از ابزارهای متن باز ا ستفاده شده ا ست تا به طور کامل روباتی از صفر طراحی و پیاده سازی شود. این ربات که با نام فراز شـــناخته می شـــود از تکنیک ها و الگوریتم های مختلف ناوبری اســـتفاده می کند تا به صورت خودمختار مسیر خود را از موانع پیدا کرده و با امنیت بالا و بدون برخورد به موانع اطراف آن به مقصد تعیین شده برسد.

واژه‌های کلیدی: ناوبری، خودمختار، شبیه ساز، ROS، طراحی

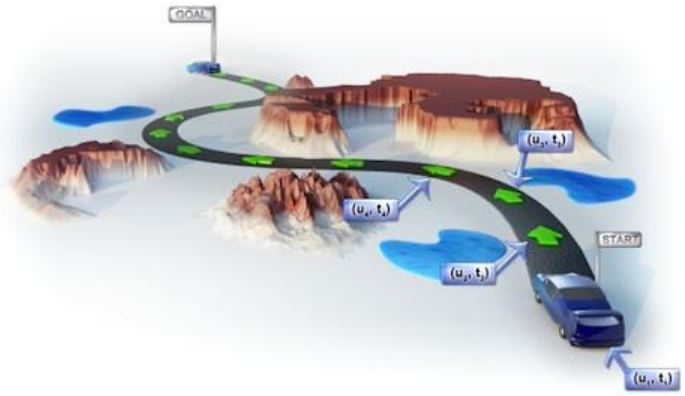
فصل اول

کلیات

۱-۱- مقدمه

امروزه انجام امور تکراری کاری بیهوده و بی ثمر اســـت و باید از فرآیندهای خودکار جهت انجام امور تکراری سود برد و اینگونه از زمان در برابر اتلاف به عنوان یک سرمایه ی گران بها جلوگیری نمود. هدف این پروژه، طراحی و ساخت سیستمی است که با اضافه شدن به هر ماشینی می تواند آن ها را به ماشین هایی با توانایی تصـــمیم گیری خودمختار و هدایت و ناوبری هوشـــمند تبدیل نموده و عملیات مورد نظر را به خوبی هدایت کند. منظور از ناوبری، هدایت و راهیابی وسایط نقلیه از مبدا به مقصد است و منظور از هوشمندی در این پروژه هدایت این سامانه به صورت خودمختار و توسط خود سیستم و در نهایت بدون دخالت انسان می باشد. گرچه ممکن است در ابتدا سیستم از کمک دانش انسانی برخوردار باشد اما در نهایت انسان نقشی در هدایت وســـیله به جز تعیین مقصـــد ندارد. به همین منظور این پروژه به چند فاز و در چند مرحله اجرایی خواهد شد. قسمت اول این پروژه به عنوان پروژه ی کارشناسی و در این فصول قابل دسترسی است.

دنیای فناوری و تکنولوژی، امروزه از ربات ها اســـتفاده های فراوانی می کند. دیر یا زود ربات های خودمختار با سطوح هو شمندی متفاوت به ان سان ها در امور مختلف کمک خواهند کرد و جان ان سان های زیادی را نجات خواهند داد. انســـان جانشـــین خود را در محیط های خطرناک کاری نظیر معادن، مکان های آلوده به امواج رادیواکتیو و آلودگی های مختلف شیمیایی، در شرایط بحرانی، در عملیات جستجو و نجات در حوادث و بلایای طبیعی پیدا کرده ا ست و زمانی فرا خواهد ر سید که از این ربات ها به صورت رایج ا ستفاده خواهد نمود. ربات ها به زودی انجام کارهای تکراری و روتین نظیر رانندگی را به عهده خواهند گرفت. این ربات ها برای اینکه بتوانند عملکرد هوشـــمندانه و بهتری را به نمایش بگذارند می بایســـت از محیط اطراف و محل دقیق خود اطلاع دا شته با شند. به این منظور، روش های مختلفی برای شناخت محیط اطراف و تعیین مسیر به کار گرفته می شود.



شکل (۱-۱) ناوبری خودمختار و ماشین های خود هدایت شونده

اگر ربات نقشه ی از پیش آماده ای داشته باشد و در حقیقت نسبت به مکان اطراف خود آگاه باشد، پیدا کردن موقعیت ربات کار آسانی است. نظیر این روش استفاده از فناوری GPS است که به سادگی موقعیت ربات را با دقت مناسبی تخمین می زند. اما استفاده از GPS به دلایل امنیتی و عدم دسترسی همیشگی برای ماشین های خود هدایت شونده روشی نیست که مورد اعتماد قرار گیرد. گرچه از این روش به دلیل سادگی و بهینگی به صورت ترکیبی استفاده می شود اما در محیط واقعی به دلیل تغییرات زیاد در محیط و جاده و همچنین عدم ثبات در محیط، به خ صوص در ما شین های خودمختار و خود هدایت شونده عملا نمی توان فقط از این روش ا ستفاده کرد. باید روش هایی را به کار گرفت که به صورت زنده و از طریق سن سورهای مختلف از محیط پیرامون خود آگاه می شوند و همزمان موقعیت ربات را با دقت بالایی تخمین می زنند. ربات ها معمولا از روش نقشه برداری و مکان یابی همزمان[1] (SLAM) برای پیدا کردن موقعیت فعلی شان استفاده می کنند. گرچه امروزه نقشه برداری و مکان یابی به صورت جدا کار آسانی برای یک ربات به شمار می رود اما به صورت همزمان رسیدن به هدف چندان آسان نیست. منظور از نقشه برداری به دست آوردن مدلی از محیط روبات و منظور از مکان یابی یافتن وضعیت فعلی ربات در نقشه است. از روش های احتمالاتی مکان یابی می توان از مدل مارکوف، فیلتر کالمن، مونت کارلو و فیلتر ذرات نام برد.

اینکه فناوری ماشین های خود هدایت شونده چه زمانی در اختیار عموم قرار خواهد گرفت و این فناوری در نهایت به چه شکلی پیاده سازی خوادهد شد، کاملا مشخص نیست. گر چه بعضی از پیش بینی ها سال ۲۰۱۷ و یا سال ۲۰۲۰ را برای عر ضه ی عمومی این تکنولوژی ب سیار پی شرفته م شخص کرده اند اما هنوز هم آینده ی این فناوری بسیار زیبا و دو ست انسان و طبیعت در هاله ای از ابهام ا ست. در نهایت از چه سنسورهایی ا ستفاده خواهد شد و یا این فناوری کاملا خود مختار خواهد بود یا هنوز هم به انسان وابستگی خواهد داشت مشخص نیست اما می توان شاهد رقابت بسیار جدی خودروسازان و شرکت های تکنولوژی در این سال ها بود. و این نویدی است که در آینده ای نزدیک شاهد شکلی از این فناوری و خودروهای خود هدایت شونده در خیابان ها و جاده های عمومی و در دسترس عموم خواهیم بود.

---

[1] Simultaneous Localization And Mapping

۱-۲- ساختار پایان نامه

در فصل اول توضیحاتی خلاصه در مورد موضوع پروژه و پایان نامه داده شده است. تا خواننده با موضوع کلی طرح آشنا شده و ارتباط بهتری برقرار کند. در فصل دوم پروژه های موجود در این زمینه که دارای ویژگی های متفاوتی بوده اند معرفی شده و سپس جهت آشنایی اطلاعاتی در مورد ابزار و شبیه سازهای مورد استفاده در این پروژه آورده شده است. در فصل سوم، الگوریتم های موجود در ناوبری خودمختار و مورد استفاده در این پروژه شرح داده شده اند. در فصل چهارم روند پیاده سازی و طراحی ربات به طور کامل آورده شده و نتایج تست و ارزیابی ربات نیز در فصل پنجم به طور کامل نوشته شده است.

فصل دوم

پیشینه مطالعات

۲-۱- مقدمه

پروژه ها و تلاش های زیادی صـورت گرفته تا ماشـین ها بتوانند بدون دخالت انسـان و با سـطوحی از هو شمندی و خودمختاری مـسیر را بدون برخورد و تـصادف و با امنیت و سرعت بالا طی کنند. در ادامه ابتدا برخی از این تلاش ها معرفی شده و سپس بعضی از ابزار و شبیه سازهای موجود و مورد استفاده در این پروژه بررسی و معرفی خواهند شد.

۲-۲- پروژه ها و تلاش های مشابه

در این بخش، سعی شده است بیشتر پروژه های متفاوت و موجود در مورد خودروهای بدون سرنشین و خود هدایت شونده[۱] همراه با هدف و معرفی تیم تحقیقاتی در سطح جهان معرفی گردد.

۲-۲-۱- گوگل

گوگل بعد از مسـابقه ی GrandChallenge2005 که توسـط سـازمان پروژه‌های تحقیقاتی پیشـرفته ی دفاعی ارتش آمریکا (دارپا)[۲] برگزار شـد، بهترین دانشـمندان و محققین در حوزه ی ناوبری خودمختار را گرد هم آورد. [1] براساس گزارش ماهانه[2] پروژه ی ماشین خود هدایت شونده گوگل[۳]، گرچه این پروژه پیشرفت های بسیار زیاد و بزرگی داشته است اما هنوز برای نهایی سازی و فروش این فناوری به عموم آماده نیست. گوگل در این گزارش آورده است که در شش سال گذشته پیشرفت های زیادی در این فناوری داشته است و هنوز در حال یادگیری است. هر روز در خیابان های عمومی، نرم افزار گوگل در برابر مسائل مختلف قرار می

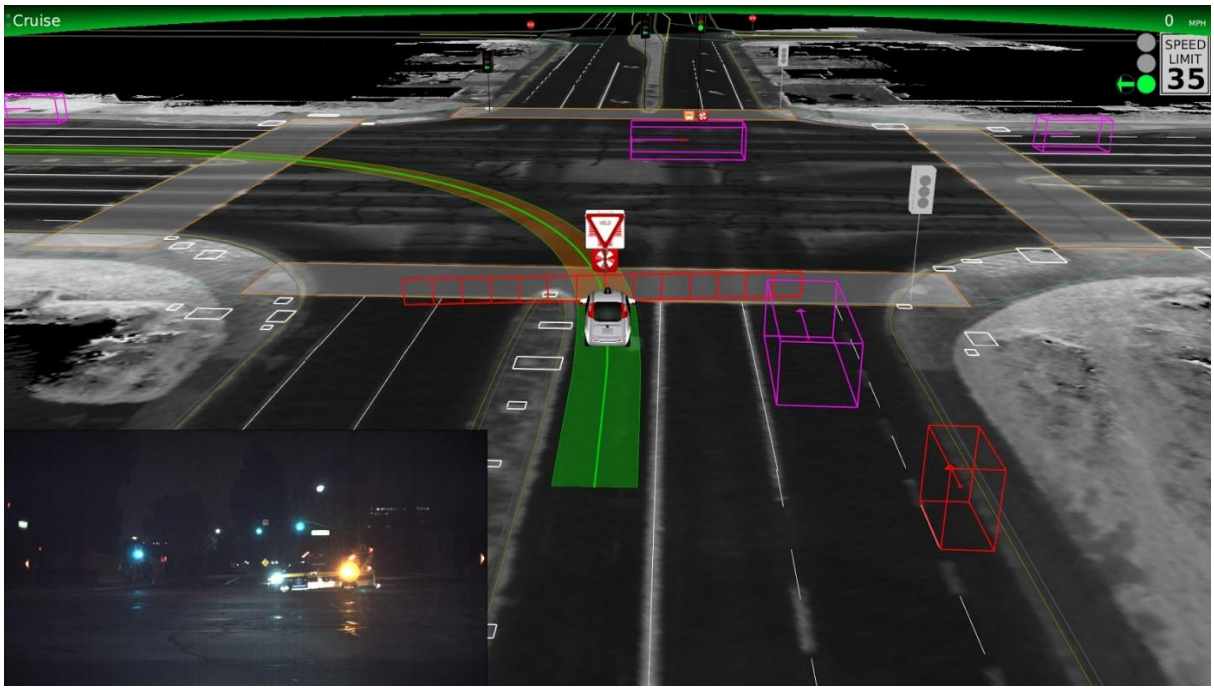
---

[۱] Self-driving Vehicle

[۲] Defense Advanced Research Projects Agency (DARPA)

[۳] Google

گیرد و پالایش های لازم صورت می گیرد. گوگل دارای ۲۳ ماشین از نوع Lexus RX450h می باشد که هم اکنون در جاده ها و خیابان ها همراه با سایر ماشین ها و انسان ها در ایالت کالیفرنیا در تعامل و رفت و آمد هستند. ۹ نمونه ی دیگر نیز از این ماشین های خود هدایت شونده، در مرحله ی تست و ارزیابی قرار دارد. از سال ۲۰۰۹ و شروع پروژه، تا تاریخ June 3, 2015 این ماشین ها توانسته اند ۱۰۱۱۳۳۸ مایل (کیلومتر) بدون دخالت راننده جایگزین و ۷۹۶۲۵۰ مایل[۱] (کیلومتر) در حالت دستی و با هدایت راننده طی کنند. نکته ی جالب موجود در این گزارش در تصاویر زیر آمده است. ماشین گوگل حتی نسبت به ماشین هایی نظیر آمبولانس و ماشین آتش نشانی و ... رفتار مناسبی نشان می دهد. همانطور که در شکل ۱-۱ مشاهده می کنید، چراغ راهنمایی سبز شده است اما ماشین گوگل به دلیل هم جهت شدن با مسیر ماشین آمبولانس که از سمت چپ به راست تصویر در حرکت است، در جای خود متوقف شده است. ماشین آمبولانس در این شکل با دو چراغ قرمز بر بالای مکعب مستیل بنفش رنگ، روبروی ماشین گوگل مشخص است.

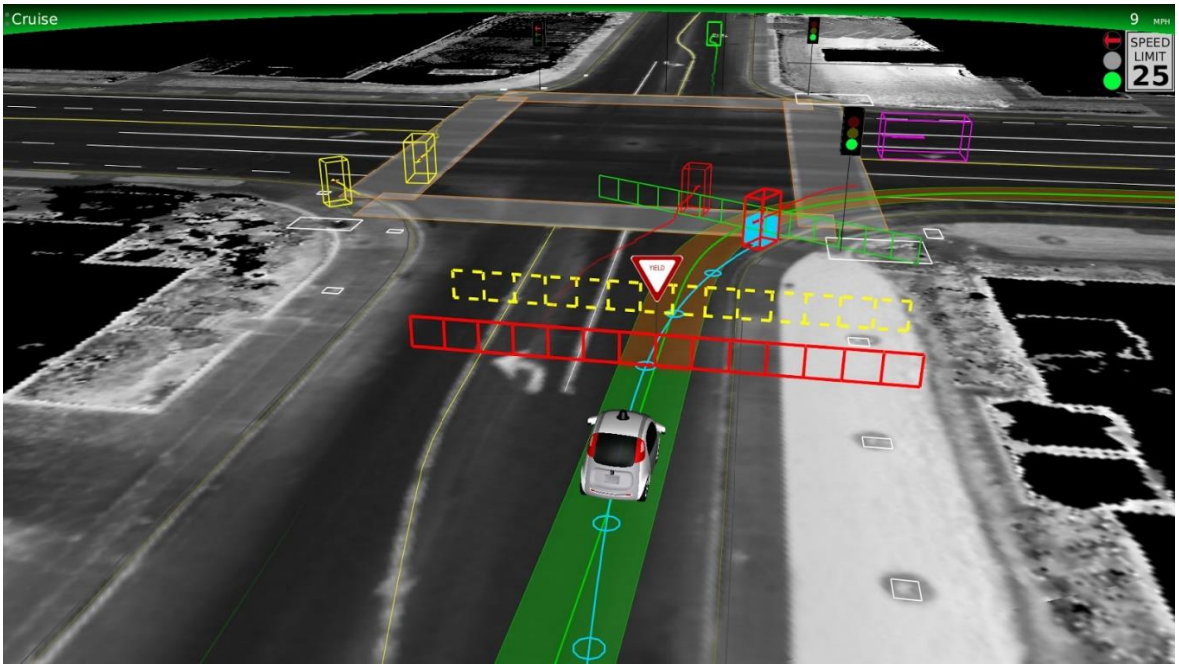


شکل (۲-۱) ماشین گوگل در خیابان ها و جاده های عمومی

به راحتی می توان در تصویر زیر مشاهده کرد که ماشین گوگل تا چه میزان هوشمند شده است. این

[۱] Miles

ماشین می تواند چندین شی متحرک را حتی در شب در لحظه دنبال کند[1]. مکعب های قرمز کوچک در برابر ماشین گوگل دوچرخه سوار هستند که مسیر طی شده توسط آن ها در تصویر با رنگ قرمز مشخص است. دوچرخه سوار سمت چپ ابتدا به خطوط چرخش به چپ[2] رفته است اما تغییر مسیر داده و در مسیر ماشین گوگل قرار گرفته و قصد عبور از تقاطع را دارد. در همان لحظه، دوچرخه سوار سمت راست، برخلاف جهت ترافیک و از سمت راست وارد تقاطع می شود و به صورت ناگهانی به سمت چپ می چرخد و به سمت ماشین گوگل رفته است. ماشین گوگل به خوبی می تواند در چنین شرایطی قرار گرفته و واکنش مناسب نشان دهد.



شکل (۲-۲) ماشین گوگل – تعامل با چندین شی ثابت و متحرک

نکته ی جالب در اینجاست که این اتفاقات در شب حتی برای انسان نیز خیلی سخت و پیچیده است. اما ماشین گوگل به خوبی در چنین شرایطی واکنش مناسب نشان داده است. براساس این گزارش، هزاران تصادف هر روز در خیابان های آمریکا رخ می دهد که عامل ۹۴ درصد این تصادفات، اشتباه و خطای انسانی است و بیشتر از ۵۵ درصد این تصادفات اصلا گزارش نمی شود. تا کنون ماشین گوگل در ۱۲ تصادف کوچک

[1] Track

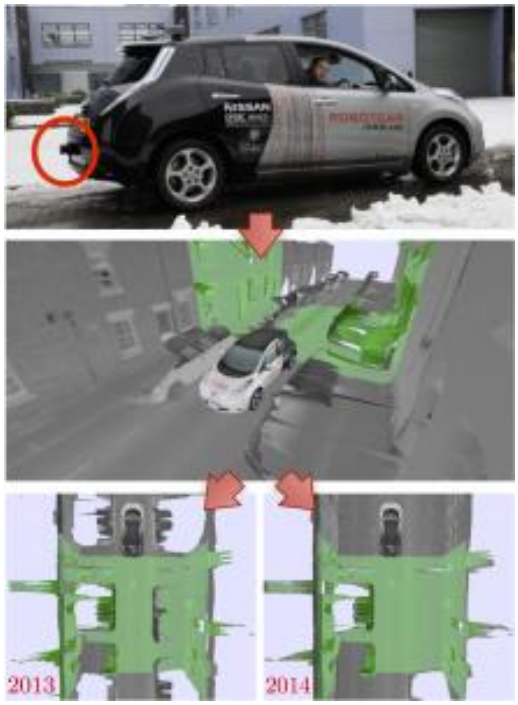
[2] The Left Turn Lane

در طی بیش از ۱٫۸ مایل (کیلومتر) در حالت دستی و خودمختار نقش داشته است. براساس این گزارش عامل هیچکدام از این تصادفات، ماشین خود هدایت شونده نبوده است. اکثر این تصادفات توسط ماشین های انسانی و برخورد از پشت به ماشین گوگل اتفاق افتاده است.

۲-۲-۲- تیم تحقیقاتی دانشگاه اکسفورد

ماشین خود هدایت شونده ی ROBOTCAR توسط گروه تحقیقاتی MRG دانشگاه اکسفورد[۱] طراحی شده است. برتری این ماشین در قیمت ارزان تر این فناوری نسبت به سایر تکنولوژی های موجود است. این تیم در طراحی این ماشین از نحوه ی رانندگی و زاویه ی دید انسان ها الهام گرفته و توانسته است با محدود کردن زاویه ی دید ماشین، فناوری ارزان تری را طراحی کند.

از سال ۲۰۰۴ تا کنون، تلاش های زیادی برای طراحی و ساخت خودروهای خود هدایت شونده شده است. موفق ترین روشی که تا قبل از طرح نوآورانه ی تیم تحقیقاتی آکسفورد به کار گرفته می شد و شاید به دلایلی بعدا هم استفاده شود، استفاده از ترکیب اسکنر LIDAR سه بعدی، سیستم GPS و INS بسیار دقیق برای مکان یابی با دقت سانتی متر و تهیه ی نقشه از محیط بوده است.[3] که علاوه بر هزینه ی بسیار گران سنسورهای مورد استفاده، هزینه ی پردازشی زیادی هم مورد نیاز است. روش پیشنهادی این گروه از آن جهت مورد توجه است که با ترکیب داده های به دست آمده از سیستم Odometry و سنسور اسکنر لیزر دو بعدی ثابت، توانسته اند یکی از ملزومات بسیار گران قیمت و ضروری یک سیستم خودمختار را حذف کنند. به عبارت دیگر این روش را به عنوان جایگزینی برای سنسور LIDAR سه بعدی پیشنهاد داده اند.

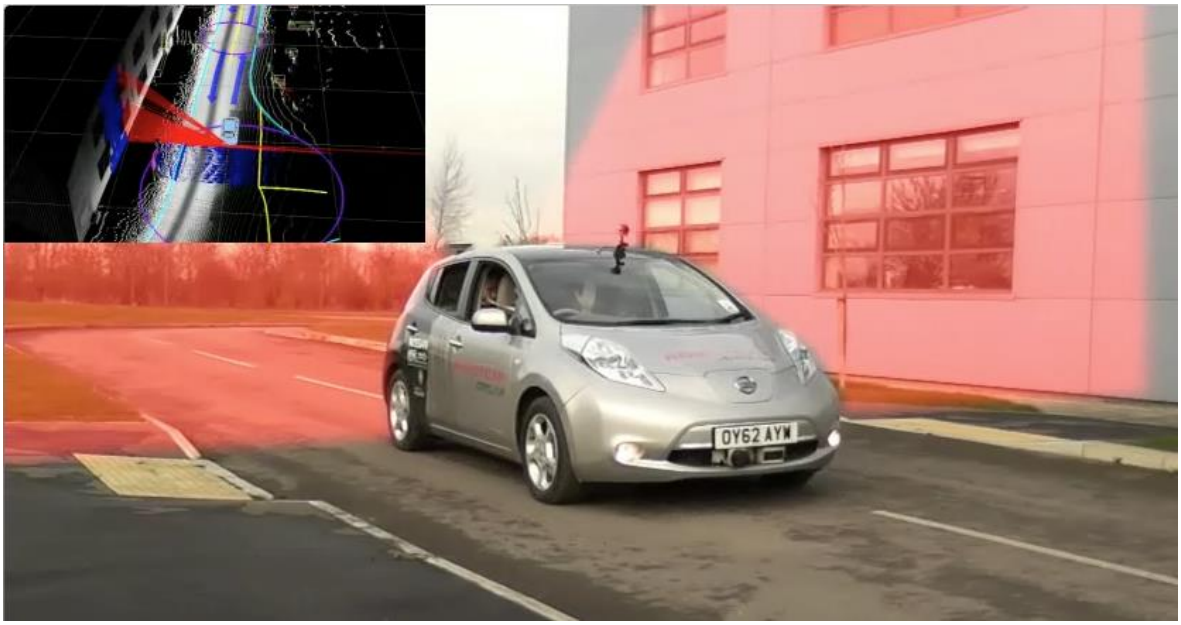


شکل (۲-۳) ماشین گروه تحقیقاتی دانشگاه

[۱] Oxford Mobile Robotics Group

اکسفورد + سیستم نقشه برداری و محل یابی

در اینجا منظور از Odometry برآورد تقریبی از تغییر موقعیت ربات در طول زمان با استفاده از داده های سنسورها می باشد. در این روش پیشنهادی، ماشین پس از طی هر مسیر تجربه ای جدید را کسب می کند و بر حسب تطابق با اطلاعات و تجربه های پیشین، از دقت داده ها اطمینان حاصل می کند. در صورتی که تمام ملزومات هدایت خودمختار در سطح مطلوب باشد، می توان این حالت را فعال کرد. [3]



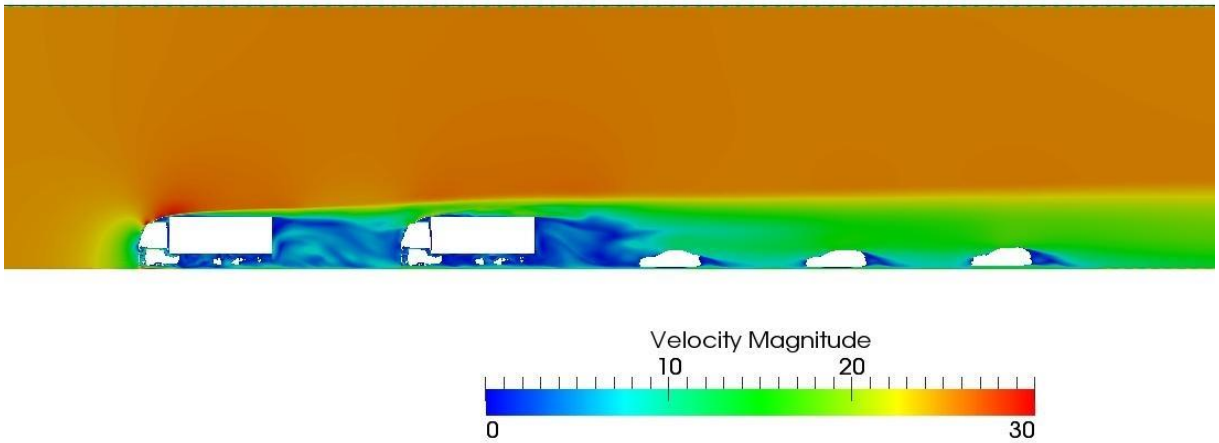
شکل (۲-۴) ماشین گروه تحقیقاتی دانشگاه اکسفورد + سیستم نقشه برداری و محل یابی

۲-۲-۳- پروژه ی SARTRE و Drive Me

برای رسیدن به فناوری ناوبری خودمختار نباید فقط به چارچوب های محدود ذکر شده فکر کرد. قطعا آینده شکل نهایی این تکنولوژی را به همگان نشان خواهد داد. یکی دیگر از پروژه ها و فناوری های جالب در این زمینه توسط اتحادیه ی اروپا و با همکاری شرکت Volvo حمایت شده است. شرکت های زیادی در این پروژه شرکت دارند از جمله Ricardo UK Ltd، Idiada، شرکت ا سپانیایی Robotiker-Tecnalia، شرکت آلمانی Institut for Kraftfahrwesen Aachen (IKA)، SP Technical Research Institute از سوئد و همچنین شرکت Volvo که با هدف تغییر در خطوط حمل و نقل و ساخت گروه ها و ریل های ماشین های

خود هدایت شونده در جاده های واقعی با هم مشارکت و همکاری کرده اند. اصل این پروژه در سال ۲۰۰۹ شروع شده و در اواخر سال ۲۰۱۲ با موفقیت به اتمام رسیده است.[4] البته تا کنون به دلایل مختلف این پروژه برای پیاده سازی در سطح وسیع و ارائه ی این فناوری به عموم عرضه نشده است.

این پروژه که با نام SARTRE[۱] می باشد و یا به فارسی قطار یا ریل جاده های امن برای محیط زیست ترجمه می شود، ایده ی ساده و جالبی را برای ورود فناوری ماشین های بدون سرنشین در جاده های عمومی به کار گرفته است. اصل ایده ی این پروژه به سادگی داشتن گروهی از ماشین ها است که با هم سفر می کنند و دارای یک ماشین رهبر با یک راننده ی حرفه ای هستند. با این ایده سایر ماشین ها با فاصله ی بسیار کم تر و مصرف سوخت پایین تر می توانند به صورت کاملا خودمختار هدایت شوند. مصرف سوخت، به دلیل کاهش نیروی پسا[۲] بین ۷ تا ۱۵ درصد کاهش می یابد.[5] تصویر زیر به خوبی تاثیر این روند در کاهش نیروی پسا در سایر ماشین های خود هدایت شونده را نشان می دهد.



شکل (۲-۵) شبیه سازی آیرودینامیکی دینامیک سیالات محاسباتی (CFD) پروژه SARTRE [5]

این طرح در اواخر سال ۲۰۱۲ در یکی از جاده های عمومی اسپانیا با به کار گیری سه ماشین مختلف از Volvo و دو کامیون، مورد ارزیابی قرار گرفت. ماشین اول به عنوان رهبر و توسط یک راننده ی حرفه ای هدایت شده است. دیگر ماشین ها به صورت کاملا خودمختار و بدون دخالت راننده، مسیر مورد نظر را می

[۱] Safe Road Trains for the Environment

[۲] Drag

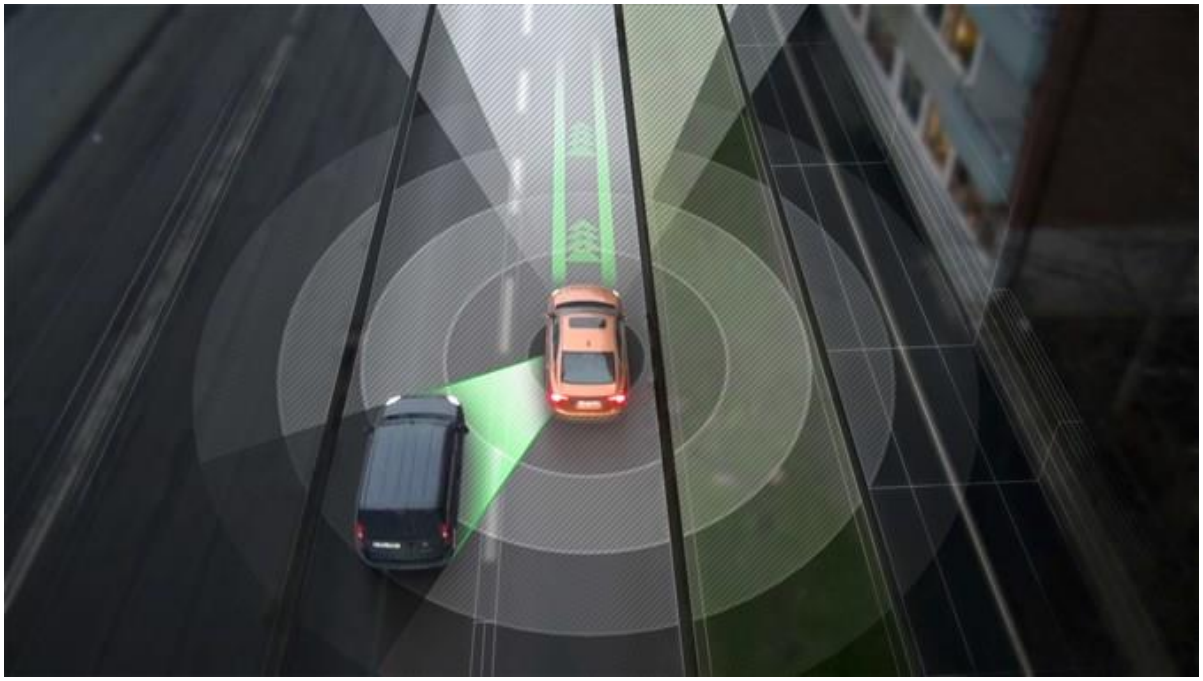
پیمایند. این پروژه با فرض اینکه در جاده هیچ تغییری ایجاد نشود طراحی شده و فقط از فناوری های موجود در ماشین ها استفاده کرده است. از دشواری های طراحی این سیستم می توان از مدیریت ایجاد یک ریل از ماشین ها، اضافه یا جدا شدن ماشین ها از ریل، نگهداری و حفظ ریل، مدیریت ایمنی و جلوگیری از تصادف، هدایت و مسیریابی و همچنین نحوه ی مدل تجاری و توجیه هزینه های پیاده سازی و نگهداری هر ریل نام برد. [5] در حقیقت، هر ماشین که از این فناوری برخوردار است می تواند در جاده با ریل های موجود هم مسیر شده و بدون دخالت انسان به مقصد مورد نظر هدایت شود. اگر راننده به هر دلیل تصمیم به جدا شدن از ریل و رفتن به مقصد دیگری کند به راحتی می تواند کنترل ماشین را در اختیار گرفته و از ریل جدا شود.

به عنوان مثال راننده قصد دارد در رستوران موجود در مسیر برود و بنابراین فقط کافی است از ریل جدا شده و ماشین را به مقصد مورد نظر خود هدایت کند و یا حتی زمانی که به مقصد خود می رسد باید از ریل جدا شود. این فناوری نیز از مزایا و معایب خاص خود برخوردار است ولیکن ایده ی بسیار جالبی است.

شکل (۲-۶) ارزیابی اولیه ی پروژه SARTRE در محیط واقعی و آزمایشی

گر چه نمی توان نام این فناوری را ما شین های کاملا خودمختار نامید اما رو شی جالب و نو آورانه به کار گرفته شده است که انسان بتواند بدون رانندگی با ماشین خودش سفر کند. از آن جهت سطح هوشمندی و خودمختاری این پروژه پایین محسوب می شود که ما شین در ا صل یک پیرو ا ست و خود توانایی تصمیم گیری ندارد. شاید با ترکیب ایده ی این پروژه و سایر پروژه های موجود بتوان روشی جدید برای هدایت ما شین های خود هدایت شونده ایجاد کرد. زیرا انسان ها هم، در ا صل به هنگام رانندگی گاهی اوقات پیرو راننده ی دیگری هستند. به هر حال نتایج این پروژه نشان می دهد، به کار گیری این فناوری باعث کاهش مصرف سوخت، افزایش ایمنی و افزاش آسایش و راحتی راننده می شود. [5] علاوه بر آن یکی از مزایای این فناوری استفاده از انواع ماشین های سبک و سنگین است که تا قبل از این همه ی پروژه ها فقط از ماشین های خاصی استفاده می کردند.

در این اواخر یعنی در سال ۲۰۱۵ شرکت Volvo فناوری دیگری را با نام Drive Me معرفی کرده و قرار است تا سال ۲۰۱۷ در مرحله ی اول ۱۰۰ ماشین خودهدایت شونده از این نوع را در اختیار مشتریان برای عرضه و حرکت در جاده های مشخصی از گوتنبرگ قرار دهد. [6]

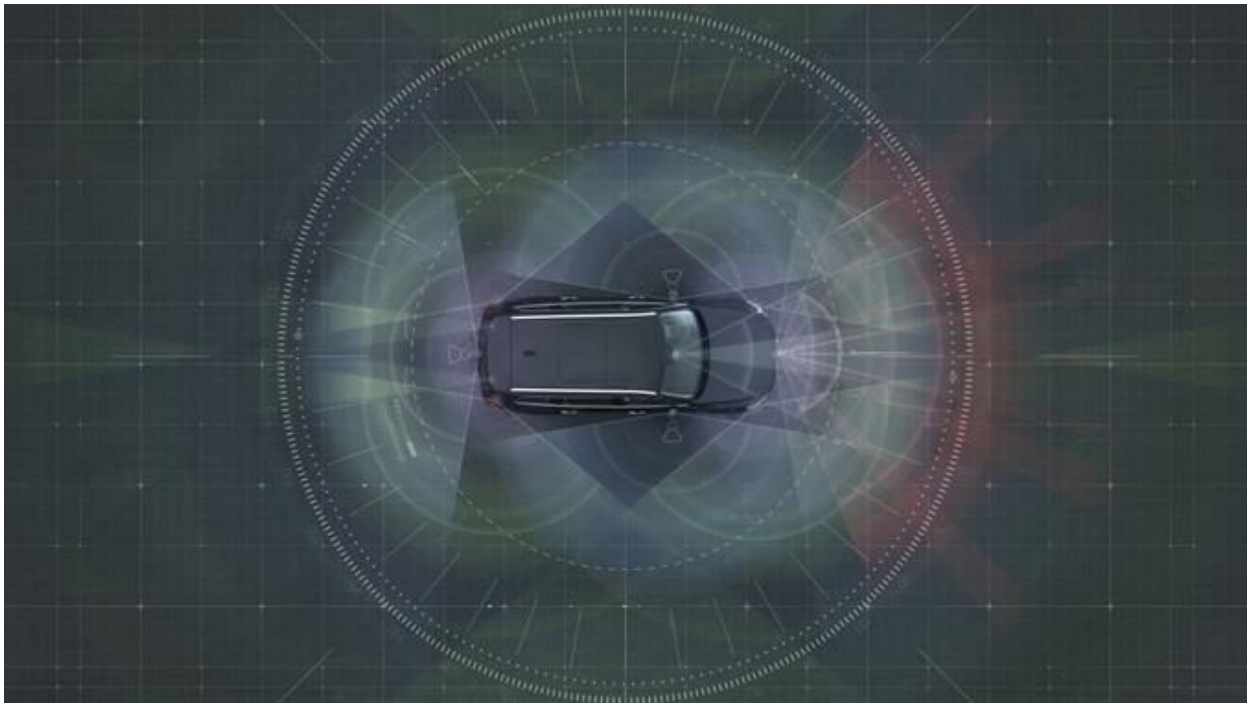


شکل (۲-۷) پروژه ی Drive Me شرکت Volvo [7]

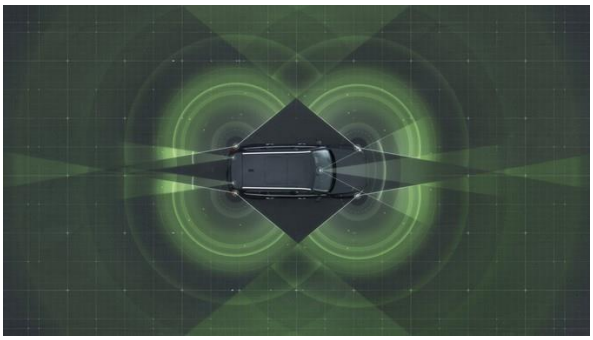
تفاوت این فناوری در سنسورهای به کار گرفته شده و نحوه ی تعامل با اطلاعات به دست آمده از این سنسورهاست. در حال حاضر به نظر می رسد قیمت این فناوری بسیار ارزان تر و قابل به کارگیرتر از فناوری شرکت گوگل برای عموم باشد. اما واقعا به طور کامل نمی توان این پروژه را با پروژه ی گوگل مقایسه کرد. به طور کلی شرکت Volvo به دنبال یک فناوری است که بتواند در سال ۲۰۲۰ از مرگ انسان ها به دلیل تصادف و در محصولات و ماشین های تولیدی خود جلوگیری کند و شاید فعلا ناوبری خودمختار تنها فناوری می باشد که می تواند در آینده ای نه چندان دور به این هدف واقعیت بخشد.

همانطور که می توان در تصاویر زیر مشاهده کرد، از سنسورهای بسیار زیاد و متنوعی استفاده شده است تا بتوان تمام زوایای ۳۶۰ درجه ی اطراف ما شین را پو شش داد. این فناوری با ترکیب اطلاعات دوربین ها و رادارهای با فرکانس ۷۶ گیگاهرتز به کار گرفته شده، اشیا و جسم های اطراف ماشین را تشخیص می دهد. همچنین از دوربین ها برای تشخیص خطوط و علائم جاده ها نیز استفاده می شود. اسکنر لیزری به کار گرفته شده نیز می تواند با زاویه ی دید ۱۴۰ درجه، ۱۵۰ متر را پوشش دهد. از سنسورهای UltraSonic نیز

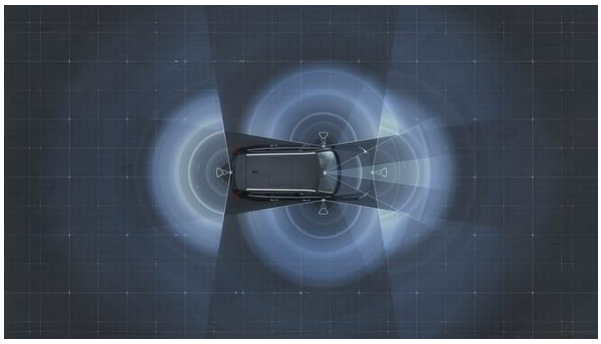
برای تشخیص اجسام بسیار نزدیک به ماشین و یا افراد و همچنین در شرایط خطر استفاده می شود. این سنسورها برای هدایت خودمختار ماشین در سرعت های بسیار پایین نیز به کار برده می شوند. به عنوان مثال برای پارک و یا در جاده های پر ترافیک و شلوغ که حرکت به سختی است.



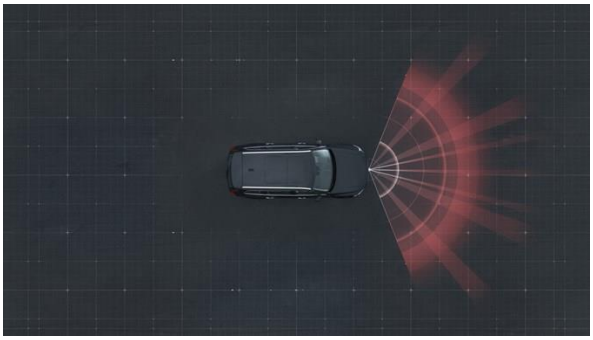
شکل (۲-۸) دید ۳۶۰ درجه ی مجموع سنسورها در پروژه ی Drive Me [8]



شکل (۲-۹) رادارها در پروژه ی Drvie Me [9]



شکل (۲-۱۰) دوربین ها در پروژه ی Drive Me [10]



شکل (۲-۱۱) لیزر چندپرتویی در Drive Me [11]

شکل (۲-۱۲) سنسورهای UltraSonic در Drive Me [11]

۲-۳- ابزار و شبیه سازها

در این بخش، ابزار و شبیه سازهای موجود و مورد استفاده در این پروژه و ناوبری خودمختار معرفی و بررسی می شود.

۲-۳-۱- سیستم عامل ربات ROS

سیستم عامل ربات (ROS[۱]) یک چارچوب[۲] انعطاف پذیر برای نوشتن نرم افزار ربات ها می باشد. از این پس جهت سادگی از عبارت سیستم عامل رز به جای سیستم عامل ربات و یا ROS استفاده خواهد شد. این سیستم عامل مجموعه ای از ابزار، کتابخانه ها و سایر ابزار مفید را برای ساده کردن توسعه ی نرم افزارهای با رفتار پیچیده و مقاوم برای گستره ای متنوع از پلتفرم های مختلف ربات فراهم کرده است. [1] یا

[۱] Robot Operating System

[۲] Framework

به عبارتی مجموعه ای از چارچوب نرم افزارها برای توســعه ی نرم افزاری ربات ها می باشــد که همانند یک سیستم عامل بر روی رایانه ی خوشه ای[1] ناهمگون، عمل کرده و یک میان افزار[2] قوی برای ربات ها است. سیستم عامل رز تمام سرویس های ارائه شده توسط یک سیستم عامل شامل لایه انتزاعی سخت افزار[3]، کنترل سطح پایین دستگاه، پیاده سازی توابع رایج و مشترک، انتقال پیام های بین فرآیندی و مدیریت بسته ها[4] و ... را ارائه می دهد. این سیستم عامل از چندین روش مختلف ارتباطی پشتیبانی می کند. می توان از syn-chronous RPC-style بر روی سرویس ها، asynchronous streaming روی تاپیک ها و ذخیره ی داده ها بر روی پارامترهای سروری نام برد. [13]

طراحی و ساخت یک ربات مطمئن و با اهداف عمومی آسان نیست. به خصوص اینکه اغلب کارهایی که برای انسان بسیار ساده هستند اما ربات باید از فرآیند پیچیده ای برای حل مسئله استفاده کند. این سیستم عامل با هدف گسترش توسعه ی مشارکتی نرم افزارهای رباتیک طراحی و ساخته شده است و می توان گفت پتانسیل و توانایی گروه ها، آزمایشگاه ها و افراد مختلف در این سیستم عامل به صورت یک جا جمع آوری شده است. شاید یکی از ویژگی های منحصر به فرد این سیستم عامل توانایی توسعه ی یک نرم افزار برای محیط شبیه ساز و محیط واقعی با اندکی تغییرات است که بیشتر این فرآیند را کنترل کرده و کار توسعه را بسیار آسان می کند.

شکل (۲-۱۳) سیستم عامل رز در یک نگاه

---

[1] Computer Cluster : رایانش خوشه‌ای یا کلاستر نوعی از سیستم‌های پردازش موازی و توزیع شده است که متشکل از مجموعه‌ای از کامپیوترهای مستقل بوده که تمام این گره‌ها در ارتباط تنگاتنگ با کل سیستم هستند و به عنوان یک منبع یکپارچه کار می‌کنند.

[2] Middleware

[3] Hardware Abstraction

[4] Package Management

رز یک سیستم عامل متن باز است که در پایین ترین سطح، یک واسط برای ارسال و دریافت پیام بین فرآیندی[1] دارد که این میان افزار سیستم عامل رز امکانات زیر را ارائه می کند:

- ارسال و دریافت پیام ناشناس عضویت/انتشار[2] (Topic)
- ذخیره و بازپخش پیام ها [3] (rosbag)
- فراخوانی های رویه ای کنترل از راه دور درخواست/پاسخ[4] (Services)
- سیستم پارامتری توزیع یافته[5] (Parameter Server)

علاوه بر این سیستم عامل رز، کتابخانه ها و ابزار مخصوص رباتیک را فراهم نموده است که به سرعت به توسعه دهنده اجازه می دهد یک ربات را راه اندازی و پیاده کند. در ادامه فقط به بخشی از این توانایی ها اشاره می شود:

- تعاریف پیام های رایج برای ربات ها
- کتابخانه هندسه ربات (Robot Geometry Library)
- زبان توصیف ربات (RDL)
- فراخوان های رویه ای کنترل از راه دور قابل انحصار[6]
- عیب یابی
- تخمین وضعیت
- مکان یابی [7]
- نقشه برداری [8]
- ناوبری

یکی از مهمترین ویژگی های سیستم عامل رز، ارائه ی ابزار قدرتمند برای توسعه است. این ابزار از خطایابی[9]، نمایش وضعیت سیستم، introspecting و plotting پشتیبانی می کنند. این سیستم عامل فوق

---

[1] Process

[2] Publish/Subscribe anonymous message passing

[3] Recording and playback of messages

[4] Request/Response remote procedure calls

[5] Distributed parameter system

[6] Preempt able Remote Procedure Calls

[7] Localization

[8] Mapping

[9] Debugging

العاده قدرتمند بوده و به لطف ویژگی متن باز بودن به سرعت رشد کرده و هر روز توانایی های جدید و قدرتمند تری به آن اضافه خواهد شد.

سطوح مختلف سیستم فایل[1] در سیستم عامل رز در ادامه با توضیحات آورده شده است: [14]

- Packages: پکیج ها واحد اصلی برای ساماندهی نرم افزارها در سیستم عامل رز هستند. این پکیج ها می توانند شامل ROS runtime processes و یا به عبارتی همان گره ها باشند. همچنین کتابخانه های وابسته، مجموعه داده ها یا Dataset ها، فایل های حاوی تنظیمات و یا هر چیز دیگه ای که به طور مفید در کنار هم ساماندهی شده باشد.
- Metapackages: متا پکیج ها، پکیج های خاصی هستند که مجموعه ای از پکیج های به هم مرتبط را به صورت یک جا ارائه می دهند.
- Package Manifests: هر پکیج دارای یک مانیفست با نام package.xml است که شامل فراداده ها[2] یا همان متادیتا در رابطه با آن پکیج می باشد. به عنوان مثال هر مانیفست شامل نام پکیج، ورژن، توضیحات، اطلاعاتی در مورد پروانه و License به کار گرفته شده، پکیج های وابسته و سایر اطلاعات و داده ها در مورد آن پکیج می باشد.
- Repositories: مخازن، مجموعه ای از پکیج ها هستند که دارای سیستم VCS مشترکی باشند. در حقیقت پکیج هایی که چنین ویژگی ای داشته باشند، دارای ورژن مشابهی هستند و می توانند توسط ابزار اتوماسیون ارائه[3] در Catkin و با نام Bloom با هم ارائه[4] شوند. البته این مخازن می توانند فقط شامل یک پکیج هم باشند.
- Message (msg) types: توصیف پیام ها و تعریف ساختار داده های جدید برای ارسال پیام در سیستم عامل رز، در فایل/msg/MyMessageType.msg در هر پکیج باید مشخص شود.
- Service (srv) types: توصیف سرویس ها و تعریف داده ساختارهای درخواست[5] و پاسخ[6] برای سرویس ها باید در فایل /srv/MyServiceType.srv برای هر پکیج مشخص گردد.

---

[1] SystemFile Level

[2] MetaData

[3] Release Automation Tool

[4] Release

[5] Request

[6] Response

سطوح گراف محاسباتی در سیستم عامل رز، یک شبکه ی نقطه به نقطه (P2P)[1] از فرآیندهای سیستم عامل رز می باشد که با هم بر روی داده ها پردازش می کنند. مفاهیم پایه در گراف محاسباتی رز nodes, Master, Parameter Server, messages, services, topics و bags هستند که هر کدام به روش متفاوتی داده ها را برای گراف مهیا می کنند. [14]

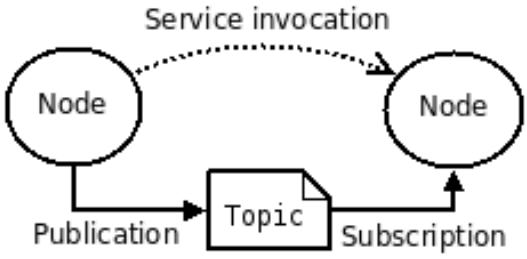
- Nodes: گره ها فرآیندی هستند که محاسباتی را انجام می دهند. سیستم عامل رز طراحی شده است تا در سطح بالایی از طراحی پودمانی و یا پیمانه ای[2] پشتیبانی کند. یک سیستم کنترل ربات معمولا از تعداد زیادی گره استفاده می کند. به عنوان مثال یک گره اسکنر لیزر را کنترل کرده، یکی دیگر هدایت و کنترل چرخ ها را به عهده دارد، یکی دیگر دوربین، یکی دیگر محل یابی و ... را و هر کدام وظیفه ای به عهده دارند. در مجموع با ترکیب این گره ها ربات می تواند وظایف بسیار پیچیده ای را با موفقیت انجام دهد. این ویژگی بسیار خوب به توسعه دهندگان کمک می کند تا از موازی کاری و اختراع مجدد چرخ خودداری کنند.
- Master: مستر در سیستم عامل رز قابلیت ثبت نام و جستجو در سایر گراف محاسبات را فراهم می آورد. بدون مستر، گره ها قادر نیستند تا همدیگر را پیدا کرده و یا حتی پیامی برای هم ارسال کنند.
- Parameter Server: سرور پارامترها در حقیقت جزئی از مستر است که اجازه می دهد داده ها به صورت کلید در یک مکان مرکزی ذخیره شوند و دسترسی به این کلید ها را آسان تر می کند.
- Messages: گره ها با هم از طریق ارسال پیام ها ارتباط برقرار می کنند. هر پیام یک داده ساختار است که می تواند حاوی فیلدهای مختلفی باشد. سیستم عامل رز همانند C، از ساختارها، آرایه ها و ساختارهای تو در تو، انواع استاندارد نوع داده ها نظیر integer floating point, Boolean و ... پشتیبانی می کند.
- Topics: تاپیک ها کانالی بین دو یا چند گره هستند. که هر گره می تواند در هر تاپیک عضو شود و یا تاپیکی منتشر کند. در یک تاپیک می تواند چندین منتشر کننده و چندین

[1] Peer-to-Peer

[2] Modular Design

عضو به صورت همزمان وجود داشته باشد. و هر گره نیز می تواند در چندین تاپیک عضو باشد و حتی چندین تاپیک مختلف منتشر کند.

- Services: تاپیک ها جاده های یک طرفه ای هستند و با تمام مزایایی که دارند نمی توانند گفتگوی دو طرفه ای بین گره ها ایجاد کنند. در صورت نیاز به یک روش درخواست[1]/ پاسخ[2] در سیستم عامل رز از سرویس ها استفاده می شود.
- Bags: بگ یا کیسه ها در سیستم عامل رز فرمتی برای ذخیره و اجرای مجدد داده ها و پیام ها هستند که مکانیزم بسیار مهمی برای ذخیره ی داده ها محسوب می شوند. به عنوان مثال جمع آوری اطلاعات سنسورها ممکن است کار بسیار مشکلی باشد اما واقعا برای توسعه، تست و ارزیابی الگوریتم ها ضروری هستند. در این مواقع می توان از این روش برای ذخیره ی اطلاعات استفاده کرد.



شکل (۲-۱۴) مفاهیم پایه در سیستم عامل رز[15]

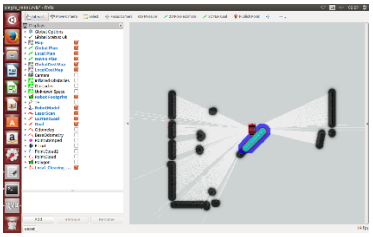
در این قسمت به صورت خیلی خلاصه در مورد برخی از بخش ها و قسمت های سیستم عامل رز توضیحاتی داده شد تا خواننده با مفاهیم به کار گرفته شده آشنا شود. توصیف و معرفی تمام بخش ها و قابلیت های این سیستم عامل از حوصله و موضوع این پایان نامه خارج است.

[1] Request

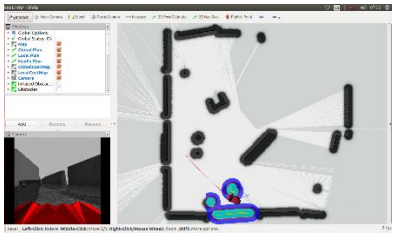
[2] Reply

۲-۳-۲- شبیه ساز Rviz

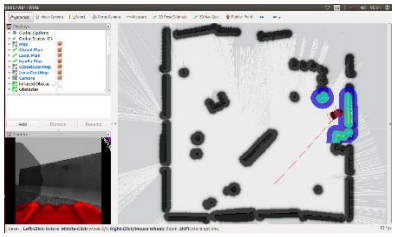
در سیستم عامل رز، Rviz یک محیط سه بعدی برای نمایش اطلاعات سنسورها و روبات ها است که همه چیز را از زاویه ی دید و نگاه ربات به توسعه دهنده و کاربر نشان می دهد.



شکل (۲-۱) محیط Rviz–ت۱



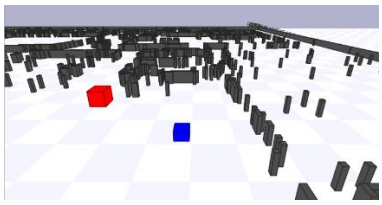
شکل (۲-۲) محیط Rviz–ت۲



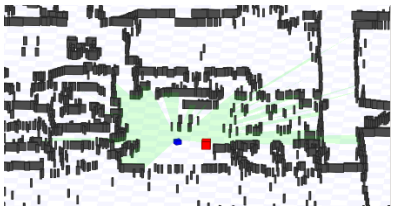
شکل (۲-۳) محیط Rviz–ت۳

۲-۳-۳- شبیه ساز Stage

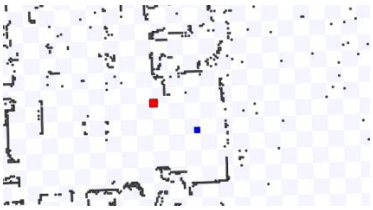
شبیه ساز *Stage* حرکت روبات های متحرک و درک این روبات ها از یک محیط دو بعدی (2D) را شبیه سازی می کند. *Stage* یک کتابخانه ی نرم افزاری C++ است که در این شبیه ساز، مدل سنسورهای مختلفی از جمله سونار، لیزر، دوربین و ... تهیه و آماده شده است و کاملا با سیستم عامل رز سازگار است و در بسیاری جهات بهینه می باشد و از سرعت بسیار خوبی برخوردار است. [2] [1]



شکل (۲-۴) محیط Stage–ت۱



شکل (۲-۵) محیط Stage–ت۲

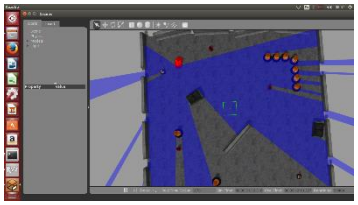


شکل (۲-۶) محیط Stage–ت۳

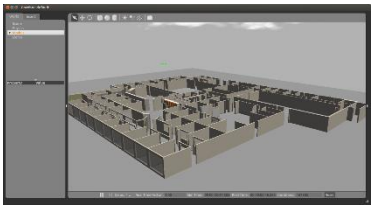
۲-۳-۴- شبیه ساز Gazebo

در این شبیه ساز که کاملا با سیستم عامل رز سازگار است می توان در آن هر رباتی را در محیط های

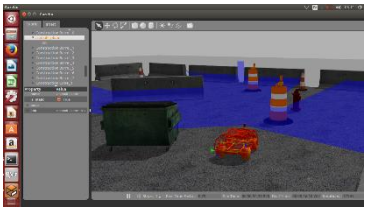
دلخواه شبیه سازی کرد. بر خلاف Stage این شبیه ساز، سنسورها و روبات ها را در یک محیط و جهان سه بعدی (3D) شبیه سازی می کند و از نیروی جاذبه و سایر مشخصه های فیزیکی و مکانیکی کاملا پشتیبانی می کند.



شکل (۲-۷) محیط Gazebo–ت۱



شکل (۲-۸) محیط Gazebo–ت۲



شکل (۲-۹) محیط Gazebo–ت۳

۲-۴- زبان های توصیفی و برنامه نویسی

در این بخش، زبان های توصیفی و برنامه نویسی مختلف که در این پروژه از آن ها استفاده شده است، معرفی می شوند و به صورت خلاصه به ویژگی ها و کاربردهای این زبان ها در رباتیک اشاره می شود.

۲-۴-۱- زبان نشانه گذاری URDF

فرمت متحد توصیف ربات یا *URDF* توسعه ای از زبان نشانه گذاری *XML* است که برای توصیف ربات ها استفاده می شود. این زبان فقط از ساختار درختی پشتیبانی می کند. همچنین ربات ها متشکل از مجموعه ای از لینک ها و اتصالات می باشند و از عناصر انعطاف پذیر پشتیبانی نمی کند. در این زبان می توان ویژگی های ظاهری، مشخصه های دینامیکی و حرکتی[۱]، و مدل برخورد و تصادف[۲] ربات را تشریح کرد.

جدول (۲-۱) مثالی ابتدایی از URDF

```
1:  <?xml version="1.0"?>
2:  <robot name="chugv">
```

[۱] Kinematic and dynamic

[۲] Collision model

```
3:    <link> ... </link>
4:    <link> ... </link>
5:    <link> ... </link>
6:
7:    <joint>  ....  </joint>
8:    <joint>  ....  </joint>
9:    <joint>  ....  </joint>
10:  </robot>
```

۲-۴-۲- زبان نشانه گذاری SDF

در حالی که *URDF* مفید بوده و یک فرمت استاندارد در سیستم عامل رز است اما فاقد تعداد زیادی قابلیت می باشد و برای حل نیازمندی های روباتیک نیز به روز رسانی نشده است. *URDF* فقط می تواند خود ربات را توصیف کند. در حالی که *SDF* علاوه بر ربات از توصیف محیطی نیز پشتیبانی می کند. البته باید گفت به راحتی قابلیت تبدیل از *URDF* به *SDF* نیز وجود دارد.*[18]*

SDF یک فرمت *XML*[۱] است که اشیاء و محیط را برای شبیه سازها توصیف می کند و در اصل به عنوان بخشی از شبیه ساز *Gazebo* توسعه پیدا کرده است. *SDF* به اندازه ی کافی پایدار، مقاوم و توسعه پذیر است و توانایی توصیف تمام جنبه های مختلف ربات از جمله اشیاء ایستا و پویا، زمین، روشنایی و حتی فیزیک را داراست. از این زبان برای توصیف انواع مختلف ربات به پیچیدگی ربات انسان نما و به سادگی ربات مسیریاب استفاده می شود. در ضمن *SDF* مزایا و ویژگی های متن باز بودن را نیز داراست و توسط *OSRF*[۲] نیز پشتیبانی می شود.

جدول (۲-۲) مثالی ابتدایی از SDF

```
1:  <?xml version='1.0'?>
2:   <sdf version='1.5'>
3:    <world name=CyberHandsWorld>
4:      ...
5:    </world>
6:   </sdf>
```

[۱] Extensible Markup Language

[۲] Open Source Robotics Foundation

۲-۵- نتیجه‌گیری

شرکت ها و پروژه های بسیار زیادی در زمینه ی توسعه ی ماشین های خودمختار و خود هدایت شونده وجود دارد و تلاش های زیادی هم صـــورت گرفته اســـت اما در این فصـــل فقط از پروژه هایی یاد شـــد که تکنولوژی به کار گرفته شـده در آن ها با هم تفاوت زیادی داشـته باشـد. باید به این نکته توجه کرد که تا به حال هیچ ماشین سرنشین دار و خودمختاری بدون نظارت انسان ها گزارش نشده است. گر چه پیشرفت های بزرگی در این زمینه صـورت گرفته اما هنوز هیچ شـرکتی ادعا نکرده اسـت که می تواند ماشـینی بسـازد که بدون هیچ نظارتی از جانب انسان ها در خیابان های عمومی رفت و آمد کند. شاید این دلیل خوبی باشد که با شروع به سرمایه گذاری در این فناوری عظیم، ایران نیز می تواند از چرخه ی این بازی پر سود، بین شرکت های تکنولوژی و فناوری بزرگ عقب نماند. دیر یا زود این فناوری به نتیجه خواهد رســـید و می توان گفت انقلابی دیگر، این بار در حمل و نقل صورت می گیرد و قطعا پیروز، کشور و یا مجموعه ای است که به این فناوری زودتر دست یابد. هنوز تا به نتیجه رسیدن این رویا باید مسیر بسیار طولانی و پر زحمتی را طی کرد.

فصل سوم

الگوریتم های ناوبری

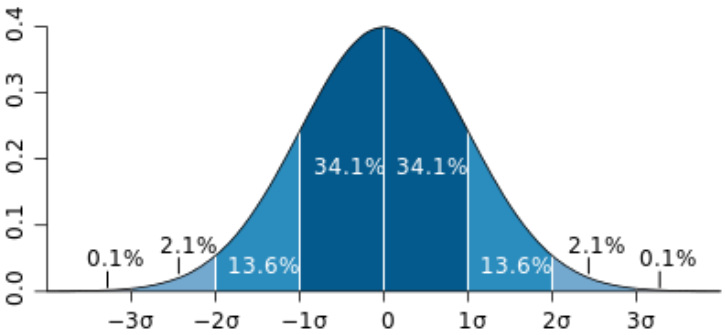
۳-۱- مقدمه

در این فصل الگوریتم های رایج در ناوبری خودمختار بررسی می شوند. تاکنون الگوریتم های زیادی به این منظور و با این هدف طراحی و پیاده سازی شده است که هر کدام مزایا و معایب خاص خود را دارند.

۳-۲- معرفی و بررسی الگوریتم های موجود

۳-۲-۱- پیش نیاز

شاید بتوان با قطعیت گفت که اساس الگوریتم های موجود در ناوبری همان توزیع نرمال گوسی است. گرچه هدف این فصل ورود به جنبه ی ریاضی و محاسباتی الگوریتم ها نیست اما جهت یادآوری به صورت خیلی مختصر توزیع گوسی شرح داده می شود. توزیع طبیعی یا گوسی، یک تابع پیوسته است که با بردار میانگین و ماتریس کواریانس آن توصیف می‌شود. توزیع نرمال استاندارد نیز توزیعی با میانگین صفر و ماتریس کواریانس واحد است. به عنوان مثال می توان بوسیله ی تابع توزیع احتمال، ویژگی های آماری مقدار خطا را نشان داد.



شکل (۳-۱) نمودار توزیع نرمال در دیاگرام انحراف معیار [19]

۳-۲-۲- فیلتر بیز و هیستوگرام

فیلتر هیســتوگرام فضــای حالات را به ناحیه های معین و محدودی تجزیه می کند و حالت بعد را با یک هیســتوگرام نمایش می دهد. این فیلتر مجموع حالت بعد[۱] برای هر ناحیه را با یک مقدار احتمال نمایش می دهد. این نوع فیلتر، در فضای گسسته با نام فیلتر بیز گسسته شناخته می شود و در فضای پیوسته به عنوان فیلتر هیستوگرام نامیده می شود. شبه کد الگوریتم بیز در فضای گسسته در ادامه آورده شده است. جهت اختصار از توضیح این شبه کدها خود داری می شود و پر واضح است که این شبه کدها به خوبی خود را توصیف می کنند و برای اطلاعات بیشتر باید به مراجع معتبر مراجعه شود.

جدول (۳-۱) شبه کد الگوریتم فیلتر بیز گسسته

Algorithm Discrete Bayes filter ({$p_{k,t-1}$}, u_t, z_t):

for all k do

$$\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$$

$$p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$$

endfor

return {$p_{k,t}$}

۳-۲-۳- فیلتر کالمن

فیلتر کالمن یک تقریب زن بازگشتی بهینه است که پارامترهای مورد نظر را به صورت غیر مستقیم و از مشاهدات غیر دقیق و غیر قطعی نتیجه می گیرد. این فیلتر از آن جهت بهینه است که اگر همه ی نویز[۲] ها از نوع گوسی باشد خطای میانگین مربع[۳] را برای پارامتر مورد نظر به حداقل می رساند. [19] فیلتر کالمن نیز یک فیلتر پیش بینی - تصحیح[۴] است یعنی بعد از هر پیش بینی، نتیجه ی حاصل را تصحیح می کند. [20] فیلتر کالمن را می توان به صورت زیر خلاصه کرد:

---

[۱] Cumulative posterior

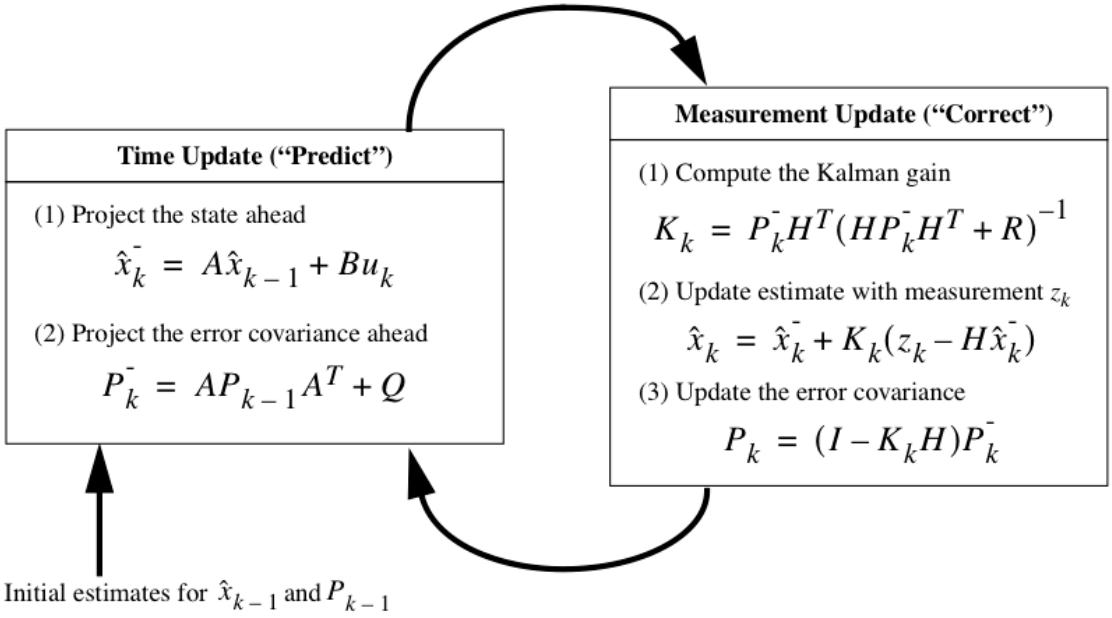
[۲] Noise

[۳] Mean Square Error

[۴] Prediction - Correction

- تخمین وضعیت با توجه به حالت قبل
- تشخیص خطا
- پیش بینی وضعیت بعدی
- پیش بینی خطای بعدی
- اندازه گیری سیستم
- به روز رسانی محسابات وضعیت
- به روز رسانی محاسبات خطا

در تصویر زیر معادلات و پارامترهای مورد استفاده در فیلتر کالمن آورده شده اند. همانطور که در تصویر دیده می شود فیلتر کالمن یک حلقه بین دو حالت پیش بینی و تصحیح است.



شکل (۳-۲) معادلات فیلتر کالمن [21]

فیلتر کالمن به دلیل توانایی و پشتیبانی از تخمین و پیش بینی وضعیت گذشته، حال و آینده و همچنین امکان پیش بینی حتی در نبود مدل دقیق از سیستم، یک فیلتر بسیار قوی محسوب می شود. [21] در ادامه شبه کد الگوریتم فیلتر کالمن آورده شده است.

جدول (۳-۲) شبه کد الگوریتم فیلتر کالمن

```
Algorithm Kalman filter ( X_{t-1}, u_t, z_t ):
  Set R; //measurement noise
  Set Q; //process noise, tuned value
  x(1,1)=z(1);
  p(1,1)=R;
  for k=2:n;
      x(k,k-1)=x(k-1,k-1);
      p(k,k-1)=p(k-1,k-1)+Q;
      kg(k)=p(k,k-1)/(p(k,k-1)+R);
      x(k,k)=x(k,k-1)+kg(k)*(z(k)-x(k,k-1));
      p(k,k)=(1-kg(k))*p(k,k-1);
end;
```

۳-۲-۴- فیلتر ذرات

فیلتر ذرات، یک پیاده سازی غیر پارامتری[۱] و جایگزین برای فیلتر بیز [۲] می باشد. فیلترهای غیرپارمتری، جایگزینی مشهور برای فیلترهای گوسی هستند و مثل روش های گوسی[۳] به یک تابع ثابت از حالت بعد وابسته نمی باشند. فیلتر ذرات[۴] همانند فیلتر هیستوگرام، حالت بعد[۵] را با تعدادی پارامتر معین تقریب می زند اما در روش تولید این پارامترها و گسترش در فضای حالات متفاوت است. [1] الگوریتم فیلتر ذرات در جدول زیر آورده شده است.

جدول (۳-۳) شبه کد الگوریتم فیلتر ذرات

```
Algorithm Particle filter ( X_{t-1}, u_t, z_t ):
  X̄_t = X_t = ∅
  for m = 1 to M do
        sample x_t^[m] ~ p(x_t | u_t, x_{t-1}^[m])
```

[۱] Nonparametric
[۲] Bayes Filter
[۳] Gaussian
[۴] Particle Filter
[۵] Posterior

15:	$w_t^{[m]} = p(z_t \| x_{t-1}^{[m]})$
16:	$\bar{X}_t = \bar{X}_t + (x_t^{[m]}, w_t^{[m]})$
17:	endfor
18:	for m = 1 to M do
19:	draw i with probability $\propto w_t^{[i]}$
20:	add $x_t^{[i]}$ to X_t
21:	endfor
22:	return X_t

۳-۲-۵- الگوریتم AMCL[۱]

محل یابی تطبیق پذیر مونته کارلو یا به صورت کوتاه AMCL یک سیستم احتمالی محل یابی[۲] برای یک ربات در محیط دو بعدی ا ست. محل یابی ربات متحرک در حقیقت حل م سئله ی ت شخیص مکان روبات از داده های به دســت آمده از ســنســورها اســت.[3] به روش هایی انطباقی گفته می شــود که بتوانند تعداد پارامترها برای نمایش حالت بعد را به صورت برخط تطبیق دهند. [4] این الگوریتم به دلیل ا ستفاده از فیلتر ذرات با نام الگوریتم محل یابی فیلتر ذرات نیز شــناخته می شــود. شــبه کد الگوریتم AMCL در ادامه آورده شده است. گر چه با نتیجه ی نهایی پیاده سازی در سیستم عامل رز اندکی متفاوت است اما به دلیل شباهت الگوریتم ها در اصل و اساس از این شبه کد استفاده شده است.

جدول (۳-۴) شبه کد الگوریتم AMCL [24]

1:	**Algorithm Augmented MCL(**X_{t-1}, u_t, z_t, m**):**
2:	static w_{slow}, w_{fast}
3:	$\bar{X}_t = X_t = 0$
4:	for m = 1 to M do
5:	$x_t^{[m]}$ = sample_motion_model($u_t, x_{t-1}^{[m]}$)
6:	$w_t^{[m]}$ = measurement_model($z_t, x_t^{[m]}, m$)

[۱] Adaptive Monte Carlo Localization

[۲] Localization

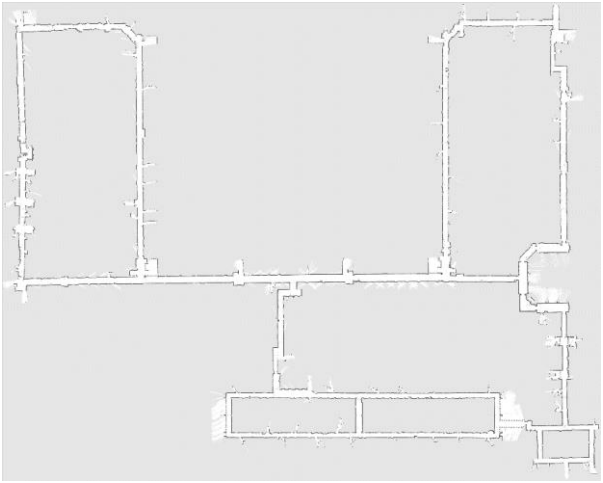
```
        $\overline{X}_t = \overline{X}_t + (x_t^{[m]}, w_t^{[m]})$
        $w_{avg} = w_{avg} + \frac{1}{M} w_t^{[m]}$
    endfor
    $w_{slow} = w_{slow} + \alpha_{slow} (w_{avg} - w_{slow})$
    $w_{fast} = w_{fast} + \alpha_{fast} (w_{avg} - w_{fast})$
    for m = 1 to M do
        with probability max(0.0; 1.0 - $w_{fast}/w_{slow}$) do
            add random pose to $X_t$
        else
            draw i ∈ {1, … ,N} with probability $\alpha \; w_t^{[i]}$
            add $x_t^{[i]}$ to $X_t$
        endwith
    endfor
    return $X_t$
```

۳-۲-۶- الگوریتم GMapping

این الگوریتم یک (RBPF) فیلتر ذرات Rao-Blackwell شـــده ی بســـیار بهینه اســـت که از آن برای تشخیص و درست کردن نقشه های شبکه ای[۱] از داده های به دست آمده از سنسورهای لیزر استفاده می شود. ساختن نقشه ها یکی از بنیادی ترین وظیفه هایی است که یک ربات متحرک باید با آن روبرو شود. در گذشته بسیاری از پژوهشگران بر روی حل این مسئله تمرکز کرده اند. مسئله ی اصلی در رویکردهای Rao-Blackwell شده، پیچیدگی و تعداد ذرات مورد نیاز برای ساختن نقشه است. [6] احتمالا Gmapping یکی از بهینه ترین و رایج ترین الگوریتم های SLAM باشد.

[۱] Grid Maps

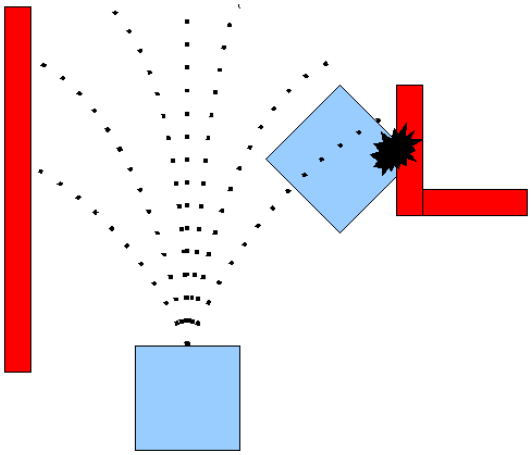
شکل (۳-۳) نقشه ی فضای باز دانشگاه فرایبورگ[۱]



شکل (۳-۴) نقشه ی MIT Killian Court

۳-۲-۷- الگوریتم DWA

رویکرد پنجره ی دینامیک یا DWA یک Local Planner برپایه ی سرعت است و سرعت ربات برای رسیدن به مقصد از مسیری بهینه، بدون برخورد و تصادف با اجسام را محاسبه می کند. این الگوریتم همچنین برای ربات موقعیت دکارتی (x,y) هدف را به سرعت و سرعت زاویه ای (v,w) ترجمه می کند.[27]



شکل (۳-۵) الگوریتم DWA

به زبان ساده، این الگوریتم برای ربات مسیری را می یابد که بدون برخورد به موانع بتواند به هدف تعیین شده برسد.

اصل ایده در این الگوریتم به شکل زیر خلاصه می شود:

- نمونه گیری گسسته در فضای کنترلی ربات (dx,dy,dtheta)

---

[۱] Universität Freiburg

- برای هر نمونه شبیه سازی و پیش بینی وضعیت آینده و شروع از وضعیت جاری
- ارزیابی و امتیازدهی به نتایج حاصل از پیش بینی و شبیه سازی
- حذف مسیرهای غیر قابل قبول
- انتخاب بهترین مسیر بر حسب امتیاز بندی
- ارسال فرامین و پارامتر سرعت های مورد نظر به ربات
- حرکت و تکرار حلقه

گر چه برای پیاده سازی این الگوریتم بر روی ربات فراز، تغییراتی در پ شته ی DWA_Local_Planner به خ صوص برای فرآیند خطایابی انجام و سورس کد این الگوریتم به صورت خاص کامپایل شد اما به دلیل حجم زیاد کدهای این الگوریتم و همچنین عدم تغییر بنیادی در این سیستم از گذاشتن سورس کد در پیوست این پایان نامه خودداری شده است. در ادامه شبه کد الگوریتم DWA آورده شده است:

جدول (۳-۵) شبه کد الگوریتم DWA [25]

```
BEGIN DWA(robotPose,robotGoal,robotModel)
  desiredV = calculateV(robotPose,robotGoal)
  laserscan = readScanner()
  allowable_v = generateWindow(robotV, robotModel)
  allowable_w  = generateWindow(robotW, robotModel)
  for each v in allowable_v
    for each w in allowable_w
    dist = find_dist(v,w,laserscan,robotModel)
    breakDist = calculateBreakingDistance(v)
    if (dist > breakDist)  //can stop in time
      heading = hDiff(robotPose,goalPose, v,w)
      clearance = (dist-breakDist)/(dmax - breakDist)
      cost = costFunction(heading,clearance, abs(desired_v - v))
      if (cost > optimal)
        best_v = v
        best_w = w
        optimal = cost
   set robot trajectory to best_v, best_w
END
```

۳-۲-۸- نتیجه گیری

الگوریتم های بسیار زیادی توسط دانشمندان و پژوهشگران مختلفی در سطح جهان ارائه شده است تا انسان بتواند روزی به رویای ماشین های بدون سرنشین و به طور کامل تر خودمختاری روبات ها برسد. توضیح کامل و آوردن تمام این الگوریتم ها در این پایان نامه تقریبا غیر ممکن است و فقط سعی شده است به صورت مختصر خواننده را با الگوریتم های به کار گرفته شده آشنا کند.

فصل چهارم

طراحی و پیاده سازی

مقدمه

در این فصل ابتدا روند طراحی ربات فراز شرح داده شده و سپس کدها، برنامه ها و پیاده سازی های مختلف و مورد نیاز آورده شده است.

۴-۱- طراحی پلتفرم / ربات

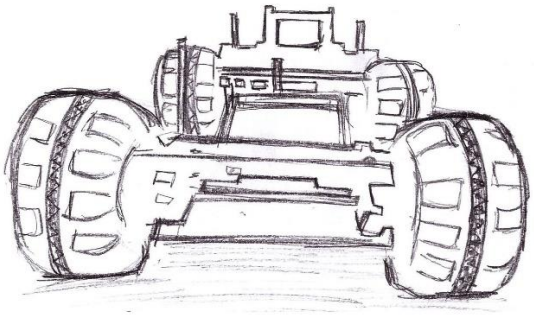
۴-۱-۱- طراحی روبات فراز سایبرهندس

در این بخش روند کلی طراحی و شبیه سازی ربات فراز آورده شده است. ابتدا طرح اولیه ی ربات از سه زاویه ی مختلف طراحی شده و سپس پس از بررسی و رفع مشکلات و نواقص موجود و برطرف سازی نیازهای طراحی، روبات فراز در نرم افزار متن باز و سه بعدی بلندر[۱] مدل سازی شد. در ادامه طرح های اولیه آورده شده است. پس از طراحی و ساخت مدل سه بعدی ربات فراز، مراحل مقدماتی انتقال این ربات به سیستم عامل رز از جمله بهینه سازی و کاهش حجم مدل و ... انجام و سپس مدل *SDF* و *URDF* ربات نوشته و آماده شد. می توان فرآیند طراحی و مدلسازی را در تصاویر آورده شده به خوبی مشاهده کرد.

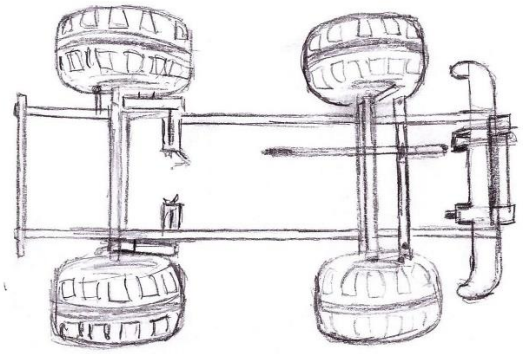
در طراحی می بایست چند نکته در نظر گرفته می شد. این ماشین برای همکاری با ربات های پرنده طراحی شده است لذا باید بتواند میزبانی برای ربات های پرنده باشد و پرنده ها در هنگام اتمام باتری و یا احساس خطر بر روی ماشین یا درون آن قرار گیرند. در ضمن لوگوی سایبرهندس نیز از نمای بالا باید قابل مشاهده باشد. بنابراین همانطور که در تصاویر دیده می شود لوگوی سایبرهندس به شکل یک درب و با لولا در قسمت فوقانی ماشین قرار دارد. علاوه بر اینکه ربات پرنده می تواند بر روی این ماشین سوار شود، با باز شدن درب امکان جاگیری ربات پرنده در ماشین نیز وجود دارد. البته ابعاد ماشین قابل تغییر است و هر چه بزرگ تر ساخته شود مخزن بزرگ تری برای حمل وسایل مختلف خواهد داشت. علاوه بر این می توان از این درب برای سهولت دسترسی به باتری و سایر سیستم های کنترلی ربات استفاده کرد. طراحی سبک، زیبا و امن از سایر مشخصه های در نظر گرفته شده در بخش طراحی است که البته بعضی از پارامترها و فاکتورهای مورد نظر به دلیل افزایش قیمت ساخت، در مرحله ی اول در نظر گرفته نشده است. برای سیستم های پیشرفته تر می توان از این مشخصه ها بهره برد. گرچه این ربات، از نوع بدون سرنشین طراحی شده است اما با اندکی تغییر و بهینه سازی قابل استفاده برای ساخت یک نمونه ی سرنشین دار نیز است.

---

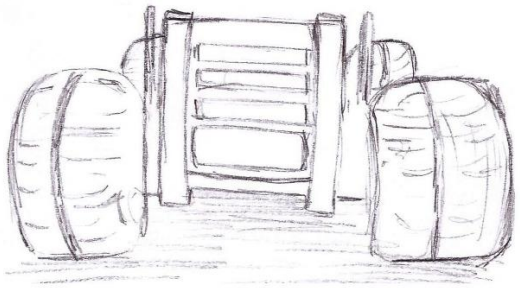
[۱] Blender



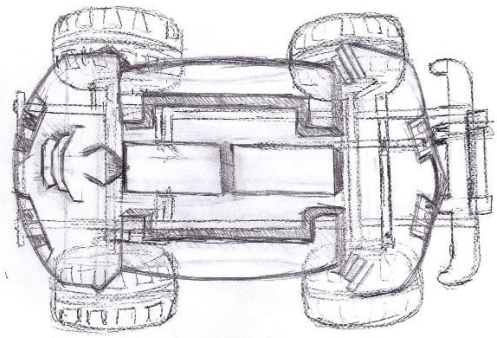
شکل (۴-۱) روبات فراز از نمای عقب + شاسی



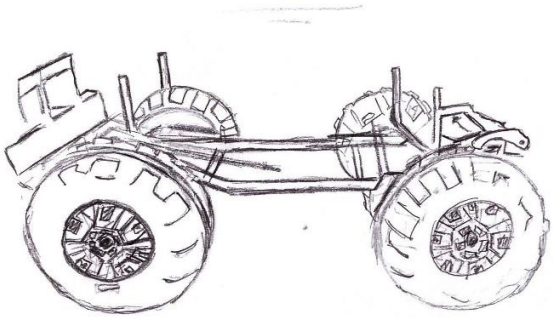
شکل (۴-۴) روبات فراز از نمای بالا + شاسی



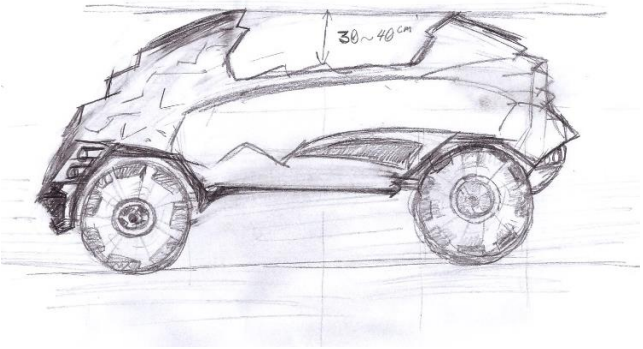
شکل (۴-۲) روبات فراز از نمای جلو + شاسی



شکل (۴-۵) روبات فراز از نمای بالا + بدنه



شکل (۴-۳) روبات فراز از نمای چپ + شاسی



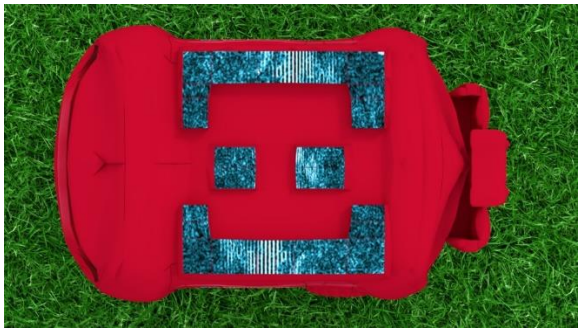
شکل (۴-۶) روبات فراز از نمای چپ + بدنه

در ادامه برخی از تصاویر مدل سازی سه بعدی و نهایی برای استفاده در شبیه ساز آورده شده است. همانطور که قبلا به آن اشاره شد، برای مدلسازی سه بعدی از نرم افزاری به نام بلندر استفاده شده است. که یکی از بزرگ ترین مزایای آن متن باز بودن این پروژه ی بزرگ می باشد. در طراحی این ربات سعی شده است فقط از نرم افزارهای متن باز و قابل دسترس استفاده شود تا هم هزینه ی ساخت کمتر شده و هم استفاده از فرهنگ متن باز ترویج داده شود.

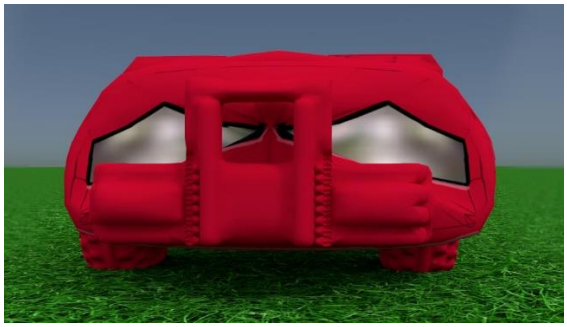
شکل (۴-۷) مدل سه بعدی فراز سایبرهندس – Cycles Render در بلندر

شکل (۴-۸) مدل سه بعدی فراز – نمای پشت

شکل (۴-۹) مدل سه بعدی فراز – نمای راست



شکل (۴-۱۰) مدل سه بعدی فراز – نمای بالا



شکل (۴-۱۱) مدل سه بعدی فراز – نمای جلو

در مدلســـازی و طراحی ربات فراز علاوه بر تکنیک های رایج در مدلســـازی از تکنیک UV Mapping جهت متریال دهی به کل طرح و همچنین از Cycles Render به عنوان موتور پردازشـــی و رندرگیری اصـــلی استفاده شده است. همچنین متریال استفاده شده ابتدا در نرم افزارهای گرافیکی رایج طراحی شده و سپس در مدل سه بعدی فراز به کار رفته است.

۴-۲- طراحی و پیاده سازی نرم افزار

۴-۲-۱- توصیف فراز برای شبیه سازها

با توجه به حجم بالای کد توصـــیف ربات برای شـــبیه ســـازها از آوردن این کد در متن پایان نامه خودداری شده است. برای دسترسی به کدهای توصیف ربات فراز سایبرهندس دانشگاه صنعتی جندی شاپور دزفول به زبان URDF جهت استفاده در شبیه سازها می توان در پیوست الف به فایل chugv.urdf.xacro و ضیمه های مورد نیاز آن مراجعه کرد.

۴-۲-۲- پیاده سازی، نصب و تنظیم پارامترهای ناوبری

اصل ایده ی ناوبری همچون سایر تکنولوژی های بسیار پیشرفته و پیچیده ساده است. ورودی سیستم، اطلاعات دریافتی از ادومتری، اطلاعات ســـنســـورها و موقعیت مقصـــد می باشـــد. بر اســـاس اطلاعات ورودی، خروجی سیستم، دستوراتی برای حرکت ربات به سمت مقصد است. در عمل پیاده سازی چنین سیستمی

سـختی های خاص خود را دارد. برای اسـتفاده از سـیسـتم ناوبری موجود در سـیسـتم عامل رز حداقل باید درخت ارتباط فریم های موجود در ربات تهیه شده باشد، پیام و اطلاعات سنسورها به درستی منتشر شود و تنظیمات مورد نیاز برای شکل و سایر ویژگی های ربات به درستی تنظیم شده باشند.

از انواع مختلف سنسورها نظیر لیدار[۱]، کینکت[۲] و لیزر می توان برای دریافت، تشخیص و ادراک ربات از محیط اطرافش استفاده کرد. هر کدام مزایا و معایب خاص خود را داراست. کینکت اصلا برای این کار ساخته نشده است و نهایتا تا سه متر از محیط را به خوبی پوشش خواهد داد و اسکن می کند. لیزر و لیدار نسبت به کینکت چنین محدودیتی ندارند. لیزر محیط اطراف خود را به صـــورت دوبعدی دریافت می کند. در مقابل لیدار گرچه می تواند محیط را به صورت سه بعدی دریافت کند و همچنین زاویه ی دید ۳۶۰ درجه داشته باشد، اما قیمت آن بسیار زیاد است و بنابراین فعلا برای این پروژه از لیزر استفاده خواهد شد. علاوه بر آن برای شبیه سازی بهتر از دوربین نیز استفاده خواهد شد. برای افزایش کیفیت ناوبری می توان از سنسورهای دیگری نظیر IMU، GPS، Sonar و سایر سنسورهای مرتبط استفاده کرد. برای ساده سازی در مرحله ی اول شبیه سازی، این سنسورها در ربات استفاده نخواهند شد.

به طور کلی پشته ی ناوبری در سیستم عامل رز شامل پکیج های مختلف و متنوعی می شود که در مجموع و با هم از ناوبری دو بعدی پشتیبانی می کنند. مهمترین این پکیج ها در ادامه آورده شده است:

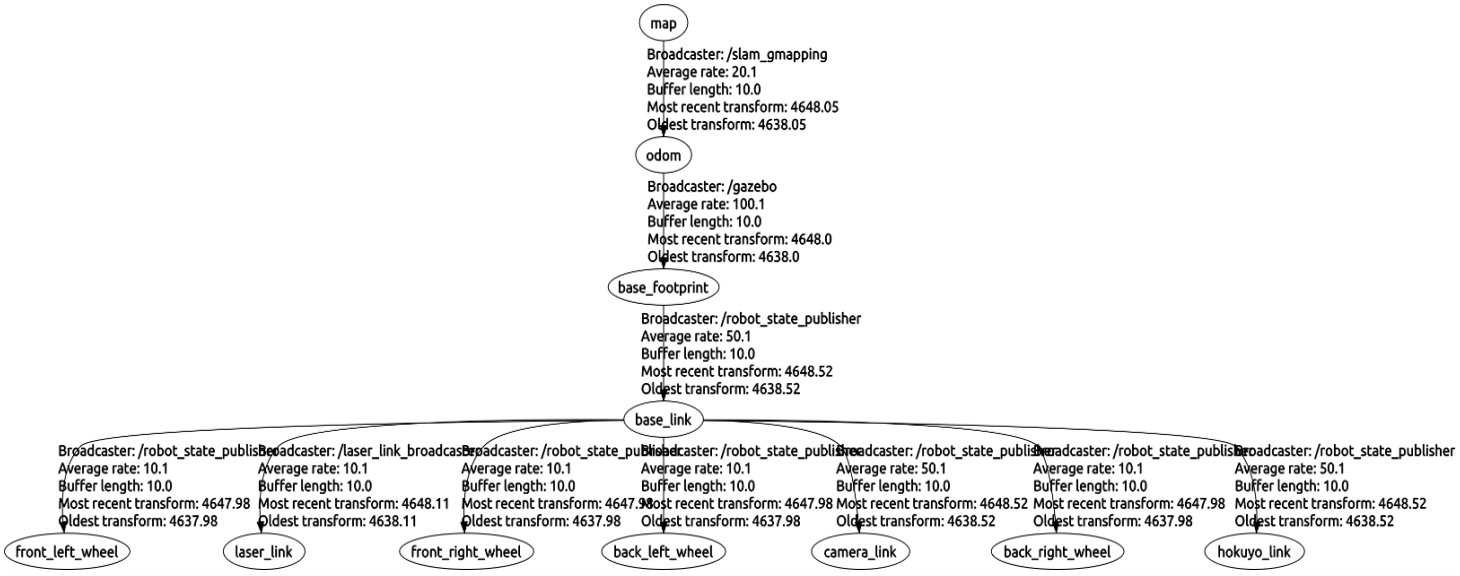
- Amcl
- base_local_planner
- carrot_planner
- clear_costmap_recovery
- costmap_2d
- dwa_local_planner
- fake_localization
- global_planner
- map_server
- move_base
- move_base_msgs
- move_slow_and_clear
- nav_core
- navfn
- robot_pose_ekf
- rotate_recovery
- voxel_grid

---

[۱] LiDAR

[۲] Kinect

در مرحله ی اول، درخت ارتباط فریم های موجود در ربات باید به شـکل صـحیح ایجاد شـود. شـما می توانید این درخت را در پیو ست ب به صورت بزرگ تر م شاهده کنید. در اینجا فقط از نمونه ای کوچک برای ساده سازی استناد و سهولت دسترسی استفاده شده است.



شکل (۴-۱۲) درخت ارتباط فریم ها در ربات فراز

همانطور که در ت صویر بالا م شخص ا ست، باید بین گره های map، odom و سایر فریم ها و لینک های ربات ارتباط در ستی وجود دا شته با شد. برای این مق صود، تمام زیر مجموعه های گره ی base_link و base_footprint در URDF/XACRO ربات به درســتی تشــریح شــده اند. می توان به کد منبع کامل این تو صیف در پیو ست الف د ستر سی دا شت. به فایل chugv.urdf.xacro و سایر فایل های لینک شده در آن مراجعه شود. همه ی این زیر مجموعه ها تو سط گره ای با نام robot_state_publisher منت شر می شوند و وظیفه ی ت شخیص ارتباط بین لینک ها و فریم ها و همچنین انت شار و ضعیت به عهده ی این گره می با شد. ما با فراخوانی ســه گره ی زیر به تمام اهداف خود برای دســترســی به تمام joint ها و فریم های ربات می رســیم. در حقیقت گره ی robot_state_publisher ارتباط بین لینک ها و فریم ها را منتشــر می کند. و Joint_state_publisher و ضعیت تمام ات صالات و joints های موجود در ربات را منت شر کرده و از گره ی با نام laser_link_broadcaster برای ایجاد ارتباط بین base_link و laser_link اســتفاده شــده اســت. در این درخت Gazebo مسئولیت انتشار اطلاعات ادومتری ربات را بر عهده دارد و نقشه یا Map محیط، توسط گره ی Slam_Gmapping منتشر می شود. که البته در تصویر بالا به خوبی مشخص است.

جدول (۴-۱) کد انتشار وضعیت فریم ها، اتصالات و لینک های ربات

```
<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher"/>
<node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher" output="screen"/>
<node pkg="tf" type="static_transform_publisher" name="laser_link_broadcaster" args="0 0 1 0 0 0 base_link laser_link 100"/>
```

برای فراخوانی گره ی SLAM_Gmapping بعد از نصب و یا کامپایل آن می توان از کد زیر استفاده کرد. البته همانطور که می توان دید، پارامترها و تنظیمات مربوط به این گره در لحظه ی فراخوانی وارد شده اند. این پارامترها برای استفاده در ربات فراز سایبرهندس بهینه شده و ممکن است برای استفاده بر روی یک ربات دیگر نیاز به تغییراتی وجود داشته باشد. البته این تنظیمات برای اکثر ربات های استاندارد کافی خواهد بود.

جدول (۴-۲) کد فراخوانی گره ی Slam_Gmapping و تنظیمات آن

```
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
    <remap from="scan" to="base_scan"/>
    <param name="odom_frame" value="odom"/>
    <param name="map_update_interval" value="30.0"/>
    <param name="maxUrange" value="16.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="srr" value="0.01"/>
    <param name="srt" value="0.02"/>
    <param name="str" value="0.01"/>
    <param name="stt" value="0.02"/>
    <param name="linearUpdate" value="0.5"/>
    <param name="angularUpdate" value="0.436"/>
```

```
20:         <param name="temporalUpdate" value="-1.0"/>
21:         <param name="resampleThreshold" value="0.5"/>
22:         <param name="particles" value="80"/>
23:         <param name="xmin" value="-50.0"/>
24:         <param name="ymin" value="-50.0"/>
25:         <param name="xmax" value="50.0"/>
26:         <param name="ymax" value="50.0"/>
27:         <param name="delta" value="0.05"/>
28:         <param name="llsamplerange" value="0.01"/>
29:         <param name="llsamplestep" value="0.01"/>
30:         <param name="lasamplerange" value="0.005"/>
31:         <param name="lasamplestep" value="0.005"/>
32:       </node>
```

باید به این نکته توجه شـود که این گره می تواند با توجه به اطلاعات ورودی وظیفه ی محل یابی و نقشه برداری را به صورت همزمان انجام دهد. بنابراین در صورتی که توسط گره ی Map_Server از نقشه ی از پیش آماده اســتفاده شــود، باید به جای فراخوانی Slam_Gmapping از گره ی AMCL برای محل یابی ربات استفاده گردد. زیرا Map_Server فقط داده های مربوط به نقشه را بازیابی می کند و محل یابی ربات را انجام نمی دهد.

جدول (۴-۳) کد فراخوانی Map_Server و انتشار نقشه ی ایستا

```
1:    <node name="map_server" pkg="map_server" type="map_server" args="$(find
      chugv_navigation)/maps/CHScanMap.yaml"/>
```

جدول (۴-۴) کد فراخوانی AMCL برای انجام محل یابی ربات

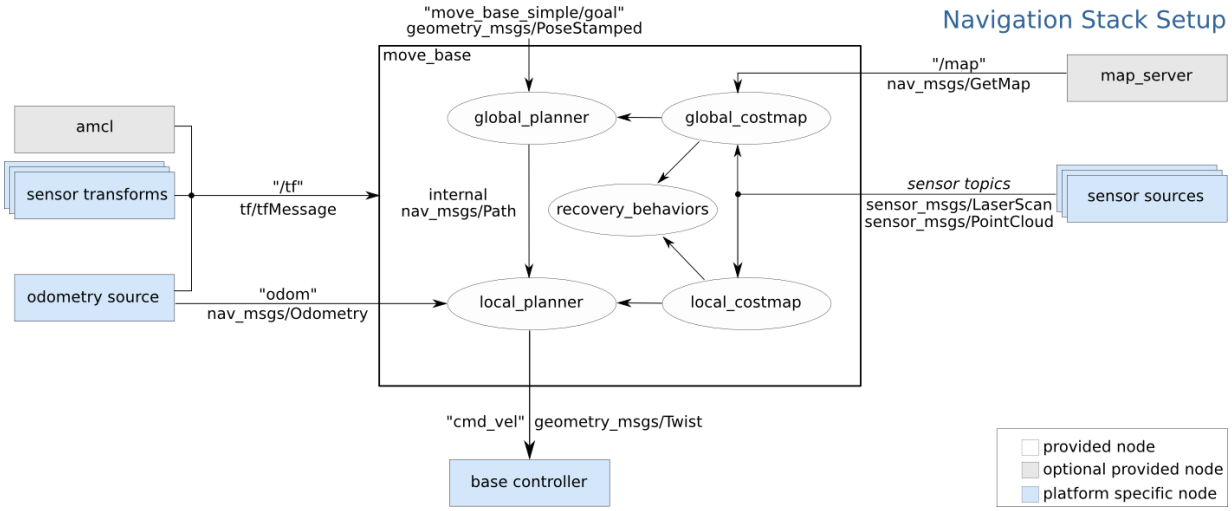
```
2:    <node pkg="amcl" type="amcl" name="amcl" respawn="true">
3:      <remap from="scan" to="base_scan" />
4:      <param name="odom_model_type" value="omni"/>
5:      <param name="odom_alpha5" value="0.1"/>
6:      <param name="transform_tolerance" value="0.2" />
7:      <param name="gui_publish_rate" value="10.0"/>
8:      <param name="laser_max_beams" value="30"/>
9:      <param name="min_particles" value="500"/>
10:     <param name="max_particles" value="5000"/>
```

```
11:     <param name="kld_err" value="0.05"/>
12:     <param name="kld_z" value="0.99"/>
13:     <param name="odom_alpha1" value="0.2"/>
14:     <param name="odom_alpha2" value="0.2"/>
15:     <param name="odom_alpha3" value="0.8"/>
16:     <param name="odom_alpha4" value="0.2"/>
17:     <param name="laser_z_hit" value="0.5"/>
18:     <param name="laser_z_short" value="0.05"/>
19:     <param name="laser_z_max" value="0.05"/>
20:     <param name="laser_z_rand" value="0.5"/>
21:     <param name="laser_sigma_hit" value="0.2"/>
22:     <param name="laser_lambda_short" value="0.1"/>
23:     <param name="laser_lambda_short" value="0.1"/>
24:     <param name="laser_model_type" value="likelihood_field"/>
25:     <param name="laser_likelihood_max_dist" value="2.0"/>
26:     <param name="update_min_d" value="0.2"/>
27:     <param name="update_min_a" value="0.5"/>
28:     <param name="odom_frame_id" value="odom"/>
29:     <param name="resample_interval" value="1"/>
30:     <param name="transform_tolerance" value="0.1"/>
31:     <param name="recovery_alpha_slow" value="0.0"/>
32:     <param name="recovery_alpha_fast" value="0.0"/>
33:   </node>
```

همانطور که در جدول بالا می توان مشاهده کرد، پارامترها و تنظیمات مربوط به این گره نیز در هنگام فراخوانی ارسال شده است. قبل از هر چیز باید از انتشار اطلاعات سنسور ها نظیر لیزر و دوربین مطمئن شد. سپس در ادامه ابتدا گره ی ناوبری در سیستم عامل رز توضیح داده می شود و در نهایت تنظیمات مورد نظر برای اجرای بهتر سیستم ناوبری بر روی روبات فراز تشریح می گردد.

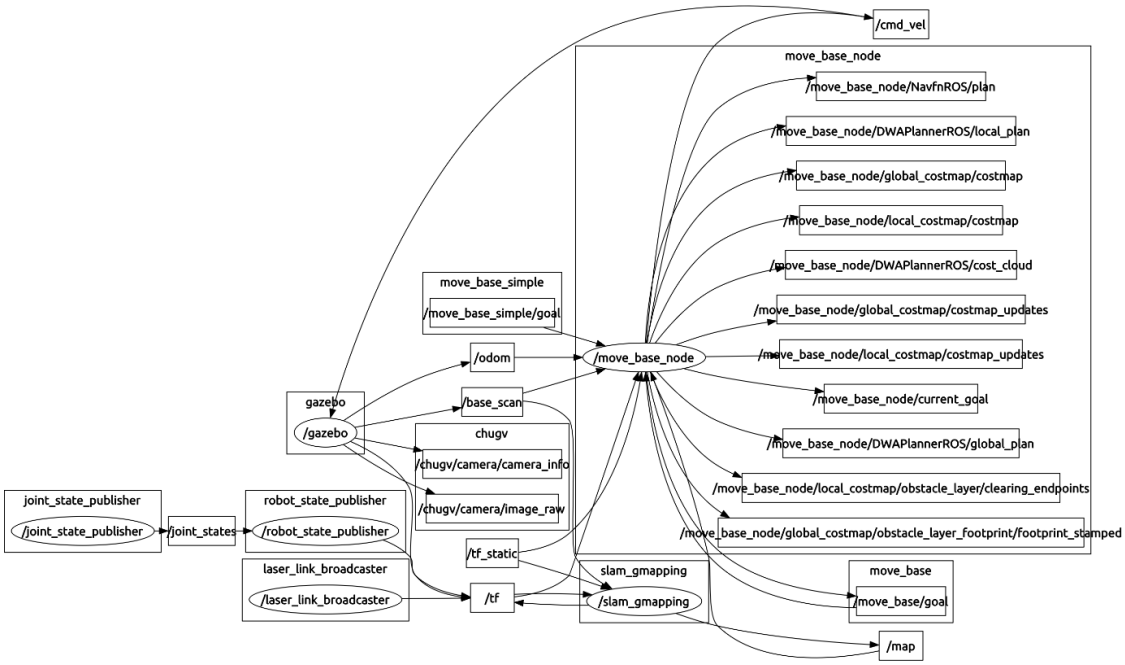
در شکل زیر دیاگرام تنظیمات و نحوه ی ارتباط بین تمام گره های مورد نیاز برای راه اندازی سیستم ناوبری در سیستم عامل رز آورده شده است. همانطور که قبلا توضیح داده شد، باید اطلاعات map، tf و odom توسط گره های مختلفی که در همین فصل شرح داده شد منتشر گردند. همچنین باید تاپیک های سنسورهای اسکن نظیر لیزر برای سیستم ناوبری مشخص شود. اصل سیستم ناوبری به عهده ی move_base می باشد. این گره موقعیت هدف را دریافت کرده و بوسیله ی map و اطلاعات به دست آمده از سنسور نقشه ی هزینه ی کلی را محاسبه می کند. در این دیاگرام با نام global_costmap مشخص شده است. و سپس داده های مورد نیاز global_planner را مهیا می کند. این مسیرساز و یا نقشه کش، با توجه به موقعیت هدف و global_costmap مسیر رسیدن به هدف را به دست می آورد و آن را برای local_planner ارسال می کند.

مسیرساز محلی نیز با توجه به اطلاعات دریافتی و local_costmap به دست آمده از اطلاعات سنسورها، سرعت و زاویه ی حرکت را برای ربات از طریق cmd_vel ارسال می کند. اگر به هر دلیل ربات با مشکلی برای حرکت و رسیدن به مقصد روبرو شود، به حالت بازیابی رفته و رفتارهای از پیش تعیین شده اجرا خواهند شد. این رفتارها در دیاگرام با نام recovery_behaviors آمده اند. بر حسب تنظیمات می توان این رفتارها را غیرفعال کرده و یا حتی رفتار جدیدی را برنامه نویسی و معرفی کرد.



شکل (۴-۱۳) تنظیمات و نصب پشته ی ناوبری در سیستم عامل رز[7]

در تصویر زیر تمام گره های به کار گرفته شده، سیستم ناوبری، شبیه سازی و تمام جنبه های مختلف ربات آورده شده است. تصویر کامل و بزرگ تر در پیوست ج وجود دارد.

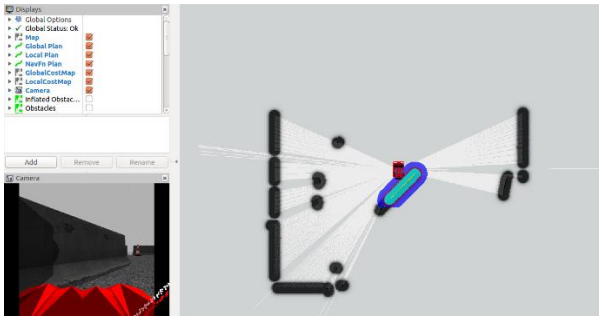


شکل (۴-۱۴) دیاگرام ارتباط بین تمام گره ها در ربات فراز سایبرهندس

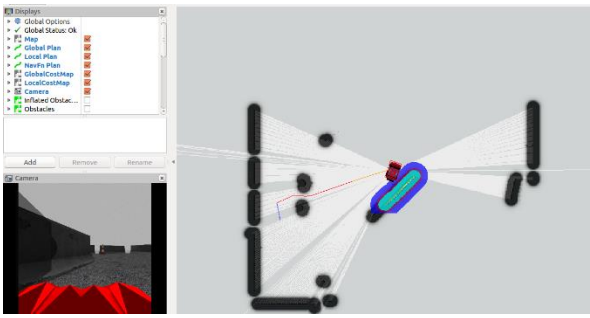
۴-۳- نتیجه گیری و ارزیابی نهایی

۴-۳-۱- تست و ارزیابی عملکرد نهایی ربات فراز سایبرهندس

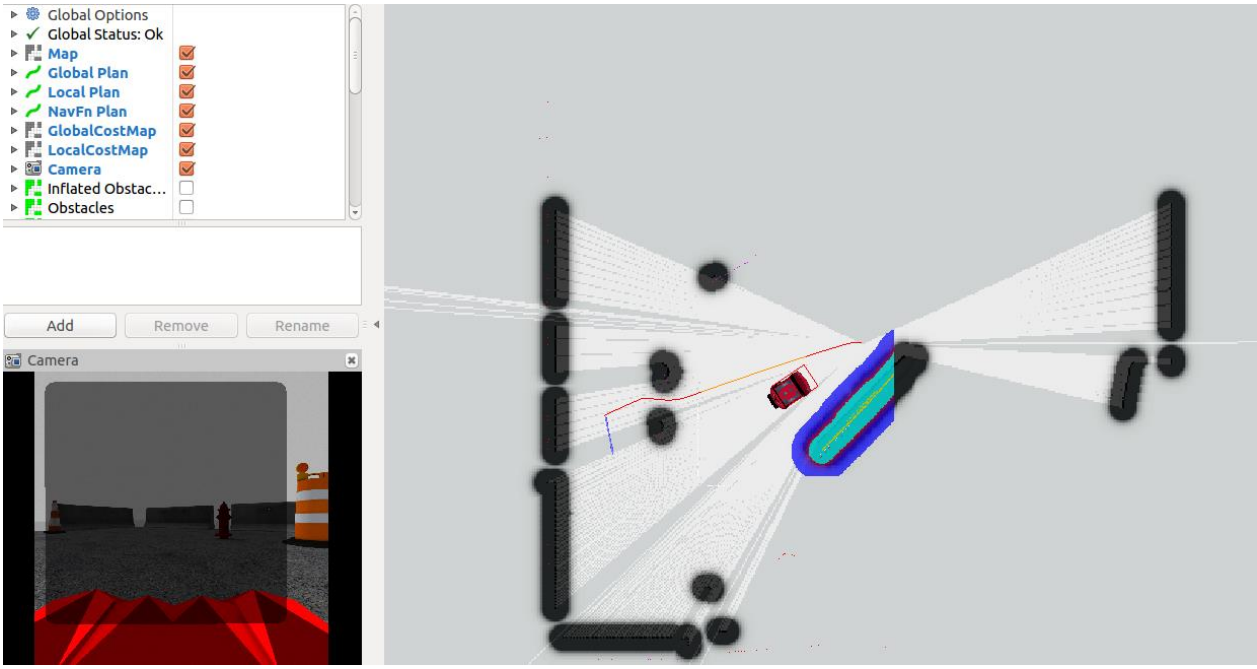
در این بخش سیستم ناوبری ربات فراز به طور کامل مورد ارزیابی قرار می گیرد. در تصاویری که در ادامه آورده شده است، چندین ماموریت مختلف به ربات داده شده است و همانطور که مشخص است ربات به خوبی وظیفه ی خود را انجام داده و با موفقیت و بدون برخورد به موانع به مقصد تعیین شده می رسد.



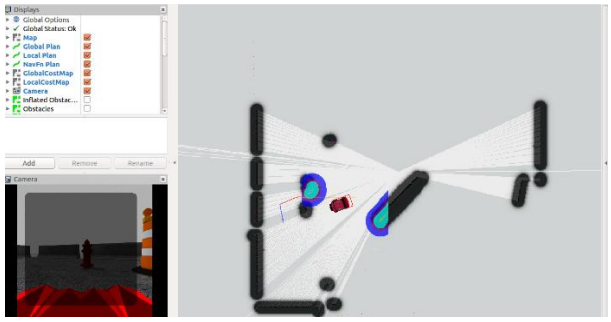
شکل (۴-۱۵) ماموریت ۱ – ربات فراز – ت ۱



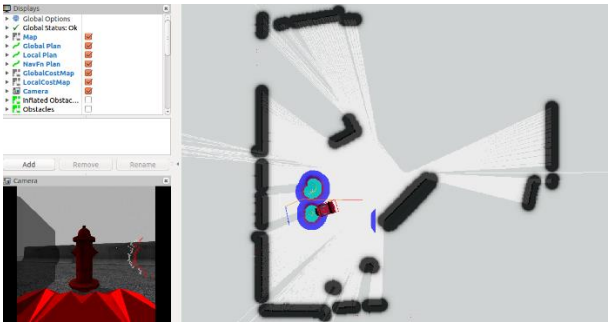
شکل (۴-۱۶) ماموریت ۱ – ربات فراز – ت ۲



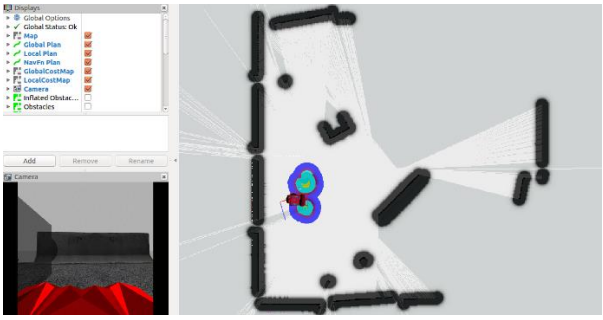
شکل (۴-۱۷) ماموریت ۱ – ربات فراز – ت ۳



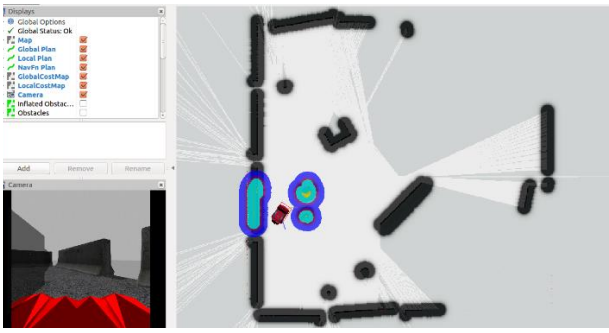
شکل (۴-۱۸) ماموریت ۱ – ربات فراز – ت ۴



شکل (۴-۱۹) ماموریت ۱ – ربات فراز – ت ۵



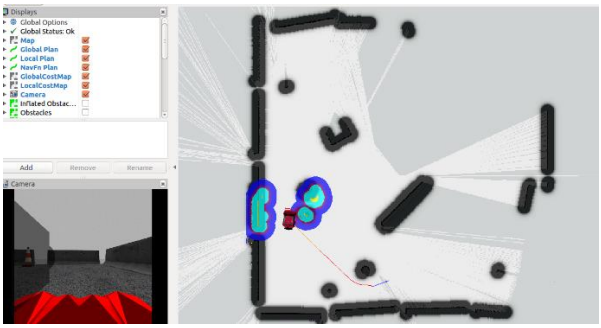
شکل (۴-۲۰) ماموریت ۱ – ربات فراز – ت ۶



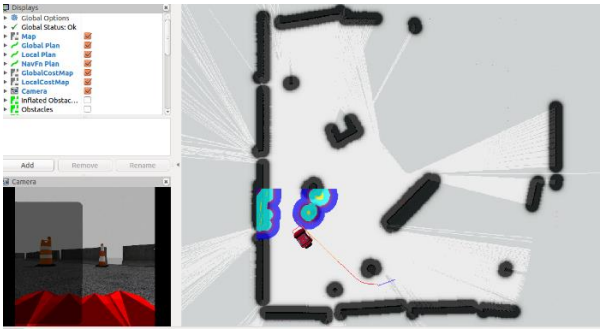
شکل (۴-۲۱) ماموریت ۱ – ربات فراز – ت ۷

همانطور که در تصاویر بالا مشخص است، موقعیت هدف به ربات داده شده است. سپس Global_Planner با توجه به نقشه ی بدست آمده توسط سنسور لیزر و Global_Costmap بهترین مسیر رسیدن به هدف را توسط الگوریتم Dijkstra محاسبه کرده و اطلاعات مسیر را برای دیگر گره ها منتشر می کند. و در نهایت Local_Planner با توجه به موانع موجود در اطراف ربات سعی می کند با توجه به مسیر مهیا شده توسط Global_Planner به مقصد نهایی برسد. موانع و Local_Costmap در تصویر بالا با رنگ آبی و بنفش به خوبی مشخص است و میزان ادراک ربات از محیط اطراف خود را نشان می دهد. در تصاویر بالا می توان مشاهده کرد که ربات از بین دو مانع بسیار نزدیک به هم می گذرد و بدون برخورد به این موانع سخت تصمیم صحیح را گرفته و مسیر درست را می پیماید.

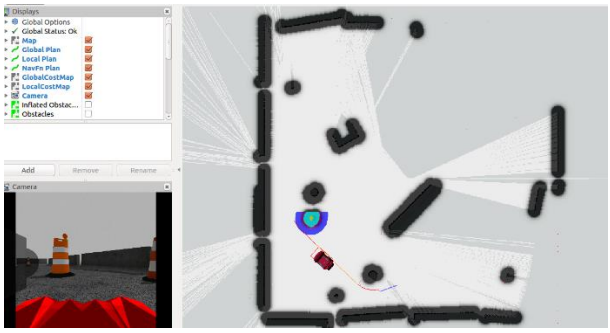
در تصاویری که در ادامه آورده شده است، ماموریت های مختلفی به ربات داده شده است و ربات با این سیستم به خوبی ماموریت خود را انجام می دهد و بدون برخورد به موانع موجود به موقعیت هدف می رسد.



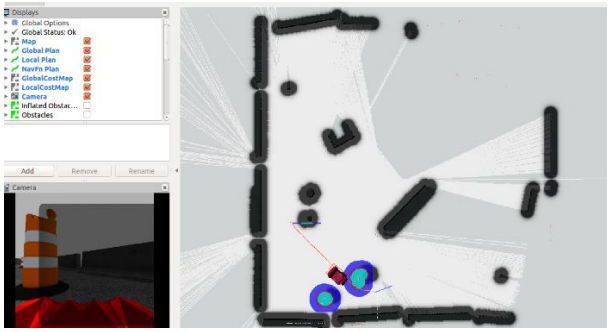
شکل (۴-۲۲) ماموریت ۲ – ربات فراز – ت ۱



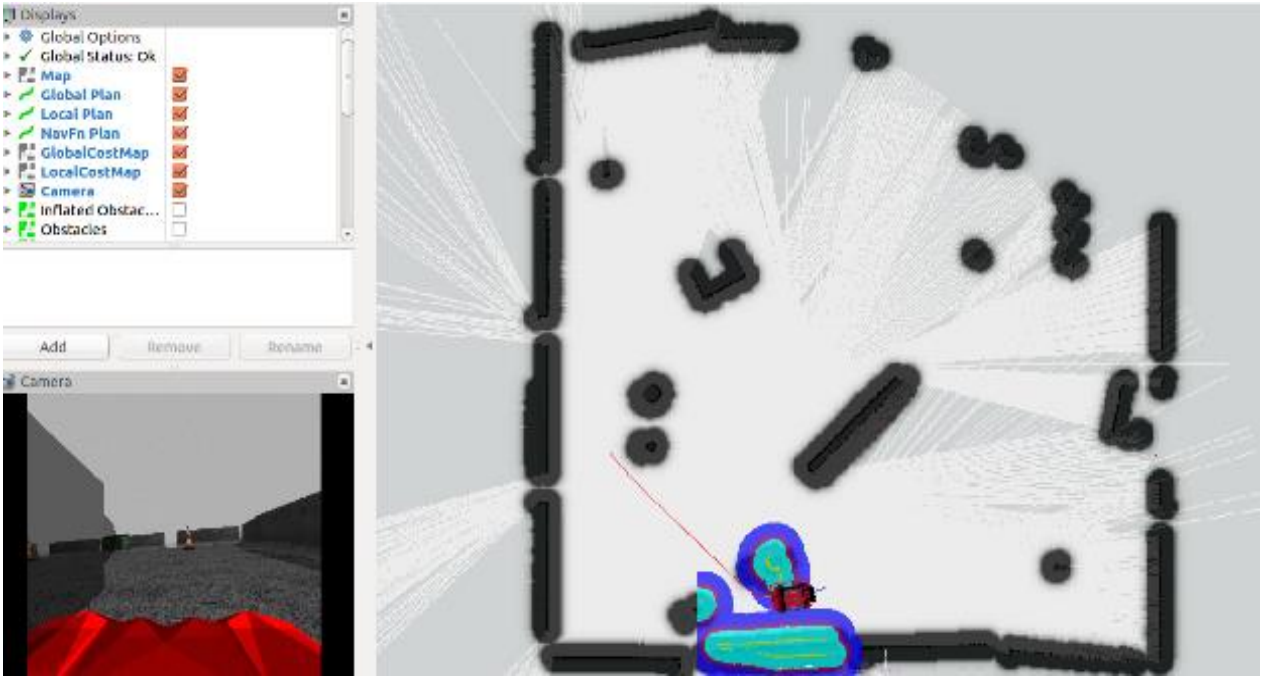
شکل (۴-۲۳) ماموریت۲ – ربات فراز – ت ۲



شکل (۴-۲۴) ماموریت ۲ – ربات فراز – ت ۳

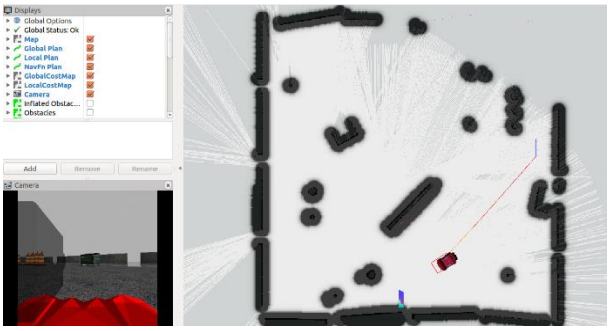


شکل (۴-۲۵) ماموریت ۲ – ربات فراز – ت ۴

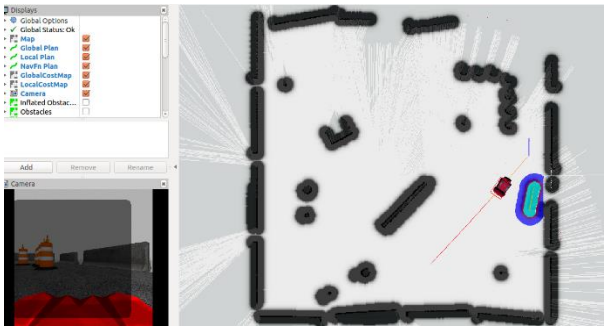


شکل (۴-۲۶) ماموریت ۲ – ربات فراز – ت ۵

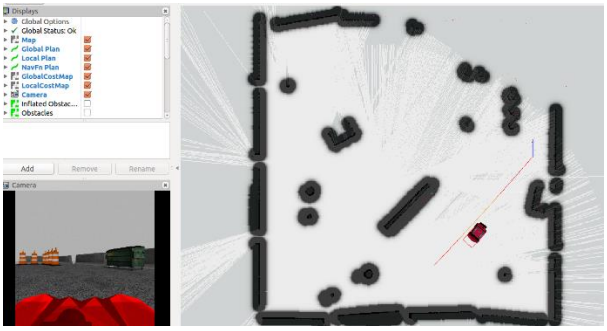
همزمان با انجام ماموریت های مختلف، به مرور زمان، نقشه ی محیط نیز کامل می شود و به خوبی می توان توان نقشه برداری این ربات را مشاهده کرد. شاید سخت ترین و در عین حال کاربردی ترین کاری که این ربات و ربات های مشابه آن انجام می دهند همان محل یابی و نقشه برداری همزمان باشد.



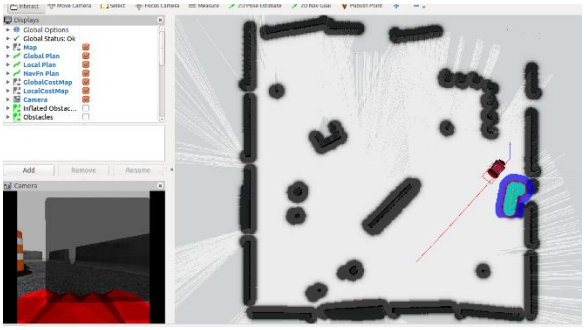
شکل (۴-۲۷) ماموریت ۳ – ربات فراز – ت ۱



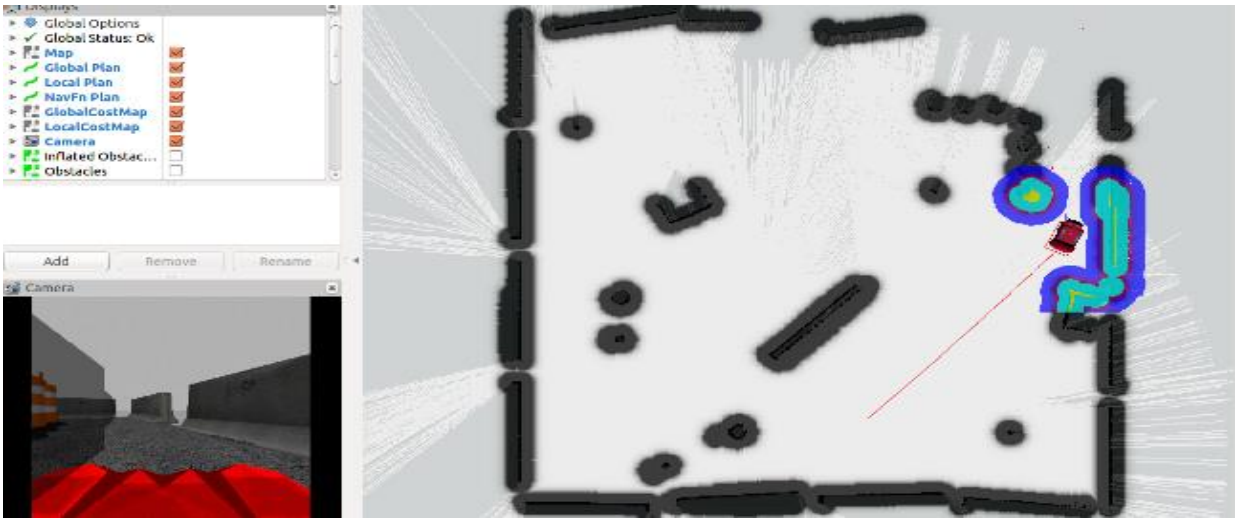
شکل (۴-۲۹) ماموریت ۳ – ربات فراز – ت ۳



شکل (۴-۲۸) ماموریت ۳ – ربات فراز – ت ۲



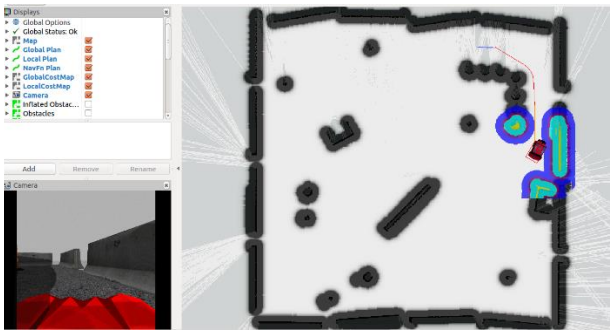
شکل (۴-۳۰) ماموریت ۳ – ربات فراز – ت ۴



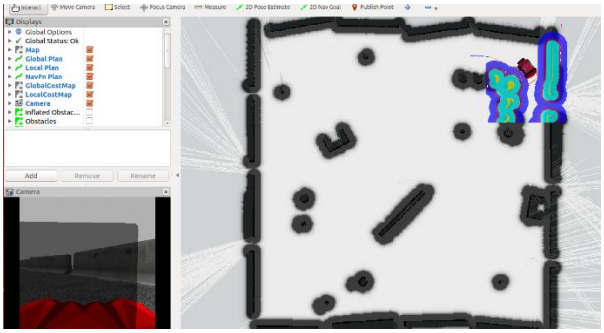
شکل (۴-۳۱) ماموریت ۳ – ربات فراز – ت ۵

قطعا سخت ترین ماموریتی که این ربات می تواند انجام بدهد، ماموریت شماره ۴ ا ست. هم م سیر عبوری باریک است و موانع در هر دو طرف مسیر وجود دارند. و هم ربات باید از یک پیچ سخت بدون برخورد به موانع گذر کند. تقریبا نقشـــه، در این مرحله به طور کامل تکمیل شـــده و ربات از تمام محیط اطراف خود

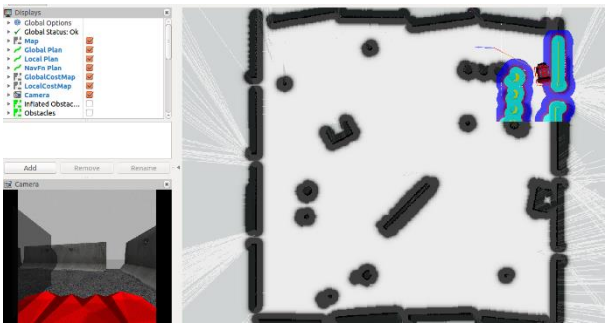
آگاه می شود.



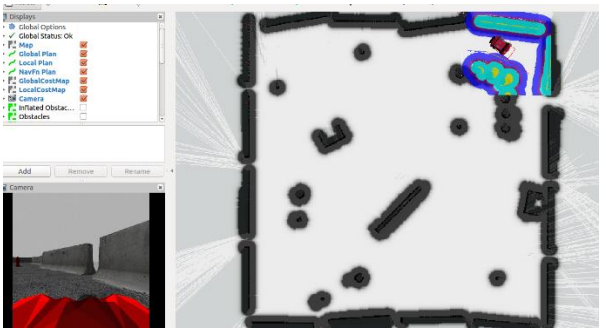
شکل (۴-۳۲) ماموریت ۴ – ربات فراز – ت ۱



شکل (۴-۳۴) ماموریت ۴ – ربات فراز – ت ۳

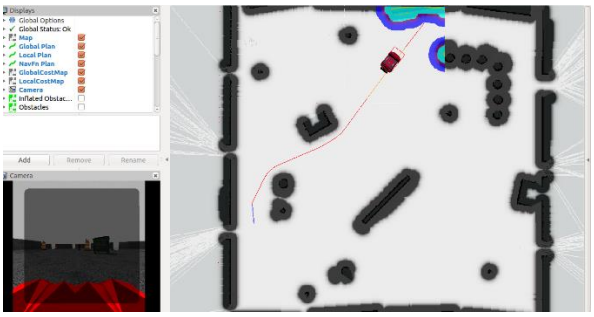


شکل (۴-۳۳) ماموریت ۴ – ربات فراز – ت ۲

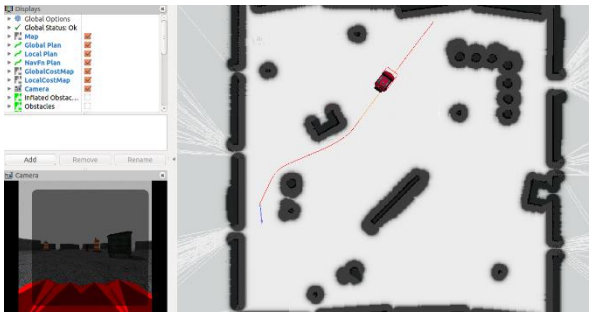


شکل (۴-۳۵) ماموریت ۴ – ربات فراز – ت ۴

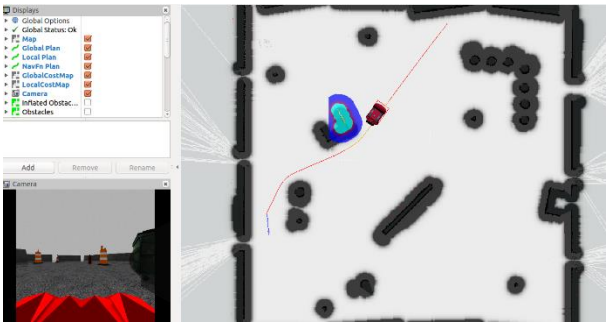
ماموریت پنجم و ماموریت آخر نیز به عنوان ماموریت تکمیلی جهت تســـت و ارزیابی قابلیت های ربات انجام شد.



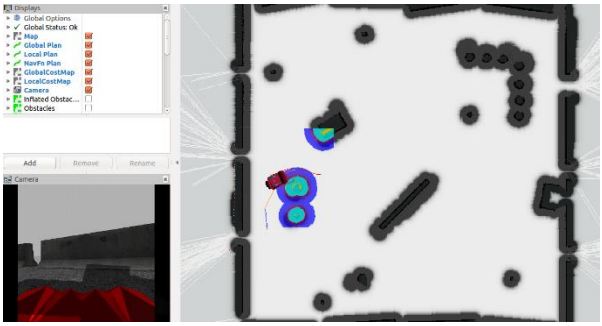
شکل (۴-۳۶) ماموریت ۵ – ربات فراز – ت ۱



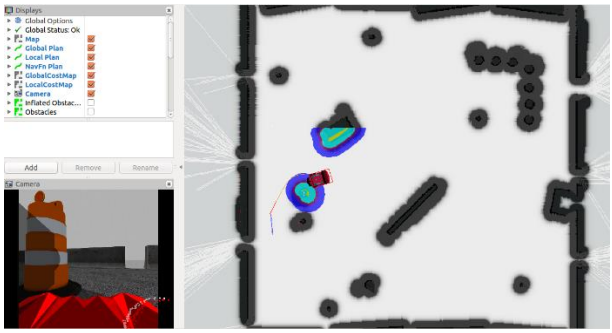
شکل (۴-۳۷) ماموریت ۵ – ربات فراز – ت ۲



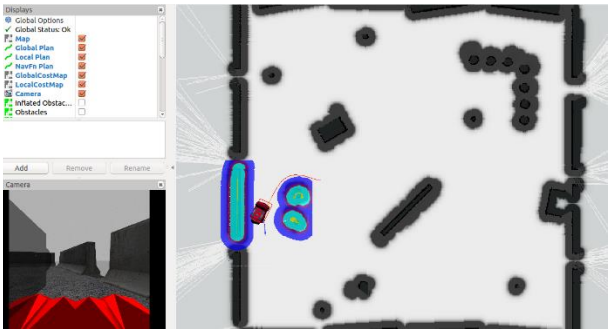
شکل (۴-۳۸) ماموریت ۵ – ربات فراز – ت ۳



شکل (۴-۴۰) ماموریت ۵ – ربات فراز – ت ۵



شکل (۴-۳۹) ماموریت ۵ – ربات فراز – ت ۴



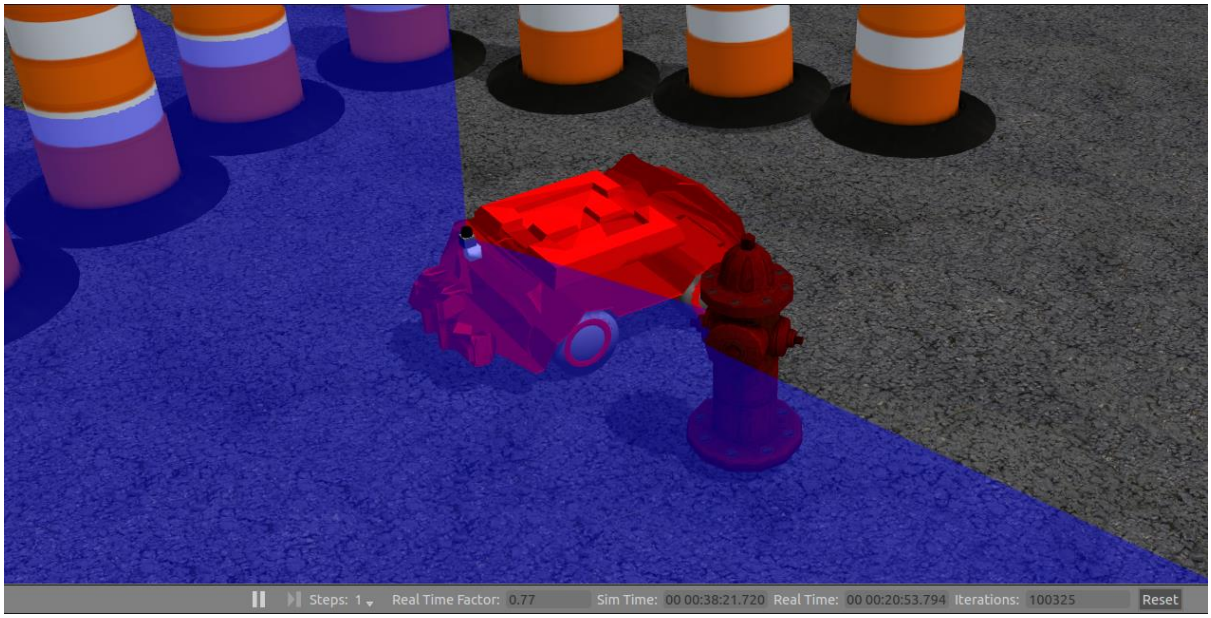
شکل (۴-۴۱) ماموریت ۵ – ربات فراز – ت ۶

گر چه این سیستم ناوبری، کاملا برای نیاز فعلی کافیست اما جای پیشرفت نیز دارد. با بهینه سازی کلی سیستم و پشتیبانی از عقب رفتن با سطح هوشمندی بالاتر و همچنین اضافه کردن سخت افزار های قوی تر برای پردازش سریعتر می توان در کل سیستم، پیشرفت چشم گیری مشاهده کرد. با سخت افزار فعلی در سرعت های خیلی بالا خطر برخورد به موانع متحرک وجود دارد. بنابراین با تجهیزات فعلی حداکثر سرعت محدود شده و نکات ایمنی در نظر گرفته شده است.

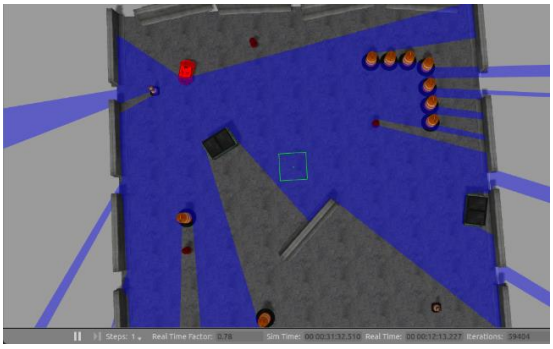
برای مشاهده ی بهتر ربات در محیط شبیه ساز سه بعدی تصاویر زیر آورده شده است. ربات در حالت ها و شکل های مختلف، برای بررسی و تحقیق بیشتر و یا حتی بهینه سازی گرافیک شبیه ساز به کار رفته و تصاویر تهیه شده اند تا خواننده بتواند ارتباط بهتری با مطالب موجود برقرار کند. در این تصاویر لیزر، دوربین، مدل سه بعدی و بهینه شده ی ربات، مدل برخورد و تصادف[۱] ربات، مدل شبکه ای و خطی، مدل سه بعدی جهان، و همچنین مرکز ثقل ربات مشخص است.

---

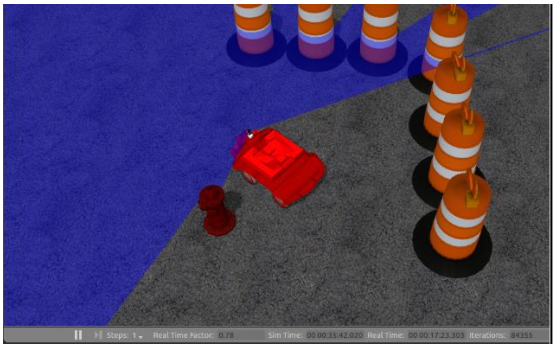
[۱] Collision



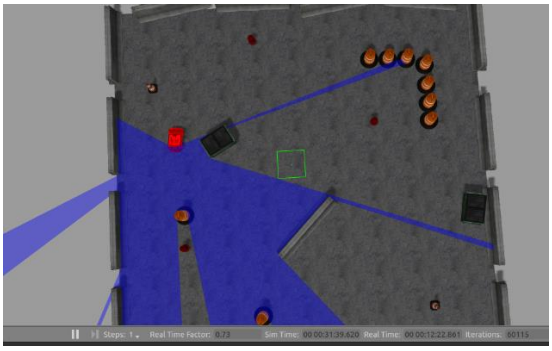
شکل (۴-۴۲) فراز در محیط شبیه ساز سه بعدی



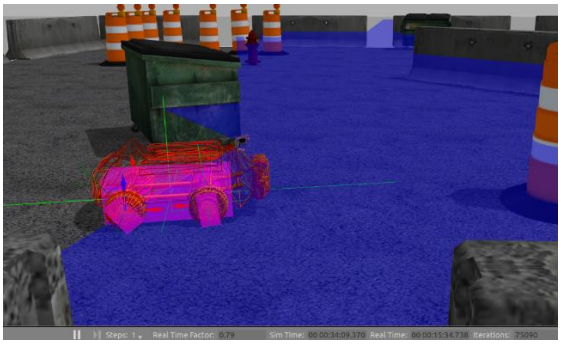
شکل (۴-۴۳) فراز در محیط شبیه ساز سه بعدی



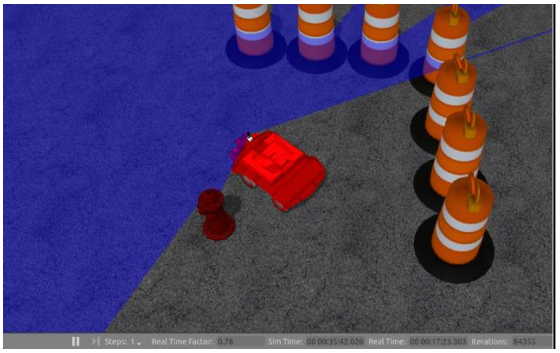
شکل (۴-۴۵) فراز در محیط شبیه ساز سه بعدی



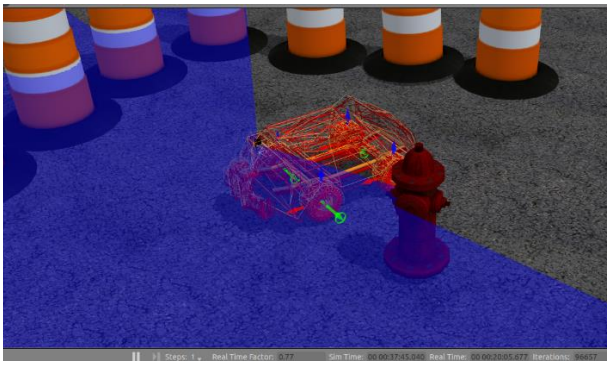
شکل (۴-۴۴) فراز در محیط شبیه ساز سه بعدی



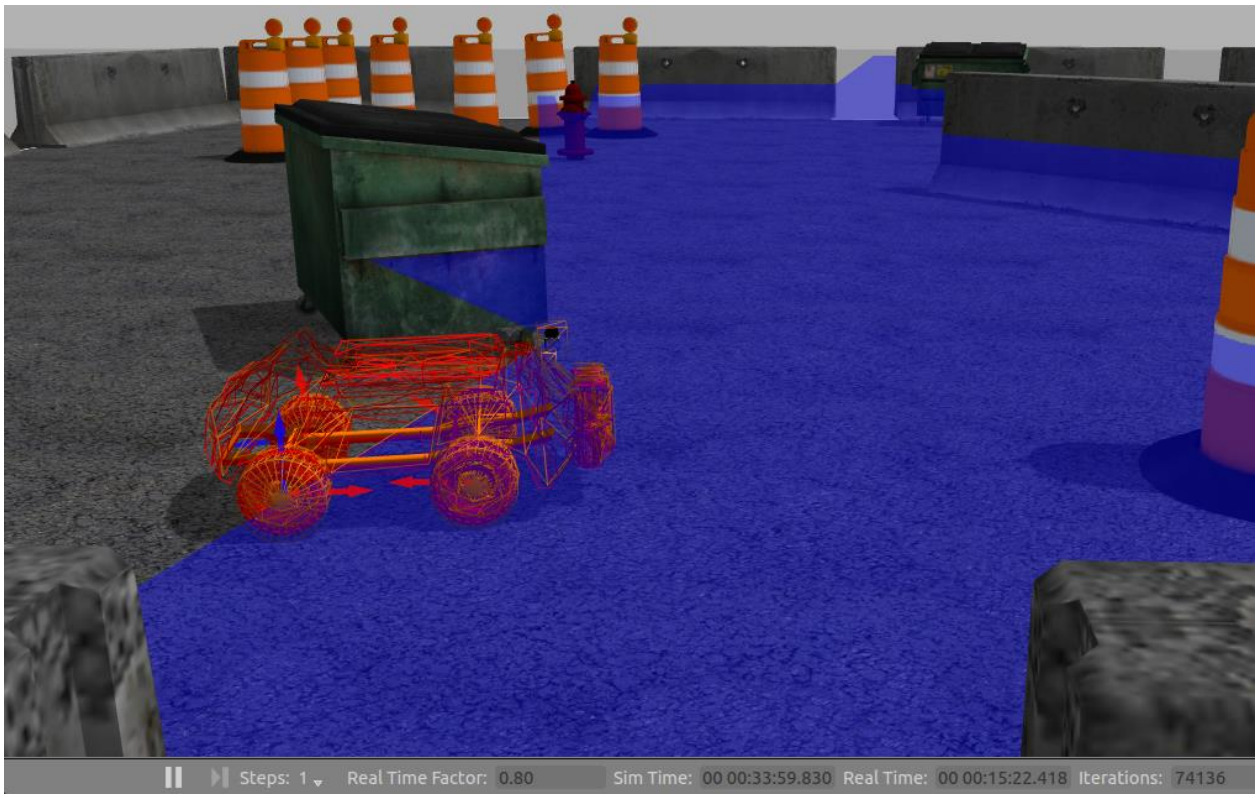
شکل (۴-۴۶) فراز در محیط شبیه ساز سه بعدی



شکل (۴-۴۷) فراز در محیط شبیه ساز سه بعدی



شکل (۴-۴۸) فراز در محیط شبیه ساز سه بعدی



شکل (۴-۴۹) فراز در محیط شبیه ساز سه بعدی

فصل پنجم

نتیجه‌گیری و پیشنهادها

مقدمه

در این فصل دستآوردهای این پروژه تشریح شده و به صورت خلاصه نتایج جمع بندی می شود.

۵-۱- نتیجه گیری و پیشنهادهای پژوهشی

۵-۱-۱- جمع‌بندی

در این پروژه، سطح پیچیده ای از طراحی ربات به کار گرفته شده ا ست. روبات از صفر طراحی شده و سپس مدل سه بعدی آن آماده شده است. از توانایی های سیستم های مختلف متن باز استفاده شده است و می توان گفت در کل طرح فقط نرم افزارها و اطلاعات متن باز به کار برده شـــده اند. با بخش های مختلف سیستم عامل رز به طور کامل کار شده و روبات بعد از توصیف به زبان URDF وارد شبیه ساز شده و عملکرد سیستم مورد ارزیابی قرار گرفته شد. همچنین سیستم ناوبری برای یک ربات جدید با موفقیت پیاده سازی شـده اسـت. در آخر می توان گفت در این پروژه از مجموع توانایی های هنری و مهندسـی اسـتفاده شـده تا خروجی بهتری به دست آید.

۵-۱-۲- نوآوری

خوشبختانه این تحقیق جنبه ها و موضوعات زیادی را پشتیبانی می کند. در بخش طراحی ماشین، تماما نوآوری و خلاقیت قابل مشـاهده اسـت. ربات فراز به طور کامل از صـفر و برحسـب نیاز ابتدا بر روی کاغذ طراحی شده و سپس در نرم افزارهای سه بعدی متن باز، مدلسازی شده است. در تمام این پروژه سعی شده ا ست فقط از نرم افزارها و الگوریتم های متن باز جهت ترویج این فرهنگ ا ستفاده گردد. همچنین سی ستم ناوبری کاملا برای فراز بهینه سازی شده است. از چندین Global_Planner و Local_Planner مختلف نظیر Base_Local_Planner، DWA_Local_Planner، SBPL و Lattice_Planner و ... برای اسـتفاده در این ر بات تسـت و ارز یابی صـورت گرف ته اسـت که ن هایـ تا به دلا یل مختلف و بهینگی بیشـتر از DWA_Local_Planner استفاده شد.

۵-۱-۳- پیشنهادها

در مجموع می توان گفت فرصت های بسیار زیاد و متنوعی برای ادامه ی این پروژه ی تحقیقاتی وجود دارد. پیشنهاد می شود ابتدا سنسورهای دیگر نظیر IMU، GPS و ... به ربات اضافه شوند. سیستم ناوبری بهینه تر شود و با استفاده از سنسور Sonar از عقب رفتن به صورت هوشمندتری پشتیبانی شود. از سنسورهایی نظیر لیدار[۱] برای درک و مدلسازی محیط اطراف به صورت سه بعدی استفاده شود. می توان از کتابخانه هایی نظیر OpenCV و PCL به ترتیب برای پردازش تصاویر و نقاط ابری استفاده کرده و ویژگی هایی نظیر تشخیص اجسام، انسان و .. به ربات اضافه گردد. به این ترتیب می توان سیستمی را پیاده سازی کرد که پس از شناسایی طراح و یا شخص مورد نظر از او پیروی کرده و به دنبال او حرکت کند. در بخش دیگر می توان از کتابخانه های موجود برای پردازش صوتی استفاده کرد و قابلیت فرامین صوتی نظیر خاموش/روشن، ایست/حرکت و ... به ربات فراز سایبرهندس اضافه شود. به طور کلی فرصت ها و موضوعات تحقیقاتی بسیار زیادی وجود دارد که می توان در این رابطه بر روی آن ها متمرکز شد و به نتایج جالبی دست یافت.

---

[۱] LiDAR

فهرست منابع

منابع غیر فارسی

[۱] S. Thrun“ ,What we’re driving at ”,Google, 9 October 2010 .Available: http://googleblog.blogspot.co.uk/2010/10/what-were-driving-at.html.

[۲] “Google Self-Driving Car Project Monthly Report ”,Google, Mountain View, CA, May 2015.

[۳] Will Maddern, Geoffrey Pascoe and Paul Newman“ ,Leveraging Experience for Large-Scale LIDAR Localisation in Changing Cities در ”,*Robotics and Automation (ICRA), 2015 IEEE International Conference on* , Seattle, WA , 2015 .

[۴] |“FAQ | Current status and next step ”,SARTRE-Consortium, 30 January 2011 . Available: http://www.sartre-project.eu/en/faq/current_status/Sidor/default.aspx .

[۵] Eric Chan, Andreas Ekfjorden, Paviter Jootel, Joshua Gidney, Arturo Dávila, Mattias Brännström, Daniel Skarin, Linda Wahlström“ ,SATRE PROJECT FINAL REPORT V14.,

[۶] “Volvo Cars presents a unique system solution for integrating self-driving cars into real traffic ”,Volvo Car Group, 19 February 2015 .Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/158276/volvo-cars-presents-a-unique-system-solution-for-integrating-self-driving-cars-into-real-traffic..

[۷] Volvo Group“ ,Volvo Car Group initiates world unique Swedish pilot project with self-driving cars on public roads ”,December 2013 ..Available: https://www.media.volvocars.com/global/en-gb/media/photos/136183/volvo-car-group-initiates-world-unique-swedish-pilot-project-with-self-driving-cars-on-public-roads1..

[۸] “Autonomous drive technology – Complete system solution ”,February 2015 .Available: https://www.media.volvocars.com/global/en-gb/media/photos/158389/autonomous-drive-technology-complete-system-solution .

[۹] “Autonomous drive technology - Surround radars ”,February 2015 ..Available: https://www.media.volvocars.com/global/en-gb/media/photos/158393/autonomous-drive-technology-surround-radars .

[۱۰] “Autonomous drive technology – Surround vision ”,February 2015 .Available: https://www.media.volvocars.com/global/en-gb/media/photos/158392/autonomous-drive-technology-surround-vision .

[۱۱] “Autonomous drive technology – Multiple beam laser scanner ”,February 2015 .
Available: https://www.media.volvocars.com/global/en-gb/media/photos/158391/autonomous-drive-technology-multiple-beam-laser-scanner

[۱۲] “Autonomous drive technology – Ultrasonic sensors ”,February 2015 .Available:
https://www.media.volvocars.com/global/en-gb/media/photos/158390/autonomous-drive-technology-ultrasonic-sensors

[۱۳] J. E. R. Rosales“ ,Obstacle avoidance for mobile robots using lidar sensors ”,September 2011.

[۱۴] A. M. Romero“ ,ROS Concepts ”,ROS, 21 June 2014 .Available:
http://wiki.ros.org/ROS/Concepts..

[۱۵] ROS“ ,attachment:ROS_basic_concepts.dia of ROS/Concepts ”,August 2009 .Available:
http://wiki.ros.org/ROS/Concepts?action=AttachFile&do=view&target=ROS_basic_concepts.dia ۳۱ دستیابی در] .August 2015.[

[۱۶] R. Vaughan“ ,Massively multi-robot simulation in stage ”,*Swarm Intell* ,p. 189,–

[۱۷] Brian Gerkey, Richard T. Vaughan and Andrew Howard“ ,The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems در ”,*Proceedings of the 11th International Conference on Advanced Robotics* ,Coimbra, Portugal, 2003 .

[۱۸] J. H. N. Koenig“ ,Gazebo در ”,*ROSCon* 2012 .

[۱۹] Wikipedia“ ,File:Standard deviation diagram.svg ۰۱ ”,June 2015 .Available:
https://commons.wikimedia.org/wiki/File:Standard_deviation_diagram.svg?uselang=en .

[۲۰] L. Kleeman ,*Understanding and Applying Kalman Filtering* ,p. 37.

[۲۱] A. Boeing“ ,Kalman Filters ”,May Available:
http://adrianboeing.blogspot.com/2010/05/kalman-filters.html .

[۲۲] A. Boeing“ ,Kalman Filters ”,May Available:
http://4.bp.blogspot.com/_nHcDA3NyAms/S9wgQYzoE6I/AAAAAAAAAQk/zR9YElumxlU/s1600/kalmaneq.png..

[۲۳] Greg Welch, Gary Bishop ,*An Introduction to the Kalman Filter* ,Chapel Hill: University of North Carolina at Chapel Hill, 2006 .

[۲۴] Thrun, S. and Fox, D. and Burgard, W. and Dellaert, F“ ,.Robust Monte Carlo Localization for Mobile Robots ”,*Artificial Intelligence* , ,pp. 99--141, 2000 .

[۲۵] Sebastian Thrun, Dieter Fox, Wolfram Burgard, PROBABILISTIC ROBOTICS, 2000 .

[۲۶] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard“ ,Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling ”,*IEEE International Conference on Robotics and Automation (ICRA) .,*

[۲۷] A. Boeing“ ,Dynamic Window Algorithm motion planning ”,May .Available: http://adrianboeing.blogspot.com/2012/05/dynamic-window-algorithm-motion.html .

[۲۸] A. Boeing“ ,Dynamic Window Algorithm motion planning ”,May 2012" Available: http://adrianboeing.blogspot.com/2012/05/dynamic-window-algorithm-motion.html] .

[۲۹] “Attachment 'overview_tf.png ۲۵ ”,'September 2009 .Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup?action=AttachFile&do=view&target=overview_tf.png .

[۳۰] Open Source Robotic Foundation“ ,About ROS ”,20 July 2015 .Available: http://www.ros.org/about-ros./

[۳۱] “The Player Project ”,16 February 2014 ..Available: http://playerstage.sourceforge.net.

پیوست‌ها

پیوست الف- کدهای منبع نرم افزاری پروژه ی فراز سایبرهندس

جهت استفاده، توزیع و یا انتشار کد لطفا به متن زیر توجه کنید. تمام کدهای منبع موجود در این پایان نامه حداقل شامل این قانون می شوند مگر اینکه به طور مستقیم در فایل، ذکر شده باشد که از لایسنس دیگری استفاده شده است.

```
/*********************************************************************
*
* Software License Agreement (BSD License)
*
*  Copyright (c) 2015, Mojtaba Válipour @ CyberHands.org.
*  All rights reserved.
*
*  Redistribution and use in source and binary forms, with or without
*  modification, are permitted provided that the following conditions
*  are met:
*
*   * Redistributions of source code must retain the above copyright
*     notice, this list of conditions and the following disclaimer.
*   * Redistributions in binary form must reproduce the above
*     copyright notice, this list of conditions and the following
*     disclaimer in the documentation and/or other materials provided
*     with the distribution.
*     Neither the name of CyberHands.org nor the names of its
*     contributors may be used to endorse or promote products derived from
*     this software without specific prior written permission.
*
*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
*  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
*  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
*  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
*  COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
*  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
*  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
*  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
*  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
*  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
*  ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
*  POSSIBILITY OF SUCH DAMAGE.
*
* Author: Mojtaba Válipour @ CyberHands.org
*
*********************************************************************/
```

نام فایل: chugv_navigation_gmapping.launch

```
<launch>
    <param name="/use_sim_time" value="true" />

    <!-- Launch Gazebo and World -->

        <include file="$(find
chugv_navigation)/launch/chugv_gazebo.launch"/>
```

```
  <!-- CyberHands FARAZ UGV description loaded on the Parameter Server -->
  <param name="robot_description" command="$(find xacro)/xacro.py '$(find
chugv_description)/urdf/chugv.urdf.xacro'" />

  <!-- spawn CyberHands FARAZ UGV's model -->
  <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model"
output="screen"
    args="-urdf -param robot_description -model chugv" />

  <!—These are Robot’s Joints and TF and other links that should publish-->
    <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher"/>
    <node pkg="robot_state_publisher" type="robot_state_publisher"
name="robot_state_publisher" output="screen"/>
    <node pkg="tf" type="static_transform_publisher"
name="laser_link_broadcaster" args="0 0 1 0 0 0 base_link laser_link 100"/>

    <!-- Run Navigation Stack with SLAM Gmapping OR AMCL & Map SERVER -->

    <!--node name="map_server" pkg="map_server" type="map_server"
args="$(find chugv_navigation)/maps/CHScanMap.yaml"/-->

  <include file="$(find
chugv_navigation)/move_base_config/slam_gmapping.xml"/>

  <include file="$(find chugv_navigation)/move_base_config/move_base.xml"/>

    <!--include file="$(find
chugv_navigation)/move_base_config/amcl_node.xml"/>
      <param name="amcl/initial_pose_x" value="0"/>
      <param name="amcl/initial_pose_y" value="0"/-->

    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
chugv_navigation)/single_robot.rviz" />
</launch>
```

نام فایل: chugv_gazebo.launch

```
<launch>
  <!-- roslaunch and RVIZ arguments -->
  <arg name="show_rviz" default="true"/>
  <arg name="paused" default="false"/>
  <arg name="debug" default="false"/>
  <arg name="gui" default="true"/>
  <!-- We resume the logic in empty_world.launch, changing only the name of
    the world to be launched -->
  <include file="$(find chugv_navigation)/launch/empty_world.launch">
    <arg name="world_name" value="$(find
chugv_navigation)/worlds/clearpath_playpen.world" />
    <arg name="paused" value="$(arg paused)" />
    <arg name="debug" value="$(arg debug)" />
    <arg name="gui" value="$(arg gui)" />
    <arg name="use_sim_time" value="true" />
    <arg name="headless" value="false" />
  </include>
</launch>
```

نام فایل: empty_world.launch

```
<launch>
  <!-- these are the arguments you can pass this launch file, for example
paused:=true -->
```

```
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="extra_gazebo_args" default=""/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>
  <arg name="verbose" default="false"/>
  <arg name="world_name" default="worlds/empty.world"/> <!-- Note: the
world_name is with respect to GAZEBO_RESOURCE_PATH environmental variable -
->
  <!-- set use_sim_time flag -->
  <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

  <!-- set command arguments -->
  <arg unless="$(arg paused)" name="command_arg1" value=""/>
  <arg     if="$(arg paused)" name="command_arg1" value="-u"/>
  <arg unless="$(arg headless)" name="command_arg2" value=""/>
  <arg     if="$(arg headless)" name="command_arg2" value="-r"/>
  <arg unless="$(arg verbose)" name="command_arg3" value=""/>
  <arg     if="$(arg verbose)" name="command_arg3" value="--verbose"/>
  <arg unless="$(arg debug)" name="script_type" value="gzserver"/>
  <arg     if="$(arg debug)" name="script_type" value="debug"/>

  <!-- start gazebo server-->
  <node name="gazebo" pkg="gazebo_ros" type="$(arg script_type)"
respawn="false" output="screen"
    args="$(arg command_arg1) $(arg command_arg2) $(arg command_arg3) -e
$(arg physics) $(arg extra_gazebo_args) $(arg world_name)" />

  <!-- start gazebo client - This one is commented for optimizing process
for poor and old Hardware - For New computer with powerful hardware
uncomment it to see Robots in 3D World. -->
  <!--
  <group if="$(arg gui)">
    <node name="gazebo_gui" pkg="gazebo_ros" type="gzclient"
respawn="false" output="screen"/>
  </group>
  -->
</launch>
```

نام فایل: chugv.urdf.xacro

```
<?xml version="1.0"?>
<robot name="chugv" xmlns:xacro="http://www.ros.org/wiki/xacro">

  <!—This is a URDF/XACRO description of CyberHands FARAZ UGV -->

  <!-- Included URDF/XACRO Files -->
  <xacro:include filename="$(find
chugv_description)/urdf/materials.urdf.xacro" />
  <xacro:include filename="$(find chugv_description)/urdf/wheel.urdf.xacro"
/>

  <!-- PROPERTY LIST -->
  <!--All units in m-kg-s-radians unit system -->
  <property name="M_PI" value="3.1415926535897931" />

  <!-- Main Body-base -->
```

```
  <property name="base_x_size" value="1" /> <!-- 1.0 0.5 0.25-->
  <property name="base_y_size" value="0.5" /> <!-- 1.25 0.82 0.42 -->
  <property name="base_z_size" value="0.25" />
  <property name="base_mass" value="35" /> <!-- in kg--> <!-- 35 -->

  <property name="camera_link" value="0.05" /> <!-- Size of square 'camera'
box -->

  <!-- Sensors - Hokuyo -->
  <!--property name="height3" value="1" /-->
  <property name="axel_offset" value="0.15" />

  <!--Inertial macros for the box and cylinder. Units are kg*m^2-->
  <macro name="box_inertia" params="m x y z">
    <inertia  ixx="${m*(y*y+z*z)/12}" ixy = "0" ixz = "0"
              iyy="${m*(x*x+z*z)/12}" iyz = "0"
              izz="${m*(x*x+z*z)/12}" />
  </macro>

  <!-- BASE-FOOTPRINT -->
  <!-- base_footprint is a fictitious link(frame) that is on the ground
right below base_link origin -->
  <link name="base_footprint">
    <!--inertial>
      <mass value="0.0001" />
      <origin xyz="0 0 0" />
      <inertia ixx="0.0001" ixy="0.0" ixz="0.0"
          iyy="0.0001" iyz="0.0"
          izz="0.0001" />
    </inertial-->
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
            <box size="0.001 0.001 0.001" />
        </geometry>
    </visual>
  </link>

  <gazebo reference="base_footprint">
    <turnGravityOff>false</turnGravityOff>
  </gazebo>

  <joint name="base_footprint_joint" type="fixed">
  <origin xyz="0 0 ${wheel_radius - base_z_origin_to_wheel_origin}" rpy="0
0 0" />
    <parent link="base_footprint"/>
    <child link="base_link" />
  </joint>

  <!-- BASE-LINK -->
  <!--Actual body/chassis of the robot-->
  <link name="base_link">
    <inertial>
      <mass value="${base_mass}" />
      <origin xyz="0 0 0" />
      <!--The 3x3 rotational inertia matrix. -->
      <box_inertia  m="${base_mass}" x="${base_x_size}" y="${base_y_size}"
z="${base_z_size}"/>
    </inertial>
    <visual>
```

```
      <!--origin xyz="0 0 0" rpy="0 0 0" /-->
      <origin xyz="0.05 0 -0.095" rpy="0 0 0" />
      <!-- 0.35 0.3129348 0.2671349 -->
      <geometry>
      <mesh filename="package://chugv_description/meshes/chugv.dae"/>
        <!--box size="${base_x_size} ${base_y_size} ${base_z_size}"/--> <!-
- 1 0.5 0.25-->
      </geometry>
      <material name="Red" /> <!-- Yellow -->
    </visual>
    <collision>
      <!--origin xyz="0 0 0" rpy="0 0 0" /-->
      <!--origin xyz="0 0 0.17" rpy="0 0 0 " /-->
      <origin xyz="0.08 0 -0.095" rpy="0 0 0" />
      <geometry>
<mesh filename="package://chugv_description/meshes/chugv_coll.dae"/>
        <!--box size="${base_x_size} ${base_y_size} ${base_z_size}"/--> <!-
- 1 0.5 0.25 -->
      </geometry>
    </collision>
  </link>
  <gazebo reference="base_link">
    <material>Gazebo/Red</material>     <!-- Yellow -->
    <turnGravityOff>false</turnGravityOff>
  </gazebo>

  <!-- WHEELs -->
  <wheel fb="front" lr="right" parent="base_link" translateX="1"
translateY="-1" flipY="-1"/>
  <wheel fb="front" lr="left" parent="base_link" translateX="1"
translateY="1" flipY="-1"/>
  <wheel fb="back" lr="right" parent="base_link" translateX="-1"
translateY="-1" flipY="-1"/>
  <wheel fb="back" lr="left" parent="base_link" translateX="-1"
translateY="1" flipY="-1"/>

  <joint name="hokuyo_joint" type="fixed">
    <axis xyz="0 1 0" />
    <origin xyz="0.58 0 ${base_z_size + axel_offset}" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="hokuyo_link"/>
  </joint>

  <!-- Hokuyo Laser -->
  <link name="hokuyo_link">
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="0.1 0.1 0.1"/>
      </geometry>
    </collision>

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh
filename="package://chugv_description/meshes/sensors/hokuyo.dae"/>
      </geometry>
    </visual>
```

```
    <inertial>
      <mass value="1e-5" />
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
    </inertial>
  </link>

  <!-- hokuyo -->
  <gazebo reference="hokuyo_link">
    <sensor type="ray" name="head_hokuyo_sensor">
      <pose>0 0 0 0 0 0</pose>
      <visualize>true</visualize>
      <update_rate>40</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>720</samples>
            <resolution>1</resolution>
        <!--min_angle>-1.570796</min_angle-->
            <min_angle>-2</min_angle>
            <max_angle>2</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.10</min>
          <max>30.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <!-- Noise parameters based on published spec for Hokuyo laser
achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m and stddev of
0.01m will put 99.7% of samples within 0.03m of the true
reading. -->
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin name="gazebo_ros_head_hokuyo_controller"
filename="libgazebo_ros_laser.so">
        <topicName>base_scan</topicName>
        <!--frameName>hokuyo_link</frameName-->
        <frameName>laser_link</frameName>
      </plugin>
    </sensor>
  </gazebo>

  <!-- Camera -->
  <joint name="camera_joint" type="fixed">
    <axis xyz="0 1 0" />
    <origin xyz="${base_x_size/2-camera_link/2+0.09} 0 ${base_z_size/2 +
camera_link/2+0.18}" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="camera_link"/>
  </joint>

  <link name="camera_link">
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
```

```
        <box size="${camera_link} ${camera_link} ${camera_link}"/>
      </geometry>
    </collision>

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="${camera_link} ${camera_link} ${camera_link}"/>
      </geometry>
      <material name="black"/>
    </visual>

    <inertial>
      <mass value="1e-5" />
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
    </inertial>
  </link>


  <gazebo>
<!-- Control Mobile Robot -->
    <plugin name="skid_steer_drive_controller"
filename="libgazebo_ros_skid_steer_drive.so">
      <alwaysOn>true</alwaysOn>
      <updateRate>100.0</updateRate>
      <leftFrontJoint>front_left_wheel_joint</leftFrontJoint>
      <rightFrontJoint>front_right_wheel_joint</rightFrontJoint>
      <leftRearJoint>back_left_wheel_joint</leftRearJoint>
      <rightRearJoint>back_right_wheel_joint</rightRearJoint>
      <wheelSeparation>${base_y_size}</wheelSeparation>
      <wheelDiameter>${2*wheel_radius}</wheelDiameter>
      <torque>35</torque>
      <broadcastTF>1</broadcastTF>
      <odometryFrame>odom</odometryFrame>
      <commandTopic>cmd_vel</commandTopic>
      <odometryTopic>odom</odometryTopic>
      <robotBaseFrame>base_footprint</robotBaseFrame>
    </plugin>
<!-- A plugin for control robot - The basic control -->
<plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so" />
      <plugin name="p3d_base_controller" filename="libgazebo_ros_p3d.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>100.0</updateRate>
        <bodyName>base_link</bodyName>
        <topicName>base_pose_ground_truth</topicName>
        <gaussianNoise>0</gaussianNoise>
        <frameName>/map</frameName>
        <xyzOffsets>0 0 0</xyzOffsets>
        <rpyOffsets>0 0 0</rpyOffsets>
      </plugin>
</gazebo>

  <gazebo reference="camera_link">
    <sensor type="camera" name="camera1">
      <update_rate>30.0</update_rate>
      <camera name="head">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>800</width>
```

```
          <height>800</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.02</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <!-- Noise is sampled independently per pixel on each frame.
               That pixel's noise value is added to each of its color
               channels, which at that point lie in the range [0,1]. -->
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
        <cameraName>chugv/camera</cameraName>
        <imageTopicName>image_raw</imageTopicName>
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>camera_link</frameName>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
    </sensor>
  </gazebo>
</robot>
```

نام فایل: materials.urdf.xacro

```
<?xml version="1.0"?>
<robot>
  <material name="Black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>

  <material name="Blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>

  <material name="Green">
    <color rgba="0.0 0.8 0.0 1.0"/>
  </material>

  <material name="Gray">
    <color rgba="0.7 0.7 0.7 1.0"/>
  </material>

  <material name="DarkGray">
    <color rgba="0.3 0.3 0.3 1.0"/>
  </material>

  <material name="Red">
    <color rgba="0.8 0.0 0.0 1.0"/>
  </material>
```

```
  <material name="White">
    <color rgba="1.0 1.0 1.0 1.0"/>
  </material>

  <material name="Yellow">
    <color rgba="0.8 0.8 0.0 1.0"/>
  </material>
</robot>
```

نام فایل: wheel.urdf.xacro

```
<?xml version="1.0"?>
<robot name="wheel" xmlns:xacro="http://www.ros.org/wiki/xacro">

  <!-- Wheels -->
  <property name="wheel_radius" value="0.14" /> <!-- 0.2 0.14 -->
  <property name="wheel_height" value="0.14" />
  <property name="wheel_mass" value="2.5" /> <!-- in kg-->

  <property name="base_x_origin_to_wheel_origin" value="0.3" />
<!-- ORIGIN 0.25 0.3 0.0-->
  <property name="base_y_origin_to_wheel_origin" value="0.3" />
  <property name="base_z_origin_to_wheel_origin" value="0.0" />

  <macro name="cylinder_inertia" params="m r h">
    <inertia  ixx="${m*(3*r*r+h*h)/12}" ixy = "0" ixz = "0"
              iyy="${m*(3*r*r+h*h)/12}" iyz = "0"
              izz="${m*r*r/2}" />
  </macro>

  <xacro:macro name="wheel" params="fb lr parent translateX translateY
flipY"> <!--fb : front, back ; lr: left, right -->
    <link name="${fb}_${lr}_wheel">
      <visual>
        <origin xyz="0 0 0" rpy="${flipY*M_PI/2} 0 0" />
        <geometry>
          <cylinder length="${wheel_height}" radius="${wheel_radius}" />
        </geometry>
        <material name="DarkGray" />
      </visual>
      <collision>
        <origin xyz="0 0 0" rpy="${flipY*M_PI/2} 0 0" />
        <geometry>
          <cylinder length="${wheel_height}" radius="${wheel_radius}" />
        </geometry>
      </collision>
      <inertial>
        <mass value="${wheel_mass}" />
        <origin xyz="0 0 0" />
        <cylinder_inertia  m="${wheel_mass}" r="${wheel_radius}"
h="${wheel_height}" />
      </inertial>
    </link>

    <gazebo reference="${fb}_${lr}_wheel">
      <mu1 value="1.0"/>
      <mu2 value="1.0"/>
      <kp  value="10000000.0" />
      <kd  value="1.0" />
```

```
      <fdir1 value="1 0 0"/>
      <material>Gazebo/Grey</material>
      <turnGravityOff>false</turnGravityOff>
    </gazebo>

    <joint name="${fb}_${lr}_wheel_joint" type="continuous">
      <parent link="${parent}"/>
      <child link="${fb}_${lr}_wheel"/>
      <origin xyz="${translateX * base_x_origin_to_wheel_origin}
${translateY * base_y_origin_to_wheel_origin}
${base_z_origin_to_wheel_origin}" rpy="0 0 0" />
      <axis xyz="0 1 0" rpy="0 0 0" />
      <limit effort="100" velocity="100"/>
      <joint_properties damping="0.0" friction="0.0"/>
    </joint>

    <!-- Transmission is important to link the joints and the controller --
>
    <transmission name="${fb}_${lr}_wheel_joint_trans">
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="${fb}_${lr}_wheel_joint" />
      <actuator name="${fb}_${lr}_wheel_joint_motor">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

  </xacro:macro>
</robot>
```

نام فایل: slam_gmapping.xml

```
<launch>
    <!—This Parameter used for configuration of SLAM Gmapping ROS Node -->
    <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"
output="screen">
      <remap from="scan" to="base_scan"/>
      <param name="odom_frame" value="odom"/>
      <param name="map_update_interval" value="30.0"/>
      <param name="maxUrange" value="16.0"/>
      <param name="sigma" value="0.05"/>
      <param name="kernelSize" value="1"/>
      <param name="lstep" value="0.05"/>
      <param name="astep" value="0.05"/>
      <param name="iterations" value="5"/>
      <param name="lsigma" value="0.075"/>
      <param name="ogain" value="3.0"/>
      <param name="lskip" value="0"/>
      <param name="srr" value="0.01"/>
      <param name="srt" value="0.02"/>
      <param name="str" value="0.01"/>
      <param name="stt" value="0.02"/>
      <param name="linearUpdate" value="0.5"/>
      <param name="angularUpdate" value="0.436"/>
      <param name="temporalUpdate" value="-1.0"/>
      <param name="resampleThreshold" value="0.5"/>
      <param name="particles" value="80"/>
      <param name="xmin" value="-50.0"/>
      <param name="ymin" value="-50.0"/>
```

```
      <param name="xmax" value="50.0"/>
      <param name="ymax" value="50.0"/>
      <param name="delta" value="0.05"/>
      <param name="llsamplerange" value="0.01"/>
      <param name="llsamplestep" value="0.01"/>
      <param name="lasamplerange" value="0.005"/>
      <param name="lasamplestep" value="0.005"/>
    </node>
</launch>
```

نام فایل: move_base.xml

```
<launch>
<!-- Move_Base configuration -->
  <node pkg="move_base" type="move_base" respawn="false"
name="move_base_node" output="screen">
    <!--param name="footprint_padding" value="0.04" /-->
    <param name="controller_frequency" value="6.5" /> <!-- Default: 10.0
GOOD: 5.0 Best: 6.5-->
    <param name="controller_patience" value="35.0" /><!-- Default: 3.0
GOOD: 35.0   Best: -->
    <param name="oscillation_timeout" value="40.0" /><!-- Default: 30.0
GOOD: 40.0  Best: -->
    <param name="oscillation_distance" value="0.25" /><!-- Default: 0.5
GOOD: .0  Best: -->
    <param name="recovery_behavior_enabled" value="false" /><!-- GOOD:
false -->
    <param name="clearing_rotation_allowed" value="true" /><!-- GOOD: false
-->

    <param name="base_local_planner"
value="dwa_local_planner/DWAPlannerROS" />
    <!--param name="base_global_planner"
value="global_planner/GlobalPlanner"/-->
    <!--param name="base_global_planner"
value="RAstar_planner/RAstarPlannerROS"/-->
    <!--param name="base_global_planner"
value="carrot_planner/CarrotPlanner"/-->


    <rosparam file="$(find
chugv_navigation)/move_base_config/costmap_common_params.yaml"
command="load" ns="global_costmap" />
    <rosparam file="$(find
chugv_navigation)/move_base_config/costmap_common_params.yaml"
command="load" ns="local_costmap" />
    <rosparam file="$(find
chugv_navigation)/move_base_config/local_costmap_params.yaml"
command="load" />
    <rosparam file="$(find
chugv_navigation)/move_base_config/global_costmap_params.yaml"
command="load" />

    <!--rosparam file="$(find
chugv_navigation)/move_base_config/base_local_planner_params.yaml"
command="load" /-->
```

```
    <rosparam file="$(find
chugv_navigation)/move_base_config/dwa_local_planner_params.yaml"
command="load" />
    <!--
    <rosparam file="$(find
turtlebot_navigation)/param/navfn_global_planner_params.yaml"
command="load" />
    -->

  </node>
</launch>
```

نام فایل: costmap_common_params.yaml

```
# This file contains common configuration options for the two costmaps used
in the ROS navigation stack. This is a good Configuration for CyberHands
FARAZ UGV.

#configure the costmap in voxel-grid mode
map_type: voxel

#Voxel grid specific parameters
origin_z: 0.0
z_resolution: 0.2
z_voxels: 10
unknown_threshold: 9
mark_threshold: 0

#Set the tolerance we're willing to have for tf transforms
transform_tolerance: 0.3

#Obstacle marking parameters
obstacle_range: 2.5
max_obstacle_height: 2.0
raytrace_range: 3.0

#The footprint of the robot and associated padding
footprint: [[-0.6, -0.4], [-0.6, 0.4], [0.6, 0.4], [0.6, 0.4], [0.6, -0.4]]
footprint_padding: 0.01 #Default: 0.01

#Cost function parameters
inflation_radius: 10.55 #Default: 0.55
cost_scaling_factor: 10.0 #Default: 10.0

#The cost at which a cell is considered an obstacle when a map is read from
the map_server
lethal_cost_threshold: 100

#Configuration for the sensors that the costmap will use to update a map
observation_sources: base_scan

#STAGE - If you want run STAGE instead of GAZEBO, You should uncomment this
and comment GAZEBO Block here.
#base_scan: {data_type: LaserScan, expected_update_rate: 0.4,
observation_persistence: 0.0, marking: true, clearing: true,
max_obstacle_height: 0.4, min_obstacle_height: 0.08}

#GAZEBO
base_scan: {sensor_frame: laser_link, data_type: LaserScan, topic:
base_scan, marking: true, clearing: true}
```

```
#use_dijkstra: false
```

نام فایل: local_costmap_params.yaml

```
#Independent settings for the local planner's costmap. This is a good
Configuration for CyberHands FARAZ UGV.

local_costmap:
  # publish the voxel grid used by this costmap
  publish_voxel_map: true

  # Set the global and robot frames for the costmap
  global_frame: odom
  robot_base_frame: base_link

  #Set the update and publish frequency of the costmap
  update_frequency: 5.0
  publish_frequency: 2.0

  # configure this costmap to be a rolling window... meaning it is always
  #centered at the robot. If you want Static map instead then you must
change rolling_window parameter to false. (This is not recommended for
local planner)
  static_map: false
  rolling_window: true
  width: 6.0
  height: 6.0
  resolution: 0.025
  origin_x: 0.0
  origin_y: 0.0
```

نام فایل: global_costmap_params.yaml

```
#Independent settings for the global planner's costmap. This is a good
Configuration for CyberHands FARAZ UGV.
global_costmap:
  #Set the global and robot frames for the costmap
  global_frame: /map
  robot_base_frame: base_link

  #Set the update and publish frequency of the costmap
  update_frequency: 5.0
  publish_frequency: 2.0

  # use a static map
  static_map: true
  rolling_window: false

  #footprint_padding: 0.02 #Default: 0.02
```

نام فایل: base_local_planner_params.yaml

```
# This is a good Configuration for CyberHands FARAZ UGV.
# parameters Configuration for TrajectoryPlannerROS

TrajectoryPlannerROS:
  #Set the acceleration limits of the robot
  acc_lim_th: 3.2
  acc_lim_x: 2.5

  acc_lim_y: 0 #Default: 2.5
```

```
  max_vel_y: 0
  min_vel_y: 0
  vy_samples: 0

  #Set the velocity limits of the robot
  max_vel_x: 0.65
  min_vel_x: 0.1
  max_rotational_vel: 1.0
  min_in_place_rotational_vel: 0.4

  #The velocity the robot will command when trying to escape from a stuck
situation
  escape_vel: -0.1

  # CyberHands FARAZ UGV is not  Holonomic ROOBOT
  holonomic_robot: false

  #Set the tolerance on achieving a goal
  xy_goal_tolerance: 0.1
  yaw_goal_tolerance: 0.05

  #We'll configure how long and with what granularity we'll forward
simulate trajectories
  sim_time: 1.7
  sim_granularity: 0.025
  vx_samples: 3
  vtheta_samples: 20

  #Parameters for scoring trajectories
  goal_distance_bias: 0.8
  path_distance_bias: 0.6
  occdist_scale: 0.01
  heading_lookahead: 0.325

  # use the Dynamic Window Approach to control instead of Trajectory
Rollout for this example
  dwa: true

  #How far the robot must travel before oscillation flags are reset
  oscillation_reset_dist: 0.05

  #Eat up the plan as the robot moves along it
  prune_plan: true
```

نام فایل: dwa_local_planner_params.yaml

```
# This is a good Configuration for CyberHands FARAZ UGV.
# DWAPlannerROS Parameters Configuration

DWAPlannerROS:
  acc_lim_th: 3.2
  acc_lim_x: 2.5
  acc_lim_y: 0 #Default: 2.5

  max_vel_x: 0.55 #Default: 0.65
  min_vel_x: -0.25 #Default: 0.0

  # Our Robot is not a Holonomic Robot, He doesn’t need y Velocity.
  max_vel_y: 0
  min_vel_y: 0
```

```
  max_trans_vel: 0.65 #Default: 0.65
  min_trans_vel: 0.1

  max_rot_vel: 1.0 #Default: 1.0
  min_rot_vel: 0.4

  sim_time: 1.0 #Default: 1.7 GOOD: 1.0
  sim_granularity: 0.025 #Default: 0.025
  angular_sim_granularity: 0.025  #Default: 0.025

  goal_distance_bias: 24.0 #Default: 32.0
  path_distance_bias: 92.0 #Default: 24.0 GOOD: 52.0
  occdist_scale: 0.01 #Default: 0.01

  stop_time_buffer: 0.2 #Default: 0.2
  oscillation_reset_dist: 0.05

  forward_point_distance: 0.525 #Default: 0.325 GOOD: 0.625 Best:0.525

  scaling_speed: 0.25
  max_scaling_factor: 0.2 #Default: 0.2

  vx_samples: 10 #Default: 3 GOOD: 10
  vy_samples: 0 #Default: 10
  vtheta_samples: 20 #Default: 20 GOOD:20

  sim_period: 0.1

  xy_goal_tolerance: 0.2 #Default: 0.2
  yaw_goal_tolerance: 0.17 #Default: 0.17 GOOD: 10.17
  #latch_xy_goal_tolerance: true #Default: false

  rot_stopped_vel: 0.01
  trans_stopped_vel: 0.01

  use_dwa: true
  prune_plan: true
```

نام فایل: amcl_node.xml

```
<launch>
<!--
  amcl configuration
-->
<node pkg="amcl" type="amcl" name="amcl" respawn="true">
  <remap from="scan" to="base_scan" />
  <!-- Publish scans from best pose at a max of 10 Hz -->
  <param name="odom_model_type" value="omni"/>
  <param name="odom_alpha5" value="0.1"/>
  <param name="transform_tolerance" value="0.2" />
  <param name="gui_publish_rate" value="10.0"/>
  <param name="laser_max_beams" value="30"/>
  <param name="min_particles" value="500"/>
  <param name="max_particles" value="5000"/>
  <param name="kld_err" value="0.05"/>
  <param name="kld_z" value="0.99"/>
  <param name="odom_alpha1" value="0.2"/>
  <param name="odom_alpha2" value="0.2"/>
```

```
  <!-- translation std dev, m -->
  <param name="odom_alpha3" value="0.8"/>
  <param name="odom_alpha4" value="0.2"/>
  <param name="laser_z_hit" value="0.5"/>
  <param name="laser_z_short" value="0.05"/>
  <param name="laser_z_max" value="0.05"/>
  <param name="laser_z_rand" value="0.5"/>
  <param name="laser_sigma_hit" value="0.2"/>
  <param name="laser_lambda_short" value="0.1"/>
  <param name="laser_lambda_short" value="0.1"/>
  <param name="laser_model_type" value="likelihood_field"/>
  <!-- <param name="laser_model_type" value="beam"/> -->
  <param name="laser_likelihood_max_dist" value="2.0"/>
  <param name="update_min_d" value="0.2"/>
  <param name="update_min_a" value="0.5"/>
  <param name="odom_frame_id" value="odom"/>
  <param name="resample_interval" value="1"/>
  <param name="transform_tolerance" value="0.1"/>
  <param name="recovery_alpha_slow" value="0.0"/>
  <param name="recovery_alpha_fast" value="0.0"/>
</node>
</launch>
```

نام فایل: CyberHandsWorld.world

جهانی کاملا مبتدی برای تست اولیه ی ربات

```
<?xml version="1.0" ?>
<sdf version="1.4">
  <world name="default">
    <!--physics type="ode"-->
      <!--gravity>0 0 0</gravity-->
      <!--ode>
        <solver>
          <type>quick</type>
          <iters>40</iters>
          <sor>1.0</sor>
        </solver>
        <constraints>
          <cfm>0.0</cfm>
          <erp>0.2</erp>
          <contact_max_correcting_vel>100.0</contact_max_correcting_vel>
          <contact_surface_layer>0.0</contact_surface_layer>
        </constraints>
      </ode>
      <max_step_size>0.001</max_step_size>
    </physics-->
    <!-- A ground plane -->
    <include>
      <uri>model://ground_plane</uri>
    </include>
    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>

<!--include>
      <uri>model://willowgarage</uri>
    </include-->
```

```
    <!--model name="gasstation">
      <include>
        <uri>model://gas_station</uri>
        <pose>-2.0 7.0 0 0 0 0</pose>
        <static>true</static>
      </include>
    </model-->
    <model name="barrier1">
      <include>
        <uri>model://jersey_barrier</uri>
        <pose>-3 0 0 0 0 0</pose>
        <static>true</static>
      </include>
    </model>
    <model name="barrier2">
      <include>
        <uri>model://jersey_barrier</uri>
        <pose>0 -3 0 0 0 1.57075</pose>
        <static>true</static>
      </include>
    </model>
    <model name="barrier3">
      <include>
        <uri>model://jersey_barrier</uri>
        <pose>3 6 0 0 0 0</pose>
        <static>true</static>
      </include>
    </model>
    <model name="barrier4">
      <include>
        <uri>model://jersey_barrier</uri>
        <pose>6 3 0 0 0 1.57075</pose>
        <static>true</static>
      </include>
    </model>
  </world>
</sdf>
```

نام فایل: single_robot.rviz

```
Panels:
  - Class: rviz/Displays
    Help Height: 78
    Name: Displays
    Property Tree Widget:
      Expanded: ~
      Splitter Ratio: 0.5
    Tree Height: 213
  - Class: rviz/Selection
    Name: Selection
  - Class: rviz/Tool Properties
    Expanded:
      - /2D Pose Estimate1
      - /2D Nav Goal1
      - /Publish Point1
    Name: Tool Properties
    Splitter Ratio: 0.588679
  - Class: rviz/Views
    Expanded:
      - /Current View1
    Name: Views
```

```
    Splitter Ratio: 0.5
  - Class: rviz/Time
    Experimental: false
    Name: Time
    SyncMode: 0
    SyncSource: LaserScan
Visualization Manager:
  Class: ""
  Displays:
    - Alpha: 0.7
      Class: rviz/Map
      Color Scheme: map
      Draw Behind: false
      Enabled: true
      Name: Map
      Topic: /map
      Value: true
    - Alpha: 1
      Buffer Length: 1
      Class: rviz/Path
      Color: 255; 149; 0
      Enabled: true
      Line Style: Lines
      Line Width: 0.03
      Name: Global Plan
      Offset:
        X: 0
        Y: 0
        Z: 0
      Topic: /move_base_node/DWAPlannerROS/global_plan
      Value: true
    - Alpha: 1
      Buffer Length: 1
      Class: rviz/Path
      Color: 0; 0; 255
      Enabled: true
      Line Style: Lines
      Line Width: 0.03
      Name: Local Plan
      Offset:
        X: 0
        Y: 0
        Z: 0
      Topic: /move_base_node/DWAPlannerROS/local_plan
      Value: true
    - Alpha: 1
      Buffer Length: 1
      Class: rviz/Path
      Color: 255; 0; 0
      Enabled: true
      Line Style: Lines
      Line Width: 0.03
      Name: NavFn Plan
      Offset:
        X: 0
        Y: 0
        Z: 0
      Topic: /move base node/NavfnROS/plan
      Value: true
    - Alpha: 0.7
```

```
    Class: rviz/Map
    Color Scheme: map
    Draw Behind: false
    Enabled: true
    Name: GlobalCostMap
    Topic: /move_base_node/global_costmap/costmap
    Value: true
  - Alpha: 0.7
    Class: rviz/Map
    Color Scheme: costmap
    Draw Behind: false
    Enabled: true
    Name: LocalCostMap
    Topic: /move_base_node/local_costmap/costmap
    Value: true
  - Class: rviz/Camera
    Enabled: true
    Image Rendering: background and overlay
    Image Topic: /chugv/camera/image_raw
    Name: Camera
    Overlay Alpha: 0.5
    Queue Size: 2
    Transport Hint: raw
    Value: true
    Visibility:
      BaseOdometry: true
      CurrentGoal: true
      DWACostCLoud: true
      Effort: true
      Global Plan: true
      GlobalCostMap: true
      Goal: true
      Inflated Obstacles: true
      LaserScan: true
      Local Plan: true
      LocalCostMap: true
      Local_Clearing_EndPoints: true
      Map: true
      NavFn Plan: true
      Obstacles: true
      Odometry: true
      PointCloud: true
      PointCloud2: true
      PointStamped: true
      Polygon: true
      Robot Footprint: true
      RobotModel: true
      TF: true
      Unknown Space: true
      Value: true
    Zoom Factor: 1
  - Alpha: 1
    Class: rviz/GridCells
    Color: 0; 0; 255
    Enabled: false
    Name: Inflated Obstacles
    Topic: /move_base_node/local_costmap/inflated_obstacles
    Value: false
  - Alpha: 1
    Class: rviz/GridCells
```

```
      Color: 255; 0; 0
      Enabled: false
      Name: Obstacles
      Topic: ""
      Value: false
    - Alpha: 1
      Class: rviz/GridCells
      Color: 0; 255; 0
      Enabled: false
      Name: Unknown Space
      Topic: /move_base_node/local_costmap/unknown_space
      Value: false
    - Alpha: 1
      Class: rviz/Polygon
      Color: 255; 0; 0
      Enabled: true
      Name: Robot Footprint
      Topic:
/move_base_node/global_costmap/obstacle_layer_footprint/footprint_stamped
      Value: true
    - Class: rviz/TF
      Enabled: false
      Frame Timeout: 15
      Frames:
        All Enabled: true
      Marker Scale: 1
      Name: TF
      Show Arrows: true
      Show Axes: true
      Show Names: true
      Tree:
        {}
      Update Interval: 0
      Value: false
    - Alpha: 1
      Class: rviz/RobotModel
      Collision Enabled: false
      Enabled: true
      Links:
        All Links Enabled: true
        Expand Joint Details: false
        Expand Link Details: false
        Expand Tree: false
        Link Tree Style: Links in Alphabetic Order
        back_left_wheel:
          Alpha: 1
          Show Axes: false
          Show Trail: false
          Value: true
        back_right_wheel:
          Alpha: 1
          Show Axes: false
          Show Trail: false
          Value: true
        base_footprint:
          Alpha: 1
          Show Axes: false
          Show Trail: false
          Value: true
        base_link:
```

```
      Alpha: 1
      Show Axes: false
      Show Trail: false
      Value: true
    camera_link:
      Alpha: 1
      Show Axes: false
      Show Trail: false
      Value: true
    front_left_wheel:
      Alpha: 1
      Show Axes: false
      Show Trail: false
      Value: true
    front_right_wheel:
      Alpha: 1
      Show Axes: false
      Show Trail: false
      Value: true
    hokuyo_link:
      Alpha: 1
      Show Axes: false
      Show Trail: false
      Value: true
  Name: RobotModel
  Robot Description: robot_description
  TF Prefix: ""
  Update Interval: 0
  Value: true
  Visual Enabled: true
- Alpha: 1
  Autocompute Intensity Bounds: true
  Autocompute Value Bounds:
    Max Value: 10
    Min Value: -10
    Value: true
  Axis: Z
  Channel Name: intensity
  Class: rviz/LaserScan
  Color: 255; 255; 255
  Color Transformer: Intensity
  Decay Time: 0
  Enabled: true
  Invert Rainbow: false
  Max Color: 255; 255; 255
  Max Intensity: 999999
  Min Color: 0; 0; 0
  Min Intensity: 0
  Name: LaserScan
  Position Transformer: XYZ
  Queue Size: 10
  Selectable: true
  Size (Pixels): 3
  Size (m): 0.01
  Style: Flat Squares
  Topic: /base_scan
  Use Fixed Frame: true
  Use rainbow: true
  Value: true
- Alpha: 1
```

```
    Axes Length: 1
    Axes Radius: 0.1
    Class: rviz/Pose
    Color: 85; 85; 255
    Enabled: true
    Head Length: 0.3
    Head Radius: 0.1
    Name: CurrentGoal
    Shaft Length: 1
    Shaft Radius: 0.05
    Shape: Arrow
    Topic: /move_base_node/current_goal
    Value: true
  - Alpha: 1
    Axes Length: 1
    Axes Radius: 0.1
    Class: rviz/Pose
    Color: 255; 25; 0
    Enabled: true
    Head Length: 0.3
    Head Radius: 0.1
    Name: Goal
    Shaft Length: 1
    Shaft Radius: 0.05
    Shape: Arrow
    Topic: /move_base_simple/goal
    Value: true
  - Angle Tolerance: 0.1
    Class: rviz/Odometry
    Color: 255; 25; 0
    Enabled: false
    Keep: 5
    Length: 1
    Name: Odometry
    Position Tolerance: 0.1
    Topic: /odom
    Value: false
  - Angle Tolerance: 0.1
    Class: rviz/Odometry
    Color: 255; 25; 0
    Enabled: false
    Keep: 100
    Length: 1
    Name: BaseOdometry
    Position Tolerance: 0.1
    Topic: /base_pose_ground_truth
    Value: false
  - Alpha: 1
    Class: rviz/PointStamped
    Color: 204; 41; 204
    Enabled: false
    History Length: 1
    Name: PointStamped
    Radius: 0.2
    Topic: /clicked_point
    Value: false
  - Alpha: 1
    Class: rviz/Effort
    Enabled: false
    History Length: 1
```

```
    Joints:
      {}
    Name: Effort
    Robot Description: robot_description
    Scale: 1
    Topic: ""
    Value: false
    Width: 0.02
  - Alpha: 1
    Autocompute Intensity Bounds: true
    Autocompute Value Bounds:
      Max Value: 10
      Min Value: -10
      Value: true
    Axis: Z
    Channel Name: intensity
    Class: rviz/PointCloud2
    Color: 255; 255; 255
    Color Transformer: ""
    Decay Time: 0
    Enabled: false
    Invert Rainbow: false
    Max Color: 255; 255; 255
    Max Intensity: 4096
    Min Color: 0; 0; 0
    Min Intensity: 0
    Name: PointCloud2
    Position Transformer: ""
    Queue Size: 10
    Selectable: true
    Size (Pixels): 3
    Size (m): 0.01
    Style: Flat Squares
    Topic: /move_base_node/TrajectoryPlannerROS/cost_cloud
    Use Fixed Frame: true
    Use rainbow: true
    Value: false
  - Alpha: 1
    Autocompute Intensity Bounds: true
    Autocompute Value Bounds:
      Max Value: 10
      Min Value: -10
      Value: true
    Axis: Z
    Channel Name: intensity
    Class: rviz/PointCloud
    Color: 255; 255; 255
    Color Transformer: Intensity
    Decay Time: 0
    Enabled: false
    Invert Rainbow: false
    Max Color: 255; 255; 255
    Max Intensity: 4096
    Min Color: 0; 0; 0
    Min Intensity: 0
    Name: PointCloud
    Position Transformer: XYZ
    Queue Size: 10
    Selectable: true
    Size (Pixels): 3
```

```
      Size (m): 0.01
      Style: Flat Squares
      Topic:
/move_base_node/global_costmap/obstacle_layer/clearing_endpoints
      Use Fixed Frame: true
      Use rainbow: true
      Value: false
    - Alpha: 1
      Class: rviz/Polygon
      Color: 25; 255; 0
      Enabled: false
      Name: Polygon
      Topic:
/move_base_node/global_costmap/obstacle_layer_footprint/footprint_stamped
      Value: false
    - Alpha: 1
      Autocompute Intensity Bounds: true
      Autocompute Value Bounds:
        Max Value: 10
        Min Value: -10
        Value: true
      Axis: Z
      Channel Name: intensity
      Class: rviz/PointCloud
      Color: 255; 255; 255
      Color Transformer: Intensity
      Decay Time: 0
      Enabled: true
      Invert Rainbow: false
      Max Color: 255; 255; 255
      Max Intensity: 4096
      Min Color: 0; 0; 0
      Min Intensity: 0
      Name: Local_Clearing_EndPoints
      Position Transformer: XYZ
      Queue Size: 10
      Selectable: true
      Size (Pixels): 3
      Size (m): 0.01
      Style: Flat Squares
      Topic:
/move_base_node/local_costmap/obstacle_layer/clearing_endpoints
      Use Fixed Frame: true
      Use rainbow: true
      Value: true
    - Alpha: 1
      Autocompute Intensity Bounds: true
      Autocompute Value Bounds:
        Max Value: 10
        Min Value: -10
        Value: true
      Axis: Z
      Channel Name: intensity
      Class: rviz/PointCloud2
      Color: 255; 255; 255
      Color Transformer: ""
      Decay Time: 0
      Enabled: true
      Invert Rainbow: false
      Max Color: 255; 255; 255
```

```
      Max Intensity: 4096
      Min Color: 0; 0; 0
      Min Intensity: 0
      Name: DWACostCLoud
      Position Transformer: ""
      Queue Size: 10
      Selectable: true
      Size (Pixels): 3
      Size (m): 0.01
      Style: Flat Squares
      Topic: /move_base_node/DWAPlannerROS/cost_cloud
      Use Fixed Frame: true
      Use rainbow: true
      Value: true
  Enabled: true
  Global Options:
    Background Color: 48; 48; 48
    Fixed Frame: map
    Frame Rate: 30
  Name: root
  Tools:
    - Class: rviz/Interact
      Hide Inactive Objects: true
    - Class: rviz/MoveCamera
    - Class: rviz/Select
    - Class: rviz/FocusCamera
    - Class: rviz/Measure
    - Class: rviz/SetInitialPose
      Topic: /initialpose
    - Class: rviz/SetGoal
      Topic: /move_base_simple/goal
    - Class: rviz/PublishPoint
      Single click: true
      Topic: /clicked_point
  Value: true
  Views:
    Current:
      Class: rviz/Orbit
      Distance: 22.4857
      Enable Stereo Rendering:
        Stereo Eye Separation: 0.06
        Stereo Focal Distance: 1
        Swap Stereo Eyes: false
        Value: false
      Focal Point:
        X: -0.646631
        Y: -0.580352
        Z: -0.331905
      Name: Current View
      Near Clip Distance: 0.01
      Pitch: 1.5698
      Target Frame: <Fixed Frame>
      Value: Orbit (rviz)
      Yaw: 6.28268
    Saved: ~
Window Geometry:
  Camera:
    collapsed: false
  Displays:
    collapsed: false
```

```
  Height: 744
  Hide Left Dock: false
  Hide Right Dock: true
  QMainWindow State:
000000ff00000000fd0000000400000000000000164000002a2fc0200000009fb00000012005
30065006c006500630074006900 6f006e00000001e10000009b0000006400ffffffb000000
1e0054006f006f006c002000500072006f007000650072007400690065007302000002540 00
0011a0000020c000000a3fb0000001200560069006500770073002000540 06f006f02000001
df000002110000018500000122fb0000002000540 06f006f006c002000500072006f0070006
500720074006900650073003203000002880000011d000002210000017afb00000010004400
6900730070006c006100790073010000002800000164000000dd00ffffffb0000002000730
065006c006500630074006900 6f006e002000620075006600660065007202000001380000 00
aa0000023a00000294fb0000001400570069006400650053007400650072006500 6f0200000
0e6000000d2000003ee0000030bfb0000000c004b0069006e006500630074020000018600 00
01060000030c00000261fb0000000c00430061006d006500720061010000019200000138000
0001600ffffff000000010000010f0000027efc0200000003fb0000000a0056006900650077
007300000000280000027e000000b000ffffffb0000001e0054006f006f006c00200050007
2006f007000650072007400690065007301000000410000007800000000000000000fb000000
1200530065006c006500630074006900 6f006e010000025a000000b2000000000000000000
00002000004bf000000b0fc0100000001fb0000000a0056006900650077007303000000 4e00
000080000002e1000001970000000300000 4bf0000003efc0100000002fb000000080054006
9006d006500000000000000000 4bf000002f600ffffffb0000000800540069006d0065010000
000000004500000000000000000000000 3ab000002a2000000040000000400000008000000 0
8fc0000000100000002000000010000000a0054006f006f006c00730100000000ffffffff00
00000000000000
  Selection:
    collapsed: false
  Time:
    collapsed: false
  Tool Properties:
    collapsed: false
  Views:
    collapsed: true
  Width: 1301
  X: 65
  Y: 24
```

نام فایل: move_base_gmapping_5cm.launch

```
<launch>
  <master auto="start"/>
  <param name="/use_sim_time" value="true"/>

  <include file="$(find chugv_navigation)/move_base_config/move_base.xml"/>

  <node pkg="stage_ros" type="stageros" name="stageros" args="$(find
chugv_navigation)/stage_config/worlds/CH-UGV-5cm.world" respawn="false" >
    <param name="base_watchdog_timeout" value="0.2"/>
  </node>
  <include file="$(find
chugv_navigation)/move_base_config/slam_gmapping.xml"/>
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
chugv_navigation)/single_robot_stage.rviz" />
</launch>
```

نام فایل: CH-UGV-5cm.world

```
define block model
(
  size [0.5 0.5 0.75]
  gui_nose 0
```

```
)

define topurg ranger
(
  sensor(
    range_max 30.0
    fov 270.25
    samples 1081
  )
  # generic model properties
  color "black"
  size [ 0.050 0.050 0.100 ]
)

define chugv position
(
  #size [0.650 0.650 0.250]
  size [1.200 0.800 0.250]
  origin [-0.050 0.000 0.000 0.000]
  gui_nose 1
  drive "diff"
  topurg(pose [ 0.275 0.000 0.000 0.000 ])
)

define floorplan model
(
  # sombre, sensible, artistic
  color "gray30"

  # most maps will need a bounding box
  boundary 1

  gui_nose 0
  gui_grid 0

  gui_outline 0
  gripper_return 0
  fiducial_return 0
  ranger_return 1.000
)

# set the resolution of the underlying raytrace model in meters
resolution 0.02

interval_sim 100  # simulation timestep in milliseconds


window
(
  size [ 745 448 ]

  rotate [ 0.000 -1.060 ]
  scale 18.806
)

# load an environment bitmap
floorplan
(
  name "chworld"
  bitmap "../maps/CHScanMap-0.05.pgm"
```

```
  size [58.250 47.250 1.000]
  #pose [ -23.625 29.125 0 90.000 ]
  pose [ 0.000 -3.000 0.000 90.000 ]
)

# throw in a robot
#chugv( pose [ -28.610 13.562 0 99.786 ] name "CHUGV" color "blue")
#block( pose [ -25.062 12.909 0 180.000 ] color "red")
#block( pose [ -25.062 12.909 0 180.000 ] color "red")
#block( pose [ -25.062 12.909 0 180.000 ] color "red")

chugv( pose [ 0 0 0.000 0 ] name "CHUGV" color "blue")
```

نام فایل: random_walk.cpp

```
#include <ros/ros.h>
#include <tf/transform_listener.h>
#include <iostream>
#include <vector>
#include "geometry_msgs/Twist.h"
#include "sensor_msgs/LaserScan.h"

using namespace std;

#define MIN_SCAN_ANGLE_RAD -60.0/180*M_PI
#define MAX_SCAN_ANGLE_RAD +60.0/180*M_PI

void readSensorCallback(const sensor_msgs::LaserScan::ConstPtr
&sensor_msg);

bool obstacleFound =false;

int main(int argc, char **argv){
    ros::init(argc, argv, "random_walk_node");
    ros::NodeHandle nh;

    ros::Publisher cmd_vel_pub =
nh.advertise<geometry_msgs::Twist>("cmd_vel",10);
    ros::Subscriber base_scan_sub =
nh.subscribe<sensor_msgs::LaserScan>("base_scan", 1, &readSensorCallback);

    geometry_msgs::Twist moveForwardCommand;
    moveForwardCommand.linear.x=0.5;

    geometry_msgs::Twist turnCommand;
    turnCommand.angular.z=1.0;

    ros::Rate loop_rate(10);

    while(ros::ok()){
        if(obstacleFound){
            cmd_vel_pub.publish(turnCommand);
            cmd_vel_pub.publish(turnCommand);
        } else {
            cmd_vel_pub.publish(moveForwardCommand);
        }
        ros::spinOnce(); // let ROS Process incoming messages
        loop_rate.sleep();
    }
    return 0;
}
```

```
void readSensorCallback(const sensor_msgs::LaserScan::ConstPtr &scan){
    bool isObstacle = false;

    int minIndex = ceil((MIN_SCAN_ANGLE_RAD - scan->angle_min) / scan-
>angle_increment);
    int maxIndex = floor((MAX_SCAN_ANGLE_RAD - scan->angle_min) / scan-
>angle_increment);

    for(int i=minIndex; i<= maxIndex; i++){
        if(scan->ranges[i] < 0.5){
            isObstacle = true;
        }
    }

    if(isObstacle){
        ROS_INFO("Obstacle found! Turning around");
        obstacleFound = true;
    } else {
        obstacleFound = false;
    }
}
```

نام فایل: random_walk.launch

```
<launch>
    <param name="/use_sim_time" value="true" />

    <!-- Launch World -->
    <include file="$(find chugv_gazebo)/launch/chugv_gazebo.launch"/>

    <!-- Launch Random Walk Node -->
    <node name="random_walk_node" pkg="chugv_random_walk"
type="random_walk_node" output="screen"/>
</launch>
```

نام فایل: simple_navigation_goals.cpp

```
/*
 * SendGoals.cpp
 *
 *  Created on: Dec 7, 2013
 *      Author: roiyeho
 *  Customize & Edit on: August 10, 2015
 *      Editor: Mojtaba Válipour @ CyberHands.org
 */

#include <ros/ros.h>
#include <move_base_msgs/MoveBaseAction.h>
#include <actionlib/client/simple_action_client.h>
#include <tf/transform_datatypes.h>

typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction>
MoveBaseClient;

int main(int argc, char** argv) {
    ros::init(argc, argv, "send_goals_node");

    // Read x, y and angle parameters
    double x, y, theta;
    ros::NodeHandle nh;
    nh.getParam("goal_x", x);
    nh.getParam("goal_y", y);
```

```
    nh.getParam("goal_theta", theta);

    // create the action client
    // true causes the client to spin its own thread
    MoveBaseClient ac("move_base", true);

    // Wait 60 seconds for the action server to become available
    ROS_INFO("Waiting for the move_base action server");
    ac.waitForServer(ros::Duration(60));

    ROS_INFO("Connected to move base server");

    // Send a goal to move_base
    move_base_msgs::MoveBaseGoal goal;
    goal.target_pose.header.frame_id = "map";
    goal.target_pose.header.stamp = ros::Time::now();

    //goal.target_pose.pose.position.x = 18.174;
    //goal.target_pose.pose.position.y = 28.876;
    //goal.target_pose.pose.orientation.w = 1;

    goal.target_pose.pose.position.x = x;
    goal.target_pose.pose.position.y = y;

    // Convert the Euler angle to quaternion
    double radians = theta * (M_PI/180);
    tf::Quaternion quaternion;
    quaternion = tf::createQuaternionFromYaw(radians);

    geometry_msgs::Quaternion qMsg;
    tf::quaternionTFToMsg(quaternion, qMsg);

    goal.target_pose.pose.orientation = qMsg;

    ROS_INFO("Sending goal to: x = %f, y = %f, theta = %f", x, y, theta);
    ac.sendGoal(goal);

    // Wait for the action to return
    ac.waitForResult();

    if (ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
        ROS_INFO("You have reached the goal!");
    else
        ROS_INFO("The base failed for some reason");

    return 0;
}
```

نام فایل: key_teleop.py

```
#! /usr/bin/env python
# rosrun chugv_control key_teleop.py
import rospy, math
import numpy as np
import sys, termios, tty, select, os
from geometry_msgs.msg import Twist

class KeyTeleop(object):
  cmd_bindings = {'q':np.array([1,1]),
                  'w':np.array([1,0]),
                  'e':np.array([1,-1]),
```

```
                 'a':np.array([0,1]),
                 'd':np.array([0,-1]),
                 'z':np.array([-1,-1]),
                 'x':np.array([-1,0]),
                 'c':np.array([-1,1])
                 }
set_bindings = { 't':np.array([1,1]),
                 'b':np.array([-1,-1]),
                 'y':np.array([1,0]),
                 'n':np.array([-1,0]),
                 'u':np.array([0,1]),
                 'm':np.array([0,-1])
               }
def init(self):
  # Save terminal settings
  self.settings = termios.tcgetattr(sys.stdin)
  # Initial values
  self.inc_ratio = 0.1
  self.speed = np.array([0.5, 1.0])
  self.command = np.array([0, 0])
  self.update_rate = 10   # Hz
  self.alive = True
  # Setup publisher
  self.pub_twist = rospy.Publisher('/cmd_vel', Twist)

def fini(self):
  # Restore terminal settings
  termios.tcsetattr(sys.stdin, termios.TCSADRAIN, self.settings)

def run(self):
  try:
    self.init()
    self.print_usage()
    r = rospy.Rate(self.update_rate) # Hz
    while not rospy.is_shutdown():
      ch = self.get_key()
      self.process_key(ch)
      self.update()
      r.sleep()
  except rospy.exceptions.ROSInterruptException:
    pass
  finally:
    self.fini()

def print_usage(self):
  msg = """
  Keyboard Teleop that Publish to /cmd_vel (geometry_msgs/Twist)
  Copyright (C) 2013
  Released under BSD License
  ---------------------------------------------------
  H:       Print this menu
  Moving around:
    Q   W   E
    A   S   D
    Z   X   Z
  T/B :   increase/decrease max speeds 10%
  Y/N :   increase/decrease only linear speed 10%
  U/M :   increase/decrease only angular speed 10%
  anything else : stop
```

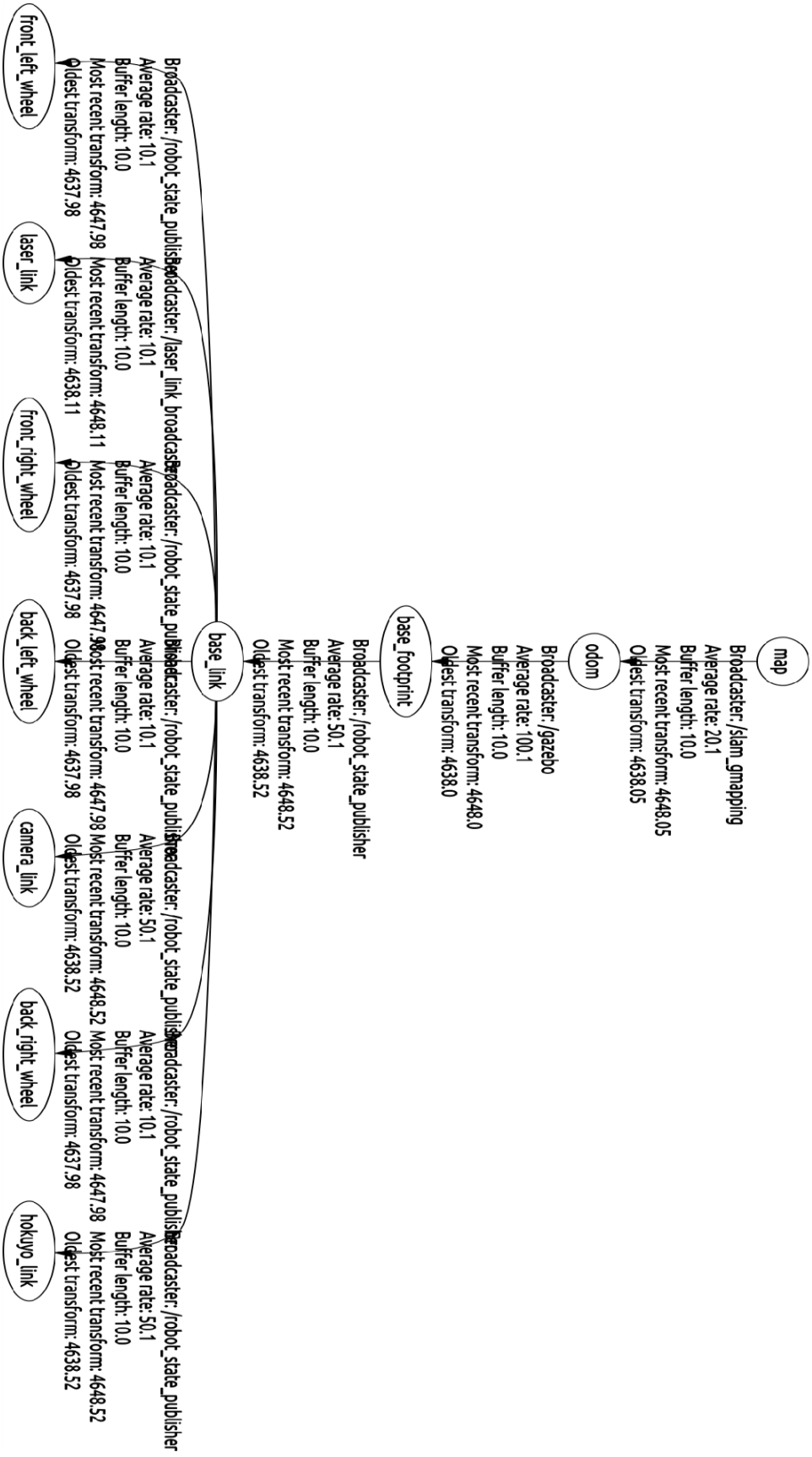
```
    G :   Quit
    ---------------------------------------------------
    """
    self.loginfo(msg)
    self.show_status()

  # Used to print items to screen, while terminal is in funky mode
  def loginfo(self, str):
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, self.settings)
    print(str)
    tty.setraw(sys.stdin.fileno())

  # Used to print teleop status
  def show_status(self):
    msg = 'Status:\tlinear %.2f\tangular %.2f' %
(self.speed[0],self.speed[1])
    self.loginfo(msg)

  # For everything that can't be a binding, use if/elif instead
  def process_key(self, ch):
    if ch == 'h':
      self.print_usage()
    elif ch in self.cmd_bindings.keys():
      self.command = self.cmd_bindings[ch]
    elif ch in self.set_bindings.keys():
      self.speed = self.speed * (1 + self.set_bindings[ch]*self.inc_ratio)
      self.show_status()
    elif ch == 'g':
      self.loginfo('Quitting')
      # Stop the robot
      twist = Twist()
      self.pub_twist.publish(twist)
      rospy.signal_shutdown('Shutdown')
    else:
      self.command = np.array([0, 0])

  def update(self):
    if rospy.is_shutdown():
      return
    twist = Twist()
    cmd  = self.speed*self.command
    twist.linear.x = cmd[0]
    twist.angular.z = cmd[1]
    self.pub_twist.publish(twist)

  # Get input from the terminal
  def get_key(self):
    tty.setraw(sys.stdin.fileno())
    select.select([sys.stdin], [], [], 0)
    key = sys.stdin.read(1)
    return key.lower()

if __name__ == '__main__':
  rospy.init_node('keyboard_teleop')
  teleop = KeyTeleop()
  teleop.run()
```

نام فایل: joy_teleop.py

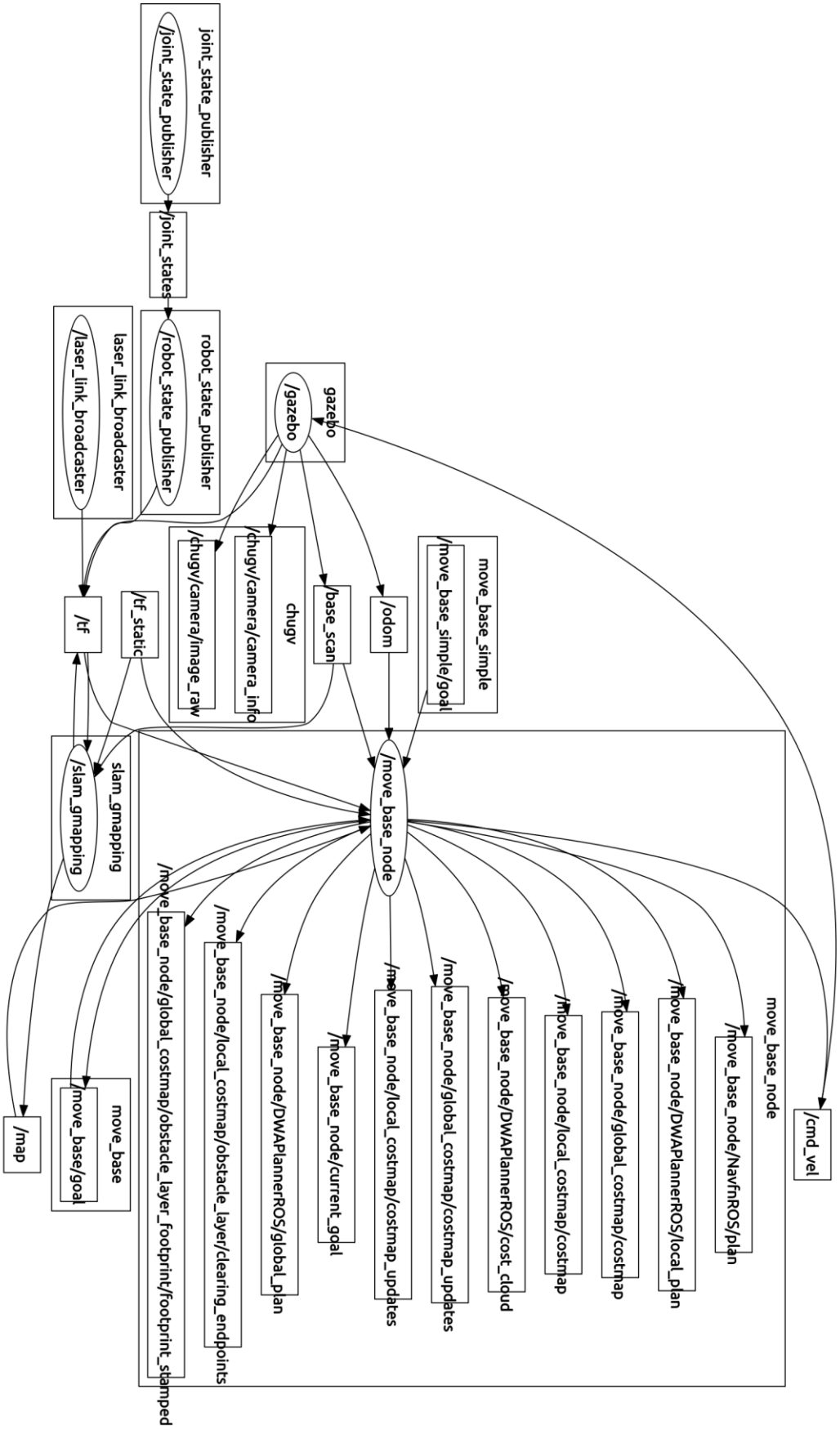
```
#!/usr/bin/env python
#rosrun joy joy_node
```

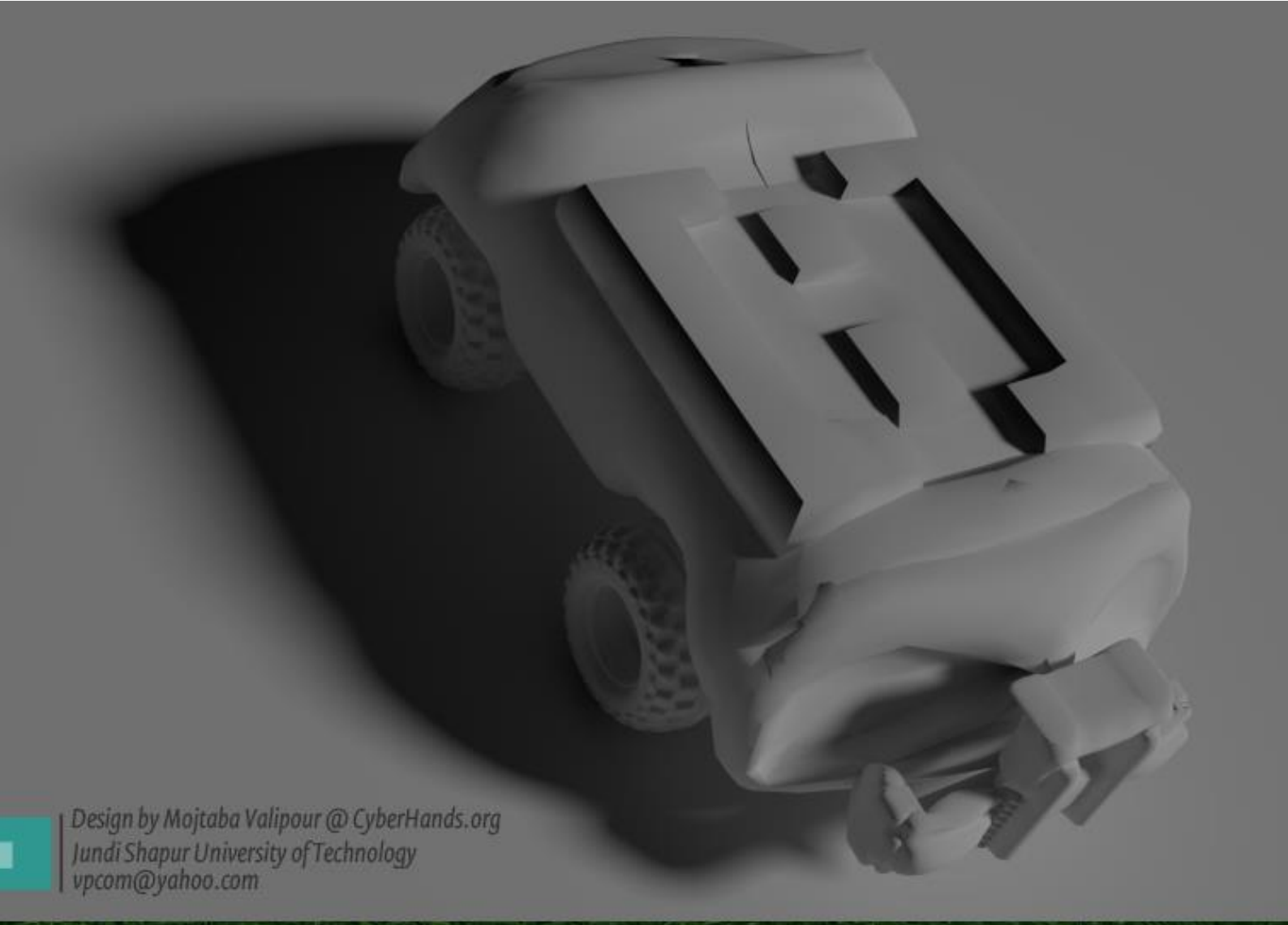
```python
#rosrun chugv_control joy_teleop.py
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import Joy

def callback(data):
    twist = Twist()
    twist.linear.x = 4*data.axes[1]
    twist.angular.z = 4*data.axes[2]
    pub.publish(twist)

def joy_teleop():
    rospy.init_node('Joy_teleop')
    rospy.Subscriber("joy", Joy, callback)
    global pub
    pub = rospy.Publisher('/cmd_vel', Twist)

    r = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        r.sleep()

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    joy_teleop()
```

پیوست ب- درخت ارتباط فریم های موجود در ربات فراز سایبرهندس



پیوست ج- دیاگرام ارتباط تمام گره های به کارگرفته شده در ربات فراز سایبرهندس





Design by Mojtaba Valipour @ CyberHands.org
Jundi Shapur University of Technology
vpcom@yahoo.com