بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

ناوبری خودمختار در محیط ناشناخته و خارج از جاده مبتنی بر خانواده الگوریتم‌های باگ

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر-هوش مصنوعی

سید نوید حسینی ایزدی

استاد راهنما

دکتر مازیار پالهنگ

۱۳۹۲

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد رشته‌ی مهندسی کامپیوتر-هوش مصنوعی آقای سید نوید حسینی ایزدی تحت عنوان

ناوبری خودمختار در محیط ناشناخته و خارج از جاده مبتنی بر خانواده الگوریتم‌های باگ

در تاریخ ۱۳۹۲/۱۰/۲۳ توسط کمیته‌ی تخصصی زیر مورد بررسی و تصویب نهایی قرار گرفت.

۱- استاد راهنمای پایان‌نامه	دکتر مازیار پالهنگ
۲- استاد داور (اختیاری)	دکتر سید رسول موسوی
۳- استاد داور (اختیاری)	دکتر محمد دانش
سرپرست تحصیلات تکمیلی دانشکده	دکتر سید محمدعلی خسروی‌فرد

از پدر و مادر مهربانم که به من فرصت زندگی دادند و همواره مشوق من بوده‌اند، قدردانی می‌کنم. با تشکر و سپاس فراوان از جناب آقای دکتر مازیار پالهنگ که راهنمای بنده در انجام این پایان‌نامه بودند. به علاوه تشکر می‌کنم از اعضای آزمایشگاه هوش مصنوعی و رباتیک دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان که در پیاده‌سازی عملی این پایان‌نامه بنده را یاری کردند.

نوید حسینی ایزدی

پاییز ۱۳۹۲

کلیه‌ی حقوق مادی مترتب بر نتایج مطالعات، ابتکارات و نوآوری‌های ناشی از تحقیق موضوع این پایان‌نامه متعلق به دانشگاه صنعتی اصفهان است.

تقدیم به

پدر و مادر عزیزم...

فهرست مطالب

عنوان **صفحه**

فهرست مطالب ..... هشت

چکیده ۱

فصل اول: مقدمه

۱-۱ پیش‌گفتار ۲

۱-۲ تعریف مسئله ۳

۱-۳ چالش‌های موجود در مسئله ۴

۱-۴ هدف از انجام پایان‌نامه ۵

۱-۵ دستاوردهای پایان‌نامه ۶

۱-۶ ساختار پایان‌نامه ۶

فصل دوم: تاریخچه و مرور کارهای پیشین

۲-۱ مقدمه ۷

۲-۲ تاریخچه ربات‌های متحرک ۸

۲-۳ مرور کارهای پیشین ۱۲

۲-۳-۱ طرح‌ریزهای سراسری ۱۲

۲-۳-۲ طرح‌ریزهای محلی حسگر-مبنا ۱۴

۲-۴ نتیجه‌گیری ۲۲

فصل سوم: مفاهیم علمی پیش‌نیاز پایان‌نامه

۳-۱ مقدمه ۲۳

۳-۲ الگوریتم تنژنت‌باگ ۲۳

۳-۲-۱ گراف رویت ۲۴

۳-۲-۲ گراف مماس ۲۵

۳-۲-۳ گراف مماس محلی ۲۶

۳-۲-۴ شرح الگوریتم تنژنت‌باگ ۲۹

۳-۳ الگوریتم اجتناب از مانع میدان پتانسیل (PF) ۳۳

۳-۴ الگوریتم اجتناب از مانع میدان نیروی مجازی (VFF) ۳۵

۳-۵ الگوریتم اجتناب از مانع هیستوگرام میدان برداری (VFH) ۳۷

۳-۶ الگوریتم اجتناب از مانع +VFH ۴۰

۳-۷ خلاصه ۴۶

فصل چهارم: الگوریتم تنژنت‌باگ تغییریافته (روش پیشنهادی)

۴-۱ مقدمه ۴۸

۴-۲ محاسبه‌ی نقطه‌ی ایمن با توجه به نقطه‌ی انتخابی تنژنت‌باگ ۴۹

۴-۳ مکانیزم رفتن به نقطه ۵۰

۴-۴ تغییرات اعمال شده حین ساخت گراف مماس محلی ۵۲

۴-۵ تغییرات اعمال شده در رفتار حرکت-به-سوی-هدف ۵۴

۴-۵-۱ بررسی شرایط مورد نیاز برای عملکرد درست فاصله‌ی مکاشفه‌ای پیشنهادی ۵۸

۴-۶ بررسی تغییرات مورد نیاز در رفتار دنبال-کردن-مرز-مانع ۶۰

۴-۶-۱ روش پیشنهادی برای دنبال-کردن-مرز-مانع ۶۲

۴-۶-۲ اثبات ایمن بودن روش پیشنهادی برای دنبال کردن مرز مانع ۶۳

۴-۷ پس پردازش تصمیمات تنژنت‌باگ برای افزایش ایمنی ربات ۶۵

۴-۷-۱ در نظر گرفتن تمام موانع قابل رویت در رفتار نهایی ربات ۶۵

۴-۷-۲ درنظر گرفتن محدودیت‌های حرکتی ربات آکرمن ۶۸

۴-۸ مکان‌یابی ربات در محیط ۷۴

۴-۹ جمع‌بندی ۷۴

فصل پنجم: نتایج شبیه‌سازی و عملی

۵-۱ مقدمه ۷۶

۵-۲ نتایج حاصل از شبیه‌سازی ۷۶

۵-۲-۱ مقایسه در ساخت گراف مماس محلی ۷۷

۵-۲-۲ مقایسه در رفتار حرکت-به-سوی-هدف ۷۹

۵-۲-۳ مقایسه در رفتار دنبال-کردن-مرز-مانع ۸۲

۵-۲-۴ مقایسه‌ی سیستم دارای فاز پس‌پردازش با سیستم فاقد آن ۸۲

۵-۳ پیاده‌سازی عملی ۸۵

۵-۴ نتیجه‌گیری ۸۹

فصل ششم: نتیجه‌گیری و جمع‌بندی

۶-۱ مقدمه ۹۰

۶-۲ نقاط قوت روش ارائه شده ۹۰

۶-۳ کاستی‌های روش ارائه شده ۹۱

۶-۳-۱ عدم استفاده از GPS ۹۱
۶-۳-۲ عدم اطمینان از چگال بودن موانع قابل رویت محیط ۹۱
۶-۴ پیشنهادات برای کارهای آینده ۹۲
مراجع ۹۴
ABSTRACT ۹۷

چکیده

امروزه زمینه‌های کاربردی برای ربات‌های متحرک خودمختار رو به افزایش است. از جمله کاربردهای این ربات‌ها می‌توان به اکتشاف سیاره‌های ناشناخته (مانند مریخ)، یافتن مجروح‌های زلزله زده و امداد رسانی به آن‌ها و ... اشاره کرد. مهم‌ترین خصوصیت این دسته از ربات‌ها، توانایی ناوبری در محیط به صورت خودمختار برای رسیدن به نقطه‌ی هدف می‌باشد.

به طور کل می‌توان محیط عملیاتی ربات متحرک را به دو دسته‌ی ۱-شناخته شده و ۲-ناشناخته تقسیم کرد. در صورتی که محیط شناخته شده باشد، ربات با در اختیار داشتن نقشه‌ی محیط و با بهره‌گیری از الگوریتم‌های طرح‌ریز مسیر سراسری، می‌تواند مسیر بهینه برای رسیدن به هدف را محاسبه کند. اما داشتن شناخت از محیط همیشه میسر نمی‌باشد. از طرف دیگر با افزایش وسعت محیط، میزان حافظه و توان پردازشی مورد نیاز جهت نگه‌داری و به روز رسانی (در صورت نیاز) نقشه‌ی آن افزایش می‌یابد. از این روست که طرح‌ریزهای حسگر-مبنا مورد توجه قرار می‌گیرند. این دسته از طرح‌ریزها فرض می‌کنند که محیط کاملاً ناشناخته است و صرفاً با تکیه بر داده‌های حاصل از حسگرهای نصب شده بر روی ربات، آن را به سمت هدف هدایت می‌کنند. از آنجا که برد حسگرهای ربات محدود می‌باشد، طرح‌ریزهای حسگر-مبنا قادر به محاسبه‌ی مسیر بهینه‌ی سراسری نمی‌باشند و طرح‌ریزی مسیر را به صورت افزایشی انجام می‌دهند؛ به این صورت که در هر گام، محیط پیرامون ربات را با استفاده از حسگرهای آن به صورت محلی درک می‌کنند و با توجه به داده‌های به دست آمده، برای حرکت ربات به سمت هدف و اجتناب از موانع موجود در محیط، تصمیم مناسب می‌گیرند. ویژگی مهم این دسته از طرح-ریزها توانایی همگرا شدن به هدف علی‌رغم ناشناخته بودن محیط می‌باشد. خانواده‌ی الگوریتم‌های باگ از دسته طرح‌ریزهای حسگر-مبنا می‌باشند که به خاطر کارایی مناسب در عین داشتن ساختاری ساده، معروف هستند.

در این پایان‌نامه هدف، تغییر الگوریتم تنژنت‌باگ برای سازگار شدن با ربات متحرک از نوع آکرمن می‌باشد. محیط عملیاتی ربات مذکور، از نوع ناشناخته و خارج از جاده است. منظور از خارج از جاده بودن محیط این است که نمی‌توان در مورد ساختار محیط فرض خاصی مانند وجود جاده و یا علائم هدایت‌کننده در نظر گرفت. سیستم ناوبری ارائه شده در این پایان‌نامه بر پایه‌ی الگوریتم تنژنت‌باگ (یکی از معروف‌ترین الگوریتم‌های باگ) بنا شده است. تنژنت‌باگ علی‌رغم قدرتمند بودنش، کاستی‌هایی دارد. به عنوان مثال تنژنت-باگ، ربات را به صورت نقطه‌ای (بدون طول و عرض) فرض می‌کند و محدودیت‌های حرکتی آن را در نظر نمی‌گیرد. در این پایان‌نامه به ارائه‌ی تدابیری برای رفع کاستی‌های مذکور پرداخته می‌شود. در این راستا از مدل سینماتیک ربات آکرمن و الگوریتم‌های اجتناب از مانع میدان پتانسیل و هیستوگرام میدان برداری (VFH) بهره گرفته می‌شود. برای بررسی صحت روش ارائه شده، نتایج آن با نتایج الگوریتم تنژنت‌باگ به صورت شبیه‌سازی مقایسه شده است. علاوه بر شبیه‌سازی، روش ارائه شده بر روی یک ربات آکرمن واقعی به صورت عملی پیاده‌سازی و تست گردیده است.

کلمات کلیدی: ۱- ناوبری خودمختار ۲- محیط ناشناخته ۳- اجتناب از مانع ۴- حسگر-مبنا

فصل اول

مقدمه

۱-۱ پیش‌گفتار

امروزه کاربرد ربات‌های خودمختار در زندگی آدمی رو به افزایش می‌باشد و نقش آن‌ها در انجام کارهای دشوار، پرمخاطره و تکراری که برای انسان‌ها آزاردهنده می‌باشد، غیرقابل انکار است. ربات مانند انسان از کار کردن خسته نمی‌شود و دقت و سرعتش (مادامی که سالم است) افت نمی‌کند. در ضمن زمانی که یک ربات برای انجام کاری تخصصی طراحی و ساخته می‌شود، ساخت ربات‌های بیشتر از همان نوع نیازی به صرف وقت و هزینه‌ی زیاد برای آموزش تک تک آن‌ها ندارد، در حالی که برای به کارگیری نیروی انسانی، قبل از آن که انسان‌ها بتوانند کار مذکور را به درستی انجام دهند، باید تک به تک آموزش داده شوند که این کار نیازمند وقت و هزینه‌ی زیادی می‌باشد. موارد ذکر شده از جمله عللی است که باعث استفاده‌ی روزافزون از ربات‌ها در انجام کارهای مختلف شده است.

ربات‌ها را می‌توان به دو دسته‌ی متحرک و ثابت تقسیم کرد. ربات‌های ثابت کاربردهای متفاوتی دارند و در زمینه‌های مختلف به آدمی خدمت می‌رسانند اما برخی از کارها مانند جابه‌جا کردن قطعات در کارخانه، امداد رسانی به قربانیان زلزله، بررسی آتشفشان‌های فعال و ... فقط از عهده‌ی ربات متحرک برمی‌آید. حتی در مورد کارهایی که توسط ربات‌های ثابت انجام می‌گیرد، باید توجه کرد که اگر ربات‌هایی که کارهای مذکور را انجام می‌دهند توانائی

حرکت نیز داشته باشند، حتی مفیدتر و پرکاربردتر نیز خواهند شد زیرا که در این صورت می‌توانند به قسمت‌های مختلف محیط کاری خود بروند و در چندین مکان به انجام وظایف بپردازند، که به این ترتیب به جای استفاده از چند ربات در مکان‌های مختلف، می‌توان از یک ربات متحرک برای انجام کارها در مکان‌های مذکور استفاده کرد. از این رو تحقیق بر روی ربات‌های متحرک بسیار گسترده و پرطرفدار می‌باشد.

۱-۲ تعریف مسئله

در این پایان‌نامه هدف، طراحی و پیاده‌سازی یک سیستم ناوبری خودمختار[1] برای هدایت یک ربات متحرک از نوع آکرمن[2] (شبیه به یک خودروی سواری چهار چرخ که دو چرخ جلویی‌اش فرمان‌پذیر هستند)، از نقطه‌ی مبدأ به نقطه‌ی مقصد می‌باشد. محیطی که در اینجا به آن پرداخته می‌شود، از نوع ناشناخته و خارج از جاده[3] است. به علت ناشناخته بودن محیط، سیستم ناوبری مذکور باید به گونه‌ای طراحی شود که ربات بدون نیاز به نقشه‌ی محیط و بدون استفاده از هیچ‌گونه زمین‌نمائی[4] بتواند به نقطه‌ی هدف برسد. به عبارت دیگر روشی مطلوب خواهد بود که صرفاً حسگر-مبنا[5] باشد و ربات بتواند با اتکا به حسگرهایش، مسیر لازم تا رسیدن به هدف را (در صورت وجود) پیدا کرده، بپیماید. واضح است که پیمایش مسیر تا هدف، اجتناب از موانع با هر اندازه و شکل دلخواه (موانعی که بر سر راه ربات حین رسیدن به هدف وجود دارد) را نیز شامل می‌شود.

همان طور که اشاره شد، محیط علاوه بر ناشناخته بودن، خارج از جاده نیز می‌باشد و از ساختار خاصی پیروی نمی‌کند. در نتیجه نمی‌توان در مورد آن فرض از پیش تعیین شده‌ای داشت مثلاً نمی‌توان فرض کرد محیط دارای جاده است و ربات باید حتماً در جاده حرکت کند و یا نمی‌توان فرض کرد که موانع از نوع خاصی مانند درخت، ساختمان، صخره و ... می‌باشند.

حسگرهایی که در این پایان‌نامه در نظر گرفته شده‌اند، عبارتند از : انکودر[6]، IMU[7] و پویش‌گر لیزری[8]. انکودر و IMU در مکان‌یابی ربات با استفاده از روابط سینماتیکی حاکم بر آن کاربرد دارد و از پویشگر لیزری برای تشخیص موانع موجود در مسیر ربات بهره گرفته می‌شود.

از آنجا که ناوبری در محیط خارج از جاده و ناشناخته در حالت کلی خود با مسائل و مشکلات متعددی رو به رو است، در این پایان‌نامه فرض‌هایی برای ساده‌تر کردن مسئله در نظر گرفته شده است که در ادامه بیان می‌شود:

---

[1] Autonomous navigation
[2] Ackerman
[3] Off-road
[4] Landmark
[5] Sensor-based
[6] Encoder
[7] Inertial Measurement Unit
[8] Laser scanner

۱- بررسی و طراحی تکنیک‌های قوی مکان‌یابی در محیط‌های ناشناخته بدون استفاده از حسگرهای مکان‌یابی مطلق مانند GPS[1] بحث وسیعی می‌باشد و از حیطه‌ی این پایان‌نامه خارج است، لذا مکان‌یابی به کار گرفته شده، در اینجا از نوع dead-reckoning و صرفاً با بهره‌گیری از انکودر و IMU می‌باشد.

۲- پویشگر لیزری مورد استفاده، از نوع دو بعدی می‌باشد که به صورت افقی نسبت به زمین بر روی ربات نصب شده است. ناحیه‌ای که پویش‌گر مذکور پوشش می‌دهد، بخشی (۲۷۰ از ۳۶۰ درجه) از دایره‌ای با شعاع ۳۰ متر است که (دایره) موازی با زمین می‌باشد. به این ترتیب موانعی که ارتفاعشان کمتر از فاصله‌ی پویش‌گر لیزری از سطح زمین باشد، در دید پویش‌گر قرار نمی‌گیرند و قابل تشخیص برای ربات نمی‌باشند. به همین علت در اینجا فرض بر این است که ارتفاع موانع به اندازه‌ی کافی بلند می‌باشد. توجه شود که برای درک کردن تمام موانع موجود در محیط به پویش‌گر لیزری با قابلیت پویش سه بعدی نیاز می‌باشد که تهیه‌ی آن برای انجام این پایان‌نامه مقدور نبود.

۱-۳ چالش‌های موجود در مسئله

ناوبری در محیط‌های ناشناخته و خارج از جاده صرف از نظر از نوع رباتی که مورد استفاده قرار می‌گیرد، با مشکلات متعددی رو به رو است. یکی از مشکلات مذکور نداشتن شناخت از محیط می‌باشد. این مورد از این جهت حائز اهمّیت است که اگر محیط شناخته شده بود، سیستم ناوبری می‌توانست با استفاده از نقشه‌ی محیط و یکی از الگوریتم‌های جستجو مانند A^*[۱]، D^* [۲] و ...، به محاسبه‌ی مسیر بهینه از نقطه‌ی فعلی به سمت هدف بپردازد و سپس طبق مسیر محاسبه شده به هدف برسد. اما زمانی که محیط ناشناخته است، امکان محاسبه‌ی مسیر بهینه (قبل از شروع به حرکت) با توجه به نقطه‌ی شروع و پایان وجود ندارد و طرح‌ریزی مسیر باید به صورت افزایشی و بلادرنگ انجام گیرد. برای این کار، در هر گام با توجه به داده‌های حسگرها تصمیم مناسب برای حرکت ربات، به صورت محلی گرفته می‌شود. از جمله الگوریتم‌هایی که این گونه عمل می‌کنند، می‌توان به خانواده‌ی الگوریتم‌های باگ[2] اشاره کرد که برخی از آن‌ها همگرا شدن به هدف را (در صورت امکان) تضمین می‌کنند.

از طرف دیگر ناشناخته بودن محیط در دقت مکان‌یابی ربات نیز تأثیر مستقیم دارد از این لحاظ که در محیط ناشناخته، ربات زمین‌نمائی را از قبل نمی‌شناسد تا با مشاهده‌ی آن بتواند باور خود در مورد مکان فعلی‌اش را اصلاح کند. البته در صورتی که ربات توانائی مکان‌یابی و نقشه‌کشی هم‌زمان[3] (مونه) را داشته باشد، بعد از شناسائی و ترسیم نقشه‌ی محیط مکان زمین‌نماها را می‌داند ولی در این پایان‌نامه هدف استفاده از مونه نمی‌باشد.

---

[1] Global Positioning System
[2] Bug algorithms
[3] Simultaneous Localization and Mapping (SLAM)

همان طور که قبلاً هم اشاره شد، ربات مورد نظر در این پایان‌نامه از نوع آکرمن می‌باشد. این نوع از ربات دارای محدودیت‌های غیرهولونومیکی[1] می‌باشد. به این معنا که ربات مذکور توانایی حرکت در هر جهت دلخواه بدون لغزش چرخ‌هایش بر روی زمین را ندارد. به عنوان مثال، ربات آکرمن نمی‌تواند در هر جهت دلخواه حرکت کند بلکه همواره بر روی کمان یک دایره حرکت می‌کند که شعاع دایره‌ی مذکور در سرعت‌های پایین تابعی از زاویه‌ی چرخ‌های فرمان‌پذیر و فاصله‌ی بین محورهای عقب و جلوی[2] ربات می‌باشد. لازم به یادآوری است که میزان چرخش چرخ‌های فرمان‌پذیر محدود می‌باشد و همین امر موجب می‌شود که شعاع دوران کمینه[3] ربات آکرمن بزرگ‌تر از صفر باشد. با توجه به مطالب گفته شده، واضح است که ربات آکرمن مسیرهایی که نیازمند شعاع دورانی کمتر از شعاع دوران کمینه‌اش می‌باشد را نمی‌تواند دنبال کند که این امر در تصمیماتی که سیستم ناوبری ربات می‌گیرد، باید لحاظ شود.

۱-٤ هدف از انجام پایان‌نامه

همان طور که قبلاً اشاره شد، هدف از این پایان‌نامه طراحی و پیاده‌سازی یک سیستم ناوبری خودمختار برای ربات متحرک آکرمن در محیط‌های ناشناخته و خارج از جاده می‌باشد. از آنجا که محیط ناشناخته است و ربات در مورد نقشه و زمین‌نماهای محیط شناختی ندارد، استفاده از الگوریتم‌های حسگر-مبنا مانند خانواده‌ی الگوریتم‌های باگ مطلوب به نظر می‌رسد زیرا که، الگوریتم‌های باگ تنها با حسگرهای موجود بر روی ربات و بدون هیچ فرضی بر روی محیط، می‌توانند با تصمیم‌گیری‌های محلی (در هر گام بر اساس درکی که از محیط پیرامون موجود می‌باشد، تصمیم مناسب برای حرکت به سوی هدف گرفته می‌شود)، ربات را به هدف برسانند و اگر هم هدف غیر قابل دسترس باشد، الگوریتم‌های مذکور با اعلام دست نیافتنی بودن هدف، متوقف می‌شوند. تاکنون نسخه‌های مختلفی از الگوریتم‌های باگ ارائه شده است که برخی از آن‌ها حتی در مریخ‌نوردها نیز به کار رفته‌اند مانند الگوریتم وِج‌باگ[4] که نسخه‌ی تغییر یافته‌اش با نام رُوِرباگ[5] در مریخ‌نورد راکی۷[6] استفاده شده است [۳]. لازم به ذکر است که مریخ مثال خوبی از یک محیط ناشناخته و خارج از جاده می‌باشد. با مشاهده‌ی کاربرد مؤثر الگوریتم وج‌باگ برای ناوبری روی کره‌ی مریخ، الگوریتم پایه برای سیستم ناوبری ارائه شده در این پایان‌نامه، یکی از الگوریتم‌های باگ با نام تنژنت‌باگ[7] [۴] در نظر گرفته شد. تنژنت‌باگ قابلیت ناوبری در محیط‌های ناشناخته‌ی نامحدود را داراست که این خصوصیت بسیار حائز اهمّیت می‌باشد. اما تنژنت‌باگ مانند اکثر الگوریتم‌های هم‌خانواده‌اش، نکاتی که برای پیاده-

[1] Non-holonomic constraints
[2] Wheelbase
[3] Minimum turning radius
[4] WedgeBug
[5] RoverBug
[6] Rocky7
[7] TangentBug

سازی عملی آن‌ها (مانند در نظر گرفتن ابعاد و محدودیت‌های حرکتی ربات) ضروری می‌باشد را در نظر نگرفته است. در نتیجه برای استفاده از تنژنت‌باگ باید موارد مذکور در پیاده‌سازی لحاظ شود که هدف از این پایان‌نامه نیز بررسی مواردی است که باید به الگوریتم تنژنت‌باگ افزوده شود تا در عمل قابل استفاده گردد.

در حال حاضر در کشور عزیزمان، بر روی ناوبری ربات‌های متحرک به خصوص در محیط‌های ناشناخته کمتر کار شده است. با توجه به کاربردهای روزافزون رباتیک متحرک، نیاز به پژوهش و کسب تجربیات بیشتر در این زمینه احساس می‌شود که هدف اصلی این پایان‌نامه برداشتن گام کوچکی در این زمینه می‌باشد.

۱-۵ دستاوردهای پایان‌نامه

در قسمت‌های قبلی اشاره شد که الگوریتم تنژنت‌باگ به عنوان پایه‌ی سیستم ناوبری این پایان‌نامه در نظر گرفته شده است. ذکر شد که الگوریتم مذکور در ناوبری محیط ناشناخته بسیار قدرتمند عمل می‌کند اما نکات ضروری برای استفاده‌ی عملی این الگوریتم در طراحی آن، در نظر گرفته نشده است. در نتیجه بررسی الگوریتم مذکور و اعمال تغییرات لازم برای استفاده‌ی آن در عمل، دستاوردهای این پایان‌نامه را تشکیل می‌دهد. تغییراتی که در این پایان‌نامه بر روی الگوریتم تنژنت‌باگ اعمال شده است، به شرح زیر می‌باشد:

۱- در نظر گرفتن ابعاد ربات در تصمیم‌گیری‌های الگوریتم تنژنت‌باگ
۲- در نظر گرفتن محدویت‌های حرکتی (غیرهولونومیک بودن) ربات آکرمن در حین اجرای تصمیمات گرفته شده توسط تنژنت‌باگ
۳- بازبینی رفتارهای حرکت-به-سوی-هدف[1] و دنبال-کردن-مرز-مانع[2] برای سازگار شدن با ربات غیرهولونومیک آکرمن

۱-۶ ساختار پایان‌نامه

در فصل دوم، مروری بر کارهای گذشته در رابطه با ناوبری ربات‌های متحرک خودمختار ارائه می‌گردد. در فصل سوم مفاهیم مورد نیاز برای مطالعه‌ی روش ارائه شده در این پایان‌نامه شرح داده می‌شوند. در فصل چهارم روش ارائه شده برای ناوبری ربات آکرمن در محیط ناشناخته و خارج از جاده توضیح داده می‌شود. فصل پنجم به بررسی نتایج شبیه‌سازی و عملی روش ارائه شده می‌پردازد و در نهایت در فصل ششم بحث جمع‌بندی می‌شود و پیشنهادات برای ادامه کار در آینده بیان می‌گردد.

---
[1] Motion-to-goal
[2] Boundary-following

فصل دوم

تاریخچه و مرور کارهای پیشین

۲-۱ مقدمه

تاکنون بر روی ناوبری خودمختار ربات‌های متحرک پروژه‌های متعددی انجام شده است. صرف نظر از نوع محیطی (از محیط‌های درون‌در کاملاً شناخته شده گرفته تا محیط‌های برون‌در کاملاً ناشناخته) که ناوبری در آن انجام می‌شود، تمامی روش‌های ناوبری باید توانائی اجتناب از موانع موجود در مسیر ربات به سمت هدف را داشته باشند. در صورتی که محیط شناخته شده باشد، ربات می‌تواند با توجه به نقشه‌ی محیط، مسیری بهینه و خالی از موانع را به سمت هدف محاسبه کند و با بهره گرفتن از یک روش مکان‌یابی مناسب به هدف برسد. اما در اختیار داشتن نقشه‌ی محیط همیشه میسر نیست مانند زمانی که ربات باید در سیاره‌ای ناشناخته به اکتشاف بپردازد. از طرف دیگر هر چه محیط بزرگ‌تر باشد میزان حافظه‌ی بیشتری برای نگه‌داری نقشه‌ی آن مورد نیاز می‌باشد. به همین علت الگوریتم‌های ناوبری که صرفاً حسگر-مبنا هستند مطلوب به نظر می‌رسند زیرا که آن‌ها فقط با استفاده از حسگرهایی که روی ربات موجود است و بدون نیاز به نقشه‌ی محیط، ربات را به سوی هدف هدایت می‌کنند. از روش‌های معروف ناوبری حسگر-مبنا می‌توان به خانواده‌ی الگوریتم‌های باگ اشاره کرد که به علت سادگی و مؤثر بودنشان معروف هستند. در این فصل ابتدا تاریخچه‌ی مختصری از ربات‌های متحرک ارائه می‌شود و سپس دسته‌بندی الگوریتم‌های طرح‌ریز مسیر مورد بررسی قرار می‌گیرد.

۲-۲ تاریخچه ربات‌های متحرک

در طی جنگ جهانی دوم از سال ۱۳۱۸ (۱۹۳۹م.) تا سال ۱۳۲۴ (۱۹۴۵م.) پیشرفت‌هایی در زمینه‌های علوم کامپیوتر و سایبرنتیک[1] انجام شد که در اثر آن‌ها اوّلین ربات‌های متحرک پا به عرصه‌ی وجود گذاشتند. در آغاز، این ربات‌ها بیشتر شبیه به بمب‌های پرنده بودند که در فاصله‌ی مشخصی از هدف به طور خودکار منفجر می‌شدند. در همین دوران بود که راکت‌های وی۱ و وی۲ آلمانی پدیدار شدند. این راکت‌ها به یک سیستم هدایت خودکار[2] ساده و سیستم انفجار خودکار مجهز بودند.

از سال ۱۳۲۷ (۱۹۴۸م.) تا سال ۱۳۲۸ (۱۹۴۹م.) گری والتر[3] دو ربات هوشمند به نام‌های السی و المر[4] را ساخت [۵]. این دو ربات به گونه‌ای طراحی شده بودند که تمایل به اکتشاف محیط پیرامون خود داشتند. در ضمن هر دوی آن‌ها به حسگر نوری[5] مجهز بودند و با استفاده از آن هر کجا که نور حس می‌کردند، به آن سمت حرکت و در عین حال از برخورد به موانع نیز اجتناب می‌کردند. والتر با ساخت این دو ربات که به لاک‌پشت معروف بودند، نشان داد که یک طراحی ساده توانایی تولید رفتارهای پیچیده را دارد. از سال ۱۳۴۰ (۱۹۶۱م.) تا ۱۳۴۲ (۱۹۶۳م.) دانشگاه جان هاپکینز[6] ربات بیست[7] را ارائه کرد [۶]. بیست مجهز به حسگر سونار بود و قابلیت شارژ باتریش به صورت خودکار را داشت. به این صورت که هرگاه شارژش رو به اتمام بود، پریز برقی را پیدا می‌کرد و به آن متصل می‌شد.

از سال ۱۳۴۵ (۱۹۶۶م.) تا ۱۳۵۱ (۱۹۷۲م.) در مؤسسه‌ی تحقیقاتی استنفورد ربات شیکی[8] (شکل ۲-۱) ساخته شد که به دوربین، حسگر مسافت‌یاب، حسگر برخورد با مانع[9] و ارتباط رادیویی مجهز بود [۷]. شیکی اوّلین رباتی بود که توانایی استدلال درباره‌ی اعمال خودش را داشت. به عبارت دیگر شیکی این توانایی را داشت که دستورات سطح بالا دریافت کند و مراحل مورد نیاز برای انجام دستورات مذکور را به طور خودکار دریابد و اجرا کند. علاوه بر شیکی، در سال ۱۳۴۹ (۱۹۷۰م.) ربات دنبال‌کننده‌ی خط استنفورد کارت[10] نیز ساخته شد که با استفاده از دوربین، توانایی دنبال کردن یک خط سفید را داشت [۸]. این ربات از طریق ارتباط رادیویی به یک کامپیوتر بزرگ[11] متصل بود و محاسبات مورد نیاز برای دنبال کردن خط توسط کامپیوتر مذکور انجام می‌شد. علاوه بر ربات مذکور، در همین سال رباتی با نام لونخود[12] برای اکتشاف سطح کره‌ی ماه توسط اتحاد جماهیر شوروی ساخته شد [۹]. در

---

[1] Cybernetics
[2] Autopilot
[3] Grey Walter
[4] Elsie and Elmer
[5] Light sensor
[6] John Hopkins
[7] Beast
[8] Shakey
[9] Bump sensor
[10] Stanford Cart
[11] Mainframe
[12] Lunkhod

مشابه در سال ۱۳۵۵ (۱۹۷۶م.) سازمان ناسا دو فضاپیما به مریخ فرستاد [۱۰].

در اوایل دهه‌ی ۱۹۸۰م. تیم ارنست دیکمنز[1] در دانشگاه بوندسور[2] مونیخ اوّلین خودروهای خودمختار را ساخت که قابلیت رانندگی با سرعت ۸۸/۵ کیلومتر بر ساعت در خیابان‌های خالی را داشتند [۱۱]. در سال ۱۳۶۶ (۱۹۸۷م.) آزمایشگاه‌های تحقیقاتی هاگز[3] اوّلین ربات خودمختار با توانایی ناوبری (با استفاده از نقشه و حسگرهایش) در مناطق صحرائی را ارائه کرد [۱۲].

شکل ۲- ۱-ربات شیکی

در دهه‌ی ۱۹۹۰م. ژوزف انگلبرگر[4] (پدر بازوهای رباتیکی صنعتی) و همکارانش، ربات‌های متحرک خودمختار با کاربرد در بیمارستان‌ها را ساختند [۱۳]. در سال ۱۳۷۰ (۱۹۹۱م.) ایدو فرنزی[5]، آندره گوییگنارد[6] و فرانسسکو موندادا[7] ربات خپرا[8] را ارائه کردند [۱۴]. خپرا یک ربات خومختار کوچک است که برای اهداف تحقیقاتی مورد استفاده قرار می‌گیرد. از سال ۱۳۷۲ (۱۹۹۳م.) تا ۱۳۷۳ (۱۹۹۴م.) ربات‌های دانته ۱ و ۲ توسط دانشگاه کارنیگی ملون[9] ساخته شدند، که توانایی راه رفتن و اکتشاف آتشفشان‌های فعال را داشتند [۱۵-۱۶].

---

[1] Ernst Dickmanns
[2] Bundeswehr
[3] Hughes
[4] Joseph Engelberger
[5] Edo Franzi
[6] Andre Guignard
[7] Francesco Mondada
[8] Khepera
[9] Carnegie Mellon University (CMU)

در سال ۱۳۷۳ (۱۹۹۴م.) دو ربات به نام‌های ومپ[1] [۱۷] و ویتا-۲ [۱۸] توسط تیم ارنست دیکمنز و کمپانی دایملر-بنز[2] ساخته شدند که مسافتی بیش از ۱۰۰۰ کیلومتر را در اتوبان‌های سه بانده‌ی پاریس با وجود ترافیک سنگین استاندارد و با سرعت ۱۳۰ کیلومتر بر ساعت به صورت خودمختار طی کردند. در سال ۱۳۷۴ (۱۹۹۵م.) در اقدامی مشابه، خودروی نیمه-خودمختار آلوین طراحی و ساخته شد [۱۹]. این خودرو که توانایی کنترل فرمان به صورت خودکار را داشت، از شبکه‌های عصبی مصنوعی به عنوان مغز متفکرش استفاده می‌کرد. لازم به ذکر است که گاز و ترمز خودروی مذکور توسط یک اپراتور انسان کنترل می‌شد. در همین سال یکی از خودروهای خودمختار ارنست دیکمنز، مسافتی بیش از ۱۶۰۹ کیلومتر را به صورت خودمختار و با وجود ترافیک با حداکثر سرعت ۱۹۳ کیلومتر بر ساعت طی کرد. خودروی مذکور کنترل گاز و ترمز را نیز به صورت خودکار (برخلاف خودروی آلوین) انجام می‌داد و به غیر از چند مورد معدود که به دلایل ایمنی اپراتور انسان در تصمیماتش دخالت کرد، در بقیه مواقع به طور کاملاً خودمختار به طی کردن مسیر پرداخت. علاوه بر پیشرفت‌های حاصل شده، در سال ۱۳۷۴ (۱۹۹۵م.) ربات پایُنییر[3] به عنوان بستری آماده برای اهداف تحقیقاتی با قیمتی مناسب ارائه شد که همین رویداد باعث افزایش میزان تحقیقات پژوهشگران و دانشگاه‌ها در زمینه‌ی ربات‌های متحرک شد [۲۰]. زیرا که با استفاده از ربات مذکور انجام پروژه‌های رباتیکی تسهیل و تسریع گردید.

از سال ۱۳۷۵ (۱۹۹۶م.) تا ۱۳۷۶ (۱۹۹۷م.) سازمان ناسا، مریخ‌نوردی[4] به نام سوجرنر[5] را ساخت و به مریخ فرستاد [۲۱]. این مریخ‌نورد از زمین دستورات مأموریتی خود را دریافت می‌کرد و به سیستمی برای مقابله با مخاطره-های احتمالی در حین اکتشاف سطح ناشناخته‌ی کره‌ی مریخ مجهز بود؛ به عبارت دیگر، ربات مذکور قادر به پیدا کردن مسیری ایمن برای رسیدن به هدفش بود. در سال ۱۳۷۸ (۱۹۹۹م.) سری ربات‌های نظامی پک-بات[6] (شکل ۲-۲) توسط کمپانی آی-روبوت[7] عرضه شدند [۲۲]. این ربات‌ها از راه دور توسط یک اپراتور کنترل می‌شدند و از جمله کاربرد آن‌ها می‌توان به تشخیص و انهدام مواد منفجره، تشخیص عوامل شیمیایی و یا پرتوزا در محیط و ... اشاره کرد.

در سال ۱۳۸۰ (۲۰۰۱م.) بود که پروژه‌ی سوآرم-باتس[8] آغاز گردید [۲۳]. هدف از این پروژه به کارگیری تعداد زیادی ربات با ساختارهای ساده بود تا با همکاری با یکدیگر، کارهای پیچیده که از عهده‌ی تک‌تک آن‌ها برنمی‌آید را، به صورت گروهی انجام دهند.

---

[1] VaMP
[2] Daimler-Benz
[3] Pioneer
[4] Mars rover
[5] Sojourner
[6] PackBot
[7] iRobot
[8] Swarm-bots

یکی از رویدادهای مهمی که در سال ۱۳۸۳ (۲۰۰۴م.) بیش از بقیه به چشم می‌آید، برگزاری مسابقه‌ی دارپا گرند چالنج[۱] می‌باشد [۲۴]. این مسابقه که توسط آژانس پروژه‌های تحقیقاتی پیشرفته‌ی دفاعی[۲] ترتیب داده شد، رقابتی بود بین تعدادی از خودروهای خودمختار در مسیری صحرائی برای به دست آوردن مقام اوّل (اوّلین خودرویی که از خط پایان می‌گذشت، تیمش برنده محسوب می‌شد). مسیر مسابقه ۲۴۰ کیلومتر طول داشت که هیچ یک از تیم‌ها نتوانستند مسیر را به طور کامل بپیمایند. تیم رد[۳] از دانشگاه کارنیگی ملون بیشترین مسافت را به طول ۱۱/۷۸ کیلومتر پیمود. در سال ۱۳۸۴ (۲۰۰۵م.) دارپا گرند چالنج دیگری برگزار شد که این بار مسیر ۲۱۲ کیلومتری توسط پنج عدد از خودروهای خودمختار به طور کامل طی شد و خوردی استنلی [۲۵] (شکل ۲-۳) از دانشگاه استنفورد مقام اوّل را کسب کرد. در سال ۱۳۸۶ (۲۰۰۷م.) دارپا مسابقه‌ی دیگری به نام دارپا اربن چالنج[۴] ترتیب داد [۲۶]. این بار مسابقه در محیط شهری با وجود موانع ثابت و متحرک (خودروهای دیگر با راننده‌ی انسان)، برگزار گردید که در آن شش تیم توانستند به خط پایان برسند. مقام اوّل به تیم تارتان[۵] متعلق به دانشگاه کارنیگی ملون و کمپانی جنرال موتورز رسید.

شکل ۲- ۲- نمونه‌ای از سری ربات‌های پک-بات

در سال ۱۳۸۹ (۲۰۱۰م.) رقابتی با نام مجیک[۶] در استرالیا برگزار شد [۲۷]. در این مسابقه هر تیم شرکت‌کننده از تعدادی ربات خودمختار تشکیل شده بود که هدفشان ترسیم کردن نقشه‌ی محیطی به مساحت ۵۰۰متر ×۵۰۰متر در کمتر از ۳/۵ ساعت بود. علاوه بر ترسیم نقشه‌ی محیط، هر تیم می‌بایست تهدیدهای ثابت و یا متحرک را تشخیص

[1] DARPA grand challenge
[2] Defense Advanced Research Project Agency (DARPA)
[3] Red team
[4] DARPA urban challenge
[5] tartan
[۶] مجیک (MAGIC) مخفف Multi Autonomous Ground-Robotic International Challenge به معنای رقابت بین‌المللی رباتیک زمینی خودمختار چندتایی می‌باشد.

داده و خنثی کند.

شکل ۲- ۳- خودروی خودمختار استنلی دارنده‌ی مقام اوّل مسابقه‌ی دارپا گرند چالنج ۲۰۰۵

۲-۳ مرور کارهای پیشین

برای آن که ربات قادر به حرکت از نقطه‌ی شروع و رسیدن به هدف باشد، به الگوریتمی برای یافتن مسیری از نقطه‌ی شروع به نقطه‌ی هدف نیازمند است. به این الگوریتم طرح‌ریز مسیر[1] می‌گویند. اکثر کارهایی که در زمینه‌ی الگوریتم‌های طرح‌ریز مسیر ارائه شده است را می‌توان به دو دسته‌ی کلی ۱- طرح‌ریزهای سراسری ۲- طرح‌ریزهای طرح‌ریزهای محلی حسگر-مبنا تقسیم کرد که در ادامه به صورت مختصر شرح داده می‌شوند.

۲-۳-۱ طرح‌ریزهای سراسری

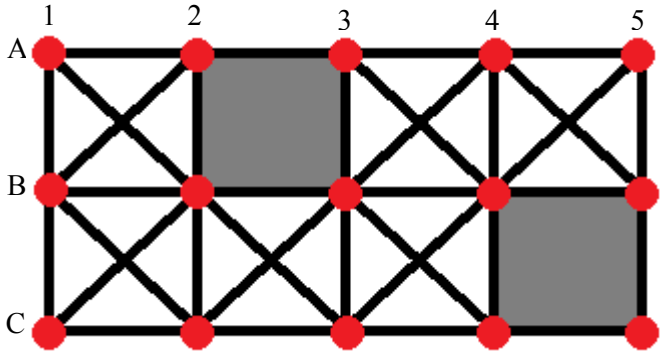
طرح‌ریزهای سراسری فرض می‌کنند که محیط کاملاً شناخته شده می‌باشد و نقشه‌ی آن در اختیار است. این طرح‌ریزها از دسته الگوریتم‌های کامل[2] محسوب می‌شوند، به عبارت دیگر الگوریتم‌های طرح‌ریز سراسری در صورتی که مسیری بین نقاط شروع و پایان موجود باشد، آن را می‌یابند و در صورت عدم وجود مسیر مذکور، با حالت شکست متوقف می‌شوند. از جمله الگوریتم‌های طرح‌ریز سراسری می‌توان به A^* [۱]، A^* PS [۲۸] و $Theta^*$ [۲۸] اشاره کرد. الگوریتم A^* در جست و جو بر روی گراف‌ها، همواره مسیرهای بهینه را پیدا می‌کند اما در صورتی که A^* بر روی نقشه‌های شبکه‌ای ۸-همسایه‌ای اجرا گردد، دیگر قادر به یافتن مسیر بهینه نخواهد بود زیراکه مسیرهای یافته شده توسط A^* تنها از یال‌های نقشه‌ی شبکه‌ای عبور خواهد کرد. در شکل ۲-۴ نمونه‌ای از یک نقشه شبکه‌ای ۸-همسایه‌ای نشان داده شده است. در این شکل هر خانه یکی از سلول‌های نقشه‌ی شبکه‌ای را نشان می‌دهد

[1] Path planner
[2] Complete algorithms

(سلول‌های خاکستری رنگ، اشغال می‌باشند). دایره‌های موجود در شکل، رأس‌ها و خطوط بین آن‌ها یال‌های گرافی محسوب می‌شوند که A^* قرار است روی آن اجرا شود. همان طور که مشاهده می‌شود هر رأس به ۸ همسایه‌ی دیگر (در صورتی که سلول‌های اطرافش آزاد باشد) به وسیله‌ی یال‌ها متصل است به همین دلیل به این نقشه، نقشه‌ی شبکه-ای ۸-همسایه‌ای گفته می‌شود.

برای کاهش طول مسیر تولید شده توسط A^* در نقشه‌های مذکور، روش A^*PS ارائه شده است. روش A^*PS ابتدا A^* را بر روی نقشه‌ی مذکور اجرا می‌کند و سپس مسیر به دست آمده را پس‌پردازش می‌کند که طی آن در صورت امکان مسیر از نظر طول بهبود داده می‌شود. اگر چه A^*PS مسیر تولید شده توسط A^* را بهبود می‌دهد، اما باز هم تضمینی برای به دست آوردن مسیر بهینه نمی‌دهد. علت آن است که مسیر بهینه ممکن است دقیقاً از یال‌های نقشه‌ی شبکه‌ای عبور نکند. این در حالی است که هر دو روش A^* و A^*PS تنها مسیرهایی را می‌یابند که حتماً از یال‌ها عبور کرده باشند. برای حل این مشکل طرح‌ریز $Theta^*$ ارائه شده است که مسیرهای محاسبه شده توسط آن محدود به یال‌های نقشه‌ی مذکور نمی‌شود و همواره بهینه است.

نکته‌ای که باید در مورد طرح‌ریزهای سراسری به آن توجه داشت این است که احتمال دارد نقشه‌ای که با استفاده از آن مسیریابی انجام می‌شود، کاملاً صحیح نباشد. به عبارت دیگر امکان دارد برخی از قسمت‌های نقشه آزاد و خالی از مانع باشند اما در واقعیت آن قسمت‌ها بسته بوده و قابل عبور نباشد. در این صورت الگوریتم‌هایی مانند A^* مسیر بهینه‌ی قبلی را به طور کامل دور می‌ریزند و مسیر جدیدی از نقطه‌ای که ربات به بن‌بست خورده است، محاسبه می‌کنند. این در حالی است که الگوریتم D^* توانایی بازسازی و اصلاح مسیری که قبلاً محاسبه کرده است را دارد. به عبارت دیگر الگوریتم D^* قسمت‌هایی از مسیر قبلی که قابل استفاده هستند را نگه می‌دارد و قسمت‌هایی که باید اصلاح شوند را مجدداً محاسبه می‌کند. به این ترتیب زمان کم‌تری صرف مسیریابی می‌شود.



شکل ۲- ۴-نمونه‌ای از نقشه‌ی شبکه‌ای ۸-همسایه‌ای [۲۸]

۲-۳-۲ طرح‌ریزهای محلی حسگر-مبنا

دسته‌ی دوم از طرح‌ریزها، طرح‌ریزهای محلی حسگر-مبنا و یا همان الگوریتم‌های اجتناب از مانع می‌باشند. این نام‌گذاری به این علت است که الگوریتم‌های عضو این دسته برای هدایت ربات به سمت هدف، تنها بر داده‌هایی

که ربات به صورت محلی از محیط با استفاده از حسگرهایش به دست می‌آورد، تکیه می‌کنند. به عبارت دیگر این الگوریتم‌ها هیچ شناختی از محیط عملیاتی ربات در اختیار ندارند. همانند طرح‌ریزهای سراسری، این دسته از الگوریتم‌ها نیز از اهمّیت زیادی برخوردار هستند. علت این است که در بسیاری از موارد محیط برای ربات از پیش شناخته شده نمی‌باشد و در نتیجه استفاده از طرح‌ریزهای سراسری حداقل تا زمان شناخته شدن محیط، ممکن نیست. به عنوان مثال ربات امدادگر، محل حادثه‌ای که باید در آن به دنبال زخمی‌ها بگردد را از قبل نمی‌شناسد و یا رباتی که باید به اکتشاف مریخ بپردازد، از محیط عملیاتی‌اش نامطلع است. در ادامه به بررسی اجمالی برخی از الگوریتم-های اجتناب از مانع می‌پردازیم. البته بر روی خانواده‌ی الگوریتم‌های باگ بیشتر تأمل خواهیم کرد زیرا که روش ارائه شده در این پایان‌نامه بر پایه‌ی الگوریتم تنزنت‌باگ بنا شده است.

الگوریتم اجتناب از مانع سرعت-انحنا

روش سرعت-انحنا[1] برای اجتناب از موانع به صورت محلی و برای محیط درون‌در در سال ۱۹۹۶م. ارائه شده است [۲۹]. روش مذکور اجتناب از مانع را به صورت یک مسئله‌ی بهینه‌سازی دارای محدودیت در فضای سرعت فرموله می‌کند. محدودیت‌هایی که روش سرعت-انحنا در نظر می‌گیرد، از دو منبع ناشی می‌شوند. منبع اوّل محدودیت‌های فیزیکی ربات (مانند سرعت) و منبع دوم نحوه‌ی قرار گرفتن موانع در محیط می‌باشد. فضای سرعتی که روش سرعت-انحنا در آن جست‌وجو می‌کند، دارای دو بعد سرعت انتقالی (v) و سرعت زاویه‌ای (w) می‌باشد. روش مذکور مقادیر v و w را به گونه‌ای انتخاب می‌کند که ربات بتواند با حداکثر سرعت ایمن به سمت هدف حرکت کند و به موانع نیز برخورد نکند. برای این کار مقدار بیشینه برای یک تابع هدف بر اساس سه پارامتر ۱- حداکثر سرعت، ۲-ایمنی و ۳-حرکت به سوی هدف[2] محاسبه می‌شود. از روش سرعت-انحنا می‌توان به عنوان پایه‌ای برای کاربردهای پیچیده‌تر مانند اکتشاف محیط و یا ناوبری نقشه-مبنا استفاده کرد. در ضمن روش مذکور برای ربات‌های غیر-هولونومیک مانند آکرمن ودیفرانسیلی قابل استفاده می‌باشد. لازم به یادآوری است که این روش تأثیر شتاب را بر روی فضای دو بعدی سرعت در نظر نمی‌گیرد. به عبارت دیگر ممکن است روش سرعت-انحنا، مقادیری برای سرعت‌های انتقالی و زاویه‌ای انتخاب کند که ربات شتاب کافی برای به رسیدن به آن‌ها در یک بازه‌ی زمانی محدود را نداشته باشد.

الگوریتم اجتناب از مانع پنجره‌ی پویا

یکی دیگر از روش‌های اجتناب از مانع بلادرنگ، الگوریتم پنجره‌ی پویا[3] ارائه شده در سال ۱۹۹۷م. می‌باشد

[1] Curvature-velociry method
[2] Goal directedness
[3] Dynamic window

[۳۰]. این روش با توجه به دینامیک حرکتی ربات سینکرو-درایو[1] ارائه شده است و مانند روش سرعت-انحنا، در فضای دو بعدی سرعت به جست‌وجوی مقادیر مناسب برای سرعت‌های انتقالی و زاویه‌ای می‌پردازد. روش پنجره‌ی پویا علاوه بر محدودیت‌های سرعتی ربات، محدودیت‌های شتابی آن را نیز (برخلاف روش سرعت-انحنا) در نظر می‌گیرد. روش مذکور محاسبه دستور بعدی جهت هدایت ربات را برای بازه‌ی زمانی کوچکی انجام می‌دهد و به این ترتیب از پیچیدگی طرح‌ریزهای کلاسیک اجتناب می‌کند. در این روش فضای دو بعدی سرعت از یک صافی (با توجه به محدودیت‌های سرعتی و شتابی) عبور داده می‌شود. که بعد از اعمال صافی مذکور، تنها مقادیری از سرعت انتقالی و زاویه‌ای در آن موجود می‌باشد که برای ربات قابل استفاده هستند؛ به عبارت دیگر اگر ربات با آن مقادیر سرعت حرکت کند، بدون برخورد به موانع می‌تواند به هدف برسد. لازم به ذکر است که اعمال محدودیت‌های شتابی روی فضای مذکور باعث می‌شود فقط مقادیری از سرعت در نظر گرفته شوند که ربات طی بازه‌ی زمانی کوچک بعدی، امکان رسیدن به آن مقادیر را داشته باشد.

حال پنجره‌ای حاوی مقادیر مجاز سرعت به مرکزیت سرعت فعلی ربات (در فضای جست‌وجو) در نظر گرفته می‌شود که ربات می‌تواند از بین مقادیر موجود در پنجره‌ی مذکور، سرعت انتقالی (v) و زاویه‌ای (w) را برای بازه‌ی زمانی بعدی انتخاب کند. معیار انتخاب مقادیر برای v و w، بیشینه کردن یک تابع هدف می‌باشد که در این مورد روش پنجره‌ی پویا به روش سرعت-انحنا شباهت دارد.

الگوریتم اجتناب از مانع پنجره‌ی پویای سراسری

پنجره‌ی پویای سراسری [۳۱] شبیه به پنجره‌ی پویا عمل می‌کند اما با در نظر گرفتن خروجی الگوریتم آتش علف[2] [۳۲] در تابع هدف، رویکردی سراسری دارا می‌باشد. لازم به ذکر است که الگوریتم آتش علف برای محاسبه‌ی مسیر بهینه در نقشه‌های سلولی با اندازه‌ی ثابت مورد استفاده قرار می‌گیرد. آتش علف از سلول هدف شروع می‌کند و فاصله‌ی بلوک-شهری[3] (با در نظر گرفتن همسایگی ۴-تایی) هر سلول تا سلول هدف را محاسبه می‌کند. این کار ادامه می‌یابد تا به سلولی (به نام S) که ربات در آن قرار دارد برسد. سپس از سلول S شروع می‌کند و از بین همسایگی ۸-تایی آن، سلولی که فاصله‌اش تا هدف از بقیه‌ی سلول‌ها کم‌تر می‌باشد را به سلول S وصل می‌کند. حال سلول انتخاب شده در نظر گرفته می‌شود و روند تکرار می‌گردد تا به سلول هدف برسیم.

باید توجه شود که برای افزایش سرعت اجرا و بلادرنگ بودن روش پنجره‌ی پویای سراسری، الگوریتم آتش علف روی تمام نقشه‌ی سلولی اعمال نمی‌شود، بلکه روی ناحیه‌ای مستطیل شکل که مکان فعلی ربات و هدف را در بر می‌گیرد اجرا می‌شود. در صورتی که در مستطیل مذکور امکان رسیدن به هدف وجود نداشته باشد، ابعاد مستطیل

[1] Synchro-drive
[2] Fire grass (NF1)
[3] City-block (Manhattan distance)

افزایش می‌یابد و دوباره امکان رسیدن به هدف بررسی می‌شود.

روش اجتناب از مانع Schlegel

این روش علاوه بر دینامیک ربات، شکل آن را نیز در نظر می‌گیرد و برای استفاده بر روی داده‌های خام پویش‌گر لیزری و نقشه‌ی سلولی محیط مناسب می‌باشد [۳۳]. به علت آن که روش مذکور شکل دقیق ربات را در تصمیماتش حین اجتناب از مانع دخیل می‌کند، محاسبات سنگینی خواهد داشت. برای کاهش محاسبات مذکور، از جدول‌هایی از پیش محاسبه شده با توجه به شکل و دینامیک ربات استفاده می‌شود. این کار موجب بلادرنگ شدن این روش می‌شود اما از طرفی حافظه‌ی مصرفی آن را به شدت افزایش می‌دهد.

روش اجتناب از مانع دیاگرام نزدیکی

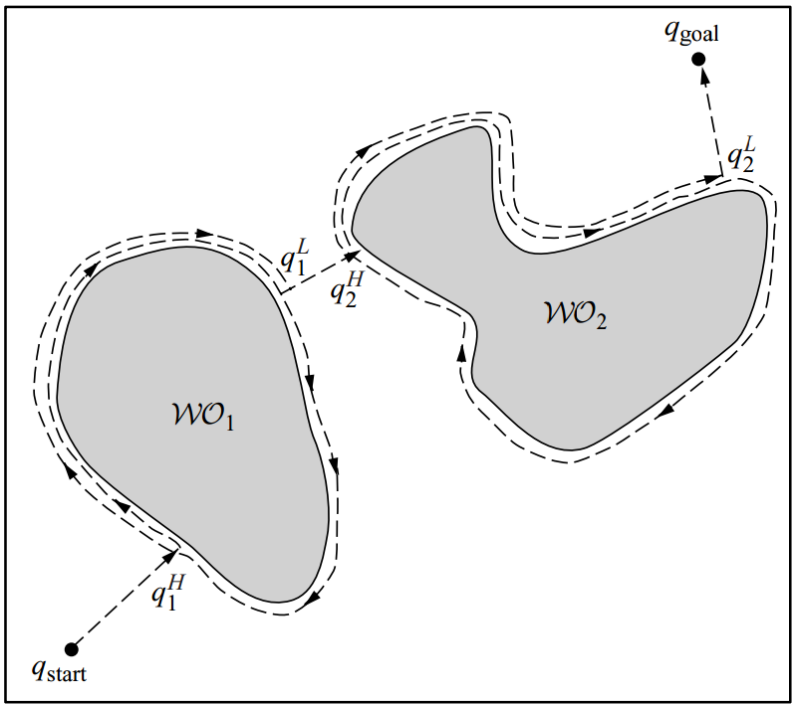
روش دیاگرام نزدیکی[1] [۳۴] دارای شباهت‌هایی با روش VFH [۳۵] (روش VFH در فصل سوم شرح داده خواهد شد) است. روش دیاگرام نزدیکی برخی از نواقص روش VFH به خصوص زمانی که محیط دارای تعداد زیادی موانع به صورت به هم ریخته می‌باشد، را برطرف کرده است. اساس کار روش دیاگرام نزدیکی به این صورت است که ابتدا جهتی ایمن برای حرکت ربات، با فرض این که شکل آن دایره‌ای می‌باشد، به دست می‌آورد. در ادامه مسیر مذکور با توجه به محدودیت‌های سینماتیکی و دینامیکی ربات اصلاح می‌شود و در نهایت شکل حقیقی ربات (در صورتی که دایره‌ای نباشد) در مسیر مذکور لحاظ می‌شود. همانند روش پنجره‌ی پویا که دارای نسخه‌ای سراسری می‌باشد، برای روش دیاگرام نزدیکی نیز نسخه‌ای با رویکرد سراسری [۳۶] ارائه شده است.

مروری بر خانواده‌ی الگوریتم‌های باگ

خصوصیتی که در تمامی الگوریتم‌های باگ مشترک می‌باشد این است که، در هر گام از اجرای الگوریتم، تصمیم مناسب برای حرکت به سوی هدف و در عین حال برخورد نکردن به موانع، به صورت محلی و با استفاده از درکی که ربات از محیط دارد، انجام می‌گیرد. از آنجا که برد حسگرهای ربات محدود می‌باشد، در هر لحظه ربات تنها قادر به درک بخشی از محیط پیرامونش می‌باشد که به همین علت است که الگوریتم‌های باگ به صورت محلی تصمیم-گیری انجام می‌دهند. خانواده‌ی الگوریتم‌های باگ، با ارائه شدن الگوریتم‌های باگ۱ و ۲ [۳۷] در سال ۱۳۶۵ (۱۹۸۶م.) متولد شد. ایده‌ی کلی الگوریتم‌های باگ بسیار ساده است: تا زمانی که مسیر مستقیم به سمت هدف باز باشد، ربات مستقیماً به طرف هدف حرکت می‌کند (رفتار حرکت-به-سوی-هدف) و اگر مانعی بر سر راهش قرار بگیرد، با دنبال کردن مرز مانع آن را دور می‌زند (رفتار دنبال-کردن-مرز-مانع). باگ۱ با برخورد به مانعی در مسیرش، ابتدا یک دور کامل حول آن می‌زند و در دور دوم از نقطه‌ای که نزدیک‌ترین فاصله تا هدف را داراست، از مانع جدا شده و به مسیر خود ادامه می‌دهد [۳۸]. نمونه‌ای از اجرای الگوریتم باگ۱ در شکل ۲-۵ نشان داده شده

[1] Nearness Diagram (ND)

است. روشی که باگ۱ برای دور زدن مانع در نظر می‌گیرد اصلاً بهینه نمی‌باشد. باگ۲ برای کوتاه‌تر کردن مسیر پیمایش شده توسط باگ۱، با برخورد به مانع حول آن می‌چرخد اما به محض آن که بتواند دوباره به طور مستقیم به سمت هدف حرکت کند، مانع را رها کرده و حرکت به سمت هدف را از سر می‌گیرد. به این ترتیب در برخی موارد مسافت پیموده شده توسط باگ۲ از باگ۱ کمتر می‌شود مانند سناریویی که در شکل ۲-۶ نشان داده شده است. اما مواردی نیز وجود دارد که باگ۲ بدتر از باگ۱ عمل خواهد کرد و طول مسیر تولید شده توسط آن از طول مسیر تولید شده توسط باگ۱ بیشتر می‌شود [۳۸]. لازم به ذکر است که هر دو الگوریتم باگ۱ و ۲ از حسگر برخورد برای درک موانع استفاده می‌کنند.

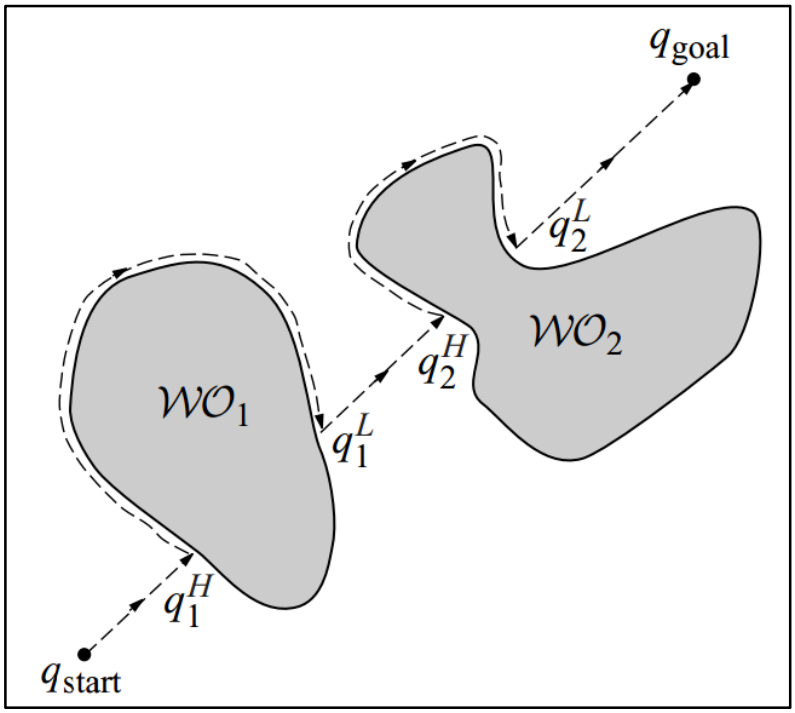


شکل ۲- ۵-نمونه‌ای از اجرای الگوریتم باگ۱: q_i^H iامین نقطه‌ی برخورد با مانع و q_i^L iامین نقطه‌ی ترک مرز مانع می‌باشد [۳۹].

بعد از باگ۲، در سال ۱۳۸۸ (۲۰۰۹م.) الگوریتم باگ۲+[1] [۴۰] به عنوان نسخه‌ی اصلاح شده‌ی آن ارائه شد. باگ۲+ با اعمال تغییراتی در شرایط سوئیچ کردن از رفتار دنبال-کردن-مرز-مانع به رفتار حرکت-به-سوی-هدف، طول مسیر پیموده شده توسط الگوریتم باگ۲ را در صورت امکان کاهش می‌دهد. به عبارت دیگر طول مسیر پیموده شده توسط الگوریتم باگ۲+ کمتر یا مساوی طول مسیر پیموده شده توسط الگوریتم باگ۲ می‌باشد. اگر چه الگوریتم باگ۲+ برای ناوبری در محیط‌های ناشناخته طراحی شده اما، از آن در طرح‌ریزی مسیر در محیط‌های شناخته شده نیز استفاده شده است. الگوریتمی که از باگ۲+ برای طرح‌ریزی در محیط شناخته شده بهره می‌برد، اِی-باگ[2] [۴۱] نام دارد که در سال ۱۳۸۸ (۲۰۰۹م.) ارائه شده است. در واقع ای‌باگ از ترکیب باگ۲+ و A^* تشکیل

[1] Bug2+
[2] ABUG

شده است و از دسته الگوریتم‌های طرح‌ریزی هر زمانه[1] به شمار می‌آید. الگوریتم‌های طرح‌ریزی هر زمانه به آن دسته از الگوریتم‌ها اطلاق می‌شود که در آن‌ها ابتدا یک مسیر غیر بهینه در زمانی کوتاه محاسبه می‌شود اما در حین طی کردن مسیر محاسبه شده، تا آنجا که وقت اجازه دهد کیفیت مسیر به مرور و به صورت افزایشی بهبود داده می‌شود. الگوریتم باگ۲+ هنگامی که به مانعی می‌رسد دو انتخاب برای دنبال کردن مرز آن دارد، یکی دنبال کردن مرز از سمت چپ و دیگری دنبال کردن مرز از سمت راست. حال با فرض این که محیط شناخته شده باشد، الگوریتم ای‌باگ برای هر مانع موجود بر سر راهش، یک گره با دو فرزند یکی برای دنبال کردن مرز مانع از چپ و دیگری برای دنبال کردن آن از راست، در نظر می‌گیرد. به این ترتیب یک درخت جستجوی دودویی تشکیل داده می‌شود. سپس الگوریتم A^* بر روی درخت مذکور برای محاسبه‌ی مسیرهای ممکن از نقطه‌ی شروع به هدف اجرا می‌شود و در ادامه مسیرهای به دست آمده به صورت افزایشی و تا آنجا که وقت اجازه می‌دهد بهبود داده می‌شوند و بهترین آن‌ها برای ادامه‌ی حرکت به سمت هدف انتخاب می‌شود.



شکل ۲- ۶-نمونه‌ای از اجرای الگوریتم باگ۲: q_i^H iامین نقطه‌ی برخورد با مانع و q_i^L iامین نقطه‌ی ترک مرز مانع می‌باشد [۳۹].

از دیگر الگوریتم‌های خانواده‌ی باگ می‌توان به سی‌باگ[2] [۴۲] ارائه شده در سال ۱۳۸۷ (۲۰۰۸م.) اشاره کرد. اساس کار این الگوریتم به این صورت است که ابتدا یک بیضی با مساحت A0 حول نقاط شروع و پایان در نظر گرفته می‌شود به طوری که نقاط شروع و پایان، نقاط کانونی بیضی را تشکیل دهند. حال درون بیضی مذکور الگوریتم باگ۱ از نقطه‌ی شروع اجرا می‌شود و سعی می‌کند به هدف برسد. در صورتی که الگوریتم نتواند هدف را

[1] Anytime planner
[2] CBUG

پیدا کند، روند قبلی تکرار می‌شود که در هر بار تکرار، بیضی مذکور مساحتش برابر $2^i A0$ در نظر گرفته می‌شود. که در آن i شماره تکرار می‌باشد. این روند آنقدر ادامه می‌یابد تا ربات به هدف برسد و یا متوجه شود که هدف دست نیافتنی است.

در [۴۲] نشان داده شده است که سی‌باگ حد پایین طول مسیر تولید شده توسط الگوریتم‌های برخط[1] را رعایت می‌کند و در نتیجه جزء الگوریتم‌های برخط بهینه به شمار می‌آید و در ضمن حافظه‌ی مصرفی‌اش مقداری ثابت است. طبق [۴۲] حد پایین طول مسیر تولید شده به صورت برخط، تابعی درجه دو از طول مسیر بهینه‌ی تولید شده به صورت برون‌خط[2] می‌باشد که این مطلب در معادله‌ی ۲-۱ آورده شده است. در این معادله l طول مسیر تولید شده‌ی برخط و l_{opt} طول مسیر بهینه‌ی تولید شده‌ی برون‌خط می‌باشد. ضرائب c_2، c_1 و c_0 ثابت و مثبت می‌باشند و با توجه به اندازه‌ی ربات به گونه‌ای انتخاب می‌گردند که واحد (متر) دو طرف نامساوی یکسان باشد.

$$l \le c_2 l_{opt}^2 + c_1 l_{opt} + c_0 \qquad (۱-۲)$$

یکی از مهم‌ترین الگوریتم‌های باگ که بعدها الگوریتم‌های دیگر مانند وج‌باگ [۳] و باگِ محتاط[3] [۴۳] بر پایه‌ی آن بنا شدند، تنژنت‌باگ [۴] ارائه شده در سال ۱۳۷۷ (۱۹۹۸م.) می‌باشد. این الگوریتم از حسگر مسافت‌یاب با میدان دید ۳۶۰ درجه (برخلاف باگ۱و۲ که از حسگر برخورد برای تشخیص موانع استفاده می‌کردند) بهره می‌برد. طول مسیری که ربات با استفاده از این روش پیمایش می‌کند کمتر از باگ۱و۲ خواهد بود زیرا که برای درک مانع نیازی به پیمودن مسیر تا مانع و برخورد با آن نیست و همین که مانع در برد عملیاتی حسگر مسافت‌یاب قرار گیرد، برای ربات قابل تشخیص می‌باشد و هنگامی که مانعی بر سر راه ربات تشخیص داده شود، تنژنت‌باگ طی محاسباتی نقاط ابتدا و انتهای مانع مذکور را به دست می‌آورد و به سمت یکی ازآن‌ها حرکت می‌کند. لازم به ذکر است که برد حسگر مسافت‌یاب (پویش‌گر لیزری) ۳۰ متر می‌باشد که این میزان، زمان کافی برای تصمیم‌گیری را، در اختیار تنژنت‌باگ قرار می‌دهد. تنژنت‌باگ علاوه بر آن که طول مسیر پیموده شده تا هدف را کاهش می‌دهد، مزیت دیگری نیز دارد و آن حفظ ایمنی ربات می‌باشد، از آن جهت که ربات نیازی به برخورد به موانع برای حس کردنشان ندارد و همواره می‌تواند فاصله‌ی خود را از آن‌ها رعایت کند. در فصل سوم الگوریتم تنژنت‌باگ به طور کامل شرح داده خواهد شد.

اکثر الگوریتم‌های خانواده‌ی باگ مسائلی که در پیاده‌سازی عملی باید مورد توجه قرار گیرد را در نظر نمی-گیرند. از جمله‌ی این مسائل می‌توان به ابعاد ربات، محدودیت‌های غیر-هولونومیکی ربات و مکان‌یابی اشاره کرد. یکی از الگوریتم‌های باگ که مسائل مذکور را در نظر گرفته است، الگوریتم وج‌باگ [۳] (نسخه‌ی توسعه یافته‌اش

[1] Online
[2] Offline
[3] CautiousBug

روربا‌گ [۳] در مریخ‌نورد راکی۷ استفاده شده است.) می‌باشد که در سال ۱۳۷۸ (۱۹۹۹م.) ارائه شده است. وج‌باگ از بینائی استریو برای تشخیص دادن موانع موجود بر سر راه ربات بهره می‌برد و به گونه‌ای طراحی شده است که مصرف انرژی، میزان محاسبات و میزان حافظه را به حداقل برساند. علت این امر مواجه بودن مریخ‌نوردها با محدودیت‌های شدید بر روی منابع مذکور می‌باشد. خصوصیت بارز وج‌باگ توانائی انجام ناوبری با استفاده از میدان دید محدود (۳۰ تا ۴۵ درجه) بینائی استریو می‌باشد. در واقع اساس کار وج‌باگ به این صورت است که قسمتی از محیط را با استفاده از دوربین‌های استریو بررسی می‌کند و در صورت نیاز دوربین‌هایش را به سمت مناسب می‌چرخاند و دوباره به بررسی ناحیه‌ی جدیدی که در دیدش قرار گرفته است، می‌پردازد. به این ترتیب حجم محاسبات به حداقل می‌رسد زیرا که فقط قسمت‌هایی از محیط مورد پردازش قرار می‌گیرد، که مورد نیاز است. این در حالی است که الگوریتم‌هایی که از پویش‌گر لیزری برای درک محیط استفاده می‌کنند، در هر بار پویش حجم زیادی داده دریافت می‌کنند که جمع‌آوری و پردازش آن‌ها زمان‌بر خواهد بود.

یکی دیگر از الگوریتم‌های باگ که سعی کرده است تا حدودی به پیاده‌سازی عملی نزدیک باشد، آی‌باگ[1] [۴۴] نام دارد. این الگوریتم برای مواقعی که داده‌های حسگرها خیلی قابل اعتماد نیستند و مکان‌یابی دقیق میسر نمی‌باشد، طراحی شده است. الگوریتم آی‌باگ فرض می‌کند که نقطه‌ی هدف از خود، یک سیگنال منتشر می‌کند و ربات حسگری برای درک سیگنال مذکور دارد، به گونه‌ای که هر چه ربات به هدف نزدیک‌تر باشد، سیگنال قوی‌تری دریافت می‌کند و هر چه از هدف دورتر باشد، سیگنال ضعیف‌تری دریافت می‌کند. در واقع حسگر سیگنال، تنها حسگری است که داده‌های قابل اعتماد تولید می‌کند. علاوه بر حسگر اندازه‌گیری قدرت سیگنال هدف، ربات به حسگر دیگری نیز مجهز می‌باشد که به کمک آن می‌تواند بفهمد که در هر لحظه رو به هدف می‌باشد یا خیر. نکته‌ی قابل توجه در مورد الگوریتم آی‌باگ این است که ربات نه تنها از موانع موجود در محیط، بلکه از موقعیت خود و موقعیت هدف نیز بی‌اطلاع است ولی باز هم می‌تواند به هدف برسد.

یکی دیگر از روش‌هایی که به جوانب عملی پیاده‌سازی ناوبری در محیط‌های ناشناخته با تأکید بر محدودیت‌های حرکتی ربات آکرمن پرداخته است، در سال ۱۳۹۱ (۲۰۱۲م.) در [۴۵] ارائه شده است که شبیه به الگوریتم‌های باگ عمل می‌کند. روش مذکور مانند الگوریتم‌های باگ دارای دو رفتار حرکت-به-سوی-هدف و دنبال-کردن-مرز-مانع می‌باشد. در این روش برای عبور از هر مانع سد کننده، دو مسیر (دور زدن به چپ و دور زدن به راست) با استفاده از الگوریتم تعقیب محض[2] [۴۶] ایجاد می‌شود، سپس مسیرهای مذکور از نظر محدودیت‌های غیر-هولونومیکی ربات آکرمن مورد بررسی قرار می‌گیرند و اگر هر دو قابل پیمایش بودند، مسیر کوتاه‌تر برای

[1] I-Bug
[2] Pure pursuit

حرکت انتخاب می‌شود و در غیر این صورت مسیری که قابل پیمایش می‌باشد (صرف نظر از طولش)، برای حرکت انتخاب می‌شود.

تمام الگوریتم‌های باگ که تا به اینجا معرفی شدند، برای ناوبری در محیط‌هایی مناسب هستند که تنها دارای موانع ساکن باشند. به عبارت دیگر هیچ یک از روش‌های بررسی شده توانایی اجتناب از موانع متحرک را ندارند. اما اخیراً (سال ۲۰۱۲م.) الگوریتمی به نام دی‌اچ‌باگ[1] [۴۷] ارائه شده است که قادر به انجام ناوبری در محیط‌هایی است که علاوه بر موانع ثابت، موانع متحرک نیز در آن‌ها وجود دارد. همگرا شدن دی‌اچ‌باگ به هدف در صورتی که محیط فقط دارای موانع ثابت باشد، تضمین شده است و اگر موانع متحرک نیز در محیط موجود باشد، دی‌اچ‌باگ با در نظر گرفتن یک سری فرضیات، تقریباً به هدف همگرا می‌شود. ارائه‌دهندگان این الگوریتم ادعا می‌کنند که تمام جوانب پیاده‌سازی عملی را در نظر گرفته‌اند. این در حالی است که در مقاله‌ی دی‌اچ‌باگ، درباره‌ی نحوه‌ی تشخیص موانع متحرک، هم‌چنین جهت حرکت و میزان سرعت آن‌ها مطلبی به طور صریح بیان نشده است. الگوریتم دی‌اچ-باگ، ربات و موانع متحرک را به صورت دایره‌ای شکل با شعاعی مشخص در نظر می‌گیرد. اساس کار این الگوریتم برای اجتناب از موانع متحرک به این صورت است که بعد از تشخیص آن‌ها، احتمال برخورد را بررسی می‌کند و در صورتی که احتمال برخورد وجود داشته باشد (که این حالت وضعیت خطرناک نام دارد)، رفتار اجتناب از مانع متحرک فعال می‌شود که طی آن، ربات ابتدا توقف کرده، به سمتی که ایمن می‌باشد می‌چرخد و دوباره به حرکت خود در جهت امن ادامه می‌دهد.

برای بررسی احتمال برخورد با موانع متحرک به این صورت عمل می‌شود که بعد از تشخیص موانع متحرک، برای هر یک از آن‌ها یک نقطه‌ی خطرناک در نظر گرفته می‌شود. در واقع نقطه‌ی خطرناک همان نقطه‌ای است که احتمال برخورد ربات با مانع متحرک در آنجا وجود دارد. در شکل ۲-۷ محاسبه‌ی نقطه‌ی خطرناک نشان داده شده است که در آن، R ربات، M مانع متحرک و D نقطه‌ی خطرناک می‌باشد. برای محاسبه‌ی D، ابتدا برآیند بردار سرعت ربات (v) و قرینه‌ی بردار سرعت مانع ($-v_O$) محاسبه می‌شود که با v' در شکل مشخص شده است. سپس پاره‌خط MD از مکان مانع (M) به خطی که در راستای v' قرار دارد، عمود می‌گردد که طول MD با d_S نمایش داده شده است. واضح است که اندازه‌ی پاره‌خط RD برابر $d_O \cos(\varphi)$ می‌باشد. حال اگر اندازه‌ی پاره‌خط RD از یک حد مشخص بزرگ‌تر باشد (معادله‌ی ۲-۲ [۴۷])، ربات در حالت ایمن قرار دارد و در غیر این صورت در حالت خطرناک می‌باشد (احتمال برخورد با مانع متحرک وجود دارد).

$$d_O \cos(\varphi) > r + r_O + r_s \qquad (۲-۲)$$

در معادله‌ی ۲-۲ r شعاع ربات، r_O شعاع مانع متحرک و r_S مسافت رزرو شده برای تغییر حالت ربات از موقعیت

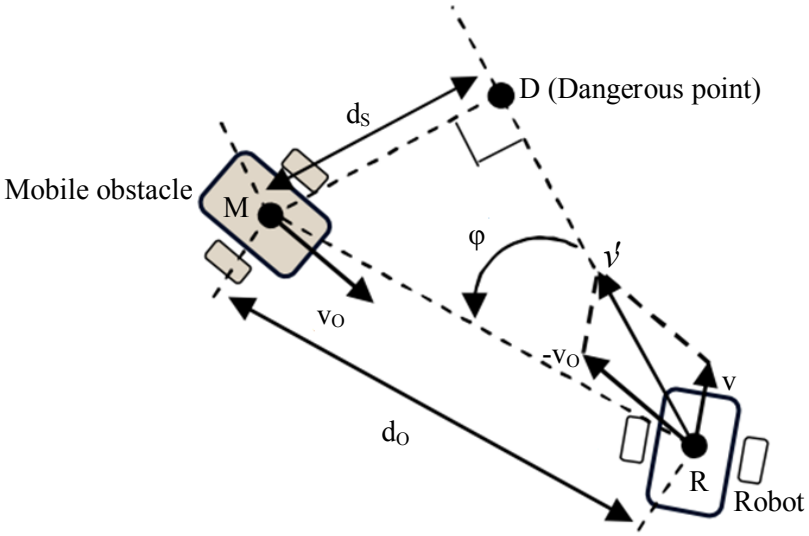
[1] DH-Bug

خطرناک به موقعیت ایمن می‌باشد. r_S مجموع سه ترم می‌باشد :

۱- مسافت تخمین زده شده که ربات قبل از ایست کامل می‌پیماید.

۲- مسافت تخمین زده شده که مانع متحرک در مدت زمان t می‌پیماید. که t زمان مورد نیاز ربات برای ایستادن و چرخش به سمت جهت ایمن می‌باشد.

۳- میزان مسافتی به نام Δr که برای جبران خطاهای تخمین در نظر گرفته شده است.



شکل ۲- ۷-شمای کلی محاسبه‌ی نقطه‌ی خطرناک [۴۷]

۲-٤ نتیجه‌گیری

در این فصل ابتدا تاریخچه‌ی ربات‌های متحرک بیان و سپس دسته‌بندی طرح‌ریزهای مسیر ارائه شد و کارهای پیشین انجام شده روی آن‌ها به صورت اجمالی مرور گردید. در فصل بعدی مفاهیم علمی که برای مطالعه‌ی این پایان‌نامه مورد نیاز می‌باشد، مورد بررسی قرار می‌گیرد.

فصل سوم

مفاهیم علمی پیش‌نیاز پایان‌نامه

۳-۱ مقدمه

در این فصل مفاهیم علمی مورد نیاز این پایان‌نامه بررسی می‌گردد. همان طور که درفصل اوّل اشاره شد، روش پیشنهادی که در این پایان‌نامه برای ناوبری در محیط‌های ناشناخته و خارج از جاده ارائه می‌شود، بر پایه‌ی الگوریتم تنژنت‌باگ می‌باشد. به همین علت در این فصل به بررسی تنژنت‌باگ خواهیم پرداخت. از طرف دیگر روش پیشنهادی این پایان‌نامه از مفاهیم میدان پتانسیل[1] و هیستوگرام میدان برداری[2] نیز بهره می‌برد که این روش‌ها نیز در این فصل بررسی خواهند شد. لازم به ذکر است که روش هیستوگرام میدان برداری خود، اصلاح شده‌ی روش میدان نیروی مجازی[3] می‌باشد به همین علت روش نیروی مجازی نیز به صورت اجمالی شرح داده می‌شود.

۳-۲ الگوریتم تنژنت‌باگ

الگوریتم تنژنت‌باگ [۴] اوّلین الگوریتم خانواده‌ی باگ می‌باشد که از حسگر مسافت‌یاب برای درک محیط استفاده کرده است. استفاده از حسگر مسافت‌یاب به جای حسگر تماس دارای دو مزیت می‌باشد :

[1] Potential Field (PF)
[2] Vector Field Histogram (VFH)
[3] Virtual Force Field (VFF)

۱- ربات برای درک کردن موانع نیازی به برخورد کردن به آن‌ها ندارد. که این امر از لحاظ ایمنی ربات حائز اهمّیت است.

۲- ربات با استفاده از حسگر مسافت‌یاب می‌تواند موانع موجود در محیط را زودتر (در مقایسه با حسگر تماس) درک کند و در نتیجه با زودتر تصمیم‌گرفتن، میزان مسافت کم‌تری را برای رسیدن به هدف طی کند و به این ترتیب مسیر بهتری را بپیماید.

الگوریتم تنژنت‌باگ از نوعی گراف به نام گراف مماس محلی[1] استفاده می‌کند. علت این نام‌گذاری این است که حسگر مسافت‌یاب قادر به درک تمام محیط نیست زیرا که دارای برد محدود می‌باشد در نتیجه گراف مذکور باید با توجه به داده‌هایی که حسگر مسافت‌یاب به صورت محلی (محیط پیرامون ربات تا آنجا که برد حسگر اجازه می‌دهد) درک کرده است، ساخته شود. در ادامه گراف مماس و سپس نسخه‌ی محلی آن یعنی همان گراف مماس محلی شرح داده می‌شود. لازم به ذکر است که برای توضیح گراف مماس ابتدا باید گراف رویت[2] معرفی گردد.

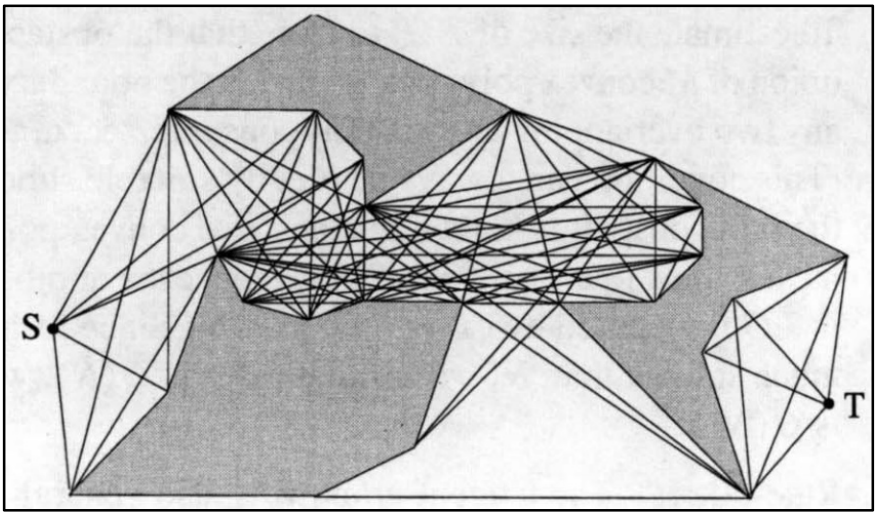
۳-۲-۱ گراف رویت

برای تعریف گراف رویت محیط پیرامون ربات، فرض کنید محیط مذکور دارای موانعی از نوع چندضلعی[3] بوده و دارای دو نقطه‌ی شروع (S) و هدف (T) باشد. حال گراف رویت محیط مذکور به صورت $VG=(V_v,E_v)$ نمایش داده می‌شود که V_v مجموعه‌ی رأس‌ها و E_v مجموعه‌ی یال‌های گراف رویت می‌باشند. نقاط S و T و رئوس موانع چندضلعی اعضای مجموعه‌ی V_v راتشکیل می‌دهند. در ضمن پاره‌خط‌هایی که با موانع موجود در محیط تلاقی ندارند و رئوس گراف رویت (مجموعه‌ی V_v) را به یکدیگر متصل می‌کنند، اعضای مجموعه‌ی E_v را تشکیل می‌دهند. نمونه‌ای از گراف رویت در شکل ۳-۱ نشان داده شده است که در آن موانع به صورت چندضلعی‌های سیاه‌رنگ نشان داده شده‌اند و خطوط بین آن‌ها یال‌های گراف رویت می‌باشند که رأس‌های چندضلعی‌های مذکور را به یکدیگر متصل کرده‌اند.

نشان داده شده است که کوتاه‌ترین مسیر ممکن بین دو نقطه‌ی S و T همواره از یال‌های گراف رویت عبور خواهد کرد [۴]. به این ترتیب با جست و جوی گراف رویت می‌توان مسیر بهینه بین دو نقطه‌ی مذکور را به دست آورد. اما همان طور که در شکل ۳-۱ هم مشخص است، گراف رویت از دسته گراف‌های چگال است. به عبارت دیگر تعداد یال‌های گراف رویت با N رأس، از مرتبه‌ی $O(N^2)$ می‌باشد [۴]. این تعداد زیاد یال‌ها، سرعت الگوریتم جست و جوی مسیر بهینه را به شدت کاهش می‌دهد که مسلماً مطلوب نمی‌باشد. با بررسی گراف رویت، این نکته نمایان می‌شود که تعداد زیادی از یال‌های گراف مذکور هیچ‌گاه در مسیر بهینه استفاده نمی‌شوند و به عبارت دیگر

[1] Local Tangent Graph (LTG)
[2] Visibility graph
[3] Polygonal

زائد هستند. پس باید یال‌های زائد گراف رویت تا حد ممکن حذف گردد. برای این کار، گراف مماس توسط رهنرت[1] ارائه گردیده است که بعدها توسط لیو[2] و آریموتو[3] [۴۸] توسعه داده شد. می‌توان نشان داد که مسیر بهینه همواره در گراف مماس موجود می‌باشد. به این ترتیب گراف مماس جایگزین مناسبی برای گراف رویت به نظر می‌رسد زیرا که خلوت‌تر از گراف رویت می‌باشد و در عین حال مسیر بهینه را نیز شامل می‌شود.



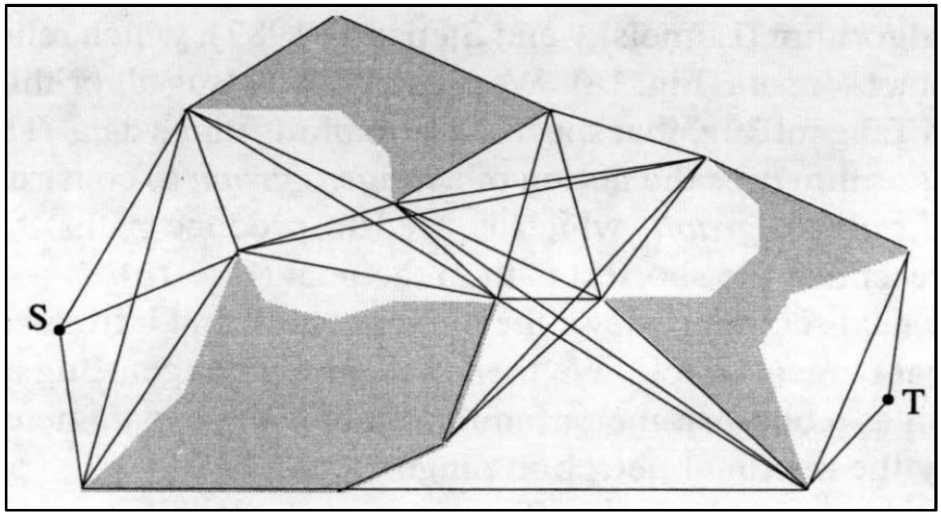
شکل ۳-۱-نمونه‌ای از گراف رویت [۴]

۳-۲-۲ گراف مماس

در واقع گراف مماس زیرگرافی از گراف رویت می‌باشد که فقط شامل رئوس محدب[4] است و یال‌هایش دو-مماس[5] بر رئوس مذکور می‌باشد. در ادامه به تعریف رأس محدب و یال دو-مماس می‌پردازیم. برای این کار یکی از موانع چندضلعی محیط را در نظر بگیرید. مسلم است که چندضلعی مذکور دارای تعدادی رأس می‌باشد. سه رأس متوالی از رئوس آن را a، p و b بنامید به طوری که apb یکی از زوایای چندضلعی باشد. حال رأس p را محدب می‌گویند هر گاه، زاویه‌ی $\widehat{apb}$، کم‌تر از ۱۸۰ درجه باشد؛ توجه شود که در غیر این صورت رأس p، غیرمحدب تلقی می‌شود [۴۸]. یال دو-مماس نیز، به یالی گفته می‌شود که هم زمان به دو رأس از رئوس موانع، مماس باشد. لازم به ذکر است که یال e را مماس به رأس p گویند هر گاه، یال e زاویه‌ی $\widehat{apb}$ را به دو قسمت تقسیم نکند و در غیر این صورت، یال e را غیر مماس گویند [۴۸]. در شکل ۳-۲ نمونه‌ای از گراف مماس آورده شده است. همان طور که مشخص است، تعداد یال‌های گراف مماس نشان داده شده در مقایسه با گراف رویتی که در شکل ۳-۱ ارائه شد، بسیار کم‌تر است. به طور کل می‌توان نشان داد که تعداد یال‌ها در گراف مماس با N رأس، از مرتبه‌ی O(N) می‌باشد

[1] Rohnert
[2] Liu
[3] Arimoto
[4] Convex
[5] Bitangent

در حالی که تعداد یال‌های گراف رویت با N رأس، از مرتبه‌ی $O(N^2)$ است [۴].



شکل ۳-۲-نمونه‌ای از گراف مماس [۴]

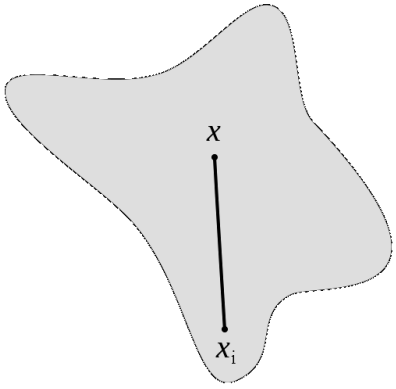
۳-۲-۳ گراف مماس محلی

همان طور که قبلاً نیز اشاره شد، الگوریتم تنژنت‌باگ از حسگر مسافت‌یاب برای درک موانع موجود در محیط استفاده می‌کند. به علاوه می‌دانیم که برد حسگر مسافت‌یاب محدود می‌باشد. به همین علت درک کل محیط برای ربات ممکن نیست و ربات تنها قادر به درک قسمتی از محیط می‌باشد که در محدوده‌ی کارکرد حسگر مسافت‌یاب قرار دارد. فرض کنید فضای آزاد محیط عملیاتی ربات مکمل فضای داخلی موانع موجود در محیط مذکور باشد. به علاوه فرض کنید موقعیت فعلی ربات با x نمایش داده شود. حال مجموعه‌ی قابل رویت[1]، مجموعه-ای است ستاره‌ای شکل[2] به مرکزیت x و با حداکثر شعاع R (R حداکثر برد حسگر مسافت‌یاب می‌باشد)، که در فضای آزاد محیط قرار دارد. به مجموعه‌ای مانند S، ستاره‌ای شکل می‌گویند هر گاه، مجموعه‌ی مذکور در فضای n بعدی اقلیدسی تعریف شده باشد به طوری که عضوی مانند x (که به آن مرکز S می‌گویند) در S موجود باشد که برای هر عضو داخل S مانند x_i، پاره‌خط بین x و x_i درون S قرار گرفته باشد. در شکل ۳-۳ نمونه‌ای از یک مجموعه‌ی ستاره‌ای شکل نشان داده شده است.

حال می‌توان تعریف گراف مماس محلی را بیان کرد. گراف مماس محلی یک گراف مماس است به طوری که فقط موانعی از محیط را در نظر می‌گیرد که در مجموعه‌ی قابل رویت واقع شده‌اند. علت وجود کلمه‌ی "محلی" در گراف مماس محلی واضح است، چرا که گراف مذکور با توجه به اطلاعاتی که حسگر مسافت‌یاب با برد محدود در اختیار ربات قرار می‌دهد، ساخته می‌شود. در واقع در ساخت گراف مماس محلی فرض می‌شود موانعی که در

[1] Visible set
[2] Star-shaped set

مجموعه‌ی قابل رویت قرار دارند، تمام موانع موجود در محیط می‌باشند. با در نظر گرفتن این فرض، الگوریتم تنژنت‌باگ به صورت محلی تصمیمات مناسب برای عبور از موانع و رسیدن به هدف را می‌گیرد. لازم به ذکر است که الگوریتم تنژنت‌باگ موانع حس شده توسط حسگر مسافت‌یاب را به صورت دیوارهایی نازک مدل می‌کند. این فرض ابعاد موانع را کم‌تر از حد واقعیشان تخمین می‌زند مثلاً قسمت قابل رویت از یک مانع مستطیل شکل (که دارای ابعاد مشخصی است) به صورت یک دیواره‌ی نازک دیده می‌شود. توجه به این نکته ضروری است که استفاده از مدل دقیق‌تر برای تخمین ابعاد موانع، با توجه به بار محاسباتی‌اش مقرون به صرفه نمی‌باشد.



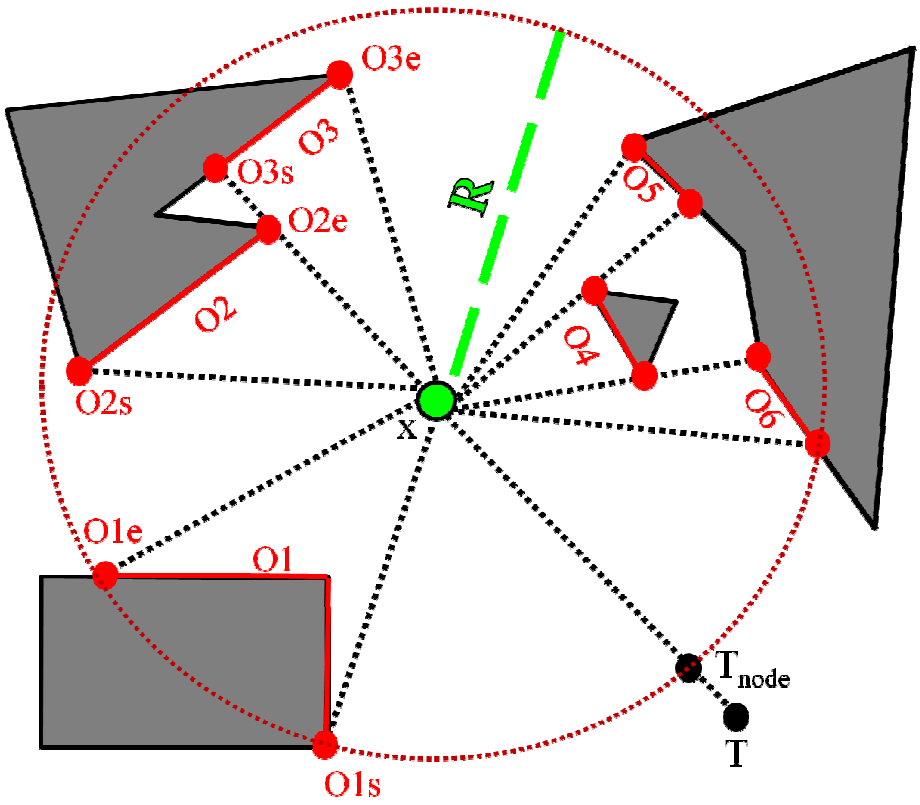
شکل ۳- ۳- نمونه‌ای از یک مجموعه‌ی ستاره‌ای شکل [۴۹].

برای ساخت گراف مماس محلی ابتدا داده‌های حاصل از حسگر مسافت‌یاب، به موانع مجزا از یکدیگر تقسیم‌بندی می‌شود. فرمت داده‌های مسافت‌یاب را می‌توان به صورت تابع $r(\theta)$ در نظر گرفت که در آن θ زاویه‌ای نسبت به یک جهت مرجع مشخص (مانند جهت سر ربات) و $r(\theta)$ میزان فاصله‌ی بین موقعیت فعلی ربات (x) تا نزدیک‌ترین نقطه‌ی مانع در جهت θ می‌باشد. تشخیص موانع مجزا از یکدیگر در داده‌های مسافت‌یاب بر این اصل استوار است که داده‌های متعلق به یک وجه قابل رویت یک مانع به صورت پیوسته تغییر می‌کنند. حال هر کجا که در داده‌های مذکور ناپیوستگی دیده شود، نشان‌دهنده‌ی نقطه‌ی تفکیک دو مانع از یکدیگر می‌باشد.

هر مانع مجزائی که تشخیص داده می‌شود، نقاط انتهائیش برای استفاده به عنوان بخشی از رئوس گراف مماس محلی ذخیره می‌گردد. برای مشخص شدن مفهوم نقاط انتهایی موانع، در شکل ۳-۴ نقاط انتهایی مانع O2 و O3 به ترتیب با نام‌های O2s، O2e و O3s، O3e مشخص شده است. لازم به ذکر است که هر مانع حس شده توسط حسگر مسافت‌یاب لزوماً معادل یک مانع در محیط نمی‌باشد زیرا لیزری که توسط حسگر مسافت‌یاب برای درک محیط تابانده می‌شود، به خط مستقیم حرکت می‌کند. از این رو ممکن است یک مانع با برآمدگی‌ها و فرورفتگی‌های مختلف به گونه‌ای در محیط قرار گرفته باشد که مسافت‌یاب توان درک آن به صورت یک مانع یکپارچه را نداشته باشد و آن را به صورت چند مانع مجزا مشاهده کند. به عنوان مثال در شکل ۳-۴ چهار عدد مانع واقعی در محیط وجود دارد اما بعد از پردازش داده‌های حسگر مسافت‌یاب، شش عدد مانع مجزا به نام‌های O1, O2, …, O6 از

محیط استخراج شده است. در شکل مذکور نقاط انتهایی شش مانع حس شده به صورت دایره‌های کوچک علامت-گذاری شده‌اند. علت به وجود آمدن دو مانع O5 و O6، قرار داشتن مانع مثلثی شکل (که یک وجه آن O4 نام گرفته است) رو به روی دید حسگر مسافت‌یاب می‌باشد. از طرفی مانعی که در بالای شکل ۳-۴ سمت چپ قرار دارد، به علت تورفتگی موجود در آن به صورت دو مانع مجزای O2 و O3 تشخیص داده شده است.

بعد از استخراج موانع مجزا از داده‌های مسافت‌یاب، می‌توان گراف مماس محلی را تشکیل داد. رئوس گراف مماس محلی عبارت است از موقعیت فعلی ربات (x)، نقاط انتهایی موانع استخراج شده از محیط و احتمالاً نقطه‌ای به نام T_{node}. که T_{node} نقطه‌ای بر روی پاره‌خط xT می‌باشد. در صورتی که پاره‌خط xT توسط مانعی مسدود نشده باشد، نقطه‌ی T_{node} روی خط مذکور و در فاصله‌ی R از ربات قرار دارد مانند سناریوی ارائه شده در شکل ۳-۴. با متصل کردن رأس موقعیت فعلی ربات (x) به سایر رئوس گراف مماس محلی ، یال‌های گراف مذکور تشکیل می‌گردد. لازم به ذکر است که گراف مورد استفاده در الگوریتم تنژنت‌باگ با گراف مماس محلی که در قسمت ۳-۲-۳ شرح داده شد کمی تفاوت دارد. به این صورت که گراف مماس محلی مورد استفاده در تنژنت‌باگ، از یال‌های دو-مماس استفاده نمی‌کند و هیچ‌گاه یال‌های مذکور را صریحاً ایجاد نمی‌کند [۴].



شکل ۳-۴-نمونه‌ای از گراف مماس محلی: x مکان ربات، R برد حسگر مسافت‌یاب و T نقطه‌ی هدف می‌باشد. موانع به صورت چندضلعی با اضلاع سیاه‌رنگ مشخص شده‌اند. مانعی که در بالای شکل سمت چپ قرار دارد به صورت دو مانع مجزای O2 و O3 و مانعی که در بالا سمت راست قرار دارد به صورت دو مانع O5 و O6 تشخیص داده شده است [۴].

۳-۲-٤ شرح الگوریتم تنژنت‌باگ

الگوریتم تنژنت‌باگ برای هدایت یک ربات نقطه‌ای (ابعاد ربات صفر فرض می‌شود) در محیطی دو بعدی و ناشناخته جهت اجتناب از موانع و رسیدن به هدف طراحی شده است. داده‌های ورودی مورد نیاز الگوریتم در هر گام عبارت است از موقعیت فعلی ربات (x)، داده‌های حسگر مسافت‌یاب با برد R و موقعیت نقطه‌ی هدف (T). توجه شود که تنژنت‌باگ هیچ گونه اطلاعاتی از مکان موانع موجود در محیط ندارد. در ادامه ابتدا شمای کلی الگوریتم مذکور شرح داده می‌شود و سپس هر قسمت از آن مفصل‌تر بررسی می‌گردد.

ساختار کلی الگوریتم تنژنت‌باگ

شبه کد الگوریتم تنژنت‌باگ در جدول ۳-۱ آورده شده است. تنژنت‌باگ مانند تمام الگوریتم‌های باگ دارای دو رفتار حرکت-به-سوی-هدف و دنبال-کردن-مرز-مانع می‌باشد. در هر گام ربات ابتدا با استفاده از داده-های حسگر مسافت‌یاب، گراف مماس محلی را تشکیل می‌دهد و سپس از آن برای تصمیم‌گیری در حالت‌های حرکت-به-سوی-هدف و دنبال-کردن-مرز-مانع استفاده می‌کند. الگوریتم تنژنت‌باگ با رفتار حرکت-به-سوی-هدف شروع به کار می‌کند و در طی آن ربات همواره در جهت کوتاه‌ترین مسیر تا هدف (در جهت بهینه‌ی محلی) حرکت می‌کند. رفتار حرکت-به-سوی-هدف ادامه می‌یابد تا یکی از دو شرط الف و یا ب (جدول ۳-۱) برقرار گردد. در صورت برقراری شرط الف، ربات به هدف رسیده است و الگوریتم متوقف می‌شود. در صورت برقراری شرط ب، ربات به رفتار دنبال-کردن-مرز-مانع سوئیچ می‌کند.

در حالت دنبال-کردن-مرز-مانع، ربات در حین حرکت همواره فاصله‌ی نزدیک‌ترین نقطه متعلق به قسمت قابل رویت از مرز مانع تا نقطه‌ی هدف را در متغیر $d_{Followed}$ ذخیره می‌کند. این رفتار تا زمانی که یکی از سه شرط پ، ت و یا ث (جدول ۳-۱) برقرار گردد، ادامه می‌یابد. شرط ث زمانی برقرار می‌گردد که رأسی مانند v_{leave} درگراف مماس محلی پیدا شود که در رابطه‌ی ۳-۱ صدق کند. در رابطه‌ی مذکور $d(v_{Leave},T)$ فاصله‌ی اقلیدسی بین v_{Leave} و T می باشد که به آن d_{Reach} نیز می‌گویند.

$$d(v_{leave},T) < d_{Followed} \qquad (۱-۳)$$

جدول ۳- ۱- شبه کد الگوریتم تنژنت‌باگ [۳۹]

۱- با توجه به گراف مماس محلی، در جهت بهینه‌ی محلی به سمت هدف (T) حرکت کن تا زمانی که یکی از دو شرط زیر برقرار گردد:
الف- ربات به هدف برسد، در این صورت ربات را متوقف کن.
ب- مقدار d(x,T) (فاصله‌ی اقلیدسی بین x و T) رو به افزایش بگذارد، در این صورت جهت دنبال کردن مرز مانع را برابر با جهت منتخب در آخرین رفتار حرکت-به-سوی-هدف قرار بده و

به رفتار دنبال-کردن-مرزمانع سوئیچ کن (به مرحله‌ی ۲ برو).

۲- مرز مانع را با استفاده از گراف مماس محلی دنبال کن و مقدار متغیرهای $d_{Followed}$ و d_{Reach} را به روز رسانی کن. این کار را ادامه بده تا یکی از سه شرط زیر برقرار شود:

پ-ربات به هدف برسد، در این صورت ربات را متوقف کن.

ت-ربات یک دور کامل به دور مانع بزند، در این صورت هدف قابل دسترس نیست پس ربات را متوقف کن.

ث-مقدار d_{Reach} کم‌تر از $d_{Followed}$ شود، در این صورت به رفتار حرکت-به-سوی-هدف سوئیچ کن (به مرحله‌ی ۱ برو).

شرح رفتار حرکت-به-سوی-هدف

همان طور که در قسمت ساختار کلی الگوریتم تنژنت‌باگ اشاره شد، هنگامی که ربات در رفتار حرکت-به-سوی-هدف به سر می‌برد، همواره در جهت کوتاه‌ترین مسیر تا هدف حرکت می‌کند. در صورت عدم وجود مانع مسدودکننده، کوتاه‌ترین مسیر از مکان فعلی ربات (x) تا هدف (T) ، دقیقاً همان پاره‌خط xT می‌باشد که نیازی به محاسبه ندارد. حال فرض کنید مانعی پاره‌خط xT را مسدود کرده است. طبق لم ۴ در [۴]، کوتاه‌ترین مسیر از x به T در صورت وجود مانع مسدودکننده، همواره از یکی از دو نقطه‌ی انتهایی مانع مذکور می‌گذرد. به این ترتیب دو مسیر نامزد $x\text{-}O_s\text{-}T$ و $x\text{-}O_e\text{-}T$ خواهیم داشت که باید یکی از آن‌ها را برای حرکت ربات انتخاب کنیم. برای این کار طول دو مسیر مذکور با استفاده از رابطه‌ی ۳-۲ تخمین زده می‌شود و مسیر با فاصله‌ی تخمینی کم‌تر انتخاب می‌گردد. به رابطه‌ی ۳-۲ فاصله‌ی مکاشفه‌ای مسیر $x\text{-}O_n\text{-}T$ می‌گویند.

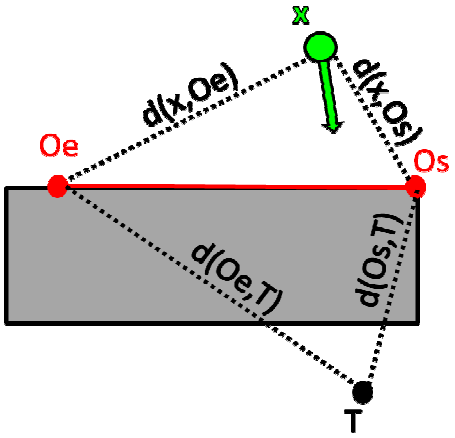
$$heuristic\_distance(x,T) = d(x,O_n) + d(O_n,T) \qquad (۳\text{-}۲)$$

که در رابطه‌ی ۳-۲ منظور از d(a,b) فاصله‌ی اقلیدسی بین دو نقطه‌ی a و b می‌باشد. در شکل ۳-۵ فاصله‌ی مکاشفه-ای مسیر $x\text{-}O_s\text{-}T$ کم‌تر از فاصله‌ی مکاشفه‌ای مسیر $x\text{-}O_e\text{-}T$ می‌باشد به همین علت مسیر $x\text{-}O_s\text{-}T$ برای دور زدن مانع انتخاب می‌شود. مادامی که d(x,T) رو به کاهش باشد، ربات رفتار حرکت-به-سوی-هدف را ادامه می‌دهد و در غیر این صورت ربات از رفتار حرکت-به-سوی-هدف به رفتار دنبال-کردن-مرز-مانع سوئیچ می‌کند.

شرح رفتار دنبال-کردن-مرز-مانع

در حین رفتار دنبال-کردن-مرز-مانع همواره فاصله‌ی نزدیک‌ترین نقطه متعلق به قسمت قابل رویت از مرز مانع تا نقطه‌ی هدف در متغیر $d_{Followed}$ ذخیره می‌گردد (مقدار متغیر $d_{Followed}$ مرتباً به روز رسانی می‌شود). علاوه بر متغیر $d_{Followed}$ الگوریتم تنژنت‌باگ مقدار متغیر d_{Reach} را نیز به روز رسانی می‌کند. اگر ربات به هدف برسد، شرط پ برقرار شده است و الگوریتم با موفقیت خاتمه می‌یابد. اما اگر ربات یک دور کامل حول مانع بچرخد، شرط ت

برقرار شده است و الگوریتم با حالت شکست متوقف می‌شود زیرا که هدف قابل دسترس نبوده است. زمانی که مقدار d_{Reach} کم‌تر از $d_{Followed}$ شود، شرط ث برقرار خواهد شد و ربات از رفتار دنبال-کردن-مرز-مانع به رفتار حرکت-به-سوی-هدف سوئیچ می کند.



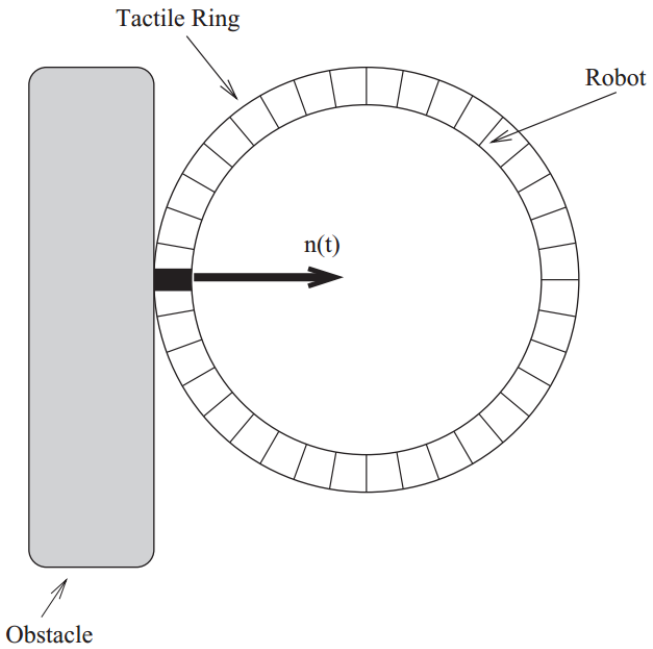
شکل ۳-۵-نحوه‌ی انتخاب بین یکی از دو مسیر کاندید x-Os-T و x-Oe-T برای عبور از مانع: فاصله‌ی مکاشفه‌ای مسیر x-Os-T کم‌تر از فاصله‌ی مکاشفه‌ای مسیر x-Oe-T می‌باشد به همین علت مسیر x-Os-T برای دور زدن مانع (مستطیل خاکستری) انتخاب می‌شود.

مشکل‌ترین قسمت از پیاده‌سازی الگوریتم تنژنت‌باگ رفتار دنبال-کردن-مرز-مانع می‌باشد زیرا که در این رفتار ربات باید مرز مانعی را دنبال کند که ناشناخته است. در نتیجه دنبال کردن مرز مانع باید به صورت افزایشی و در هر گام با توجه به داده‌های حسگر مسافت‌یاب انجام گیرد. در مقاله‌ای ([۴]) که در آن الگوریتم تنژنت‌باگ ارائه شده است، به طور صریح نحوه‌ی پیاده‌سازی رفتار دنبال-کردن-مرز-مانع بیان نشده است. اما در [۳۹] روشی برای آن، بر اساس بردار نرمال سطح مانع[1] مطرح گردیده است که در ادامه به شرح آن می‌پردازیم. اگر فرض شود موانع موجود در محیط دارای سطحی یکنواخت و صاف هستند، برای دنبال کردن مرز مانع کافی است که ربات موازی با سطح مانع حرکت کند. به عبارت دیگر ربات باید عمود بر بردار نرمال سطح مانع حرکت کند. اما موانع موجود در یک محیط واقعی لزوماً دارای سطحی یکنواخت و صاف نمی‌باشند. که در نتیجه برای آن که ربات بتواند موازی با مرز مانع حرکت کند باید در هر گام و به طور مرتب بردار نرمال سطح مانع را محاسبه کند. لازم به ذکر است که محاسبه‌ی دقیق بردار نرمال مذکور میسر نمی‌باشد اما می‌توان با روشی ساده بردار آن را با دقت قابل قبولی تخمین زد. برای این کار فرض می‌شود که سطح مانع به طور محلی صاف و یکنواخت[2] می‌باشد. با این فرض می‌توان از حسگر مسافت‌یاب برای تخمین زدن بردار نرمال سطح مانع به صورت محلی استفاده کرد. بعد از تخمین بردار نرمال در هر گام، ربات فاصله‌ی کوتاهی در راستای عمود بر بردار نرمال تخمین زده شده حرکت می کند و سپس در گام بعدی دوباره بردار نرمال با توجه به داده‌های حسگر به روز رسانی می‌شود و این روند تکرار می گردد.

[1] Obstacle surface normal vector
[2] Locally flat

حال که ایده‌ی کلی دنبال کردن مرز مانع بیان شد، به نحوه‌ی تخمین بردار نرمال سطح مانع با استفاده از حسگر مسافت‌یاب می‌پردازیم. فرض کنید که ربات به جای حسگر مسافت‌یاب دارای حسگر تماس[1] می‌باشد. به طوری که دور تا دور ربات تعداد زیادی حسگر تماس قرار داشته باشد و هر گاه ربات با مانعی برخورد می‌کند، بتواند بفهمد که کدام یک از حسگرهای مذکور مانع را حس کرده‌اند. به این ترتیب از مرکز ربات به حسگری که مانع را لمس کرده است، برداری تشکیل می‌شود که تخمینی از بردار نرمال سطح مانع می‌باشد. در شکل ۳-۶ نمونه‌ای از نحوه‌ی محاسبه‌ی بردار نرمال با استفاده از حسگرهای تماس نشان داده شده است.

توجه به این نکته ضروری است که استفاده از حسگر تماس برای تخمین بردار نرمال، مستلزم برخورد ربات با مانع می‌باشد. که این امر ایمنی ربات را به خطر می‌اندازد. پس ربات باید مرز مانع را به گونه‌ای دنبال کند که همواره فاصله‌ی خود را از مانع به اندازه‌ای مشخص مانند W^* حفظ کند. برای این کار به جای حسگرهای تماس از حسگر مسافت‌یاب استفاده می‌شود و روند کار به این صورت خواهد بود که ابتدا نزدیک‌ترین (کمینه‌ی سراسری داده‌های مسافت‌یاب) نقطه‌ی مانع به ربات مشخص می‌گردد. سپس جهت پرتویی که از حسگر مسافت‌یاب به نقطه‌ی مذکور برخورد کرده است تخمینی از بردار نرمال خواهد بود. یادآوری می‌شود که با مشخص شدن بردار نرمال می‌توان با حرکت کردن در جهت عمود بر آن، مرز مانع را دنبال کنیم.



شکل ۳-۶-نحوه‌ی تخمین بردار نرمال سطح مانع با استفاده از حسگرهای تماس، بردار نرمال با n(t) نمایش داده شده است و حسگری که مانع را لمس کرده با رنگ سیاه مشخص می‌باشد [۳۹].

[1] Tactile sensor

۳-۳ الگوریتم اجتناب از مانع میدان پتانسیل (PF)

الگوریتم میدان پتانسیل [۵۰] در ابتدا برای هدایت بازوهای رباتیک به سمت نقطه‌ی هدف و در عین حال اجتناب از برخورد آن‌ها با موانع، ارائه گردید اما به علت ساده بودن پیاده‌سازی و عملکرد قابل قبول (نسبت به ساده بودن)، در حوزه‌ی ربات‌های متحرک نیز مورد استفاده قرار گرفته است. در این روش ربات به صورت یک نقطه فرض می‌شود که تحت تأثیر یک میدان پتانسیل مجازی به نام U(q) در محیط حرکت می‌کند. میدان مذکور از برآیند میدان‌های پتانسیل جاذب[1] به نام $U_{att}(q)$ و دافع[2] به نام $U_{rep}(q)$ و با توجه به نقشه‌ی محیط (شامل موانع و نقطه-ی هدف)، محاسبه می‌گردد و ربات تحت تأثیر آن از بین موانع عبور می‌کند و به سمت هدف هدایت می‌شود. اساس کار میدان پتانسیل به این صورت است که موانع موجود در محیط بر ربات نیروی دافعه (F_{rep}) و نقطه‌ی هدف بر آن نیروی جاذبه (F_{att}) اعمال می‌کنند که برآیند نیروهای دافعه و جاذبه، نیرویی به نام F(q) است که باعث هدایت ربات به سمت هدف و اجتناب آن از موانع می‌شود:

$$F(q) = F_{att}(q) + F_{rep}(q) \qquad (۳-۳)$$

ساده‌ترین فرم الگوریتم میدان پتانسیل زمانی است که محیط به صورت دو بعدی و ربات به صورت نقطه‌ای در نظر گرفته شود. در ضمن موقعیت ربات با دوتایی q=(x,y) نشان داده شود و از جهت سر ربات (θ) نیز صرف نظر گردد. حال برای آن که ربات بتواند از موانع اجتناب کند و به هدف برسد، باید تعاریفی برای نیروهای دافعه (F_{rep}) و جاذبه (F_{att}) در نظر گرفت تا بتوان در هر گام با استفاده از آن‌ها، نیروهای دافعه و جاذبه‌ای که بر موقعیت ربات (نقطه‌ی q) اعمال می‌شوند را محاسبه کرد و برآیند گرفت. برای این کار، ابتدا باید تابع‌های میدان پتانسیل جاذب و میدان پتانسیل دافع که بر نقطه‌ی q تأثیر می‌گذارند، را تعریف کرد. در [۳۸] توابع مذکور به گونه‌ای تعریف شده‌اند که مشتق‌پذیر باشند. علت مشتق‌پذیری توابع میدان پتانسیل این است که نیروهای دافعه و جاذبه با گرفتن گرادیان توابع مذکور به دست می‌آیند. در ادامه روابط توابع میدان پتانسیل جاذب و دافع آورده شده است:

$$U_{att}(q) = \frac{1}{2} k_{att} \rho_{goal}^2(q) \qquad (۴-۳)$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & if\ \rho(q) \le \rho_0 \\ 0 & if\ \rho(q) \ge \rho_0 \end{cases} \qquad (۵-۳)$$

رابطه‌ی ۳-۴ میدان پتانسیل جاذب را تعریف می‌کند که در آن $\rho_{goal}(q)$ نشان‌دهنده‌ی فاصله‌ی اقلیدسی بین نقطه‌ی q و نقطه‌ی هدف (q_{goal}) است و k_{att} ضریب تغییر اندازه[3] می‌باشد. همان طور که مشاهده می‌شود، هر چقدر

[1] Attractive
[2] Repulsive
[3] Scaling factor

ربات به هدف نزدیک‌تر باشد (مقدار $\rho_{goal}(q)$ کوچک‌تر باشد) میزان پتانسیل جاذب نیز کاهش می‌یابد و هر چه ربات از هدف دورتر باشد میزان پتانسیل جاذب نیز افزایش می‌یابد.

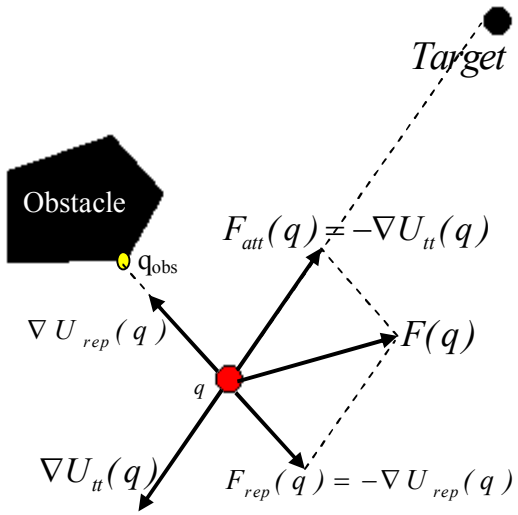
رابطه‌ی ۳-۵ میدان پتانسیل دافع را تعریف می‌کند که در آن $\rho(q)$ کم‌ترین فاصله بین مانع (مانعی که می‌خواهیم میزان میدان پتانسیل حاصل از آن را بر روی ربات به دست آوریم) و ربات می‌باشد. در ضمن k_{rep} ضریب تغییر اندازه است و ρ_0 حد آستانه‌ی فاصله بین ربات و مانع می‌باشد به این معنی که اگر میزان فاصله‌ی مانع تا ربات از حد ρ_0 بیشتر باشد، میدان پتانسیلی بر روی ربات اعمال نمی‌کند و در غیر این صورت اعمال می‌کند. همان طور که از رابطه‌ی ۳-۵ مشخص است، هر چقدر فاصله‌ی ربات تا مانع کم‌تر باشد میزان میدان پتانسیل دافعه بر ربات بیشتر خواهد بود و هر چقدر فاصله‌ی ربات تا مانع بیشتر باشد میزان میدان پتانسیل دافعه بر ربات کم‌تر می‌شود. علت این امر قرار داشتن $\rho(q)$ در مخرج کسر می‌باشد.

حال که تعاریف میدان پتانسیل دافع و جاذب بررسی گردید، می‌توان نیروهای دافعه و جاذبه را بر اساس آن‌ها تعریف کرد. همان طور که قبلاً اشاره شد میزان نیروی‌های مذکور با گرفتن گرادیان از میدان‌های پتانسیل دافعه و جاذبه به دست می‌آید. برای بررسی علت استفاده از گرادیان میدان‌های پتانسیل باید به این نکته توجه کرد که بردار گرادیان همواره به جهتی اشاره دارد که میزان تابع بیشینه گردد. حال با توجه به شکل ۳-۷ می‌توان علت استفاده از گرادیان را شرح داد. قبلاً اشاره شد که هر چقدر فاصله‌ی ربات تا مانع کم‌تر باشد میزان میدان پتانسیل دافعه بر ربات بیشتر خواهد بود. حال اگر نزدیک‌ترین نقطه‌ی مانع به ربات q_{obs} نام داشته باشد (شکل ۳-۷)، بردار $\overrightarrow{qq_{obs}}$ راستایی را نشان می‌دهد که میزان میدان پتانسیل دافعه در آن راستا بیشینه می‌گردد که در نتیجه با گرفتن گرادیان از تابع $U_{rep}(q)$ می‌توان برداری ($\nabla U_{rep}(q)$) هم‌راستا با $\overrightarrow{qq_{obs}}$ محاسبه کرد. از آنجا که دوری جستن از برخورد با مانع برای ربات مد نظر می‌باشد، با قرینه کردن جهت $\nabla U_{rep}(q)$ می‌توان بردار دافعه‌ی F_{rep} را به گونه‌ای به دست آورد که همواره ربات را از مانع دور کند.

با استفاده از شکل ۳-۷، در مورد بردار جاذبه‌ی F_{att} نیز می‌توان به طریق مشابه استدلال کرد. به یاد آورید که هر چقدر ربات از هدف دورتر باشد، میزان میدان پتانسیل جاذبه‌ای که بر آن وارد می‌شود بیشتر خواهد بود. پس با گرفتن گرادیان از میدان پتانسیل جاذبه می‌توان برداری ($\nabla U_{att(q)}$) که در راستای آن میزان نیروی جاذبه حداکثر می‌گردد را به دست آورد. که این بردار همان طور که در شکل ۳-۷ نیز مشاهده می‌شود دقیقاً در جهت عکس بردار $\overrightarrow{qTarget}$ می‌باشد. در نتیجه با ضرب کردن منفی در بردار $\nabla U_{att(q)}$ می‌توان برداری (F_{att}) که همواره ربات را به سمت هدف می‌کشاند به دست آورد. بعد از محاسبه‌ی بردار نیروی جاذبه‌ی هدف و بردارهای نیروی دافعه برای تمام موانع، با برآیندگیری از آن‌ها می‌توان بردار $F(q)$ را به دست آورد.

۳-٤ الگوریتم اجتناب از مانع میدان نیروی مجازی (VFF)

یکی از روش‌های اجتناب از مانع بلادرنگ که برای ربات‌های نسبتاً سریع ارائه شده است، میدان نیروی مجازی[1] (VFF) [۵۱] نام دارد. روش VFF در اکثر محیط‌های ناشناخته با پیکربندی‌های متفاوت (با چیدمان‌های مختلف موانع) به خوبی عمل می‌کند و حرکتی نرم، پیوسته در بین موانع را برای ربات به ارمغان می‌آورد. اساس کار VFF به این صورت است که با توجه به موانع موجود در محیط، هیستوگرامی دکارتی و در دو بعد به نام C ساخته می‌شود. در واقع هیستوگرام C نوعی نمایش موانع موجود در محیط می‌باشد به طوری که مقدار هر سلول (i,j) از هیستوگرام مذکور، میزان قطعیت وجود مانع در آن سلول را نشان می‌دهد. برای ساخت هیستوگرام C، تعدادی حسگر سونار[2] دور تا دور ربات تعبیه شده است. به ازاء هر بار خواندن یکی از سونارها، مقدار سلولی از هیستوگرام C که بر روی محور صوتیِ[3] سونار مذکور و در فاصله‌ای که سونار خوانده است، قرار دارد، یک واحد افزوده می‌شود (شکل ۳-۸). حال اگر محیط به طور مداوم و سریع در حین حرکت ربات با استفاده از سونارهای آن نمونه‌برداری شود، به مرور و به طور افزایشی تخمین مناسبی از مکان موانع، بر روی هیستوگرام C نقش می‌بندد علی‌رغم این که، داده‌های سونار مکان دقیق موانع را به ما نشان نمی‌دهند. در واقع با نمونه‌برداری مکرر و سریع، توزیع احتمال نواحی اشغال شده‌ی محیط، به صورت تخمینی به دست می‌آید.

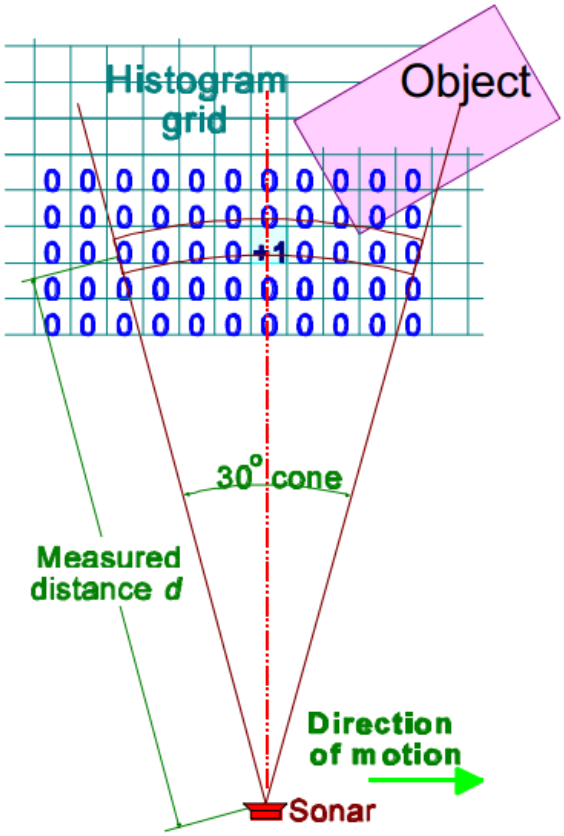


شکل ۳-۷-نحوه‌ی محاسبه‌ی برآیند نیروهای جاذبه و دافعه

روشی که VFF برای نگاشت موانع موجود در محیط بر روی هیستوگرام C استفاده می‌کند، از نظر محاسباتی بسیار کم هزینه است زیرا که به ازاء هر بار خواندن یک حسگر، تنها یک سلول از هیستوگرام به روز رسانی می‌شود.

---
[1] Virtual Force Field (VFF)
[2] Sonar
[3] Acoustic axis

این در حالی است که در روشی مشابه به نام شبکه‌ی قطعیت[1] [۵۲-۵۴]، هر بار خواندن حسگر موجب به روز رسانی چندین سلول می‌گردد که از نظر محاسباتی سنگین می‌باشد. در شکل ۳-۹ نمونه‌ای از هیستوگرام C نشان داده شده است.

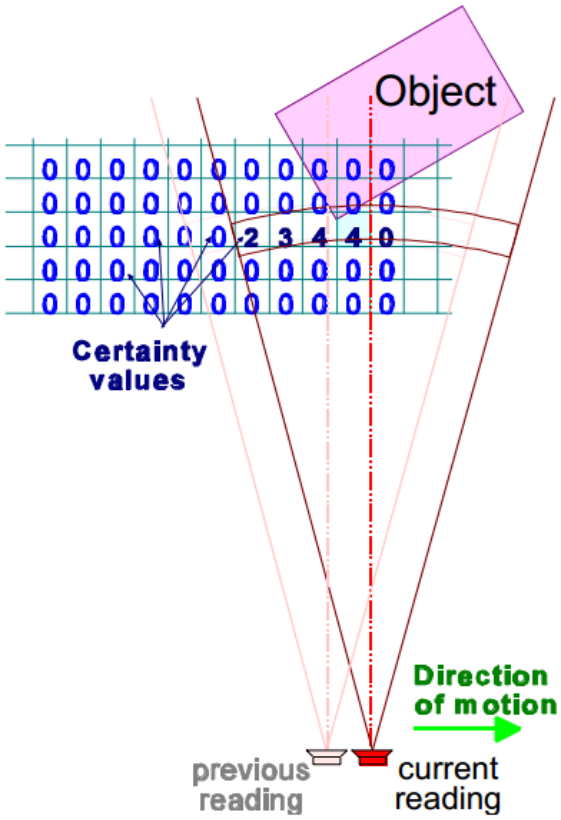


شکل ۳-۸-هر بار خواندن سونار تنها در مقدار یک سلول از هیستوگرام تغییر ایجاد می‌کند [۳۵].

بعد از ساخت هیستوگرام C، روش VFF برآیند نیروهای دافعه‌ای که از موانع حوالی ربات به آن وارد می‌شود را، محاسبه می‌کند. برای این کار، ناحیه‌ای مستطیل شکل با ابعاد مشخص بر روی هیستوگرام C در نظر گرفته می‌شود به طوری که ربات در مرکز این ناحیه واقع شده باشد. به این ناحیه، ناحیه‌ی فعال[2] می‌گویند و با C^* نشان داده می‌شود. حال با استفاده از الگوریتم میدان پتانسیل برآیند نیروهای دافعه‌ی حاصل از سلول‌های ناحیه‌ی فعال محاسبه می‌شود و در برداری به نام F_r ذخیره می‌گردد. لازم به ذکر است که میزان نیروی دافعه‌ی هر سلول با مقدار قطعیت آن رابطه‌ی مستقیم و با فاصله‌ی آن تا ربات نسبت عکس دارد. به عبارت دیگر هر چه قدر میزان قطعیت یک سلول در مورد وجود مانع بیشتر باشد، میزان نیروی دافعه‌ای که به ربات اعمال می‌کند بیشتر خواهد بود و هر چه قدر فاصله‌ی سلول مذکور از ربات بیشتر باشد، میزان نیروی دافعه‌ی مذکور کمتر خواهد بود. بعد از محاسبه‌ی F_r، بردار مجازی F_t (که

[1] Certainty grid
[2] Active window or active region

همواره اندازه‌ی ثابت دارد) بر ربات اعمال می‌شود. بردار F_t جذب‌کننده‌ی ربات به سمت هدف می‌باشد. با گرفتن برآیند دو بردار F_r و F_t، برداری حاصل می‌شود که ربات را بدون برخورد با موانع، به سمت هدف هدایت می‌کند.



شکل ۳-۹-نمونه‌برداری مکرر و سریع با استفاده از سونارها در حین حرکت ربات، موجب شکل‌گیری توزیع احتمال نواحی اشغال شده در محیط، به صورت افزایشی می‌شود [۳۵].

۳-۵ الگوریتم اجتناب از مانع هیستوگرام میدان برداری (VFH)

هیستوگرام میدان برداری[1] (VFH) [۳۵] یکی دیگر از روش‌های بلادرنگ اجتناب از مانع می‌باشد که در سال ۱۹۹۱م. ارائه شده است. این روش در واقع برای اصلاح نواقص و معایب روش VFF طراحی شده است. بارزترین عیبی که به روش VFF وارد است، عدم توانائی عبور از بعضی از موانع می‌باشد. به عنوان مثال اگر ربات امکان عبور از یک راهرو را داشته باشد، به علت نیروی دافعه‌ای که از دیوارهای دو طرف به آن وارد می‌شود، قادر به عبور از آن نخواهد بود. علت شکست روش VFF را، باید در مرحله‌ی محاسبه‌ی بردار نیروی دافعه (F_r) جست و جو کرد. همان طور که قبلاً ذکر شد، برای محاسبه‌ی بردار دافعه‌ی F_r باید برآیند نیروهای دافعه حاصل از سلول‌های ناحیه‌ی C* گرفته شود. با این کار اطلاعات مربوط به صدها نقطه از هیستوگرام C به دو مؤلفه‌ی اندازه و جهت بردار F_r کاهش می‌یابد که این کاهش شدید و ناگهانی طی یک مرحله، علت شکست روش VFF می‌باشد [۳۵]. راهکاری که در

[1] Vector Field Histogram (VFH)

روش VFH برای رفع نواقص روش VFF ارائه شده است، کاهش داده‌های هیستوگرام C طی دو مرحله می‌باشد. به این صورت که ابتدا همانند روش VFF هیستوگرام C بر اساس موانع موجود در محیط ساخته می‌شود. در مرحله‌ی بعد یک هیستوگرام قطبی (به نام H) حول مکان کنونی ربات با توجه به ناحیه‌ی فعال (در شکل ۳-۱۰ ناحیه‌ی فعال به صورت یک مربع با طول و عرضی برابر Ws و با نام Active cells مشخص شده است) تشکیل داده می‌شود. در واقع تشکیل هیستوگرام H، نگاشتی از فضای دو بعدی ناحیه‌ی فعال هیستوگرام C به فضایی تک بعدی (هیستوگرام H) می‌باشد. تک بعدی شدن نمایش موانع موجود در محیط، اوّلین مرحله‌ی کاهش داده‌ها می‌باشد. در شکل ۳-۱۰ نحوه‌ی نگاشت از ناحیه‌ی فعال به هیستوگرام قطبی نشان داده شده است. همان طور که در این شکل مشاهده می‌شود، هیستوگرام قطبی از تعدادی قطاع[1] (n عدد) تشکیل شده است که هر قطاع مانند k، α درجه می‌باشد و نشان‌دهنده‌ی میزان چگالی موانع (h_k) در محدوده‌ای از ناحیه‌ی فعال می‌باشد که توسط قطاع k پوشش داده شده است. برای محاسبه‌ی میزان چگالی موانع در هر قطاع مانند k، برای هر سلول از ناحیه‌ی فعال، یک بردار مانع[2] در نظر گرفته می‌شود. هر بردار مانع دارای دو مؤلفه‌ی جهت و اندازه می‌باشد که در روابط زیر آورده شده‌اند [۳۵]:

$$\beta_{i,j} = tan^{-1}\left(\frac{y_i - y_0}{x_i - x_0}\right) \qquad (۶-۳)$$

$$m_{i,j} = (c^*_{i,j})^2 (a - bd_{i,j}) \qquad (۷-۳)$$

در رابطه‌ی ۳-۶ $\beta_{i,j}$ نشان‌دهنده‌ی جهت بردار مانع می‌باشد و دارای بازه‌ی $[0, 2\pi]$ است. هم‌چنین (x_i,y_i) مختصات سلول (i,j) و (x_0,y_0) مختصات کنونی مرکز ربات می‌باشد. در رابطه‌ی ۳-۷ $m_{i,j}$ نشان‌دهنده‌ی اندازه‌ی بردار مانع می‌باشد، $c^*_{i,j}$ میزان قطعیت وجود مانع در سلول (i,j) می‌باشد، a و b ضرایبی ثابت و مثبت هستند و $d_{i,j}$ فاصله‌ی سلول (i,j) تا مرکز ربات می‌باشد.

با توجه به مؤلفه‌ی جهتِ $(\beta_{i,j})$ بردار مانع سلول (i,j) می‌توان قطاعی که سلول مذکور در آن واقع شده است را مشخص کرد. برای این کار می‌توان از رابطه‌ی زیر استفاده کرد [۳۵]:

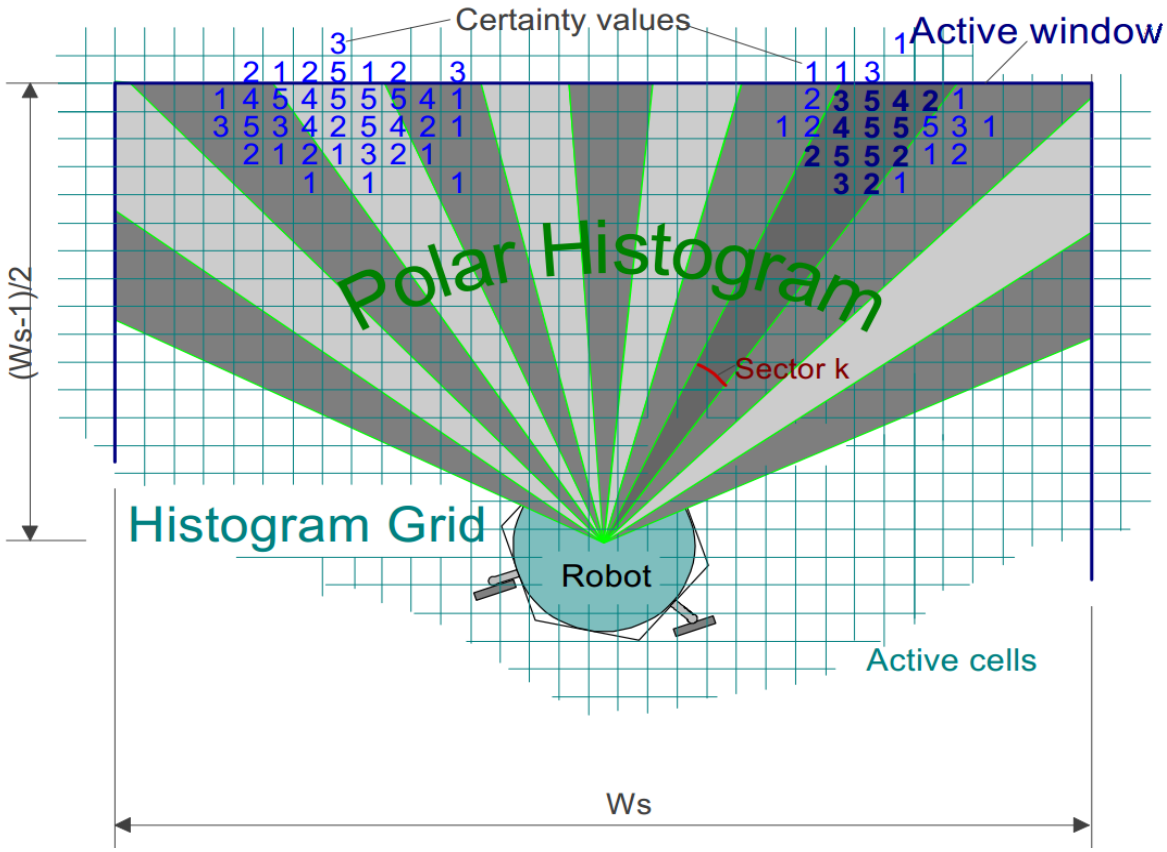
$$k = INT\left(\frac{\beta_{i,j}}{\alpha}\right) \qquad (۸-۳)$$

در رابطه‌ی ۳-۸ با استفاده از "INT" حاصل کسر به صورت عدد صحیح در می‌آید و در k ذخیره می‌گردد. بعد از مشخص شدن قطاع مذکور، مؤلفه‌ی اندازه‌ی $(m_{i,j})$ بردار مانع سلول (i,j) به میزان چگالی قطاع مذکور افزوده می‌شود. حال با مشخص شدن تمام سلول‌هایی از ناحیه‌ی فعال، که در قطاع k قرار می‌گیرند، می‌توان چگالی قطاع k را به صورت زیر محاسبه کرد [۳۵]:

---

[1] Sector
[2] Obstacle vector

$$h_k = \sum_{i,j} m_{i,j} \qquad (۹-۳)$$



شکل ۳-۱۰-نحوه‌ی نگاشت ناحیه‌ی فعال هیستوگرام C به هیستوگرام قطبی [۳۵]. هر یک از نواحی مخروطی شکل یک قطاع می‌باشد.

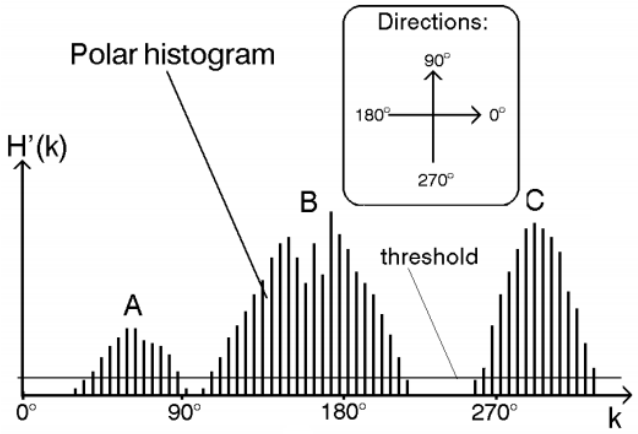
در رابطه‌ی ۳-۹ h_k میزان چگالی موانع برای قطاع k می‌باشد. با محاسبه‌ی میزان چگالی تمام قطاع‌ها، هیستوگرام قطبی آماده می‌گردد. همان طور که می‌دانیم هر هیستوگرام دارای ذاتی گسسته است، که این در مورد هیستوگرام C نیز صادق می‌باشد. این گسستگی بر روی نگاشت از هیستوگرام دکارتی C به هیستوگرام قطبی H تأثیر می‌گذارد و موجب می‌شود که هیستوگرام H حالت پاره‌پاره و غیر یکنواخت پیدا کند که این امر تصمیم‌گیری درباره‌ی سرعت و جهت حرکت را دچار مشکل می‌کند به همین علت هیستوگرام قطبی باید هموارسازی[1] می‌شود. برای هموارسازی هیستوگرام H از رابطه‌ی زیر استفاده می‌شود [۳۵]:

$$h'_k = \frac{h_{k-l} + 2h_{k-l+1} + \ldots + lh_k + \ldots + 2h_{k+l-1} + h_{k+l}}{2l+1} \qquad (۱۰-۳)$$

در رابطه‌ی ۳-۱۰ h'_k مقدار چگالی هموارسازی شده‌ی قطاع k می‌باشد. l نیز پارامتری است که میزان هموارسازی را مشخص می‌کند که در روش VFH مقدار آن برابر ۵ در نظر گرفته شده است. در شکل ۳-۱۱ نمونه‌ای از هیستوگرام قطبی هموارسازی شده نشان داده شده است. که در آن محور افقی (k) نشان‌دهنده بازه‌ی [۰, ۳۶۰] درجه‌ای دور

[1] Smoothen

ربات می‌باشد و محور عمودی ((H'(k)) نشان‌دهنده‌ی میزان چگالی موانع پیرامون ربات در زوایای مختلف می‌باشد. بعد از تشکیل شدن هیستوگرام قطبی، قسمت‌هایی که برای عبور ربات ایمن می‌باشند مشخص می‌گردند. برای این کار بر روی هیستوگرام هموارسازی شده آستانه‌گذاری می‌شود (نشان داده شده در شکل ۳-۱۱) به طوری که مقادیری از k که H'(k) متناظر با آن‌ها از آستانه‌ی مذکور کمتر باشد، به عنوان قطاع‌های امن در نظر گرفته می‌شوند. بعد از مشخص شدن قطاع‌های امن، با استفاده از آن‌ها سرعت و جهت حرکت مناسب برای ربات تعیین می‌گردد. که این دومین مرحله‌ی کاهش داده‌ها می‌باشد.



شکل ۳-۱۱-نمونه‌ای از هیستوگرام قطبی تشکیل شده توسط روش VFH [۳۵]

۳-۶ الگوریتم اجتناب از مانع +VFH

الگوریتم +VFH توسعه داده شده‌ی VFH می‌باشد که حرکتی نرم‌تر و مطمئن‌تر را برای ربات به ارمغان می‌آورد [۵۵]. الگوریتم +VFH بخشی از فرآیند تنظیم پارامتر الگوریتم VFH را حذف می‌کند و به علاوه با تخمین دقیق‌تری از نحوه‌ی حرکت ربات، قابلیت اطمینان را بالا می‌برد. منظور از بالا بردن قابلیت اطمینان، کاهش احتمال برخورد ربات با موانع و افزایش ایمنی ربات می‌باشد. ساختار کلی +VFH شبیه به VFH می‌باشد. ورودی به الگوریتم +VFH نقشه‌ای سلولی به نام شبکه‌ی هیستوگرام[1] (همان هیستوگرام دو بعدی C در روش VFH) می‌باشد. الگوریتم +VFH داده‌های هیستوگرام C را طی چهار مرحله کاهش بعد می‌دهد و به این ترتیب جهت مناسب حرکت ربات را محاسبه می‌کند. در سه مرحله‌ی اوّل، داده‌های دو بعدی دکارتی هیستوگرام C ابتدا به هیستوگرامی تک بعدی و قطبی، سپس به هیستوگرام قطبی دو سطحی و در نهایت به هیستوگرام قطبی نقاب[2] شده تبدیل می‌شود. در مرحله‌ی چهارم الگوریتم +VFH جهت حرکت مناسب را برای ربات با توجه به هیستوگرام قطبی نقاب شده و یک تابع هزینه، محاسبه می‌کند.

[1] Histogram grid
[2] mask

در اوّلین مرحله‌ی کاهش داده، هیستوگرام دو بعدی C به هیستوگرام قطبی H^p تبدیل می‌شود. تفاوت روش+VFH با روش VFH در این مرحله این است که VFH به طور صریح ابعاد ربات را در نظر نمی‌گیرد بلکه، با اعمال یک صافی پایین‌گذر بر روی هیستوگرام قطبی و هموارسازی آن، سعی دارد ابعاد ربات را به صورت ضمنی در هیستوگرام قطبی دخیل کند (در نظر بگیرد). تنظیم کردن پارامترهای صافی پایین‌گذر مذکور به صورت تجربی، چالش اصلی در پیاده‌سازی الگوریتم VFH می‌باشد. اما الگوریتم +VFH، صافی پایین‌گذر مورد نیاز برای در نظر گرفتن ابعاد ربات را به صورت نظری محاسبه می‌کند. در این راستا هر سلول اشغالی از ناحیه‌ی فعال (در روش +VFH ناحیه‌ی فعال با C_a نمایش داده می‌شود) به شعاع $r_{r+s}=r_r+d_s$ متورم[1] می‌شود. r_r شعاع ربات (ربات مانند یک دیسک به شعاع r_r در نظر گرفته شده است) است که از مرکز ربات تا دورترین نقطه روی محیط آن در نظر گرفته می‌شود. d_s میزان فاصله‌ی امن بین ربات و مانع می‌باشد. با متورم‌سازی سلول‌های اشغالی با شعاع r_{r+s}، می‌توان ربات را به صورت نقطه‌ای در نظر گرفت. لازم به ذکر است که نحوه‌ی متورم‌سازی سلول‌های اشغالی به صورت متقارن (با شعاع r_{r+s}) تنها زمانی مناسب است که بتوان شکل ربات را به صورت یک دیسک تخمین زد. اما اگر ربات دارای شکل نامتقارنی باشد، آن‌گاه سلول‌ها نیز باید به صورت نامتقارن و با توجه به مکان کنونی ربات و جهت[2] آن متورم-سازی گردند.

متورم‌سازی سلول‌های اشغالی در حین ساخت هیستوگرام قطبی انجام می‌گیرد. برخلاف روش VFH که برای هر سلول فقط یک قطاع به روز رسانی می‌شود؛ در روش +VFH تمام قطاع‌هایی که سلول متورم شده در آن‌ها قرار می‌گیرد، به روز رسانی می‌شوند. برای متورم‌سازی هر سلول (i,j) یک زاویه‌ی متورم‌سازی به نام $\gamma_{i,j}$ به صورت زیر تعریف می‌شود [۵۵]:

$$\gamma_{i,j} = sin^{-1}(\frac{r_{r+s}}{d_{i,j}}) \qquad (۱۱-۳)$$

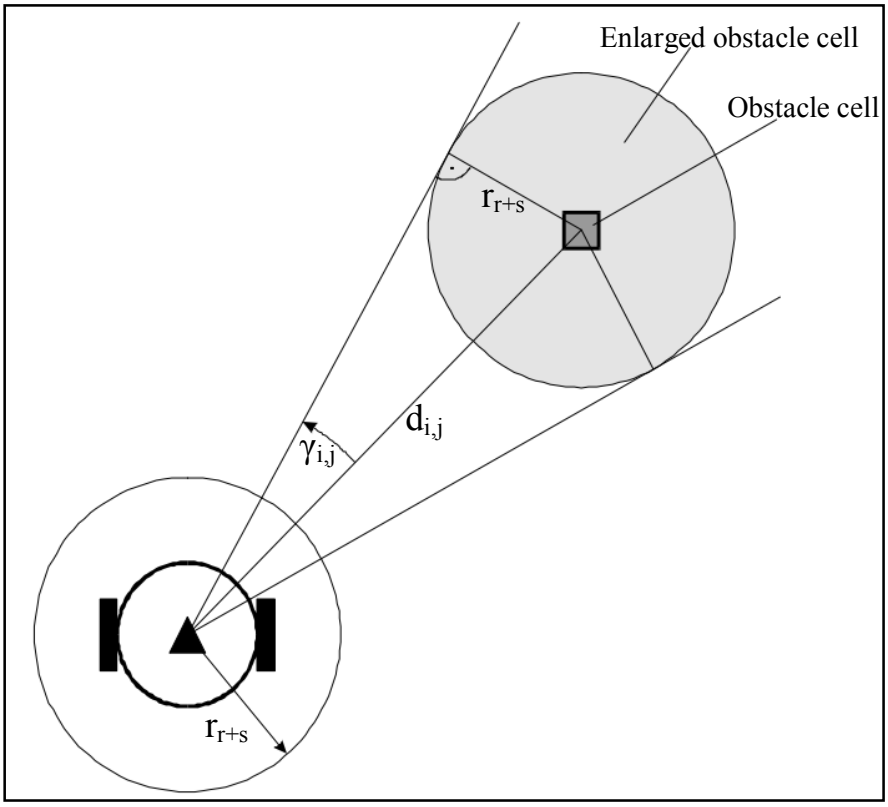
که در رابطه‌ی ۳-۱۱ $d_{i,j}$ فاصله‌ی مرکز ربات از سلول (i,j) می‌باشد. در شکل ۳-۱۲ شمای متورم‌سازی سلول (i,j) نشان شده است. حال برای هر قطاع مانند k از هیستوگرام H^p، چگالی موانع به صورت زیر محاسبه می‌شود [۵۵]:

$$H_k^p = \sum_{i,j \in C_a} m_{i,j}.h'_{i,j}$$
$$with:\ h'_{i,j} = 1 \ \ if\ k.\alpha \in [\beta_{i,j} - \gamma_{i,j}, \beta_{i,j} + \gamma_{i,j}] \qquad (۱۲-۳)$$
$$h'_{i,j} = 0 \ \ otherwise$$

هیستوگرام H^p که با استفاده از روش بالا ساخته می‌شود، ابعاد ربات را در نظر می‌گیرد. لازم به ذکر است که تابع h'

[1] Enlarge
[2] Orientation or heading

مانند یک صافی پایین‌گذر عمل می‌کند که موجب هموارسازی هیستوگرام قطبی می‌شود. در ضمن در این روش دیگر نیازی به تنظیم پارامتر صافی پایین‌گذر نمی‌باشد.



شکل ۳-۱۲- نحوه‌ی متورم‌سازی سلول (i,j) [۵۵]

در مرحله‌ی دوم از کاهش داده‌ها، هیستوگرام قطبی با استفاده از آستانه‌گذاری، به هیستوگرامی دو سطحی[1] به نام H^b تبدیل می‌شود. هر قطاع از هیستوگرام قطبی دو سطحی دارای مقدار ۱ به معنای اشغال بودن و یا دارای مقدار صفر به معنای آزاد بودن می‌باشد. واضح است که حرکت ربات در صورتی مطلوب می‌باشد که نرم و بدون نوسان[2] باشد. در روش VFH برای آستانه‌گذاری بر روی هیستوگرام قطبی از یک حد آستانه‌ی ثابت به نام τ استفاده می‌شود که گاهی اوقات حرکت نرم و بدون نوسان ربات را دچار مشکل می‌کند. مثلاً اگر محیط پیرامون ربات دارای چندین ناحیه‌ی آزاد باریک[3] باشد، در حین نمونه‌برداری از محیط با استفاده از حسگرهای سونار، وضعیت نواحی مذکور در هیستوگرام قطبی بین آزاد و اشغال بودن سوئیچ می‌کند. این وضع موجب نوسان کردن سر ربات (جهت حرکت ربات) می‌شود زیرا که انتخاب ربات بین دو یا چند عدد از نواحی مذکور (با توجه به باز و بسته بودنشان در هر گام) مرتباً عوض می‌شود. نتیجه این که، ربات قبل از انتخاب کردن یکی از نواحی آزاد به صورت نهائی، تا حد خطرناکی

[1] Binary histogram
[2] Oscillation
[3] Narrow opening

به موانع موجود نزدیک می‌شود. برای حل این مشکل روش +VFH از آستانه‌گذاری هیستِرِسیس[1] استفاده کرده است. به این صورت که به جای در نظر گرفتن یک سطح آستانه، دو سطح آستانه‌ی τ_{low} و τ_{high} در نظر گرفته می‌شود. حال هیستوگرام قطبی دو سطحی در گام iام با توجه به آستانه‌های مذکور و هیستوگرام قطبی دو سطحی در گام i-1ام ساخته می‌شود که در زیر روابط آن آورده شده است [۵۵]:

$$H^b_{k,i} = \begin{cases} 1 & if\ H^p_{k,i} > \tau_{high} \\ 0 & if\ H^p_{k,i} < \tau_{low} \\ H^b_{k,i-1} & otherwise \end{cases} \qquad (۱۳-۳)$$

که در رابطه‌ی ۳-۱۳ $H^p_{k,i}$ چگالی قطاع k از هیستوگرام قطبی در گام iام و $H^b_{k,i}$ مقدار قطاع k (صفر برای قطاع آزاد و ۱ برای قطاع اشغال) از هیستوگرام قطبی دو سطحی در گام iام می‌باشد.

مرحله‌ی سوم از کاهش داده‌ها برای در نظر گرفتن خصوصیات سینماتیکی و دینامیکی ربات به طور صریح، طراحی شده است. در این مرحله هیستوگرام قطبی دو سطحی طی فرآیندی بر اساس خصوصیات دینامیکی و سینماتیکی ربات نقاب می‌گردد؛ به این معنا که قطاع‌های آزاد هیستوگرام H^b مورد بررسی قرار می‌گیرند و آن دسته از قطاع‌هایی که ربات قادر به استفاده از آن‌ها برای عبور از موانع نمی‌باشد، به عنوان قطاع‌های مسدود قلمداد می‌شوند. به هیستوگرام حاصل، هیستوگرام قطبی نقاب[2] شده می‌گویند. به عبارت دیگر، ربات‌ها با توجه به ساختارشان دارای محدودیت حرکتی هستند و در نتیجه بسته به سرعتی که در هر لحظه دارند، قادر به دنبال کردن هر مسیر دلخواه نمی‌باشند. فاز سوم کاهش داده در روش +VFH برای افزایش ایمنی ربات حین حرکت، ناحیه‌های آزادی که ربات قادر به فرمان دادن به سمت آن‌ها و عبور از آن‌ها نمی‌باشد را حذف می‌کند (H^b_k آن‌ها را برابر ۱ در نظر می‌گیرد). روشی که +VFH برای نقاب کردن هیستوگرام قطبی دو سطحی استفاده می‌کند به این صورت است که، فرض می‌شود ربات همواره بر روی کمان یک دایره به شعاع r حرکت می‌کند. واضح است که اگر r برابر بی‌نهایت باشد، ربات بر روی یک خط مستقیم حرکت خواهد کرد. حداکثر انحنا[3]ی کمانی که ربات روی آن حرکت می‌کند معمولاً تابعی از سرعت ربات و نوع ربات است. مثلاً یک ربات دیفرانسیلی در صورتی که سرعت انتقالی[4]اش صفر باشد، شعاع پیچشی برابر صفر خواهد داشت. اما شعاع پیچش یک ربات آکرمن به علت ساختارش هیچ‌گاه برابر صفر نخواهد بود. لازم به ذکر است که در روش +VFH نوع ربات از نوع دیفرانسیلی در نظر گرفته شده است.

روش +VFH برای نقاب کردن هیستوگرام قطبی دو سطحی، در هر گام حداقل شعاع پیچش به سمت چپ (r_l) و راست (r_r) را محاسبه می‌کند. برای این کار از روابط زیر استفاده می‌کند [۵۵]:

---

[1] Hysteresis threshold
[2] Masked polar histogram
[3] Curvature
[4] Travel speed

$$r_r = \frac{1}{K_r}, \quad r_l = \frac{1}{K_l} \tag{۱۴-۳}$$

در رابطه‌ی بالا K_l و K_r به ترتیب حداکثر میزان انحنای ممکن برای چرخش ربات به سمت چپ و راست می‌باشند. از آنجا که K_l و K_r حداکثر میزان انحنا می‌باشند، r_l و r_r نیز حداقل شعاع پیچش به سمت چپ و راست می‌باشند. در مقاله‌ی روش VFH+ در مورد نحوه‌ی محاسبه‌ی K_l و K_r مطلبی ذکر نشده است اما آن چه مسلم است میزان این دو متغیر به سرعت ربات در هرلحظه بستگی دارد. به این صورت که با سرعت بیشتر، میزان انحنائی که ربات می‌تواند در چرخش به چپ و راست داشته باشد، کاهش می‌یابد.

با فرض داشتن r_l، r_r و هیستوگرام C (یا همان نقشه‌ی سلولی محیط)، می‌توان قطاع‌هایی از هیستوگرام قطبی دو سطحی که آزاد هستند اما ربات (به علت میزان سرعت کنونی‌اش) توانایی استفاده از آن‌ها را ندارد، تشخیص داده، مسدود اعلام کرد. در شکل ۳-۱۳ نمونه‌ای از نحوه‌ی کار فاز سوم روش VFH+ مشاهده می‌شود که در آن ربات به صورت دایره‌ای با دو چرخ متصل به آن نمایش داده شده و جهت سر آن نیز با یک مثلث مشخص گردیده است. برای تشخیص جهاتی که باید مسدود گردند، ابتدا سلول‌های اشغال شده (که در اینجا A و B نام دارند) به شعاع r_{r+s} متورم شده‌اند و در ادامه دایره‌هایی که با سلول‌های متورم شده برخورد دارند، تشخیص داده شده‌اند. که در شکل مذکور دایره‌ی چرخش به سمت چپ (به شعاع r_l) با سلول A تقاطع دارد. به همین علت تمام جهت‌های حرکتی آزادی که در سمت چپ مانع A قرار دارند، مسدود در نظر گرفته شده‌اند. اما دایره‌ی چرخش به سمت راست (به شعاع r_r) با مانع B تقاطع ندارد به همین علت جهات حرکتی آزاد که در سمت راست مانع B قرار دارند بدون تغییر خواهند ماند. برای محاسبه‌ی مختصات مرکز دایره‌های چرخش به چپ و راست از فرمول‌های زیر استفاده می‌شود [۵۵] که در آن‌ها θ همان جهت فعلی سر ربات می‌باشد:

$$\Delta x_l = -r_l.sin(\theta), \quad \Delta y_l = -r_l.cos(\theta) \tag{۱۵-۳}$$

$$\Delta x_r = r_r.sin(\theta), \quad \Delta y_r = r_r.cos(\theta) \tag{۱۶-۳}$$

در ضمن فاصله‌ی بین هر سلول مانند C_{ij} متعلق به ناحیه‌ی فعال (C_a) و مرکز دایره‌های چرخش چپ و راست با استفاده از روابط زیر به دست می‌آید [۵۵]:

$$d_l^2 = (\Delta x_l - \Delta x(j))^2 + (\Delta y_l - \Delta y(i))^2 \tag{۱۷-۳}$$

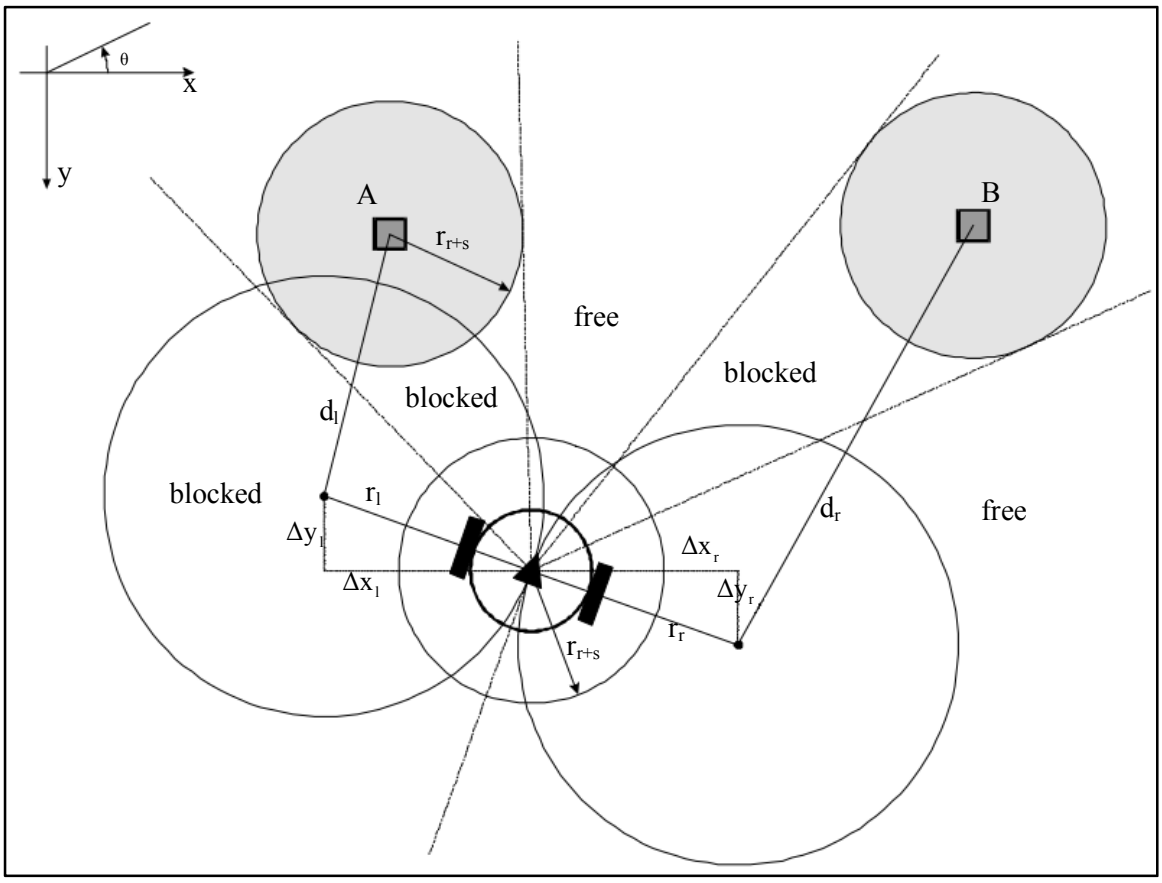
$$d_r^2 = (\Delta x_r - \Delta x(j))^2 + (\Delta y_r - \Delta y(i))^2 \tag{۱۸-۳}$$

که در روابط بالا d_r فاصله‌ی بین سلول C_{ij} و مرکز دایره‌ی چرخش به سمت راست و d_l فاصله‌ی بین سلول C_{ij} و مرکز دایره‌ی چرخش به سمت چپ می‌باشد. هم‌چنین Δx(j) میزان اختلاف بین مؤلفه‌ی x مکان ربات و مؤلفه‌ی x سلول C_{ij} و Δy(i) میزان اختلاف بین مؤلفه‌ی y مکان ربات و مؤلفه‌ی y سلول C_{ij} می‌باشد. بعد از محاسبه‌ی d_l و d_r می‌توان از روابط زیر برای تشخیص این که یک مانع مشخص با دایره‌های چرخش چپ و راست تقاطع دارد یا خیر،

استفاده کرد [۵۵]:

$$d_l^2 < (r_l + r_{r+s}) \quad [condition1] \tag{۱۹-۳}$$

$$d_r^2 < (r_r + r_{r+s}) \quad [condition2] \tag{۲۰-۳}$$



شکل ۳-۱۳-نحوه‌ی تشخیص دادن جهت‌های آزادی که به علت محدودیت‌های حرکتی ربات و سرعت فعلی آن باید مسدود اعلام شوند. مانع A با دایره به شعاع r_l برخورد دارد به همین علت تمام جهات حرکتی آزاد در سمت چپ A باید مسدود اعلام گردند [۵۵].

به این صورت که اگر شرط اوّل برقرار باشد، مانع مذکور با دایره‌ی چرخش به چپ تقاطع دارد و در نتیجه تمام جهات حرکتی در سمت چپ مانع، مسدود اعلام می‌شود. به صورت مشابه اگر شرط دوم برقرار باشد، مانع با دایره‌ی چرخش به راست تقاطع دارد و در نتیجه تمام جهات حرکتی آزاد در سمت راست مانع مسدود در نظر گرفته می‌شود. بعد از چک کردن تقاطع تمام سلول‌های اشغالی ناحیه‌ی فعال با دو دایره به شعاع‌های r_r و r_l، بازه‌ی زوایای مجاز که ربات قادر به استفاده از آن‌ها می‌باشد، مشخص می‌گردد. که این بازه به صورت $[\varphi_r, \theta]$ و $[\theta, \varphi_l]$ می‌باشد. $[\varphi_r, \theta]$ تمام زوایای مُجازی است که ربات می‌تواند برای چرخش به راست از بین آن‌ها انتخاب کند و $[\theta, \varphi_l]$ تمام زوایای مُجازی که ربات می‌تواند برای چرخش به چپ از بین آن‌ها انتخاب کند. حال می‌توان هیستوگرام قطبی نقاب شده را به صورت زیر تعریف کرد [۵۵]:

$$H_k^m = \begin{cases} 0 & if\ H_k^b = 0\ and\ (k.\alpha) \in \{[\varphi_r, \theta], [\theta, \varphi_l]\} \\ 1 & otherwise \end{cases} \qquad (۲۱-۳)$$

که در رابطه‌ی بالا مقدار قطاع kام از هیستوگرام قطبی نقاب شده برابر صفر خواهد بود (آزاد قلمداد می‌شود) اگر، قطاع kام از هیستوگرام قطبی دو سطحی آزاد باشد (مقدارش صفر باشد) و قطاع مذکور در بازه‌ی مجاز $[\varphi_r, \theta]$ و $[\theta, \varphi_l]$ قرار داشته باشد و در غیر این صورت قطاع kام مسدود قلمداد می‌شود (مقدار قطاع مذکور در هیستوگرام قطبی نقاب شده برابر ۱ می‌گردد). در شکل ۳-۱۴ هیستوگرام‌های قطبی، قطبی باینری و قطبی نقاب شده با توجه به سناریویی که در شکل ۳-۱۳ آورده شده ، ترسیم گردیده است. قسمت (b) هیستوگرام قطبی دو سطحی را نمایش می‌دهد که با توجه به هیستوگرام قطبی قسمت (a) ساخته شده است. در قسمت (c) از شکل ۳-۱۴، هیستوگرام قطبی نقاب شده نمایش داده شده است. همان طور که مشاهده می‌شود قطاع‌هایی که در سمت چپ مانع A و در درون دایره به شعاع r_l (شکل ۳-۱۳) قرار دارند مسدود گشته‌اند زیرا که مانع A با دایره‌ی چرخش به چپ برخورد دارد.

در مرحله‌ی چهارم کاهش داده‌ها، روش VFH+ با استفاده از هیستوگرام قطبی نقاب شده و با تعریف کردن تابع هزینه‌ی g(c)، بهترین جهت حرکت را انتخاب می‌کند. لازم به ذکر است که در تابع هزینه‌ی g(c) میزان اختلاف بین جهت هدف و جهت‌های کاندید، تنها معیار برای محاسبه‌ی هزینه نمی‌باشد.

۳-۷ خلاصه

در این فصل مفاهیم مورد نیاز برای مطالعه‌ی ادامه‌ی این پایان‌نامه بررسی گردید. از جمله مطالبی که شرح داده شد می‌توان به الگوریتم تنژنت‌باگ، VFF، VFH، VFH+ و میدان پتانسیل اشاره کرد. در فصل بعدی روش پیشنهادی ارائه شده در این پایان‌نامه مورد بررسی قرار می‌گیرد. در روش مذکور از روش اجتناب از مانع میدان پتانسیل استفاده می‌شود و با الهام از روش VFH+ مسیری که تنژنت‌باگ برای حرکت ربات انتخاب می‌کند، با توجه به روابط سینماتیکی ربات آکرمن، امکان‌سنجی می‌گردد و مسیری ایمن برای ربات برگزیده می‌شود.

شکل ۳- ۱۴-(a) هیستوگرام قطبی، (b) هیستوگرام قطبی دوسطحی و (c) هیستوگرام قطبی نقاب شده [۵۵]

فصل چهارم

الگوریتم تنژنت‌باگ تغییر یافته (روش پیشنهادی)

٤-١ مقدمه

همان طور که قبلاً نیز اشاره شد، هدف از انجام این پایان‌نامه پیاده‌سازی یک سیستم ناوبری (بر پایه‌ی الگوریتم تنژنت‌باگ) برای محیط ناشناخته‌ی خارج از جاده و قابل استفاده روی ربات آکرمن می‌باشد. به علاوه ذکر گردید که الگوریتم تنژنت‌باگ ربات را به صورت نقطه‌ای فرض می‌کند. این در حالی است که در دنیای واقعی یک ربات آکرمن دارای ابعاد مشخص است و برای حفظ ایمنی خود باید همواره فاصله‌اش (به اندازه‌ی کافی) را با موانع موجود در محیط حفظ کند. علاوه بر مورد مذکور، تنژنت‌باگ محدودیت‌های حرکتی ربات را نیز در نظر نمی‌گیرد. در واقع تنژنت‌باگ بدون در نظر گرفتن میزان اختلاف بین بردار جهت ربات و بردار هدف (برداری که از ربات به نقطه‌ی هدف کشیده شده است)، فرض می‌کند که ربات در هر لحظه قادر است با دنبال کردن راستای بردار هدف مستقیماً به طرف نقطه‌ی هدف حرکت کند. اما می‌دانیم که هر ربات آکرمن دارای محدودیت غیر-هولونومیکی می‌باشد که به آن اجازه‌ی حرکت در هر جهت دلخواه را نمی‌دهد.

محدودیت غیر-هولونومیک ربات آکرمن به ساختار فیزیکی آن مربوط می‌شود. شعاع چرخش به راست یا به چپ ربات آکرمن (در سرعت‌های پایین) تابعی از زاویه‌ی چرخ‌های فرمان‌پذیر و فاصله‌ی بین دو محور چرخ‌های جلو و عقب می‌باشد که به این ترتیب حداقل شعاع چرخش ربات غیر صفر خواهد بود و ربات همواره بر روی کمان

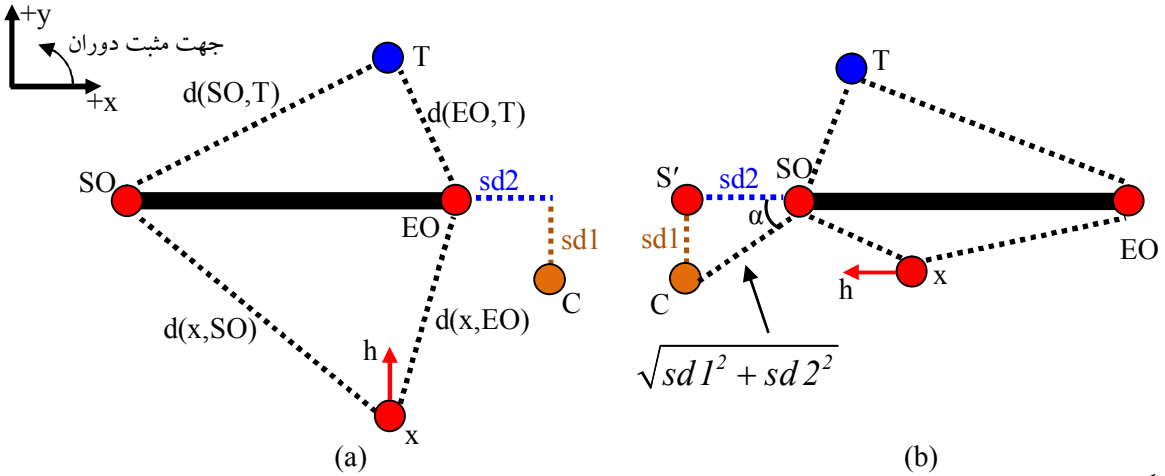
یک دایره به شعاع چرخش حرکت می‌کند. لازم به ذکر است که مرکز دایره‌ی مذکور، همان مرکز دوران لحظه‌ای[1] ربات می باشد.

با توجه به مطالب گفته شده واضح است که در هر لحظه ربات آکرمن توانایی دنبال کردن هر مسیر دلخواه را ندارد و برای استفاده از الگوریتم تنژنت‌باگ بر روی آن، باید تغییراتی در الگوریتم مذکور اعمال شود. در این فصل سیستم ناوبری ارائه شده برای ربات آکرمن شرح داده می‌شود و تغییرات اعمال شده روی تنژنت‌باگ مورد بررسی قرار می‌گیرد.

٤-۲ محاسبه‌ی نقطه‌ی ایمن با توجه به نقطه‌ی انتخابی تنژنت‌باگ

یادآوری می‌شود که طرح‌ریزی مسیر برای رسیدن به هدف در محیط ناشناخته باید به صورت افزایشی و بر اساس درکی که ربات با استفاده از حسگرهایش از محیط دارد، انجام گیرد. این دقیقاً روندی است که الگوریتم تنژنت‌باگ برای هدایت ربات به سوی هدف دنبال می‌کند. به این صورت که در هر گام محیط درک می‌شود، گراف مماس محلی تشکیل و نقطه‌ی مناسب برای حرکت ربات به سوی آن از بین رئوس گراف مماس محلی انتخاب می‌گردد. باید توجه کرد که رئوس گراف مذکور نقاط انتهایی موانع موجود در محیط می‌باشند. به این ترتیب اگر ربات به سمت آن‌ها حرکت کند، با موانع برخورد خواهد کرد. برای حل این مشکل، به جای حرکت به سمت نقطه‌ی منتخب تنژنت‌باگ، باید نقطه‌ای ایمن با توجه به نقطه‌ی منتخب محاسبه شود و ربات به سمت آن حرکت کند. محاسبه‌ی نقطه‌ی ایمن در شکل ۴-۱ نشان داده شده است که در آن x مکان ربات، $\overrightarrow{xh}$ بردار جهت ربات و T نقطه‌ی هدف می‌باشد. همان طور که در شکل مذکور مشاهده می‌شود، بر اساس این که نقطه‌ی انتهایی انتخاب شده از نوع SO باشد یا EO، نحوه‌ی محاسبه‌ی نقطه‌ی ایمن C کمی متفاوت است. در شکل مذکور sd1 و sd2 دو پارامتر قابل تنظیم هستند که برای حفظ فاصله‌ی عمودی و افقی از نقطه‌ی انتهایی منتخب در نظر گرفته شده‌اند. علت در نظر گرفتن دو نوع فاصله‌ی عمودی و افقی، حالت‌های مختلفی است که ممکن است ربات حین حرکت، نسبت به راستای مانع مسدودکننده داشته باشد. به عنوان مثال اگر بردار جهت ربات ($\overrightarrow{xh}$) به راستای مانع عمود باشد، برای جلوگیری از برخورد ربات با مانع حین دور زدن آن، باید فاصله‌ی افقی (sd2) تا نقطه‌ی منتخب حفظ شود. این سناریو در قسمت a از شکل ۴-۱ نشان داده شده است که در آن بر اساس نقطه‌ی انتهایی EO نقطه‌ی C محاسبه شده است. حال اگر بردار جهت ربات با راستای مانع موازی باشد، برای جلوگیری از برخورد، باید فاصله‌ی عمودی (sd1) تا نقطه‌ی انتهایی منتخب حفظ شود. این سناریو در قسمت b از شکل ۴-۱ مشاهده می‌شود که در آن بر اساس نقطه‌ی انتهایی SO، نقطه‌ی C محاسبه شده است.

[1] Instantaneous Center of Rotation (ICR)



شکل ۴- ۱-نحوه‌ی محاسبه‌ی نقطه‌ی ایمن (C) برای نقاط انتهایی (SO و EO) مانع مسدودکننده: در قسمت a نقطه‌ی C بر اساس نقطه‌ی EO و در قسمت b نقطه‌ی C بر اساس نقطه‌ی SO محاسبه شده است.

بار دیگر به شکل ۴-۱ قسمت a توجه کنید. برای محاسبه‌ی نقطه‌ی ایمن C با توجه به نقطه‌ی انتهایی از نوع SO، ابتدا اندازه‌ی زاویه‌ی α را با استفاده از رابطه‌ی ۴-۱ به دست می‌آوریم. سپس بردار $\overrightarrow{EOSO}$ را با استفاده از رابطه‌ی ۴-۲ محاسبه و نرمال‌سازی می‌کنیم. بر اساس بردار $\overrightarrow{EOSO}$ و با بهره‌گیری از رابطه‌ی ۴-۳ می‌توان بردار $\overrightarrow{SOS'}$ را محاسبه کرد. حال بردار $\overrightarrow{SOS'}$ را حول نقطه‌ی SO در جهت پاد-ساعت‌گرد به اندازه‌ی α درجه دوران می‌دهیم (رابطه‌ی ۴-۴) و آن را $\overrightarrow{rs}$ می‌نامیم. حال سَرِ بردار $\overrightarrow{rs}$ مختصات نقطه‌ی C را نشان خواهد داد. با توجه به دستگاه مختصات مفروض (نشان داده شده در قسمت بالا سمت چپ شکل ۴-۱)، دوران در جهت پاد-ساعت‌گرد دارای علامت مثبت خواهد بود. لازم به ذکر است که می‌توان نقطه‌ی C را برای حالتی که نقطه‌ی انتهایی از نوع EO می‌باشد، به طریقی مشابه (با روندی که برای محاسبه‌ی C بر اساس SO شرح داده شد) محاسبه کرد.

$$\hat{\alpha} = |atan2(sd1, sd2)| \qquad (۱-۴)$$

$$\overrightarrow{EOSO} = SO - EO, \quad \overrightarrow{EOSO} = \frac{\overrightarrow{EOSO}}{|\overrightarrow{EOSO}|} \qquad (۲-۴)$$

$$\overrightarrow{SOS'} = \overrightarrow{EOSO} \times \sqrt{sd1^2 + sd2^2} + SO \qquad (۳-۴)$$

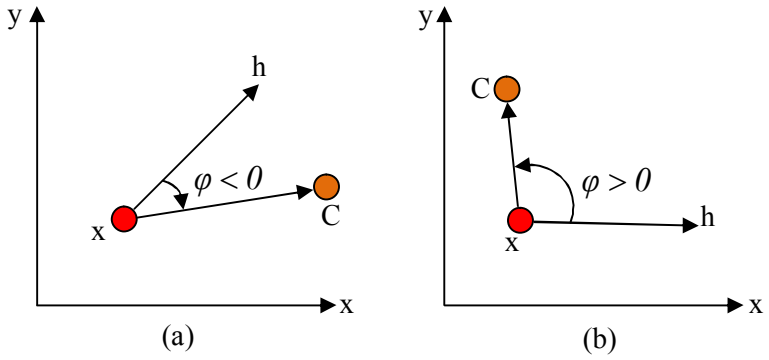
$$\begin{aligned} \overrightarrow{rs}.x &= SO.x + (S'.x - SO.x) \times cos(\alpha) - (S'.y - SO.y) \times sin(\alpha) \\ \overrightarrow{rs}.y &= SO.y + (S'.x - SO.x) \times sin(\alpha) + (S'.y - SO.y) \times cos(\alpha) \end{aligned} \qquad (۴-۴)$$

٤-۳ مکانیزم رفتن به نقطه

صرف نظر از رفتاری که تئزنت‌باگ در آن قرار دارد، بعد از محاسباتش نقطه‌ای برای حرکت ربات انتخاب می‌کند و همان طور که در بخش ۴-۲ شرح داده شد برای جلوگیری از برخورد ربات با موانع، نقطه‌ای ایمن به نام C

بر اساس نقطه‌ی منتخب محاسبه می‌شود. برای این که ربات بتواند به طرف C حرکت کند، باید زاویه فرمانی مناسب محاسبه گردد. به این صورت که ابتدا دو بردار $\overrightarrow{xC}$و $\overrightarrow{xh}$ تشکیل می‌گردد. بردار $\overrightarrow{xC}$ از رابطه‌ی ۴-۵ به دست می‌آید و بردار $\overrightarrow{xh}$ جهت ربات را نسبت به شمال مغناطیسی نشان می‌دهد که با استفاده از IMU به دست می‌آید. بعد از محاسبه‌ی دو بردار مذکور، زاویه‌ی کمینه‌ی علامت‌دار (φ) بین آن دو محاسبه می‌شود. زاویه‌یφ به گونه‌ای تعریف می‌شود که اگر با توجه به اندازه و علامت آن بردار $\overrightarrow{xh}$ را دوران دهیم تا بر راستای $\overrightarrow{xC}$ منطبق شود، کم‌ترین میزان چرخش ممکن را انجام داده باشیم.

$$\overrightarrow{xC} = C - x \qquad (۴-۵)$$



شکل ۴- ۲- نحوه‌ی محاسبه‌ی زاویه‌ی کمینه‌ی علامت‌دار(φ) بین دو بردار $\overrightarrow{xC}$و $\overrightarrow{xh}$: در قسمت a، منفی بودن زاویه‌یφنشان می‌دهد که برای هم‌راستا شدن $\overrightarrow{xh}$ با $\overrightarrow{xC}$ و رسیدن به نقطه‌ی C، ربات باید به سمت راست بِپیچَد و در قسمت b، علامت مثبتφنشان می‌دهد که ربات باید به چپ بپیچد.

همان طور که در شکل ۴-۲ مشاهده می‌شود (با توجه به جهت مثبت محورهای x و y)، اگر بردار $\overrightarrow{xC}$ در طرف راست بردار $\overrightarrow{xh}$ باشد علامت زاویه‌یφ منفی خواهد بود. این علامت منفی نشان‌دهنده‌ی این است که ربات باید به سمت راست بپیچد و اگر بردار $\overrightarrow{xC}$ در طرف چپ بردار $\overrightarrow{xh}$ باشد علامتφ مثبت خواهد بود و نشان‌دهنده این است که ربات باید به سمت چپ بپیچد. محاسبه‌ی زاویه‌یφ از رابطه‌ی ۴-۱۱ انجام می‌شود. که در آن "atan2()" مقدار تانژانت معکوس را در بازه‌ی [-π, π] محاسبه می‌کند. آرگومان اوّل atan2 مؤلفه‌ی سوم بردار حاصل از ضرب خارجی دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xC}$ و آرگومان دوم atan2 ضرب داخلی دو بردار مذکور را نشان می‌دهد. حال به شرح رابطه‌ی ۴-۱۱ می‌پردازیم. از آنجا که بردارهای $\overrightarrow{xh}$ و $\overrightarrow{xC}$ دو بعدی می‌باشند و محاسبه‌ی ضرب خارجی، به بردارهای سه بعدی نیاز دارد، بعد سوم بردارهای مذکور برابر صفر در نظر گرفته می‌شود (رابطه‌ی ۴-۶) که این امر موجب صفر شدن دو مؤلفه‌ی اوّل بردار $\vec{Z}$ (حاصل ضرب خارجی) می‌شود (رابطه‌ی ۴-۷). به این ترتیب قدر مطلق مؤلفه‌ی سوم بردار $\vec{Z}$ با اندازه‌ی بردار مذکور برابر می‌باشد (رابطه‌ی ۴-۸). توجه شود که در رابطه‌ی ۴-۸ θ زاویه‌ی بین دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xC}$ می‌باشد.

از طرفی می‌دانیم که حاصل ضرب داخلی دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xC}$ از رابطه‌ی ۴-۹ به دست می‌آید و در تابع atan2 آرگومان اوّل بر آرگومان دوم تقسیم می‌گردد. اگر رابطه‌ی ۴-۸ را به رابطه‌ی ۴-۹ تقسیم کنیم حاصل، رابطه‌ی ۴-۱۰ خواهد بود که در واقع تانژانت زاویه‌ی θ می‌باشد. درست است که با گرفتن تانژانت معکوس از رابطه ۴-۱۰ زاویه‌ی θ به دست می‌آید، اما باید توجه داشت که زاویه‌ی مذکور فاقد علامت می‌باشد و مشخص نیست که ربات برای هم سو شدن با راستای بردار $\overrightarrow{xC}$ باید به چه سمتی (چپ یا راست) بپیچد. برای حل این مشکل باید به این نکته توجه کرد که علامت مؤلفه‌ی z_Z از بردار $\vec{Z}$بسته به این که $\overrightarrow{xC}$ در کدام طرف $\overrightarrow{xh}$ قرار داشته باشد، مثبت یا منفی خواهد بود. با توجه به این مطلب برای به دست آوردن زاویه‌ی کمینه‌ی علامت‌دار بین دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xC}$، آرگومان اوّل تابع atan2 را برابر مؤلفه‌ی سوم بردار $\vec{Z}$ و آرگومان دوم آن را برابر رابطه‌ی ۴-۹ قرار می‌دهیم و به این ترتیب رابطه‌ی ۴-۱۱ به دست می‌آید.

$$\overrightarrow{xh}=(x_{xh},y_{xh},0),\quad \overrightarrow{xC}=(x_{xC},y_{xC},0) \tag{۶-۴}$$

$$\vec{Z}=\overrightarrow{xh}\times\overrightarrow{xC}=(x_Z,y_Z,z_Z)=(0,0,x_{xh}.y_{xC}-y_{xh}.x_{xC}) \tag{۷-۴}$$

$$|\vec{Z}|=\sqrt{0^2+0^2+(z_Z)^2}=|z_Z|=|\overrightarrow{xh}\times\overrightarrow{xC}|=|\overrightarrow{xh}||\overrightarrow{xC}|sin(\theta) \tag{۸-۴}$$

$$<\overrightarrow{xh},\overrightarrow{xC}>=|\overrightarrow{xh}||\overrightarrow{xC}|cos(\theta) \tag{۹-۴}$$

$$\frac{|z_Z|}{<\overrightarrow{xh},\overrightarrow{xC}>}=\frac{|\overrightarrow{xh}||\overrightarrow{xC}|sin(\theta)}{|\overrightarrow{xh}||\overrightarrow{xC}|cos(\theta)}=\frac{sin(\theta)}{cos(\theta)}=tan(\theta) \tag{۱۰-۴}$$

$$\varphi=atan2(z_Z,<\overrightarrow{xh},\overrightarrow{xC}>) \tag{۱۱-۴}$$

بعد از محاسبه‌ی زاویه‌یφ، می‌توان زاویه‌ی فرمان ربات را با توجه به آن به دست آورد. برای این کار از رابطه‌ی ۴-۱۲ استفاده می‌شود. که در آن k ضریبی ثابت و مثبت در بازه‌ی [۰, ۱] می‌باشد (در آزمایش‌های این پایان نامه مقدار ۰/۵ برای آن در نظر گرفته شده است) و δ زاویه‌ی فرمان را نشان می‌دهد. لازم به ذکر است که رابطه‌ی ۴-۱۲ در واقع نوعی کنترل‌کننده از نوع P است که در آن مقدار زاویه‌ی فرمان، ضریبی از مقدار اختلاف جهت (مقدار زاویه) بین دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xC}$ می‌باشد.

$$\delta=k\varphi \tag{۱۲-۴}$$

٤-٤ تغییرات اعمال شده حین ساخت گراف مماس محلی

در فصل سوم فرآیند تشکیل گراف مماس محلی با استفاده از داده‌های حسگر مسافت‌یاب (پویش‌گر لیزری) بیان شد. مشاهده گردید که برای ساخت گراف مذکور، ابتدا باید موانعی که از یکدیگر مجزا هستند تشخیص داده شوند تا بتوان با استفاده از نقاط انتهایی آن‌ها رئوس گراف مماس محلی را تشکیل داد. که معیار مجزا دانستن دو مانع مجاور، وجود ناپیوستگی در داده‌های پویش‌گر لیزری می‌باشد. این روند تشخیص موانع مجزا از یکدیگر زمانی

مناسب می‌باشد که ربات نقطه‌ای فرض شود زیرا که ربات نقطه‌ای دارای طول و عرض صفر می‌باشد و از هر روزنه-ای قابلیت عبور دارد. اما باید توجه شود که یک ربات واقعی دارای ابعادی غیر صفر می‌باشد به همین علت امکان عبور از روزنه‌هایی که برایش به اندازه‌ی کافی فراخ نباشد را ندارد. برای حل این مشکل ساخت گراف مماس محلی در دو فاز انجام می‌گیرد که در ادامه شرح داده می‌شود.

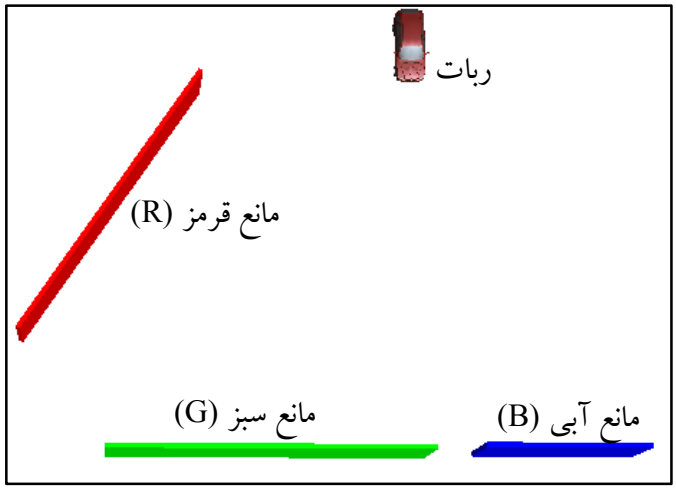
فرض کنید جهت چرخش پویش‌گر لیزری ساعت‌گرد باشد و اولین نقطه از مانع iام (O_i) که توسط پویش‌گر (در جهت ساعت‌گرد) حس شده است SO_i و آخرین نقطه‌ی حس شده از مانع مذکور EO_i نام داشته باشد. هم‌چنین فرض کنید که شکل ربات با دایره‌ی محیط به آن، به شعاع R_b تخمین زده شود. حال در فاز اوّل از ساخت گراف مماس محلی، همانند روندی که در فصل سوم بیان شد موانع از داده‌های پویش‌گر استخراج می‌شوند. در فاز دوم، هر دو مانع مجاور مانند (SO_i, EO_i) و (SO_{i+1}, EO_{i+1}) که فضای آزاد بین آن‌ها از یک حد آستانه کم‌تر باشد به صورت مانعی یکپارچه تلقی می‌شوند (دو مانع در یک مانع ادغام می‌شوند). بعد از مرحله‌ی ادغام موانع، با استفاده از نقاط انتهایی آن‌ها به عنوان رئوس گراف مماس محلی، گراف مذکور (همانند روش بیان شده در فصل سوم) تشکیل می-گردد. برای اندازه‌گیری فضای آزاد بین دو مانع مذکور از فاصله‌ی اقلیدسی بین دو نقطه‌ی EO_i و SO_{i+1} به صورتی که در رابطه‌ی ۴-۱۳ نشان داده شده است، استفاده می‌شود:

$$d(EO_i, SO_{i+1}) < R_b + s \qquad (۱۳-۴)$$

واضح است که فضای آزاد بین هر دو مانع مجاور زمانی برای عبور ربات ایمن می‌باشد که میزان فضای مذکور مقداری از شعاع (R_b) دایره‌ی محیطی ربات بیشتر باشد. به همین علت در رابطه‌ی ۴-۱۳ پارامتر s در نظر گرفته شده است. هر چقدر مقدار s بیشتر گردد، ربات در تفکیک کردن موانع از یکدیگر سخت‌گیرتر و محتاط‌تر می‌شود. به عبارت دیگر با افزایش s برای آن که دو مانع مجاور به صورت مجزا از یکدیگر در نظر گرفته شوند، باید مقدار فضای آزاد بیشتری بین آن‌ها موجود باشد.

برای آن که نحوه‌ی ادغام کردن موانع نزدیک به هم (موانعی که ربات قادر به عبور از بین آن‌ها نیست) بهتر مشخص گردد به مثالی که در شکل ۴-۳ آورده شده است توجه کنید. در شکل مذکور سه مانع با رنگ‌های قرمز، سبز و آبی اطراف ربات قرار دارند. حال فرض کنید که ربات با استفاده از پویش‌گر لیزری‌اش محیط را پویش کند و سپس موانع مجزا را از داده‌های پویش‌گر استخراج کرده به صورت دیوارهای نازک ترسیم می‌کند. طبق روشی که در فصل سوم بیان شد، سه مانع قرمز، آبی و سبز به صورت مجزا از یکدیگر تشخیص داده خواهند شد که در قسمت a از شکل ۴-۴ با سه خط مستقیم سیاه‌رنگ به نام‌های R، G و B مشخص شده‌اند. توجه شود که نقاط انتهایی موانع قرمز، سبز و آبی به ترتیب با نام‌های (SO_r, EO_r)، (SO_g, EO_g) و (SO_b, EO_b) نشان داده شده‌اند. در شکل ۴-۴ دایره‌ای مشاهده می‌شود که فقط سه-چهارم (۲۷۰ درجه) آن ترسیم شده است. این دایره در واقع همان محوطه‌ای

است که موانع موجود در آن توسط پویش‌گر لیزری قابل درک است. به عبارت دیگر شعاع دایره‌ی مذکور برد نهایی پویش‌گر لیزری می‌باشد. علت این که دایره به طور کامل ترسیم نشده است این است که پویش‌گر لیزری مورد استفاده در این پایان‌نامه از نوع Hokuyo UTM-30LX می‌باشد که فقط قادر است ۲۷۰ درجه از محیط پیرامون خود به شعاع ۳۰ متر را پویش کند. توجه شود که مرکز دایره‌ی مذکور مکانی است که ربات هم‌اکنون در آنجا قرار دارد که با دایره‌ای کوچک و توپر نشان داده شده است. خط کوچکی که به مکان فعلی ربات متصل است نشان‌دهنده‌ی بردار جهت ربات می‌باشد.



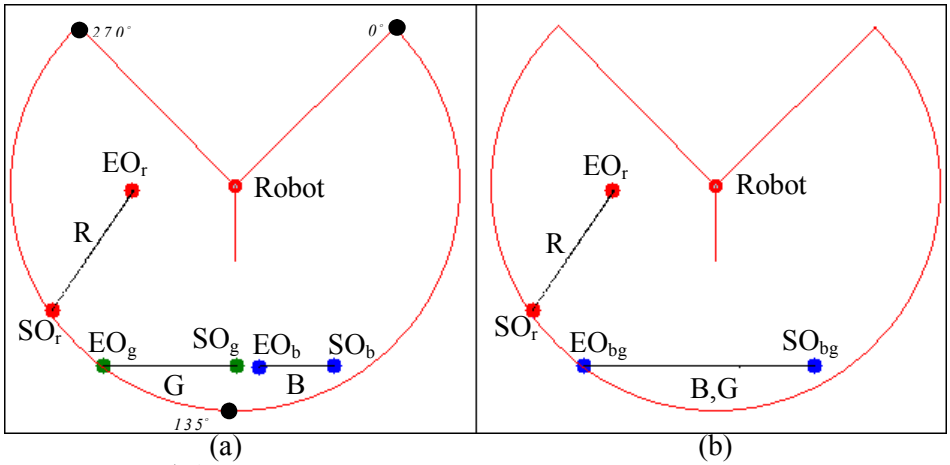
شکل ۴- ۳-سه مانع قرمز، سبز و آبی ربات را محاصره کرده‌اند. فضای بین موانع سبز و آبی برای عبور ربات کافی نیست اما فضای بین موانع سبز و قرمز کافی است.

با توجه دوباره به شکل ۴-۳، مشخص می‌گردد که فضای آزاد بین دو مانع سبز و آبی برای عبور ربات کافی نیست پس تشخیص دادن این دو مانع به صورت مجزا مشکل‌ساز می‌باشد زیرا در مراحل بعدی الگوریتم تنژنت‌باگ، در صورتی که الگوریتم تصمیم بگیرد از فضای آزاد بین این دو مانع عبور کند، ربات به موانع مذکور برخورد خواهد کرد. برای حل این مشکل، در فاز اوّل روش پیشنهادی این پایان‌نامه سه مانع قرمز (SO_r, EO_r)، آبی (SO_b, EO_b) و سبز (SO_g, EO_g) تشخیص داده می‌شوند و در فاز دوم موانع مذکور بر اساس رابطه‌ی ۴-۱۳ دو به دو مورد بررسی قرار می‌گیرند که طی آن دو مانع سبز و آبی به علت نبود فضای کافی بین آن‌ها با یکدیگر ادغام می‌شوند و مانع (SO_{bg}, EO_{bg}) را تشکیل می‌دهند. حال مانع (SO_{bg}, EO_{bg}) با مانع (SO_r, EO_r) مورد بررسی قرار می‌گیرند که فضای کافی بین آن‌ها موجود می‌باشد و با یکدیگر ادغام نمی‌شوند. خروجی فاز دوم را در قسمت b از شکل ۴-۴ مشاهده می‌کنید.

٤-٥ تغییرات اعمال شده در رفتار حرکت-به-سوی-هدف

در فصل سوم اشاره شد که تنژنت‌باگ در رفتار حرکت-به-سوی-هدف برای انتخاب بین یکی از دو نقطه‌ی انتهایی (SO و EO) مانعی که مسیرش را مسدود کرده است، به نوعی طول مسیرهای x-SO-T و x-EO-T را تخمین

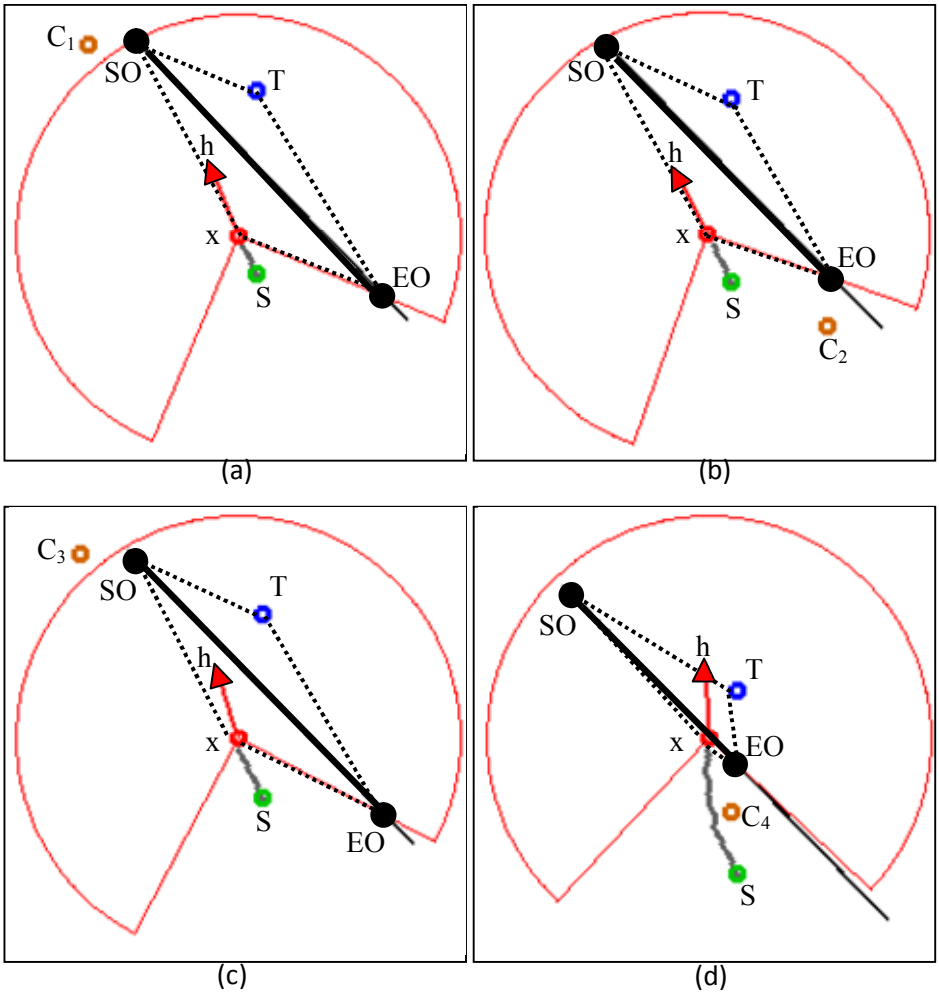
می‌زند. این مسافت تخمین زده شده، فاصله‌ی مکاشفه‌ای نامیده می‌شود. به علاوه بیان شد که از بین دو مسیر مذکور، مسیر با فاصله‌ی مکاشفه‌ای کم‌تر برای دور زدن مانع مسدودکننده برگزیده می‌شود.



شکل ۴- ۴- نحوه‌ی تشخیص موانع مجزا با استفاده از داده‌های مسافت‌یاب: قسمت a خروجی فاز اول (موانع تشخیص داده شده با توجه به شکل ۴-۳) ساخت گراف مماس محلی را نشان می‌دهد. قسمت b خروجی فاز دوم (موانع بعد از عملیات ادغام) را نشان می‌دهد. دو مانع B و G در یک مانع به نام B,G ادغام شده‌اند.

فاصله‌ی مکاشفه‌ای مورد استفاده در تئزنت‌باگ باعث می‌شود که الگوریتم مذکور تقریباً مسیری بهینه (به صورت محلی) از x به T طی کند. اما این فاصله‌ی مکاشفه‌ای در برخی موارد موجب می‌شود که ربات در انتخاب بین یکی از دو نقطه‌ی انتهایی مانع مسدودکننده دچار مشکل شود و در نهایت با مانع مذکور برخورد کند. با ذکر یک مثال به شرح مطلب می‌پردازیم.

فرض کنید که مانع مسدودکننده به طور کامل در دید پویش‌گر لیزری قرار ندارد و قسمتی از آن خارج از محدوده‌ی قابل درک برای حسگر مذکور می‌باشد. به علاوه توجه کنید که تئزنت‌باگ هیچ حافظه‌ای در مورد قسمت دیده شده از مانع در گام‌های قبلی ندارد. به عبارت دیگر تئزنت‌باگ در هر گام فرض می‌کند که تنها موانعی وجود خارجی دارند که قابل رویت برای پویش‌گر لیزری می‌باشند. حال به شکل ۴-۵ توجه کنید. در شکل مذکور مکان ربات با x، مکان شروع حرکت ربات با S، مکان هدف با T، نقاط انتهایی مانع مسدودکننده با SO و EO و نقطه‌ی ایمن با C_i نشان داده شده است. خط‌های نقطه‌چین نشان‌دهنده‌ی فاصله‌ی اقلیدسی بین نقاط می‌باشند. خط روشن خاکستری مسیری است که ربات از نقطه‌ی S تا نقطه‌ی x پیموده است. توجه نمایید که در هر قسمت از شکل ۴-۵ قسمتی از مانع مسدودکننده که در دید پویش‌گر لیزری قرار دارد، با خط سیاه ضخیم و قسمتی که از دید حسگر مذکور خارج است، با خط سیاه باریک نمایش داده شده است.



شکل ۴- ۵-سوئیچ کردن ربات در انتخاب بین SO و EO حین رفتار حرکت-به-سوی-هدف: در قسمت a، SO برای محاسبه‌ی C_1، در قسمت b، EO برای محاسبه‌ی C_2 و در قسمت c دوباره SO برای محاسبه‌ی C_3 انتخاب شده است. در قسمت d، ربات در اثر سوئیچ کردن مداوم بین دو نقطه‌ی مذکور در نهایت به مانع برخورد کرده است.

همان طور که در قسمت a از شکل ۴-۵ دیده می‌شود، فاصله‌ی مکاشفه‌ای مسیر x-SO-T از فاصله‌ی مکاشفه‌ای مسیر x-EO-T کم‌تر می‌باشد. به همین دلیل نقطه‌ی انتهایی SO برای محاسبه‌ی نقطه‌ی ایمن C_1 انتخاب می‌شود. حال هر چه ربات به سمت نقطه‌ی C_1 حرکت می‌کند، بخش بیشتری از مانع مسدودکننده از دید ربات خارج می‌شود که موجب کاهش مقدار فاصله‌ی مکاشفه‌ای مسیر x-EO-T می‌گردد. این روند ادامه پیدا می‌کند تا این که فاصله‌ی مکاشفه‌ای مسیر x-EO-T از فاصله‌ی مکاشفه‌ای مسیر x-SO-T کم‌تر شود. این حالت در قسمت b از شکل ۴-۵ دیده می‌شود. در این زمان ربات برای دور زدن مانع به جای نقطه‌ی SO، نقطه‌ی EO را برای محاسبه‌ی نقطه‌ی C_2 انتخاب می‌کند. با حرکت ربات به سمت نقطه‌ی C_2، قسمت قابل رویت مانع افزایش پیدا می‌کند. در این زمان سناریویی شبیه به آن چه در قسمت a دیده شد، رخ می‌دهد. به این ترتیب ربات دوباره نقطه‌ی SO را انتخاب می‌کند و C_3 بر اساس آن محاسبه می‌شود (قسمت c از شکل ۴-۵). سوئیچ کردن بین دو نقطه‌ی SO و EO ادامه پیدا

می‌کند و در نهایت ربات آن قدر به مانع نزدیک می‌شود که به آن برخورد می‌کند (قسمت d از شکل ۴-۵). علت مشکلی که شرح داده شد، به نحوه‌ی محاسبه‌ی فاصله‌ی مکاشفه‌ای برمی‌گردد و مستقل از نوع ربات می‌باشد. برای جلوگیری از سوئیچ کردن ربات در انتخاب بین نقاط SO و EO می‌توان رابطه‌ای که توسط آن فاصله‌ی مکاشفه‌ای محاسبه می‌گردد را تغییر داد. در ادامه نحوه‌ی این تغییر شرح داده می‌شود.

برای حل مشکل سوئیچ کردن بین نقاط انتهایی مانع مسدودکننده، دو ضریب به نام‌های K_S و K_E (که در ادامه تعریف می‌گردند) به ترتیب در فاصله‌ی مکاشفه‌ای مسیر x-SO-T و فاصله‌ی مکاشفه‌ای مسیر x-EO-T ضرب می‌شوند. با داشتن مختصات نقاط x، SO و EO در دستگاه مختصات جهانی، می‌توان بردارهای $\overrightarrow{xSO}$ و $\overrightarrow{xEO}$ را با استفاده از رابطه‌ی ۴-۱۴ محاسبه کرد. بردار نشان‌دهنده‌ی جهت ربات ($\overrightarrow{xh}$) نیز با استفاده از IMU به دست می‌آید. حال با استفاده از روابط ۴-۱۵ و ۴-۱۶ دو زاویه‌ی sAngle و eAngle را محاسبه می‌کنیم. توجه شود که sAngle زاویه‌ی بین دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xSO}$ و eAngle زاویه‌ی بین دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xEO}$ می‌باشد.در روابط مذکور <a,b> ضرب داخلی دو بردار a و b و |a| اندازه‌ی بردار a را نشان می‌دهد.

حال دو زاویه‌ی sAngle و eAngle را نرمال‌سازی می‌کنیم. منظور از نرمال‌سازی، نگاشت کردن مقادیر دو زاویه‌ی مذکور از بازه‌ی $[۰,\pi]$ (بازه‌ی تابع acos) به بازه‌ی [۰,۱] می‌باشد. برای این کار زوایای sAngle و eAngle به π تقسیم می‌شوند و حاصل به ترتیب مقدار ضرائب K_S و K_E (رابطه‌ی ۴-۱۷) را تشکیل خواهد داد.

$$\overrightarrow{xSO} = SO - x, \quad \overrightarrow{xEO} = EO - x \tag{۱۴-۴}$$

$$\angle sAngle = acos(\frac{<\overrightarrow{xh}, \overrightarrow{xSO}>}{|\overrightarrow{xh}||\overrightarrow{xSO}|}) \tag{۱۵-۴}$$

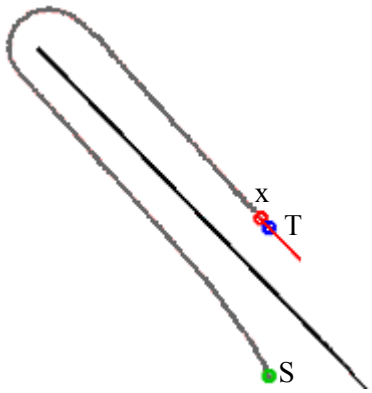
$$\angle eAngle = acos(\frac{<\overrightarrow{xh}, \overrightarrow{xEO}>}{|\overrightarrow{xh}||\overrightarrow{xEO}|}) \tag{۱۶-۴}$$

$$K_S = sAngle / \pi, \quad K_E = eAngle / \pi, \quad (0 \le K_S, K_E \le 1) \tag{۱۷-۴}$$

$$revised\ heuristic\ distance = \begin{cases} K_S \times (d(x,SO) + d(SO,T)), \ for\ SO\ endpoint \\ K_E \times (d(x,EO) + d(EO,T)), \ for\ EO\ endpoint \end{cases} \tag{۱۸-۴}$$

با محاسبه‌ی ضرائب K_Sو K_E، فاصله‌ی مکاشفه‌ای اصلاح شده برای دو مسیر x-SO-T و x-EO-T بر اساس رابطه‌ی ۴-۱۸ به دست می‌آید. ضرائب K_Sو K_E به صورت فاکتورهای وزن‌دهی به فواصل مکاشفه‌ای عمل می‌کنند. به عبارت دیگر با ضرب K_Sو K_Eدر مقدار فاصله‌های مکاشفه‌ای، هنگام انتخاب بین SO و EO، نقطه‌ای برگزیده می‌شود که حرکت به سوی آن تغییرات کم‌تری در جهت ربات ایجاد کند. به این ترتیب در مواردی مانند شکل ۴-۵ سوئیچ کردن بین دو نقطه‌ی SO و EO اتفاق نمی‌افتد. به عنوان مثال دوباره به قسمت b از شکل مذکور توجه نمایید. همان طور که پیش‌تر بیان شد، تنزنت‌باگ نقطه‌ی EO را برای محاسبه‌ی نقطه‌ی ایمن انتخاب می‌کند. اما باید توجه

کرد که انتخاب نقطه‌ی SO (به جای EO) و حرکت به سوی آن تغییرات کم‌تری در جهت ربات ایجاد می‌کند. به همین علت مقدار K_S کم‌تر از مقدار K_E خواهد بود و ضرب K_S و K_E در مقادیر فواصل مکاشفه‌ای باعث می‌شود که فاصله‌ی مکاشفه‌ای مسیر x-SO-T کم‌تر از مسیر x-EO-T گردد. در نتیجه ربات به دنبال کردن نقطه‌ی SO ادامه می‌دهد و به نقطه‌ی EO سوئیچ نمی‌کند. در شکل ۴-۶ رفتار حاصل از فاصله‌ی مکاشفه‌ای اصلاح شده نشان داده شده است. همان طور که در شکل دیده می‌شود، مسیر حرکت ربات حین رفتار حرکت-به-سوی-هدف پایدار، بدون سوئیچ کردن بین SO و EO و بدون برخورد به مانع مسدودکننده می‌باشد.



شکل ۴- ۶-رفتار پایدار ربات حین حرکت-به-سوی-هدف به علت به کارگیری فاصله‌ی مکاشفه‌ای اصلاح شده

٤-٥-١ بررسی شرایط مورد نیاز برای عملکرد درست فاصله‌ی مکاشفه‌ای پیشنهادی

یادآوری می‌شود که علت تغییر فاصله‌ی مکاشفه‌ای تنژنت‌باگ، با مشکل مواجه شدن ربات در انتخاب بین نقاط انتهایی SO و EO در حین رفتار حرکت-به-سوی-هدف و در نهایت برخورد آن با مانع مسدودکننده می‌باشد. حال باید بررسی شود که فاصله‌ی مکاشفه‌ای پیشنهادی تحت چه شرایطی به درستی عمل می‌کند.

فاصله‌ی مکاشفه‌ای مسیر x-SO-T را HD(x,SO,T) و فاصله‌ی مکاشفه‌ای مسیر x-EO-T را HD(x,EO,T) بنامید. حال فرض کنید که در زمان t_1 HD(x,SO,T) کم‌تر از HD(x,EO,T) باشد بنابراین نقطه‌ی SO برای دور زدن مانع انتخاب می‌شود (مانند قسمت a از شکل ۴-۵). حال فرض کنید در زمان t_2 مقدار HD(x,SO,T) بیشتر از مقدار HD(x,EO,T) گردد (مانند قسمت b از شکل ۴-۵). با توجه به ایستا بودن موانع در محیط، می‌دانیم که علت کم‌تر شدن HD(x,SO,T) از HD(x,EO,T)، خروج قسمتی از مانع از دید پویش‌گر لیزری می‌باشد؛ به علاوه می‌دانیم که برای جلوگیری از سوئیچ کردن بین نقاط SO و EO، باید در زمان t_2 نیز نقطه‌ی SO انتخاب شود؛ این در حالی است که (در زمان t_2) فاصله‌ی مکاشفه‌ای تنژنت‌باگ نقطه‌ی EO را پیشنهاد می‌دهد. پس باید بررسی گردد که ضرائب K_S و K_E (استفاده شده در فاصله مکاشفه‌ای پیشنهادی) تحت چه شرایطی باعث می‌شوند مقدار فاصله‌ی مکاشفه‌ای پیشنهادی برای مسیر x-SO-T (که با RHD(x,SO,T) نمایش داده می‌شود) از مقدار فاصله‌ی مکاشفه‌ای پیشنهادی برای مسیر x-EO-T (که با RHD(x,EO,T) نمایش داده می‌شود) کم‌تر شود. که در ادامه به این موضوع می‌پردازیم.

فرض:

مقدار HD(x,SO,T) بیشتر از مقدار HD(x,EO,T) می‌باشد.

هدف:

یافتن شرایطی برای K_S و K_E به طوری که مقدار RHD(x,SO,T) کم‌تر از مقدار RHD(x,EO,T) گردد.

تحلیل:

برای بررسی شرایط مورد نیاز جهت برقراری رابطه‌ی $RHD(x,SO,T) < RHD(x,EO,T)$ با فرض داشتن رابطه‌ی $HD(x,SO,T) > HD(x,EO,T)$، باید بدترین حالت ممکن را در نظر گرفت. منظور از بدترین حالت ممکن، تخمین کوچک‌ترین مقدار HD(x,EO,T) و بزرگ‌ترین مقدار HD(x,SO,T) و به دست آوردن شرایط حاکم روی K_S و K_E جهت برقراری $RHD(x,SO,T) < RHD(x,EO,T)$ می‌باشد.

حال رابطه‌ی ۴-۱۹ (فاصله‌ی مکاشفه‌ای پیشنهادی) را در نظر بگیرید. اگر رابطه‌ی مذکور را باز کنیم، به رابطه‌ی ۴-۲۰ و سپس به رابطه‌ی ۴-۲۱ می‌رسیم. همان طورکه قبلاً اشاره شد، برای در نظر گرفتن بدترین حالت ممکن باید کم‌ترین مقدار(HD(x,EO,T و بیشترین مقدار HD(x,SO,T) را تخمین بزنیم. واضح است که مقدار HD(x,EO,T) کم‌تر از اندازه‌ی بردار $|\overrightarrow{xT}|$ (فاصله‌ی اقلیدسی بین x و T) نمی‌تواند باشد (رابطه‌ی ۴-۲۲). از طرفی اندازه‌ی بردار $|\overrightarrow{xT}|$ مشخص است و به علاوه می‌دانیم که اندازه‌ی بردار $|\overrightarrow{xSO}|$ بین ۰ تا ۳۰ متر (محدوده‌ی کارکرد پویش‌گر لیزری) خواهد بود (رابطه‌ی ۴-۲۳)؛ پس کافی است حداکثر مقدار اندازه‌ی بردار $|\overrightarrow{SOT}|$ را تخمین بزنیم تا بتوانیم رابطه‌ی حاکم روی K_S و K_E را به دست آوریم. برای این کار، باید توجه کرد که بردار $\overrightarrow{SOT}$ از تفریق برداری بین $\overrightarrow{xT}$ و $\overrightarrow{xSO}$ به دست می‌آید (رابطه‌ی ۴-۲۴). پس می‌توانیم اندازه‌ی بردار $\overrightarrow{SOT}$ را مانند رابطه‌ی ۴-۲۵ بنویسیم که در آن θ زاویه‌ی بین دو بردار $\overrightarrow{xT}$ و $\overrightarrow{xSO}$ می‌باشد. حال باید مقدار بیشینه‌ی رابطه-ی ۴-۲۵ را به دست آوریم. برای این کار باید مقدار $\cos(\theta)$ را برابر ۱- در نظر گرفت که به این ترتیب به رابطه‌ی ۴-۲۶ و سپس به رابطه‌ی ۴-۲۷ خواهیم رسید.

حال با توجه به روابط ۴-۲۲، ۴-۲۳ و ۴-۲۷ می‌توان مقدار کمینه‌ی $|\overrightarrow{EOT}| + |\overrightarrow{xEO}|$ و مقدار بیشینه‌ی $|\overrightarrow{SOT}| + |\overrightarrow{xSO}|$ را تخمین زد و با جای‌گذاری مقادیر مذکور در رابطه‌ی ۴-۲۱، به رابطه‌ی ۴-۲۸ و سپس به رابطه‌ی ۴-۲۹ رسید. با توجه به رابطه‌ی ۴-۲۹، رابطه‌ی نهایی ۴-۳۰ نتیجه می‌شود که تعبیرش این است که اگر HD(x,SO,T) بزرگ‌تر از HD(x,EO,T) باشد، مادامی که K_S کوچک‌تر از K_E باشد، مقدار RHD(x,SO,T) کم‌تر از مقدار RHD(x,EO,T) خواهد بود که این امر باعث می‌شود ربات همواره برای دور زدن مانع مسدودکننده نقطه‌ی

انتهایی را به نحوی انتخاب کند که حداقل تغییرات در بردار جهت ربات ایجاد شود و این همان چیزی است که مد نظر ما می‌باشد.

$$RHD(x,SO,T) < RHD(x,EO,T) \Rightarrow \quad (۱۹-۴)$$

$$K_S \times HD(x,SO,T) < K_E \times HD(x,EO,T) \Rightarrow \quad (۲۰-۴)$$

$$\frac{sAng}{\pi} \times [|\overrightarrow{xSO}| + |\overrightarrow{SOT}|] < \frac{eAng}{\pi} \times [|\overrightarrow{xEO}| + |\overrightarrow{EOT}|] \quad (۲۱-۴)$$

$$min(HD(x,EO,T)) = min(|\overrightarrow{xEO}| + |\overrightarrow{EOT}|) = |\overrightarrow{xT}| \quad (۲۲-۴)$$

$$0 < |\overrightarrow{xSO}| < 30 \quad (۲۳-۴)$$

$$\overrightarrow{SOT} = \overrightarrow{xT} - \overrightarrow{xSO} \Rightarrow \quad (۲۴-۴)$$

$$|\overrightarrow{SOT}| = \sqrt{|\overrightarrow{xSO}|^2 + |\overrightarrow{xT}|^2 - 2.|\overrightarrow{xSO}|.|\overrightarrow{xT}|.cos(\theta)} \quad (۲۵-۴)$$

$$max(|\overrightarrow{SOT}|) = \sqrt{|\overrightarrow{xSO}|^2 + |\overrightarrow{xT}|^2 + 2.|\overrightarrow{xSO}|.|\overrightarrow{xT}|} \Rightarrow \quad (۲۶-۴)$$

$$max(|\overrightarrow{SOT}|) = \sqrt{(|\overrightarrow{xSO}| + |\overrightarrow{xT}|)^2} = |\overrightarrow{xSO}| + |\overrightarrow{xT}| \quad (۲۷-۴)$$

$$4-21, 4-22, 4-23, 4-27 \Rightarrow sAng \times [2.|\overrightarrow{xSO}| + |\overrightarrow{xT}|] < eAng \times [|\overrightarrow{xT}|] \Rightarrow \quad (۲۸-۴)$$

$$\frac{sAng}{eAng} < \frac{|\overrightarrow{xT}|}{2.|\overrightarrow{xSO}| + |\overrightarrow{xT}|} \Rightarrow \frac{eAng}{sAng} > \frac{2.|\overrightarrow{xSO}|}{|\overrightarrow{xT}|} + 1 \Rightarrow \quad (۲۹-۴)$$

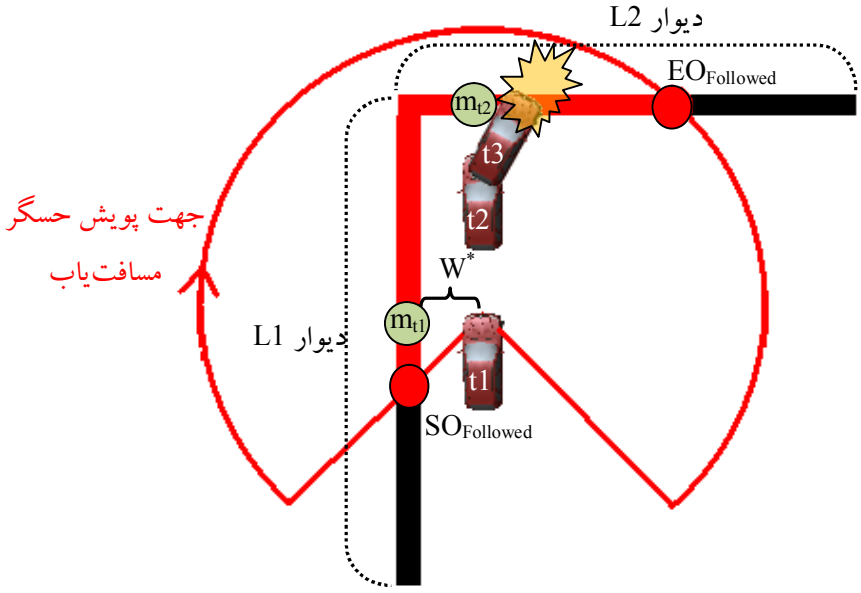
$$eAng > sAng \Rightarrow \frac{eAng}{\pi} > \frac{sAng}{\pi} \Rightarrow K_E > K_S \quad (۳۰-۴)$$

٤-٦ بررسی تغییرات مورد نیاز در رفتار دنبال-کردن-مرز-مانع

در فصل سوم روشی که برای پیاده‌سازی دنبال-کردن-مرز-مانع در [۳۹] ارائه گردیده است، بررسی شد. به طور خلاصه، روش مذکور به این صورت عمل می‌کند که ربات در هر گام کمینه‌ی سراسری داده‌های پویش‌گر لیزری را می‌یابد و آن را m می‌نامد. حال با فرض این که موقعیت ربات با x نمایش داده شود، $\overrightarrow{mx}$ بردار نرمال تخمینی سطح مانع می‌باشد. که ربات برای دنبال کردن مرز مانع همواره در جهت عمود بر بردار مذکور حرکت می‌کند. اگر چه روش مذکور برای ربات‌هایی که حداقل شعاع چرخش آن‌ها برابر صفر می‌باشد مناسب است، برای ربات آکرمن که حداقل شعاع چرخش غیر صفر دارد قابل استفاده نیست و ممکن است که در برخی مواقع موجب برخورد ربات با مانع گردد. در ادامه علت نامناسب بودن روش مذکور با ذکر یک مثال توضیح داده می‌شود.

فرض کنید مانعی که ربات باید مرز آن را دنبال کند، از تقاطع دو دیوار به نام‌های L1 و L2 تشکیل شده باشد. به طوری که L1 و L2 به یکدیگر عمود باشند. در شکل ۴-۷ این سناریو نشان داده شده است که در آن قسمتی از مانع که در زمان t1 قابل رویت برای پویش‌گر لیزری می‌باشد بین دو نقطه‌ی انتهایی به نام‌های $SO_{Followed}$ و $EO_{Followed}$ قرار دارد. در ضمن مکان ربات در زمان‌های t1، t2 و t3 در شکل مشخص می‌باشد. حال فرض کنید که

در زمان t1 ربات در حال حرکت به موازات دیوار L1 می‌باشد. به عبارت دیگر نقطه‌ی کمینه‌ی سراسری داده‌های پویش‌گر (m_{t1}) روی مرز دیوار L1 قرار دارد و ربات در جهت عمود بر بردار $\overrightarrow{m_{t1}x_{t1}}$ در حال حرکت است (x_{t1} مکان ربات در زمان t1 می‌باشد). همان طور که اشاره شد مانعی که ربات باید دنبال کند از تقاطع دو دیوار L1 و L2 تشکیل شده است، بنابراین زمانی که ربات مشغول دنبال کردن مرز دیوار L1 می باشد، دیوار L2 دقیقاً بر سر راهش قرار دارد. توجه شود که ربات همواره فاصله‌ی ایمنی را با مانعی که باید دنبال کند، به میزان مشخصی (W^*) حفظ می‌کند. در نتیجه با توجه به این که ربات در حال دنبال کردن دیوار L1 می‌باشد، فاصله‌ی بین نزدیک‌ترین نقطه از مرز دیوار L1 تا ربات حدوداً برابر W^* می‌باشد. با توضیحات داده شده می‌توان گفت که ربات دنبال کردن مرز دیوار L1 را ادامه می‌دهد تا این که فاصله‌ی ربات از نزدیک‌ترین نقطه روی مرز L2 کم‌تر از W^* شود. همان طور که در شکل ۴-۷ مشخص است این حالت در زمان t2 اتفاق می‌افتد که در نتیجه‌ی آن نقطه‌ی کمینه‌ی سراسری داده‌های پویش‌گر (m_{t2}) روی دیوار L2 قرار خواهد گرفت. در این زمان ربات باید به موازات دیوار L2 و عمود بر بردار $\overrightarrow{m_{t2}x_{t2}}$ حرکت کند (x_{t2} مکان ربات در زمان t2 می‌باشد). حال اگر حداقل شعاع چرخش (R_{min}) ربات آکرمن مساوی یا بیشتر از W^* باشد، زمانی که ربات قصد دنبال کردن دیوار L2 را می‌کند فضای کافی برای دور زدن و هم-راستا کردن خود با مرز دیوار L2 را ندارد و در نتیجه در زمان t3 به دیوار L2 برخورد خواهد کرد.



شکل ۴- ۷-سناریوی برخورد ربات به دیوار L2 حین دنبال کردن دیوار L1، ربات در گام زمانی t3 هنگامی که سعی می‌کند خود را با L2 هم‌راستا کند به آن برخورد می‌کند.

یک راه حل برای جلوگیری از برخورد ربات به مانع در سناریوهایی شبیه به آنچه در شکل ۴-۷ نشان داده شد، این است که مقدار W^* را بزرگ‌تر از R_{min} در نظر گرفت. اما در این صورت مقدار پارامتر W^* به مقدار R_{min} بستگی پیدا خواهد کرد که ممکن است مطلوب نباشد. پس باید به گونه‌ی دیگری مشکل حل گردد. توجه شود که

برد دقیق پویش‌گر لیزری مورد استفاده حدود ۳۰ متر می‌باشد. به عبارت دیگر اوّلین باری که دیوار L2 در دید پویش‌گر لیزری قرار می‌گیرد، فاصله‌اش با ربات حدود ۳۰ متر می‌باشد که این میزان فاصله برای تصمیم‌گیری مناسب قبل از برخورد با L2 کافی می‌باشد. پس برای جلوگیری از برخورد با L2 باید از اطلاعات مفیدی که تا شعاع ۳۰ متری ربات موجود است، استفاده کرد. روش پیشنهادی برای حل مشکل مذکور در ادامه بررسی می‌گردد.

٤-٦-١ روش پیشنهادی برای دنبال-کردن-مرز-مانع

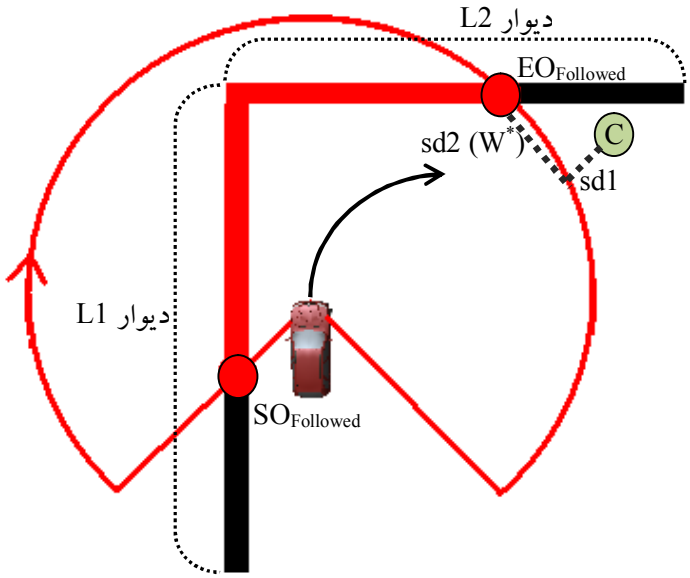
در این روش، ابتدا باید نقطه‌ی m مشخص گردد. اگر در آخرین دفعه‌ای که رفتار حرکت-به-سوی-هدف انجام شده است نقطه‌ی انتهایی انتخاب شده برای دور زدن مانع مسدودکننده از نوع SO بوده، مانع مسدودکننده (همان مانعی که باید مرز آن دنبال شود) در سمت راست ربات قرار گرفته است و اگر نوع نقطه‌ی انتهایی EO بوده، مانع در سمت چپ ربات قرار گرفته است. با توجه به مطلب گفته شده، منطقی است که اگر مانع در سمت راست ربات باشد، نیمه‌ی راست داده‌های پویش‌گر لیزری (بازه‌ی [۱۳۵, ۲۷۰] درجه) و در صورتی که مانع در سمت چپ ربات باشد، نیمه‌ی چپ داده‌های مذکور (بازه‌ی [۰, ۱۳۵] درجه) برای پیدا کردن نقطه‌ی m جست و جو شود. این کار دو مزیت دارد. مزیت اوّل این که مدت زمان جست و جو برای m به نصف کاهش پیدا می‌کند زیرا که فقط نیمی از داده‌های پویش‌گر بررسی می‌گردد. مزیت دوم این که از سردرگُمی ربات حین دنبال کردن مانع جلوگیری می‌کند زیرا که فقط بازه‌ای از داده‌های پویش‌گر لیزری را جست و جو می‌کند که انتظار می‌رود مانع مذکور در آن قرار داشته باشد. با ذکر یک مثال مورد دوم روشن‌تر خواهد شد. فرض کنید ربات در حال دنبال کردن مانعی در سمت راست خود به نام $O_{Followed}$ می‌باشد. به علاوه فرض کنید مانند روش فصل سوم برای پیدا کردن نقطه‌ی m تمام داده‌های پویش‌گر جست و جو می‌شود. مادامی که نقطه‌ی m روی مانع $O_{Followed}$ باشد مشکلی در کار نخواهد بود اما اگر حین دنبال کردن مانع مذکور، در سمت چپ ربات مانعی ظاهر گردد که نقطه‌ی کمینه‌ی سراسری داده‌های پویش‌گر (یا همان m) روی آن قرار داشته باشد، ربات به جای دنبال کردن مانع $O_{Followed}$، مانعی که در سمت چپ خود قرار دارد را دنبال می‌کند. توجه شود که ربات همواره باید همان مانعی ($O_{Followed}$) را دنبال کند که در آغاز رفتار دنبال کردن مانع مد نظر داشته است. زیرا در غیر این صورت ممکن است رفتار دنبال-کردن-مرز-مانع با مشکل مواجه شود و هیچ‌گاه خاتمه نیابد.

بعد از به دست آوردن نقطه‌ی m، به جای آن که ربات در جهت عمود بر بردار نرمال $\overrightarrow{mx}$ حرکت کند، مانعی ($O_{Followed}$) که نقطه‌ی m بر روی مرز آن قرار دارد را در نظر می‌گیرد و همواره به سمت نقطه‌ای ایمن حرکت می-کند که بر اساس یکی از دو نقطه‌ی انتهایی مانع مذکور محاسبه می‌شود. این که کدام یک از دو نقطه‌ی انتهایی برای محاسبه‌ی نقطه‌ی ایمن انتخاب می‌شود، بستگی به این دارد که مانع $O_{Followed}$ در کدام سمت (چپ یا راست) از ربات قرار دارد. اگر مانع مذکور در سمت چپ ربات باشد، نقطه‌ی انتهایی انتخاب شده $EO_{Followed}$ می‌باشد و اگر مانع

مذکور در سمت راست ربات باشد، نقطه‌ی انتهایی انتخاب شده $SO_{Followed}$ می‌باشد. برای بهتر مشخص شدن موضوع به شکل ۴-۸ توجه فرمایید. در این شکل سناریوی شکل ۴-۷ دوباره تکرار شده است اما این بار برای دنبال کردن مرز مانع از روش پیشنهادی استفاده می‌شود. مانع مذکور در سمت چپ ربات قرار دارد به همین علت نقطه‌ی انتهایی $EO_{Followed}$ برای محاسبه‌ی نقطه‌ی C انتخاب می‌گردد. بعد از محاسبه‌ی نقطه‌ی C، ربات می‌تواند به طرف آن حرکت کند که مسیر حرکتش در شکل با فلش سیاه‌رنگ نشان داده شده است.

در واقع در روش ارائه شده ربات منتظر نمی‌ماند تا فاصله‌اش از دیوار L2 کم‌تر از فاصله‌اش از دیوار L1 شود و سپس از دنبال کردن L1 به دنبال کردن L2 سوئیچ کند؛ بلکه به محض این که دیوار L2 در دیدش قرار گیرد دیوار L1 را رها می‌کند و به دنبال کردن دیوار L2 می‌پردازد.

برای حالتی که مانع در سمت راست ربات باشد نیز می‌توان به طریقی مشابه با آن چه در شکل ۴-۸ نشان داده شد، نقطه‌ی ایمن را محاسبه کرد. تنها تفاوتی که در این حالت باید به آن توجه شود این است که نقطه‌ی ایمن، بر اساس $SO_{Followed}$ (به جای $EO_{Followed}$) محاسبه می‌شود.



شکل ۴-۸-نحوه‌ی محاسبه‌ی نقطه‌ی ایمن (C) برای دنبال کردن مانع: مسیر حرکت ربات به طرف نقطه‌ی S با فلش مشخص شده است.

٤-٦-٢ اثبات ایمن بودن روش پیشنهادی برای دنبال کردن مرز مانع

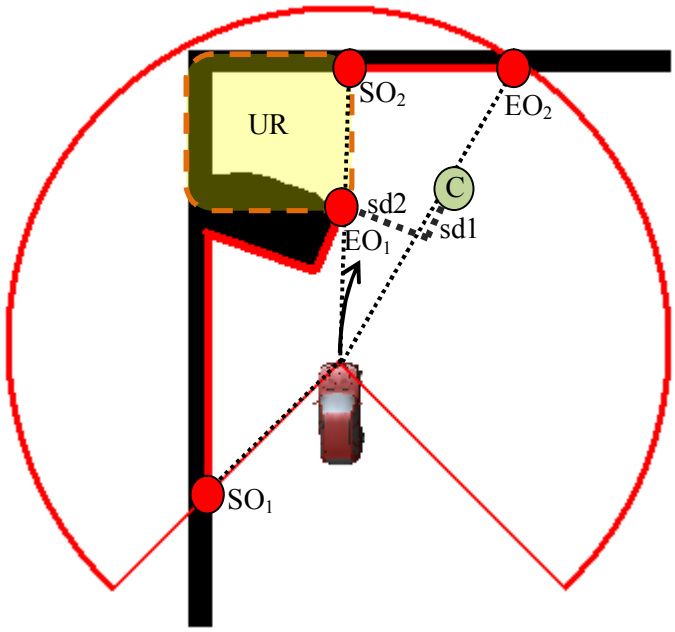
حال که نحوه‌ی دنبال کردن مرز مانع شرح داده شد، ممکن است این شُبهه پیش آید که دنبال کردن مرز مانع با توجه به یکی از نقاط انتهایی آن مناسب نیست. به عنوان مثال اگر در قسمتی از مانع بین دو نقطه‌ی انتهایی‌اش یک برآمدگی موجود باشد (مانند شکل ۴-۹)، ممکن است تصور شود که ربات با بی‌توجهی به قسمت مذکور در حین دنبال کردن مانع بر اساس نقطه‌ی انتهایی آن، به مخاطره می‌افتد و امکان برخوردش با برآمدگی مذکور وجود دارد. می‌توان نشان داد که با استفاده از روش ارائه شده برای دنبال کردن مرز مانع، هیچ گاه ربات به مانع برخورد نمی‌کند.

قضیه:

روش پیشنهادی برای دنبال کردن مرز مانع هیچ‌گاه موجب برخورد ربات با مانعی که باید مرز آن را دنبال کند، نمی‌شود.

اثبات:

برای اثبات این ادعا باید توجه شود که پویش‌گر لیزری با تابش تعدادی پرتوی لیزر به محیط و دریافت بازتاب آن‌ها فاصله‌ی موانع موجود تا ربات را محاسبه می‌کند. طبق قوانین فیزیک پرتوی لیزر به خط مستقیم حرکت می‌کند. به این معنا که اگر مانعی بر سر راه پرتوی لیزر باشد، پرتوی مذکور توانایی حرکت قوصی برای دور زدن حول مانع و رسیدن به پشت آن را ندارد. به این ترتیب هر کجا از سطح مانع که دارای برآمدگی باشد، در داده‌های پویش‌گر لیزری ناپیوستگی ایجاد می‌کند. به عنوان مثال در شکل ۴-۹، در اثر ناپیوستگی ایجاد شده در داده‌های پویش‌گر، مانعی که ربات در حال دنبال کردنش می‌باشد (علی رغم این که مانع یکپارچه می‌باشد) به صورت دو مانع مجزا به نام‌های (SO_1,EO_1) و (SO_2, EO_2) تشخیص داده شده است. حال زمانی که ربات نیمه‌ی چپ داده‌های پویش‌گر را برای پیدا کردن نقطه‌ی m جست و جو می‌کند، نقطه‌ی مذکور را روی مرز مانع (SO_1, EO_1) می‌یابد که به این ترتیب نقطه‌ی امن (C) با توجه به نقطه‌ی EO_1 محاسبه می‌شود و ربات به دیواره‌ی مانع (SO_1, EO_1) که دارای برآمدگی می‌باشد، برخورد نمی‌کند. به این صورت اثبات می‌شود که ربات هرگز به برآمدگی‌های احتمالی روی مرز مانعی که باید دنبال کند، برخورد نمی‌کند.



شکل ۴- ۹-وجود برآمدگی روی مرز مانع موجب ایجاد ناپیوستگی در داده‌های مسافت‌یاب شده، دو مانع مجزا به نام‌های (SO_1, EO_1) و (SO_2, EO_2) به وجود آورده است. در نتیجه ربات با در نظر گرفتن نقطه‌ی انتهایی EO_1، نقطه‌ی ایمن را محاسبه می‌کند. توجه شود که ناحیه‌ی UR برای ربات قابل رویت نمی‌باشد.

٤-٧ پس پردازش تصمیمات تنژنت‌باگ برای افزایش ایمنی ربات

تا به اینجا تغییراتی در روند ساخت گراف مماس محلی با هدف در نظر گرفتن ابعاد ربات و هم‌چنین تغییراتی در رفتار دنبال کردن مرز مانع با هدف سازگار کردن آن با محدودیت‌های حرکتی ربات بررسی گردید. اما موارد مذکور برای سازگاری کامل الگوریتم تنژنت‌باگ با ربات آکرمن کافی نمی‌باشد. به همین دلیل تصمیمات گرفته شده توسط تنژنت‌باگ نیاز به پس‌پردازش دارد. در زیر روند پس‌پردازش و اصلاح تصمیمات تنژنت‌باگ با هدف افزایش ایمنی ربات شرح داده می‌شود. روند مذکور دارای دو فاز می‌باشد. فاز اول از روش میدان پتانسیل بهره می‌گیرد و فاز دوم شبیه به روش VFH+ عمل می‌کند.

٤-٧-١ در نظر گرفتن تمام موانع قابل رویت در رفتار نهایی ربات

علی‌رغم این که الگوریتم تنژنت‌باگ از کل داده‌های پویش‌گر لیزری در ساخت گراف مماس محلی استفاده می‌کند، در تصمیم‌گیری برای حرکت بعدی ربات تنها یک مانع را دخیل می‌کند. که این مانع در رفتار حرکت-به-سوی-هدف همان مانع مسدودکننده و در رفتار دنبال-کردن-مرز همان مانعی است که باید مرزش دنبال شود. مادامی که ربات به صورت نقطه‌ای و بدون ابعاد فرض شود این نحوه‌ی عملکرد مشکلی نخواهد داشت اما یک ربات با ابعاد واقعی را دچار مشکل می‌کند. به عنوان مثال فرض کنید که ربات با حفظ فاصله‌ی عمودی sd2 در حال دنبال کردن مانع $O_{Followed}$ در سمت راست خود می‌باشد. به علاوه مانع O_{left} (یک مانع مجزا و مستقل از $O_{Followed}$) در سمت چپ آن قرار دارد به طوری که فاصله‌ی بین دو مانع مذکور کمی کم‌تر از sd2 باشد اما باز هم فضای کافی برای عبور ربات از بین دو مانع موجود باشد. حال اگر ربات بدون توجه به مانعی که در سمت چپش قرار دارد هم‌چنان فاصله‌ی sd2 را از مانع $O_{Followed}$ حفظ کند، با مانع O_{left} برخورد خواهد کرد. علت این برخورد این است که ربات در حین دنبال کردن مانع $O_{Followed}$ به اطراف خود هیچ توجهی نمی‌کند.

برای حل این مشکل می‌توان خروجی الگوریتم نتژنت‌باگ را پس‌پردازش و تأثیر تمام موانع موجود در محیط را روی تصمیم نهایی آن دخیل کرد که برای این کار از روش میدان پتانسیل استفاده کرده‌ایم. همان طور که در فصل سوم نیز بیان گردید، برای به کارگیری میدان پتانسیل باید بردار نیروهای دافعه و جاذبه محاسبه گردد و برآیند آن‌ها به عنوان جهتی که ربات باید به سمت آن حرکت کند در نظر گرفته شود. نقطه‌ی انتهای[1] بردار جاذبه همان موقعیت ربات (x) و سَرِ[2] آن همان نقطه‌ی ایمن (C) می‌باشد که تنژنت‌باگ برای حرکت بعدی ربات در نظر دارد. حال برای محاسبه‌ی نیروهای دافعه‌ی حاصل از موانع قابل رویت ابتدا باید مقدار آستانه‌ی ρ_0 تنظیم گردد (ρ_0 شعاع

[1] Tail
[2] Head

دایره‌ای به مرکز x است که تمام موانع داخل آن بر ربات نیروی دافعه وارد می‌کنند). با توجه به مقدار ρ_0، موانع تأثیرگذار روی ربات مشخص می‌گردند و بردار دافعه‌ی هر یک از موانع مذکور طبق روش شرح داده شده در فصل سوم محاسبه می‌شود. در نهایت برآیند حاصل از تمام بردارهای دافعه و بردار جاذبه به دست می‌آید و ربات سعی می‌کند با تنظیم زاویه فرمان، خود را هم‌راستا با بردار برآیند کند. به این ترتیب در صورتی که حداکثر زاویه‌ی فرمان ربات به آن اجازه دهد (در فاز دوم پس‌پردازش در مورد مواقعی که حداکثر زاویه فرمان برای پیچیدن ربات کافی نیست بحث خواهیم کرد)، موفق می‌شود علاوه بر دنبال کردن مسیر تعیین شده توسط تنژنت‌باگ به موانع اطراف نیز برخورد نکند. در زیر روابطه مورد نیاز برای محاسبه‌ی بردار برآیند F(x) نشان داده شده است.

$$F_{att}(x) = C - x \qquad (۳۱-۴)$$

$$f_{rep}(x,p) = \begin{cases} k_r.(\frac{1}{d(x,p)} - \frac{1}{\rho_0}).(\frac{1}{d^2(x,p)}).(\frac{x-p}{d(x,p)}) & if\ d(x,p) \le \rho_0 \\ 0 & if\ d(x,p) \ge \rho_0 \end{cases} \qquad (۳۲-۴)$$

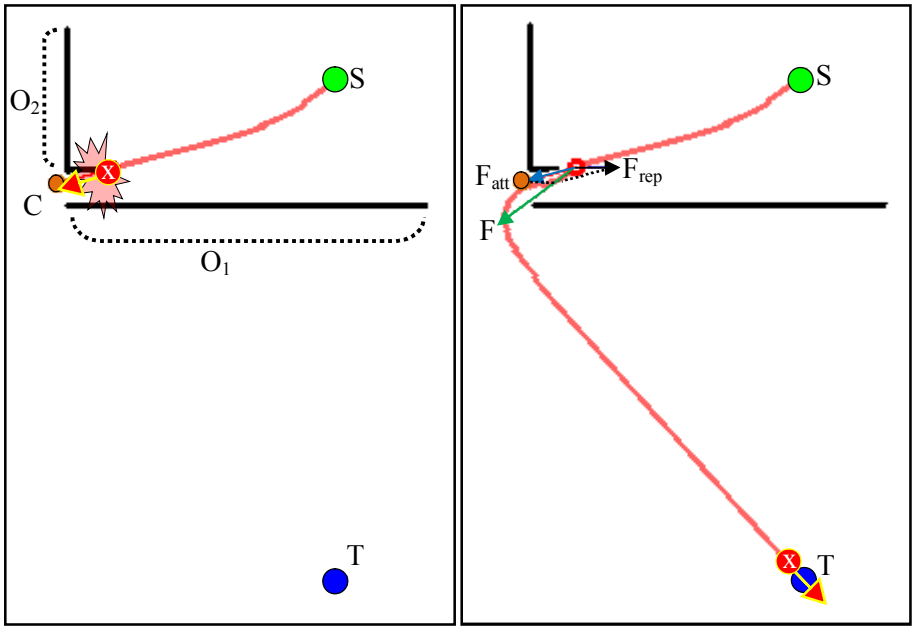
$$F_{rep}(x) = \sum_{i=1}^{b} f_{rep}(x, \arg\min_{p \in \partial O_i}[d(x,p)]) \qquad (۳۳-۴)$$

$$F(x) = K_{att}.F_{att}(x) + K_{rep}.F_{rep}(x) \qquad (۳۴-۴)$$

در رابطه‌ی ۴-۳۱ بردار نیروی جاذبه محاسبه می‌شود که در آن C نقطه‌ی ایمن و x موقعیت فعلی ربات می‌باشد. در رابطه‌ی ۴-۳۲ بردار نیروی دافعه‌ای که نقطه‌ی p بر ربات وارد می‌کند محاسبه می‌شود. لازم به ذکر است که p نزدیک‌ترین نقطه روی مرز مانعی است که می‌خواهیم نیروی دافعه‌ی حاصل از آن را محاسبه کنیم. d(x,p) فاصله‌ی اقلیدسی بین x و p می‌باشد. k_r نیز ضریب تغییر اندازه می‌باشد که برابر ۱ در نظر گرفته شده است. رابطه‌ی ۴-۳۳ برآیند تمام نیروهای دافعه‌ی وارده بر ربات را به دست می‌آورد. که در رابطه‌ی مذکور b تعداد موانع قابل رویت در گام فعلی و ∂O_i مرز مانع O_i می‌باشد. وظیفه‌ی arg min پیدا کردن نزدیک‌ترین نقطه‌ی مرز مانع O_i تا ربات می‌باشد که این نقطه برای محاسبه‌ی نیروی دافعه‌ی حاصل از O_i استفاده می‌شود. رابطه‌ی ۴-۳۴ بردار نیروی دافعه و جاذبه را برای به دست آوردن بردار نهایی F(x)، برآیندگیری می‌کند. که در آن K_{att} و K_{rep} دو ضریب برای وزن‌دهی به بردارهای جاذبه و دافعه می‌باشند به گونه‌ای که حاصل جمع ضرائب مذکور برابر ۱ گردد. بعد از انجام آزمایش‌های متعدد مقدار K_{att} و K_{rep} به ترتیب برابر ۰/۶ و ۰/۴ در نظر گرفته شدند. توجه شود که اگر مقدار K_{att} از مقدار K_{rep} بیشتر باشد، ربات بیشتر تمایل به دنبال کردن نقطه‌ی ایمن دارد و در غیر این صورت، به اجتناب از موانع پیرامونش اولویت بیشتری می‌دهد.

در واقع بردار برآیند محاسبه شده موجب می‌شود که ربات، علاوه بر مانعی که حین انجام دو رفتار تنژنت‌باگ توجهش به آن معطوف است، از سایر موانع قابل رویت نیز غافل نشود. توجه به این نکته ضروری است که با افزایش

مقدار ρ_0، شعاع دایره‌ای که موانع قابل رویت درون آن به ربات نیروی دافعه اعمال می‌کنند، افزایش پیدا می‌کند. حال اگر مقدار ρ_0 از یک حدی بیشتر گردد، موجب می‌شود موانعی که هیچ تهدیدی برای ربات ایجاد نمی‌کنند، بر آن نیروی دافعه وارد کنند. که این امر موجب افزایش محاسبات و به علاوه حرکات اضافی ربات می‌شود. در ضمن در صورتی که مقدار ρ_0 از یک حدی کم‌تر باشد، نیروی دافعه‌ی موانع پیرامون به اندازه‌ی کافی بر ربات اعمال نمی‌شود و ممکن است ربات به موانع مذکور برخورد کند. مقدار ρ_0 به صورت تجربی برابر ۳/۵ متر در نظر گرفته شده است.



(a) (b)

شکل ۴- ۱۰-مقایسه‌ی تأثیر استفاده و عدم استفاده از روش میدان پتانسیل: در قسمت (a) از میدان پتانسیل استفاده نشده اما در قسمت (b) استفاده شده است. مشاهده می‌شود که در (a) ربات در حین دنبال کردن مرز مانع O1 به نام O2 برخورد کرده است. اما در (b) نیروی دافعه‌ی مانع O2 ربات را از خود دور کرده است و ربات از بین O1 و O2 عبور کرده است.

در شکل ۴-۱۰ نمونه‌ای از تأثیر میدان پتانسیل در حفظ ایمنی ربات و عدم برخورد آن با موانع نشان داده شده است. در شکل مذکور S نقطه‌ی شروع، T نقطه‌ی هدف، x موقعیت ربات، C نقطه‌ی انتخاب شده توسط تنژنت‌باگ (یا همان نقطه‌ی امن) می‌باشد. در بخش a از شکل مذکور خروجی الگوریتم تنژنت‌باگ بدون استفاده از فاز پس-پردازش میدان پتانسیل مشاهده می‌شود. همان طور که مشخص است ربات در حین دنبال کردن مانع O_1 به مانع O_2 برخورد کرده است. که علت، بی‌توجهی الگوریتم تنژنت‌باگ به وجود مانع O_2 می‌باشد. در قسمت b از شکل ۴-۱۰ همان سناریو دوباره تکرار شده است. اما این بار خروجی تنژنت‌باگ با استفاده از میدان پتانسیل پس‌پردازش شده است. همان طور که در شکل مشاهده می‌شود هنگامی که ربات به مانع O_2 نزدیک شده است، حاصل برآیند (یا همان بردار F) بردار نیروی جاذبه‌ی نقطه‌ی انتخابی تنژنت‌باگ و بردار نیروی دافعه‌ی مانع O_2 ربات را به طرف نقطه-

ی انتخابی تنژنت‌باگ و به دور از مانع O_2 هدایت کرده است. لازم به ذکر است که در شکل ۴-۱۰ قسمت b طول بردار F (برای بهتر دیده شدن) بیش از اندازه‌ی واقعی آن کشیده شده است.

بعد از محاسبه‌ی بردار F(x)، از مکانیزم رفتن به نقطه (بخش ۴-۳ را ببینید) برای حرکت ربات و هم‌راستا کردن آن با F(x) استفاده می‌شود. برای این کار کافی است بردار $\overrightarrow{xC}$ برابر F(x) قرار داده شود و سپس با توجه به زاویه‌ی کمینه‌ی علامت‌دار بین دو بردار $\overrightarrow{xh}$ و $\overrightarrow{xC}$ مقدار مناسب برای زاویه‌ی فرمان به دست آید. مقدار محاسبه شده برای زاویه فرمان، خروجی فاز ۱ یا همان فاز میدان پتانسیل می‌باشد.

٤-۷-۲ درنظر گرفتن محدودیت‌های حرکتی ربات آکرمن

همان طور که قبلاً نیز اشاره شد، الگوریتم تنژنت‌باگ محدودیت‌های حرکتی ربات را در حین انتخاب نقاطی که ربات باید به سمت آن‌ها حرکت کند، در نظر نمی‌گیرد و همواره فرض می‌کند که ربات قادر به حرکت در هر جهت دلخواه می‌باشد. برای ربات آکرمن چنین فرضی درست نیست زیرا که حداقل شعاع چرخش آن غیر صفر می‌باشد. با توجه به این مطلب، گاهی اوقات ربات در دنبال کردن مجموعه نقاطی که تنژنت‌باگ برای حرکت آن در محیط انتخاب می‌کند، دچار مشکل می‌شود و به موانع برخورد می‌کند. به عنوان مثال ممکن است که ربات مجبور باشد برای دنبال کردن مسیر تعیین شده توسط تنژنت‌باگ، پیچی ۹۰ درجه را بپیچد و چون میزان گردش فرمان محدود می‌باشد، ربات از عهده‌ی انجام مانور مذکور برنمی‌آید و به موانع برخورد خواهد کرد. برای حل این مشکل، فاز دوم پس‌پردازش تصمیمات تنژنت‌باگ ارائه گردیده است.

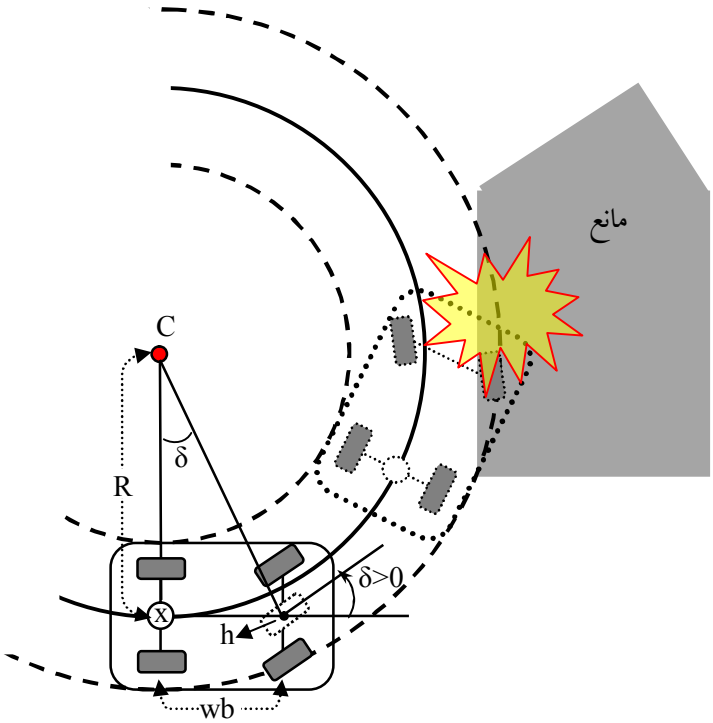
در فاز دوم از فرآیند پس‌پردازشِ تصمیمات تنژنت‌باگ، بازه‌ی مقادیر زاویه فرمان ربات با توجه به محدودیت‌های حرکتی ربات و موانع موجود در اطرافش امکان‌سنجی می‌شود و مقادیر مجاز برای زاویه‌ی فرمان مشخص می‌گردد. حال از بین مقادیر مجاز، نزدیک‌ترین مقدار به زاویه فرمان محاسبه شده در فاز اوّل پس‌پردازش انتخاب می‌شود. روند کار در ادامه شرح داده می‌شود.

متورم‌سازی موانع موجود در دید ربات

برای آن که بتوان وضعیت ربات را حین حرکت روی دایره (دایره‌ای که با توجه به زاویه فرمان تشکیل می‌گردد) از نظر برخورد با موانع بررسی کرد، باید موانع استخراج شده از داده‌های پویش‌گر لیزری را با توجه به ابعاد ربات متورم ساخت. تکنیک متورم کردن موانع برای حفظ ایمنی ربات، در روش‌های دیگر اجتناب از مانع مانند VFH+ نیز استفاده شده است. علت متورم‌سازی این است که ربات دارای ابعادی مشخص می‌باشد و عدم برخورد مختصات مکان ربات با مرز موانع، به معنای ایمن بودن مسیر نمی‌باشد. به عبارت دیگر ممکن است نقطه‌ی نشان‌دهنده‌ی مکان ربات با هیچ مانعی برخورد نداشته باشد اما بدنه‌ی ربات با موانع برخورد کند. با ذکر یک مثال مطلب

روشن‌تر خواهد شد.

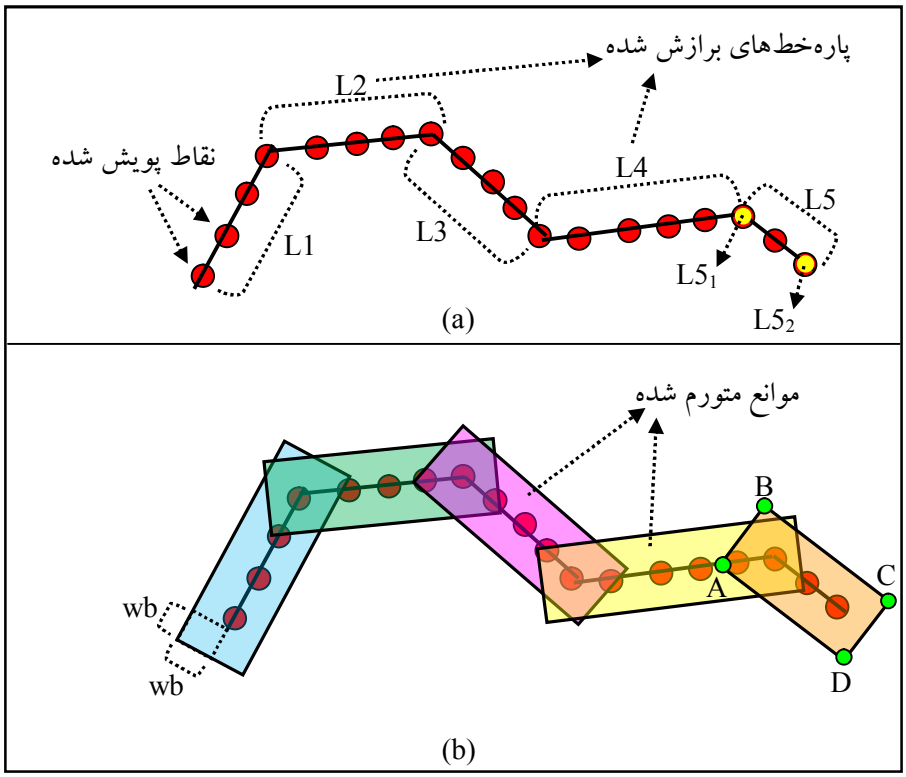
در شکل ۴-۱۱ ربات روی مسیر دایره‌ای به شعاع R و به مرکز C (همان ICR یا مرکز دوران آنی ربات) در حال حرکت است و نقطه‌ای که وسط محور عقب ربات (نقطه‌ی x) قرار دارد موقعیت ربات را نشان می‌دهد. زاویه فرمان ربات نیز با δ نمایش داده شده است. لازم به تذکر است که در مدل آکرمن زاویه فرمان برای چرخ‌های چپ و راست محور جلو یکسان نمی‌باشد اما در اینجا برای سادگی مدل آکرمن با مدل دوچرخه که دارای یک چرخ فرمان‌پذیر می‌باشد، تخمین زده می‌شود. به این صورت که دو چرخ جلوی ربات با یک چرخ در وسط محور جلو جایگزین و زاویه فرمان تک چرخ مذکور δ نام‌گذاری می‌شود. همان طور که در شکل مشخص است اگر ربات به مسیر خود روی دایره با شعاع R ادامه دهد، نقطه‌ی x با مانع خاکستری رنگ برخورد نمی‌کند اما بدنه‌ی ربات با مانع برخورد خواهد کرد. توجه شود که اگر مانع به اندازه‌ی کافی متورم گردد، هنگام برخورد بدنه‌ی ربات با مانع متورم شده، نقطه‌ی x نیز با آن برخورد خواهد کرد. پس با متورم‌سازی موانع و تست عدم برخورد نقطه x با آن‌ها، می‌توان از ایمن بودن مسیری که ربات با زاویه فرمان δ طی می‌کند، اطمینان حاصل کرد. حال که ضرورت متورم‌سازی موانع مشخص گردید، به شرح نحوه‌ی متورم‌سازی می‌پردازیم.



شکل ۴- ۱۱- ربات با مانع خاکستری رنگ برخورد کرده است؛ اگر چه نقطه‌ی نشان‌دهنده‌ی موقعیت ربات (x) با مانع برخورد ندارد. δ زاویه فرمان و R شعاع دایره‌ای است که ربات روی آن حرکت می‌کند. فاصله‌ی بین محور عقب و جلوی ربات برابر wb است.

فرمت داده‌های پویش‌گر لیزری مورد استفاده در این پایان‌نامه، به صورت آرایه‌ای تک بعدی با ۱۰۸۰ عنصر

می‌باشد. پویش‌گر مذکور ۲۷۰ درجه از مساحت دایره‌ای به شعاع ۳۰ متر را با تفکیک‌پذیری[1] ۰/۲۵ درجه به صورت گسسته پویش می‌کند (۱۰۸۰ = ۰/۲۵ ÷ ۲۷۰). در واقع موانع موجود در محیط به صورت مجموعه‌ای از نقاط در آرایه‌ی مذکور ذخیره می‌شوند. حال برای متورم‌سازی مانع O_i، ابتدا به مجموعه نقاطی از آرایه‌ی مذکور که به O_i متعلق هستند، تعدادی پاره‌خط برازش می‌گردد. برای برازش پاره‌خط از انشعاب-ادغام[2] [۵۶] استفاده می‌شود. سپس پاره‌خط‌های حاصل از برازش با توجه به ابعاد ربات به شکل مستطیل متورم می‌شوند.



شکل ۴- ۱۲-نحوه‌ی متورم‌سازی مانع: در قسمت a به نقاط مانع (دایره‌ها) خط‌های L1، L2، L3، L4 و L5 برازش شده است. در قسمت b از خط‌های مذکور، به عنوان اسکلت مستطیل‌ها (موانع متورم شده) استفاده شده است. به عنوان مثال ABCD متورم شده‌ی قسمتی از مانع است که به آن خط L5 برازش شده است. wb فاصله‌ی بین محورهای عقب و جلوی ربات می‌باشد.

در شکل ۴-۱۲ نحوه‌ی متورم‌سازی یک مانع نمونه نشان داده شده است. در قسمت a از شکل مذکور نقاط ترسیم شده همان داده‌های پویش‌گر لیزری هستند و پاره‌خط‌های سیاه رنگ (L1، L2، L3، L4 و L5) خروجی الگوریتم انشعاب-ادغام می‌باشد که با توجه به نقاط مذکور محاسبه شده‌اند. همان طور که در قسمت b از شکل ۴-۱۲ مشاهده می‌شود، هر یک از پاره‌خط‌های حاصل از برازش، نقشی شبیه به محور میانی[3] برای مستطیل متناظرشان ایفا می‌کنند. به عنوان مثال پاره‌خط L5 (در قسمت a) که دو نقطه‌ی انتهایی‌اش $L5_1$ و $L5_2$ نام دارند، محور میانی مستطیل ABCD (در قسمت b) می‌باشد. راستای مستطیل ABCD، بر اساس شیب پاره‌خط L5 تعیین می‌گردد. به

[1] Resolution
[2] Split-Merge
[3] Medial axis

عبارت دیگر پاره‌خط L5 موازی با اضلاع AD و BC و عمود بر اضلاع AB و CD می‌باشد. ابعاد مستطیل مذکور به گونه‌ای تعیین می‌گردد که فاصله‌ی بین L5 و AD و فاصله‌ی بین L5 و BC برابر wb باشد. به علاوه فاصله‌ی نقطه‌ی $L5_1$ از پاره‌خط AB و فاصله‌ی $L5_2$ از پاره‌خط CD برابر wb است.

تشکیل لیست Δ^*

فرض کنید ربات بازه‌ی $[-\delta_{max}, \delta_{max}]$ را به عنوان مقادیر زاویه‌ی فرمان در اختیار داشته باشد؛ به این معنا که ربات بتواند با حداکثر زاویه فرمان $-\delta_{max}$ به سمت راست و حداکثر زاویه فرمان δ_{max} به سمت چپ خود بپیچد. در این مرحله بازه‌ی مذکور با گام‌های مشخص گسسته‌سازی می‌شود که به این ترتیب لیستی از مقادیر زاویه فرمان به فرم $\Delta=\{\delta_0, \delta_1, \ldots, \delta_n\}$ خواهیم داشت. عنصر iام از لیست Δ با استفاده از رابطه‌ی ۴-۳۵ به دست می‌آید که در آن α گام استفاده شده جهت گسسته‌سازی می‌باشد. در رابطه‌ی ۴-۳۶ مقدار n (تعداد عناصر موجود در لیست Δ) محاسبه می‌شود. بعد از تشکیل لیست Δ، برای هر عنصر از آن مانند δ_i، شعاع (R_i) و مرکز (C_i) دایره‌ای که ربات با زاویه فرمان δ_i روی آن حرکت خواهد کرد محاسبه و به عنصر مربوطه در لیست پیوست می‌شود. لیست حاصل Δ^* نام دارد و هر عنصر از آن به فرم $(\delta_i, R_i, C_i=(x_i, y_i))$ می‌باشد. محاسبه‌ی مقدار R_i برای عنصر δ_i از رابطه‌ی ۴-۳۷ انجام می‌شود که در آن wb فاصله‌ی بین محورهای عقب و جلوی ربات می‌باشد. محاسبه‌ی C_i در ادامه شرح داده خواهد شد.

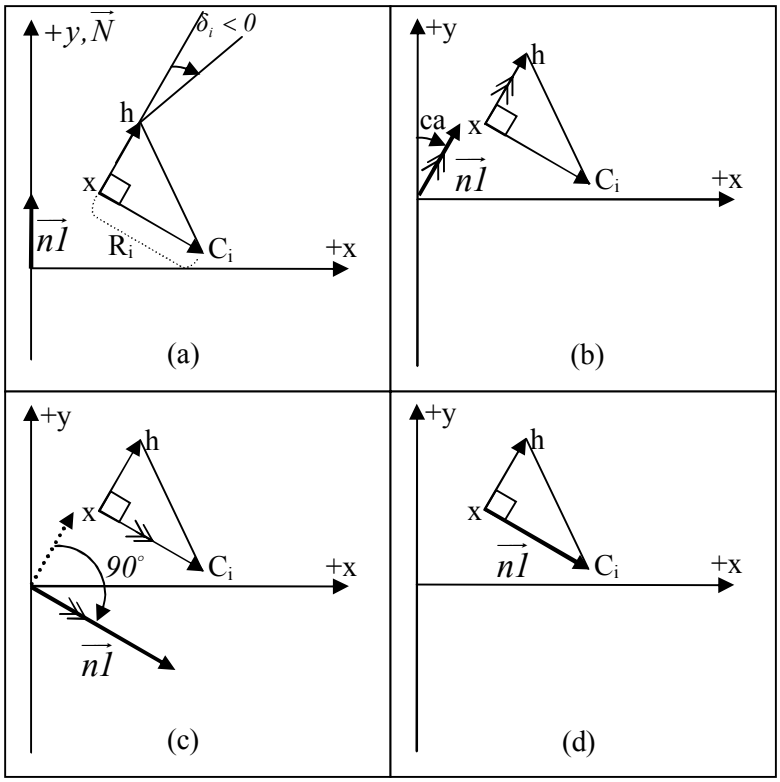
$$\delta_i = -\delta_{max} + i\alpha \qquad (۳۵-۴)$$

$$n = \frac{2\delta_{max}}{\alpha} \qquad (۳۶-۴)$$

$$R_i = \frac{wb}{tan(\delta_i)} \qquad (۳۷-۴)$$

به شکل ۴-۱۳ قسمت a توجه کنید. در شکل مذکور مثلثی به نام xC_ih همانند مثلث xCh از شکل ۴-۱۱ مشاهده می‌شود. تفاوت مثلث xC_ih با مثلث xCh در مقدار δ می‌باشد (در شکل ۴-۱۱ $\delta > 0$ اما در ۴-۱۳ $\delta < 0$ است). طول xC_i برابر R_i و هدف به دست آوردن مختصات نقطه‌ی C_i یا همان ICR_i می‌باشد. فرض کنید بردار نشان‌دهنده‌ی قطب شمال ($\overrightarrow{N}$) بر محور y از دستگاه مختصات جهانی منطبق باشد. به علاوه فرض کنید که IMU زاویه بین بردار جهت سر ربات ($\overrightarrow{xh}$) و بردار $\overrightarrow{N}$ را به صورت ساعت‌گرد و در بازه‌ی [۰, ۳۶۰] به ما می‌دهد. زاویه‌ی مذکور را ca بنامید. با توجه به منطبق بودن بردار $\overrightarrow{N}$ بر محور y، اگر برداری به نام $\overrightarrow{n1}$ به اندازه‌ی واحد هم‌راستا با $\overrightarrow{N}$ به گونه‌ای در نظر گرفته شود که مختصات انتهای آن (۰, ۰) باشد (شکل ۴-۱۳ قسمت a) و سپس حول مبدأ مختصات به اندازه‌ی ca در جهت ساعت‌گرد دوران داده شود، بردار حاصل هم‌راستا با بردار $\overrightarrow{xh}$ خواهد بود (شکل ۴-۱۳ قسمت b). حال که بردار $\overrightarrow{n1}$ هم‌راستا با $\overrightarrow{xh}$ می‌باشد، می‌توان با استفاده از آن بردار $\overrightarrow{xC_i}$ را به دست آورد .

برای این کار، بردار $\overrightarrow{n1}$ با توجه به علامت δ_i دوباره حول نقطه‌ی (۰ ,۰) دوران داده می‌شود به طوری که اگر δ_i مثبت(منفی) باشد میزان دوران $\overrightarrow{n1}$ برابر ۹۰+(۹۰-) درجه خواهد بود. بعد از دوران مذکور، $\overrightarrow{n1}$ هم‌راستا با $\overrightarrow{xC_i}$ می‌باشد (شکل ۴-۱۳ قسمت c). حال کافی است که $\overrightarrow{n1}$ نرمال‌سازی شود و سپس در R_i ضرب شده با x جمع گردد تا به این ترتیب نقطه‌ی سر بردار $\overrightarrow{n1}$ بر روی نقطه‌ی C_i منطبق گردد (شکل ۴-۱۳ قسمت d). علت جمع کردن $\overrightarrow{n1}$ با x این است که مختصات انتهای بردار $\overrightarrow{n1}$ (۰ ,۰) می‌باشد و لازمه‌ی این که سر بردار $\overrightarrow{n1}$ نقطه‌ی C_i را نشان بدهد این است که انتهای آن روی نقطه‌ی x قرار داشته باشد.

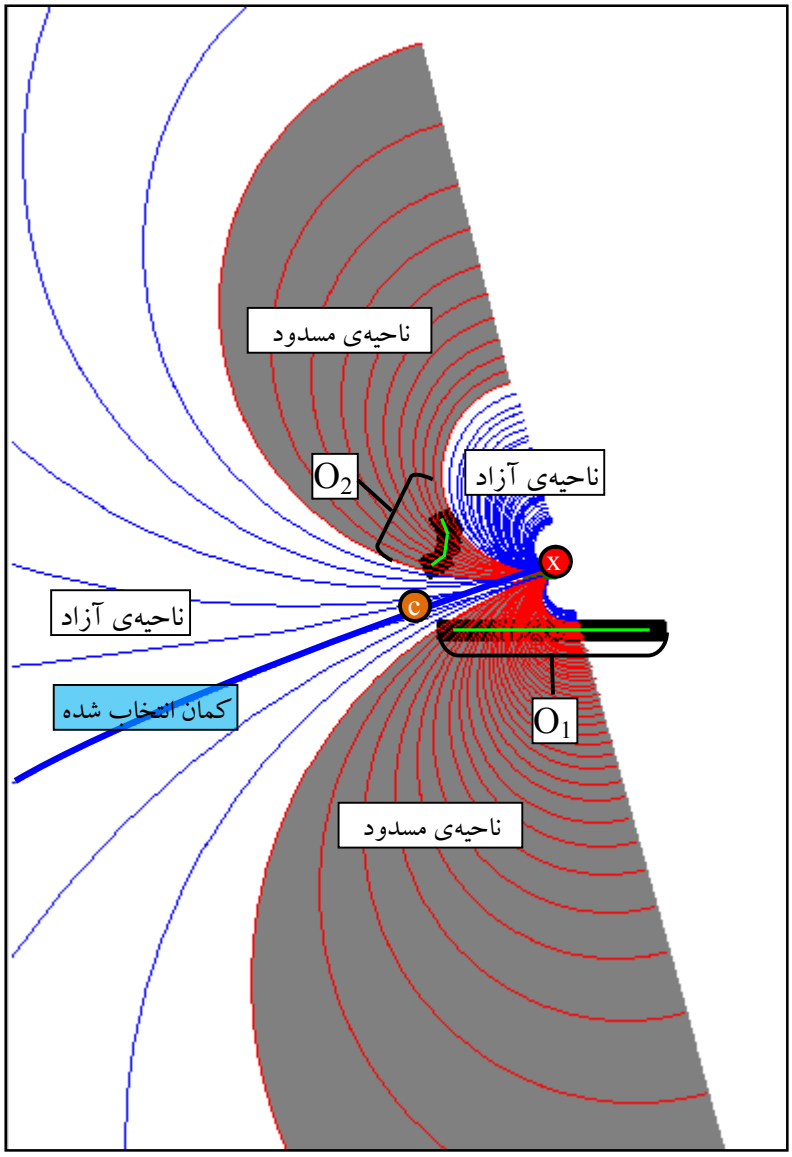


شکل ۴- ۱۳-نحوه‌ی محاسبه‌ی نقطه‌ی Ci با در دست داشتن نقطه‌ی x ، زاویه‌ی ca و با توجه به علامت زاویه فرمان (δi): در قسمت a بردار $\overrightarrow{n1}$ هم‌راستا با بردار شمال ($\overrightarrow{N}$) در نظر گرفته می‌شود. در قسمت b بردار $\overrightarrow{n1}$ به اندازه‌ی ca در جهت ساعت‌گرد دوران داده می‌شود تا با بردار $\overrightarrow{xh}$ هم‌راستا شود. در قسمت c بردار $\overrightarrow{n1}$ ۹۰ درجه‌ی دیگر دوران داده می‌شود تا با بردار $\overrightarrow{xC_i}$ هم‌راستا شود. در قسمت d انتهای بردار $\overrightarrow{n1}$ به نقطه‌ی x منتقل می‌شود تا بر بردار $\overrightarrow{xC_i}$ منطبق شود و به این ترتیب سَرِ $\overrightarrow{n1}$ مختصات نقطه‌ی Ci را نشان دهد.

تشخیص مقادیر ایمن برای زاویه‌ی فرمان

در این مرحله، با در اختیار داشتن موانع متورم شده، هر یک از دایره‌های موجود در لیست Δ^* از نظر برخورد با موانع متورم مورد بررسی قرار می‌گیرند. به عنوان مثال اگر دایره به شعاع R_i و مرکز $C_i=(x_i, y_i)$ با هیچ یک از موانع متورم شده برخورد نداشته باشد، زاویه فرمان δ_i به عنوان یکی از کاندیدهای مجاز زاویه فرمان در لیست Δ^* باقی می‌ماند. اما اگر دایره‌ی مذکور حداقل با یکی از موانع برخورد کند، زاویه فرمان متناظر با آن نامعتبر تلقی می-

شود و از لیست Δ^* حذف می‌گردد. بعد از حذف مقادیر نامعتبر زاویه‌ی فرمان، از بین کاندیدهای باقی مانده در Δ^*، آن که به مقدار زاویه فرمان محاسبه شده در فاز اولِ پس‌پردازش نزدیک‌تر می‌باشد، به عنوان مقدار نهایی زاویه‌ی فرمان در نظر گرفته می‌شود و به فرمان ربات اعمال می‌گردد.



شکل ۴- ۱۴-نمونه‌ای از خروجی فاز دوم پس‌پردازش از روش پیشنهادی: O_1 و O_2 موانع متورم شده، x مکان ربات و c نقطه‌ی انتخاب شده توسط تنژنت‌باگ می‌باشد. ناحیه‌هایی که کمان‌های ترسیم شده در آن‌ها با موانع برخورد کرده‌اند، به رنگ خاکستری نشان داده شده‌اند. کمانی که ضخیم‌تر از بقیه ترسیم شده است، توسط فاز دوم پس‌پردازش از بین کمان‌های معتبر انتخاب شده است.

در شکل ۴-۱۴ نمونه‌ای از خروجی فاز دوم پس‌پردازش تصمیمات تنژنت‌باگ نشان داده شده است. همان طور که در شکل مشاهده می‌شود دو مانع به نام‌های O_1 و O_2 در محیط قرار دارند که با استفاده از روش شرح داده شده در فاز دوم پس‌پردازش، متورم شده‌اند. مکان ربات با x و نقطه‌ی انتخاب شده توسط تنژنت‌باگ با c نشان داده شده است. کمان‌هایی که در شکل دیده می‌شوند، اعضای لیست Δ^* قبل از حذف کاندیدهای غیرمجاز می‌باشند. بعد

از بررسی کاندیدهای موجود در لیست Δ^*، ناحیه‌هایی که در آن کمان‌های غیرمجاز واقع شده‌اند، با رنگ خاکستری در شکل مشخص شده‌اند. در شکل مذکور یکی از کمان‌های ترسیم شده از بقیه ضخیم‌تر می‌باشد. که این کمان همان کاندید منتخب فاز دوم می‌باشد. توجه شود که کمان منتخب در محدوده‌ی آزاد محیط قرار دارد و تا حد امکان به نقطه‌ی c نزدیک می‌باشد.

٤-٨ مکان‌یابی ربات در محیط

همان طور که در فصل اوّل اشاره گردید، مکانیزم مکان‌یابی در محیط‌های ناشناخته و خارج از جاده بدون بهره‌گیری از حسگرهای مکان‌یابی مطلق مانند GPS، بحثی وسیع می‌باشد و از حیطه‌ی این پایان‌نامه خارج است. لذا مکانیزم مکان‌یابی انتخاب شده در این پایان‌نامه، صرفاً از نوع dead-reckoning بر اساس تخمینی از روابط سینماتیکی حاکم بر ربات آکرمن و با بهره‌گیری از حسگرهای انکودر و IMU می‌باشد. در هر گام از به روز رسانی مکان ربات، مسافت طی شده با استفاده از انکودر به دست می‌آید و IMU زاویه‌ی ساعت‌گرد نسبت به شمال مغناطیسی را به ما می‌دهد. با داشتن اطلاعات حسگرهای مذکور و با استفاده از روابط سینماتیکی می‌توان تخمینی از مکان ربات به دست آورد.

٤-٩ جمع‌بندی

در این فصل روش ارائه شده برای انجام ناوبری خودمختار در محیط ناشناخته و خارج از جاده بررسی گردید. همان طور که ذکر شد، الگوریتم نتژنت‌باگ به عنوان پایه‌ی روش ارائه شده در نظر گرفته شده که با اعمال تغییراتی با ربات آکرمن سازگار گردیده است. از جمله تغییرات اعمال شده در تنژنت‌باگ می‌توان به بازبینی رفتارهای حرکت-به-سوی-هدف و دنبال-کردن-مرز-مانع اشاره کرد. به علاوه جزئیات پیاده‌سازی پروسه‌ی ساخت گراف مماس محلی مورد بررسی قرار گرفت و در آخر تصمیمات گرفته شده توسط تنژنت‌باگ در دو فاز پس‌پردازش شد تا ایمنی ربات حین حرکت بین موانع افزایش یابد. در فاز اول از روش میدان پتانسیل بهره گرفته شد و در فاز دوم روشی شبیه به VFH+ جهت اعتبارسنجی زاویه فرمان انتخاب شده در فاز اول، ارائه گردید. در جدول ۴-۱ شبه کد روش پیشنهادی آورده شده است.

جدول ۴- ۱-شبه کد روش ارائه شده

۱- رفتار حرکت-به-سوی-هدف: با توجه به گراف مماس محلی نقطه‌ی ایمن C را جهت حرکت به سوی هدف (T) محاسبه کن. در صورت باز بودن مسیر xT نقطه‌ی T_{node} را محاسبه و در C ذخیره کن و اگر مسیر مذکور مسدود می‌باشد از فاصله‌ی مکاشفه‌ای اصلاح شده برای انتخاب یکی از دو نقطه‌ی انتهایی مانع مسدودکننده (SO و EO) جهت دور زدن آن استفاده کن. بر اساس نقطه‌ی انتخاب شده، نقطه‌ی C را محاسبه کن و سپس با توجه به بردارهای $\overrightarrow{xh}$ و $\overrightarrow{xC}$ زاویه فرمان δ را به دست بیاور. سپس δ را در

فاز پس‌پردازش مورد بررسی قرار بده و در صورت نیاز اصلاح کن. ربات را با زاویه فرمان اصلاح شده حرکت بده. به رفتار حرکت-به-سوی-هدف ادامه بده تا زمانی که یکی از دو شرط زیر برقرار شود:

الف) ربات به هدف برسد⇐ ربات را متوقف کن.

ب) فاصله‌ی ربات تا هدف رو به افزایش بگذارد⇐ به مرحله‌ی ۲ (دنبال کردن مرز مانع) برو.

۲- رفتار دنبال-کردن-مرز-مانع: با استفاده از گراف مماس محلی مانعی که باید دنبال شود را مشخص کن. با توجه به یکی از دو نقطه‌ی انتهایی مانع مذکور (هم‌نوع با نقطه‌ی انتهایی انتخاب شده در آخرین رفتار حرکت-به-سوی-هدف)، نقطه‌ی ایمن C را محاسبه کن. با توجه به بردارهای $\overrightarrow{xh}$ و $\overrightarrow{xC}$ زاویه فرمان δ را به دست بیاور. سپس δ را در فاز پس‌پردازش مورد بررسی قرار بده و در صورت نیاز اصلاح کن. ربات را با زاویه فرمان اصلاح شده حرکت بده. در حین حرکت مقدار متغیرهای $d_{Followed}$ و d_{Reach} را به روز رسانی کن. این روند را ادامه بده تا یکی از سه شرط زیر برقرار شود:

پ) ربات به هدف برسد⇐ ربات را متوقف کن.

ت) ربات یک دور کامل به دور مانع بزند⇐با چاپ پیام "هدف قابل دسترس نیست" ربات را متوقف کن.

ث) مقدار d_{Reach} کم‌تر از $d_{Followed}$ شود⇐ به مرحله‌ی ۱ برو (حرکت-به-سوی-هدف).

فصل پنجم

نتایج شبیه‌سازی و عملی

۵-۱ مقدمه

در فصل چهارم، چالش‌های استفاده از الگوریتم تنژنت‌باگ روی ربات آکرمن شرح داده و تدابیری برای رفع آن‌ها پیشنهاد شد و به این ترتیب سیستمی برای ناوبری در محیط ناشناخته‌ی خارج از جاده قابل استفاده روی ربات آکرمن ارائه گردید. در این فصل نتایج حاصل از تست سیستم ناوبری ارائه شده، به صورت شبیه‌سازی و عملی بررسی می‌گردد. کارایی سیستم مذکور و مقایسه‌ی آن با الگوریتم تنژنت‌باگ به صورت شبیه‌سازی انجام می‌شود و بعد از اطمینان حاصل کردن از مناسب بودن روش ارائه شده، نمونه‌ای از تست عملی سیستم بر روی یک ربات حقیقی نیز ارائه می‌گردد.

۵-۲ نتایج حاصل از شبیه‌سازی

برای شبیه‌سازی سیستم ارائه شده و مقایسه‌ی آن با تنژنت‌باگ، از نرم‌افزار Webots 7.0.1، محیط توسعه‌ی یکپارچه[1] Visual C++ 2008 و کتابخانه‌ی بینایی ماشین OpenCV 2.1.0 استفاده شده است. جهت تست سیستم مذکور باید یک ربات آکرمن شبیه‌سازی شده تهیه گردد. یکی از پروژه‌های نمونه[2] در نرم‌افزار Webots به نام autonomous_vehicle دارای رباتی از نوع آکرمن است که شبیه‌سازی‌های این پایان‌نامه با استفاده از آن انجام شده

[1] IDE
[2] Sample projects

است. از کتابخانه‌ی OpenCV برای ترسیم نقشه‌ی محیط بر اساس داده‌های پویش‌گر لیزری و از Visual C++ برای کامپایل کد پایان‌نامه، استفاده شده است.

در فصل چهارم تغییرات مورد نیاز برای سازگار شدن الگوریتم تنژنت‌باگ با ربات آکرمن مورد بررسی قرار گرفت. تغییرات اعمال شده عبارتند بودند از: ۱- افزودن فاز ادغام به پروسه‌ی ساخت گراف مماس محلی ۲- بازبینی رابطه‌ی فاصله‌ی مکاشفه‌ای در رفتار حرکت-به-سوی-هدف ۳- بازبینی رفتار دنبال-کردن-مرز-مانع ۴- افزودن دو فاز پس‌پردازش خروجی تنژنت‌باگ. در ادامه برای هر یک از این موارد، کارایی سیستم ارائه شده را با تنژنت‌باگ مقایسه خواهیم کرد.

۵-۲-۱ مقایسه در ساخت گراف مماس محلی

همان طور که در فصل چهارم اشاره شد، موانعی که فضای آزاد بین آن‌ها به اندازه‌ی کافی نمی‌باشد باید با یکدیگر ادغام گردند تا از برخورد ربات به آن‌ها حین عبور از بینشان، جلوگیری شود. در ادامه نمونه‌هایی برای مشخص شدن تأثیر فاز ادغام در ساخت گراف مماس محلی ارائه می‌شود.

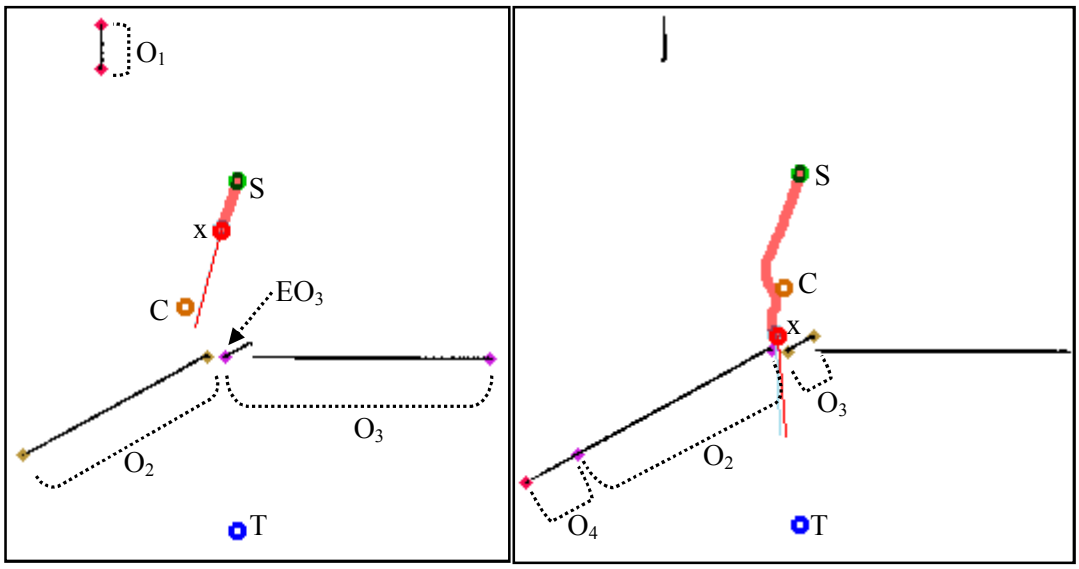
مقایسه‌ی تنژنت‌باگ و روش ارائه شده در ساخت گراف مماس محلی

در شکل ۵-۱ سناریوی اوّل برای مقایسه‌ی تنژنت‌باگ با روش ارائه شده ازنظر ساخت گراف مماس محلی نشان داده شده است. طبق معمول x مکان ربات، S نقطه‌ی شروع، T نقطه‌ی هدف، C نقطه‌ی ایمن محاسبه شده و O_iها موانع می‌باشند. توجه شود که در شکل ۵-۱ قسمتی از موانع که هم‌اکنون در دید پویش‌گر لیزری قرار دارند به صورت O_i نام‌گذاری شده‌اند و قسمت‌هایی که نام‌گذاری نشده‌اند اما به صورت خطوط سیاه رنگ مشاهده می‌شوند، صرفاً برای مشاهده‌ی خواننده رسم شده است. یادآوری می‌شود که تنژنت‌باگ هیچ‌گونه حافظه‌ای از موانعی که قبلاً مشاهده کرده است در اختیار ندارد. به این ترتیب علت عدم حضور مانع O_1 در قسمت b از شکل ۵-۱، قرار نداشتن آن در دید ربات و فراموش شدن آن می‌باشد.

در قسمت a از شکل ۵-۱ ربات در حال حرکت به سوی هدف می‌باشد که مانع O_3 را در راهش مشاهده می‌کند و نقطه‌ی C را با توجه به نقطه‌ی EO_3 برای دور زدن مانع مذکور محاسبه می‌کند. بعد از مدتی که ربات به حرکت خود ادامه می‌دهد، به مانع O_3 برخورد می‌کند (قسمت b از شکل ۵-۱). علت برخورد، نبود فضای کافی بین دو مانع O_2 و O_3 می‌باشد.

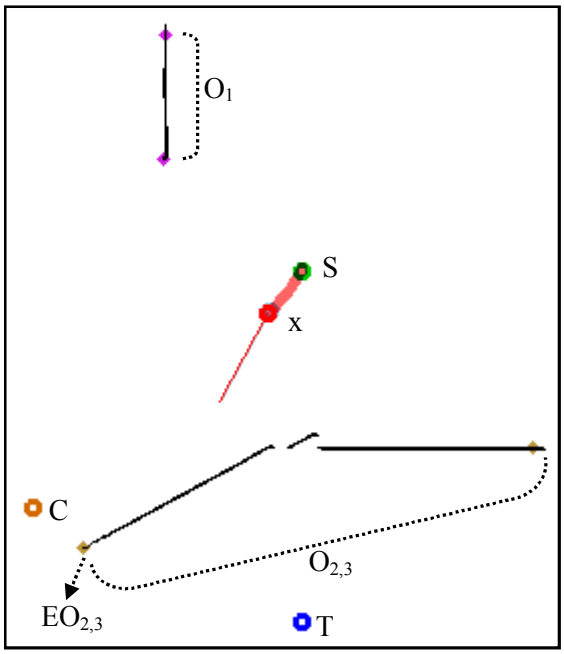
حال خروجی روش ارائه شده برای سناریوی شکل ۵-۱ را در شکل ۵-۲ مورد بررسی قرار می‌دهیم. در شکل مذکور دو مانع O_2 و O_3 با یکدیگر ادغام شده‌اند و مانع $O_{2,3}$ را تشکیل داده‌اند. به این ترتیب دیگر ربات سعی در عبور از بین آن دو نمی‌کند و در عوض بر اساس نقطه‌ی انتهایی $EO_{2,3}$، نقطه‌ی C را محاسبه می‌کند. در شکل ۵-۳

ادامه‌ی حرکت ربات در شکل ۵-۲ نشان داده شده است که ربات با بهره‌گیری از روش ارائه شده، به سلامت به هدف رسیده است.

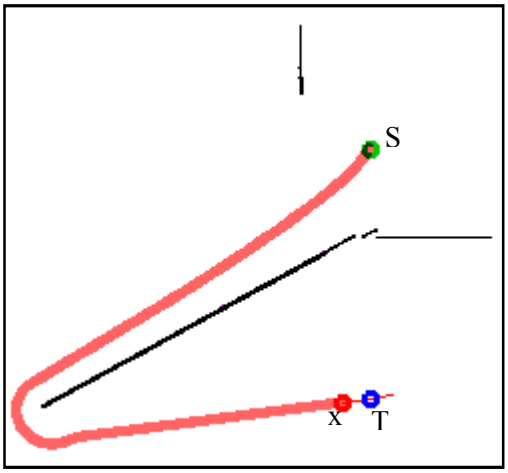


(a) (b)

شکل ۵- ۱-نحوه‌ی رفتار تنژنت‌باگ در سناریوی اوّل از ساخت گراف مماس محلی: همان طور که مشاهده می‌شود نبود فاز ادغام موانع در ساخت گراف مذکور باعث برخورد ربات با موانع شده است. قسمت a اوایل حرکت ربات را نشان می‌دهد و قسمت b برخورد ربات با مانع O_3 حین عبور از بین آن و O_2 را نشان می‌دهد.

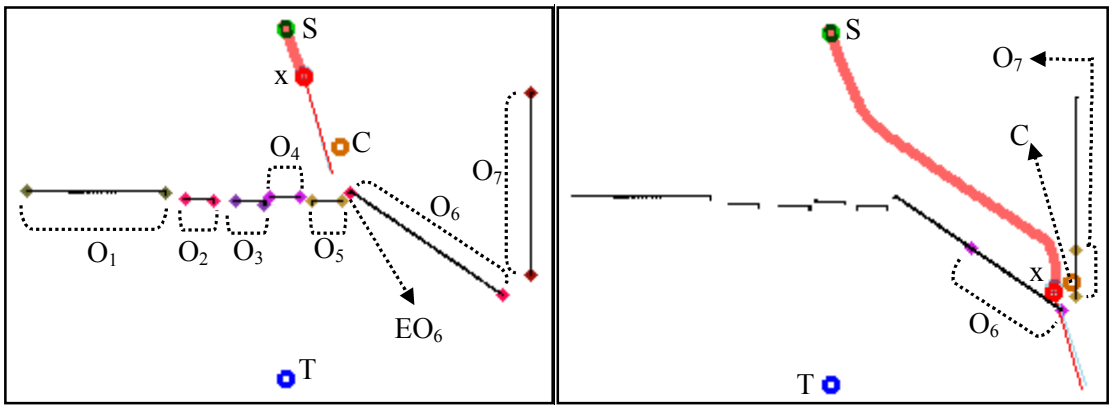


شکل ۵- ۲- نحوه‌ی رفتار روش ارائه شده در سناریوی شکل ۵-۱: همان طور که مشاهده می‌شود موانع O_2 و O_3 در فاز ادغام از ساخت گراف مماس محلی، به صورت یک مانع واحد در نظر گرفته شده‌اند. در نتیجه C با توجه به $EO_{2,3}$ محاسبه شده است.



شکل ۵- ۳-ربات با استفاده از روش ارائه شده به سلامت از نقطه‌ی S به نقطه‌ی T رسیده است.

حال به بررسی سناریوی دیگری برای مقایسه‌ی کارکرد تنژنت‌باگ و روش ارائه شده در مورد ساخت گراف مماس محلی می‌پردازیم. در شکل ۵-۴ سناریوی دوم نشان داده شده است. این بار نیز به علت عدم ادغام موانع نزدیک به هم، تنژنت‌باگ در هدایت ربات به سمت هدف با مشکل مواجه شده و ربات به مانع O_1 برخورد کرده است. اما در شکل ۵-۵ سناریوی دوم برای روش ارائه شده نشان داده شده است. همان طور که مشاهده می‌شود، ادغام موانع $O_1, \ldots, O_7$ باعث شده است تا ربات نقطه‌ی ایمن C را به درستی (بر اساس نقطه‌ی انتهایی $SO_{1,2,3,4,5,6,7}$) محاسبه کند و به موانع برخورد نداشته باشد. ادامه‌ی حرکت ربات از سناریوی دوم در شکل ۵-۶ نشان داده شده است که ربات با بهره‌گیری از روش ارائه شده به هدف رسیده است.



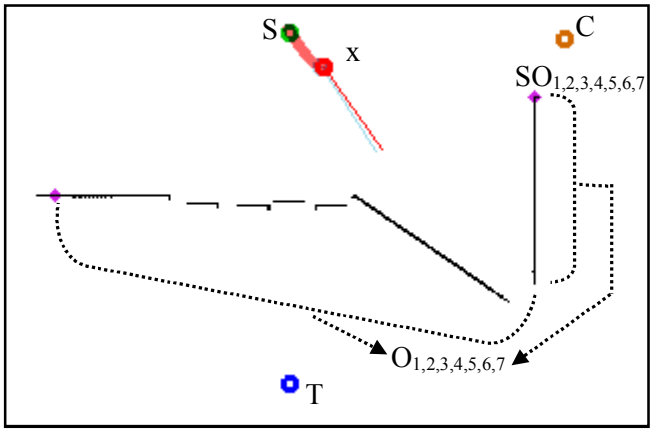
(a) (b)

شکل ۵- ۴- نحوه‌ی رفتار تنژنت‌باگ در سناریوی دوم از ساخت گراف مماس محلی: در قسمت a به علت نبود فاز ادغام موانع، نقطه C بر اساس EO_6 محاسبه شده است و در نهایت ربات در قسمت b به مانع O_6 برخورد کرده است.

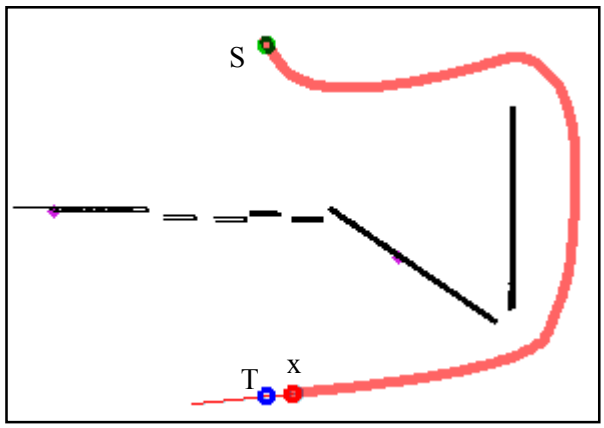
۵-۲-۲ مقایسه در رفتار حرکت-به-سوی-هدف

در این قسمت تنژنت‌باگ با روش ارائه شده از لحاظ عملکرد در رفتار حرکت-به-سوی-هدف مورد مقایسه قرار می‌گیرند. یادآوری می‌شود که مشکل تنژنت‌باگ در این رفتار، سوئیچ کردن بین انتخاب نقاط انتهایی مانع

مسدودکننده می‌باشد که در نهایت موجب برخورد ربات با مانع مذکور می‌گردد. علت این مشکل فاصله‌ی مکاشفه-ای تنژنت‌باگ می‌باشد که در روش ارائه شده مورد بازبینی قرار گرفت. در فصل چهارم یک سناریو برای نشان دادن نحوه‌ی مشکل‌ساز شدن فاصله‌ی مکاشفه‌ای تنژنت‌باگ ارائه شد. به همین دلیل در اینجا به ذکر یک نمونه‌ی دیگر اکتفا خواهیم کرد.



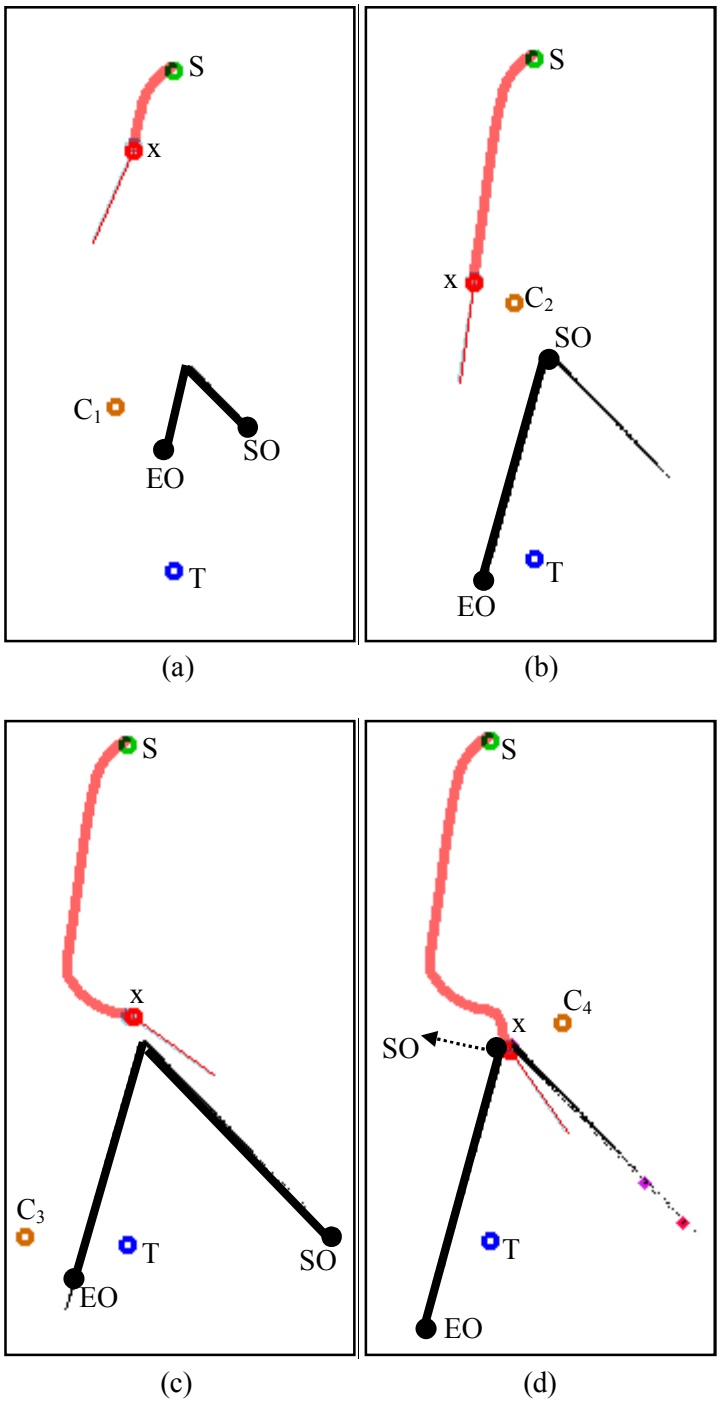
شکل ۵- ۵-روش ارائه شده، در سناریوی شکل ۵-۴ موانع $O_1, \ldots, O_7$ را با یکدیگر ادغام کرده است و نقطه‌ی C بر اساس نقطه‌ی $SO_{1,2,3,4,5,6,7}$ محاسبه شده است.



شکل ۵- ۶- ربات با استفاده از روش ارائه شده موفق شده است از نقطه‌ی S به نقطه‌ی T برسد.

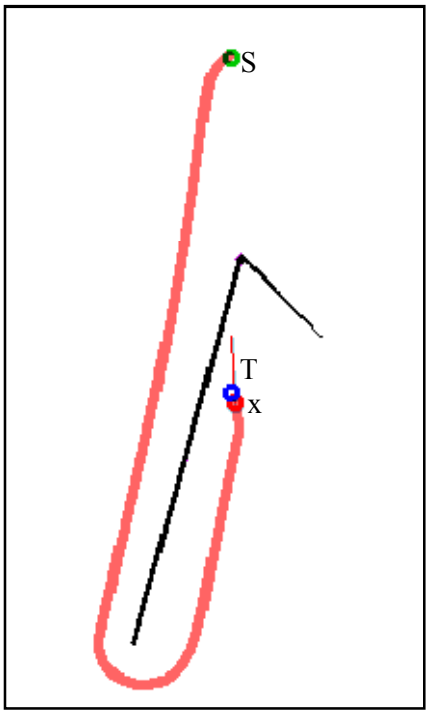
در شکل ۵-۷ نحوه‌ی انتخاب نقاط انتهایی برای دور زدن مانع مسدودکننده توسط تنژنت‌باگ نشان داده شده است. در قسمت a طبق معیار فاصله‌ی مکاشفه‌ای نقطه‌ی EO برای محاسبه‌ی نقطه‌ی C_1 انتخاب شده است. پس از مدتی حرکت به سمت C_1، قسمتی از مانع (از سمت SO) از دید ربات خارج می‌شود و در نتیجه فاصله‌ی مکاشفه‌ای مسیری که از نقطه‌ی SO می‌گذرد کم‌تر از فاصله‌ی مکاشفه‌ای مسیری که از EO می‌گذرد، می‌شود. در نتیجه نقطه‌ی SO برای محاسبه‌ی C_2 انتخاب می‌شود (قسمت b). با حرکت ربات به سمت C_2 دوباره قسمت خارج شده از دید ربات به داخل دید آن برمی‌گردد و باعث می‌شود که دوباره نقطه‌ی EO برای محاسبه‌ی نقطه‌ی C_3 انتخاب شود (قسمت c). در قسمت d نیز ربات در نهایت به مانع برخورد کرده است که شکست تنژنت‌باگ در دور زدن مانع

مسدودکننده را نشان می‌دهد.



شکل ۵-۷ -سناریوی شکست تنژنت‌باگ در رفتار حرکت-به-سوی-هدف

در شکل ۵-۸ خروجی حاصل از رفتار حرکت-به-سوی-هدف با استفاده از فاصله‌ی مکاشفه‌ای اصلاح شده نشان داده شده است. همان طور که مشاهده می‌شود، ربات بدون سوئیچ کردن بین نقاط انتهایی مانع مسدودکننده توانسته است به عبور از مانع و رسیدن به هدف دست یابد.



شکل ۵- ۸-استفاده از فاصله‌ی مکاشفه‌ای اصلاح شده موجب پایداری رفتار ربات حین حرکت به سوی هدف می‌شود.

۵-۲-۳ مقایسه در رفتار دنبال-کردن-مرز-مانع

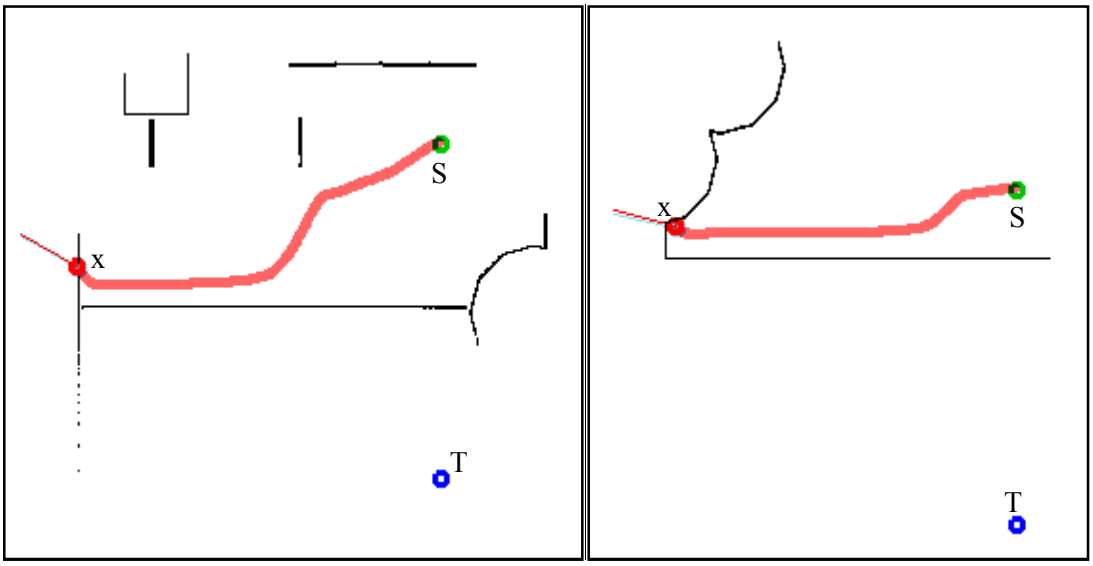
همان طور که در فصل چهارم شرح داده شد، نقص تنژنت‌باگ در دنبال کردن مرز مانع، حرکت کردن در جهت عمود بر بردار نرمال مرز آن می‌باشد که باعث برخورد ربات با مانع در برخی از موارد می‌شود. در این قسمت دو سناریو برای مقایسه‌ی عملکرد خروجی تنژنت‌باگ و روش ارائه شده در رفتار دنبال-کردن-مرز-مانع بررسی می-گردد. در قسمت a از شکل ۵-۹ سناریوی اوّل و در قسمت b سناریوی دوم آورده شده است. همان طور که مشاهده می‌شود، در هر دو سناریو، تنژنت‌باگ در حین دنبال کردن مرز مانع به آن برخورد کرده است. در شکل ۵-۱۰ سناریوی قسمت a از شکل ۵-۹ آورده شده است اما این بار روش ارائه شده هدایت ربات را به عهده دارد. همان طور که مشاهده می‌شود روش ارائه شده توانسته است با محاسبه‌ی نقطه‌ی ایمن C بر اساس نقطه‌ی انتهایی مانعی که باید دنبال شود، به موقع برای جلوگیری از برخورد ربات با مانع اقدام کند.

شکل ۵-۱۱ سناریوی قسمت b از شکل ۵-۹ را برای روش ارائه شده نشان می‌دهد. این بار نیز ربات با تکیه بر روش دنبال کردن مرز مانع اصلاح شده، با موفقیت مرز مانع را بدون برخورد با آن دنبال کرده و به هدف رسیده است.

۵-۲-٤ مقایسه‌ی سیستم دارای فاز پس‌پردازش با سیستم فاقد آن

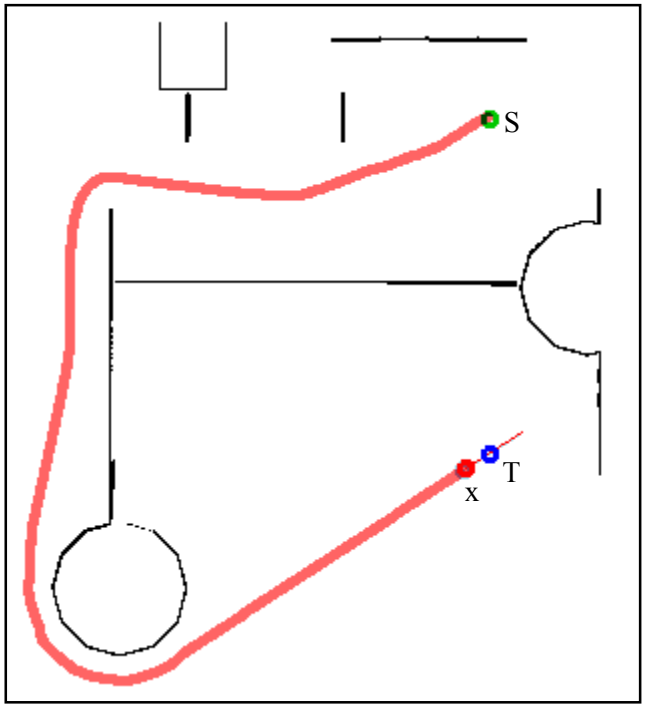
حال به بررسی تأثیر فاز پس‌پردازش تصمیمات تنژنت‌باگ می‌پردازیم. یادآوری می‌شود که علت افزودن فاز پس‌پردازش به سیستم ارائه شده، در نظر نگرفتن محدودیت‌های غیر-هولونومیکی ربات آکرمن توسط الگوریتم

تنژنت‌باگ می‌باشد. همان طور که در فصل چهارم نیز اشاره شد، برای رفع این نقص، باید نقطه‌ی ایمن محاسبه شده توسط تنژنت‌باگ را مورد پس‌پردازش قرار داد. که فاز اوّل پس‌پردازش استفاده از روش میدان پتانسیل می‌باشد و فاز دوم شبیه به روش VFH+ عمل می‌کند. علت اضافه کردن فاز دوم این است که روش میدان پتانسیل به تنهایی برای جبران عدم توجه تنژنت‌باگ به محدودیت‌های غیر-هولونومیکی ربات کافی نیست. توجه شود که نقش فاز دوم در حفظ ایمنی ربات بیشتر از فاز اوّل می‌باشد.

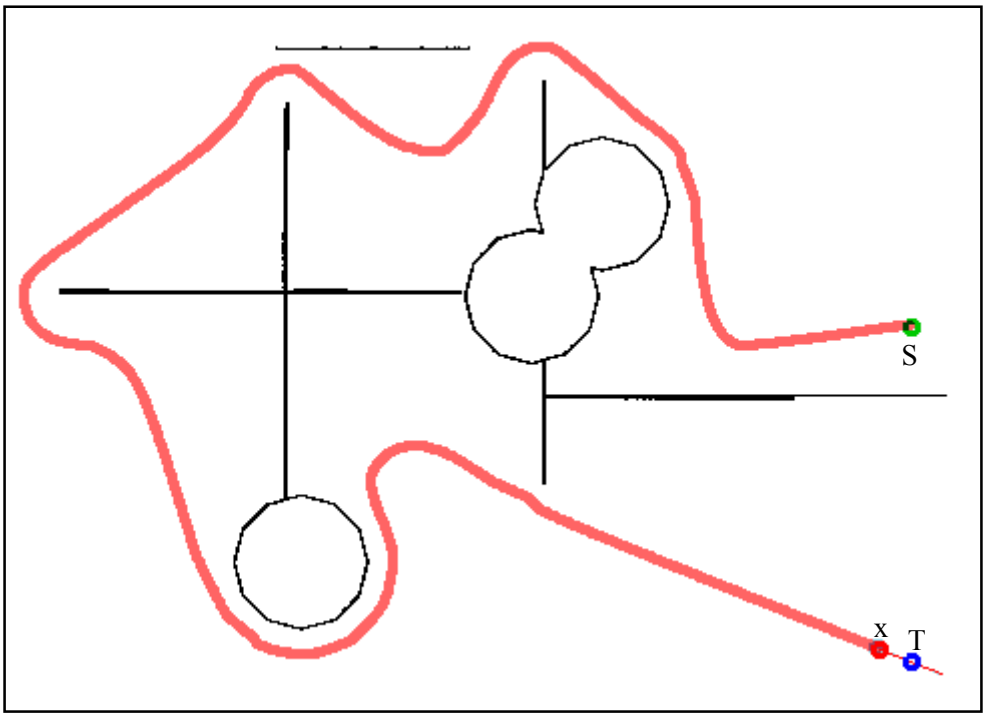


(a) (b)

شکل ۵- ۹-تنژنت‌باگ در هر دو سناریوی a و b حین دنبال کردن مرز مانع به آن برخورد کرده است.



شکل ۵- ۱۰-روش ارائه شده با بهره‌گیری از روش دنبال کردن مرز مانع اصلاح شده توانسته است در سناریوی قسمت a از شکل ۵-۹ ربات را بدون برخورد به مانع به هدف برساند.

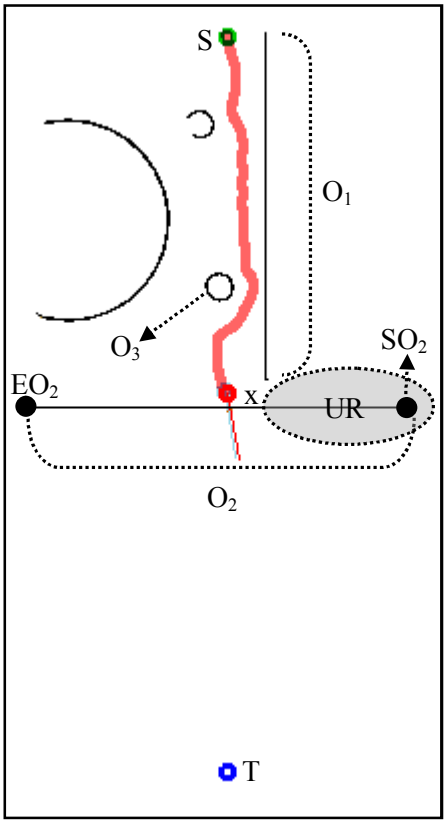


شکل ۵- ۱۱-روش ارائه شده با بهره‌گیری از متد دنبال کردن مرز مانع اصلاح شده توانسته است در سناریوی قسمت b از شکل ۵-۹ ربات را بدون برخورد به مانع به هدف برساند.

در شکل ۵-۱۲ نمونه‌ای از سناریوهایی که تنژنت‌باگ در آن‌ها به علت عدم توجه به محدودیت‌های حرکتی ربات آکرمن شکست می‌خورد، نشان داده شده است. در شکل مذکور ربات در ابتدا شروع به حرکت می‌کند تا این که مانع O_3 را بر سر راه خود می‌بیند و آن را دور می‌زند. سپس مانع O_2 بر سر راهش قرار می‌گیرد اما چون سمت چپ ربات مانع O_1 قرار دارد، بخشی (به نام UR) از مانع O_2 مشاهده نمی‌شود و ربات تقریباً به موازات مانع O_1 حرکت می‌کند تا زمانی که بخش پنهان مانع O_2 برای ربات آشکار می‌گردد. اما در این زمان فضای کافی برای پیچیدن ربات و حرکت به سمت نقطه‌ی SO_2 وجود ندارد و ربات به مانع O_2 برخورد می‌کند. توجه شود که اگر ربات قادر به اجرای هر مانور دلخواه بود، برخورد با مانع O_2 صورت نمی‌گرفت اما ربات مورد نظر از نوع آکرمن است و توان اجرای هر حرکتی را ندارد. در ادامه تأثیر فاز پس‌پردازش روش ارائه شده بر روی سناریوی شکل ۵-۱۲ مورد بررسی قرار می‌گیرد.

در شکل ۵-۱۳ دوباره سناریوی شکل ۵-۱۲ تکرار شده است اما این بار روش ارائه شده کنترل ربات را به دست دارد. همان طور که در قسمت a از شکل ۵-۱۳ مشاهده می‌شود، تنژنت‌باگ نقطه‌ی C را با توجه به مانع مسدودکننده‌ی O_4 محاسبه کرده است. توجه شود که در این سناریو هیچ یک از موانع، در دایره به شعاع ρ_0 (دایره‌ای که موانع موجود در آن بر ربات نیروی دافعه وارد می‌کنند) قرار ندارند به همین دلیل فاز اوّل پس‌پردازش (میدان پتانسیل) روی خروجی تنژنت‌باگ تأثیری ندارد. لازم به ذکر است که با حذف کمان‌های غیرمجاز در فاز دوم پس-

پردازش، کم‌تر شرایطی پیش می‌آید که ربات به قدری به موانع نزدیک شود که آن‌ها در شعاع ρ_0 قرار گیرند. در فاز دوم پس‌پردازش با توجه به موانع متورم شده کمان‌های مجاز مشخص می‌گردند. از آنجایی که هیچ یک از کمان-های مجاز از نقطه‌ی C عبور نمی‌کند، نزدیک‌ترین کمان مجاز (به کمان غیر مجازی که از C عبور می‌کند) انتخاب شده است. کمان منتخب در قسمت a از شکل ۵-۱۳ A نام‌گذاری شده است.



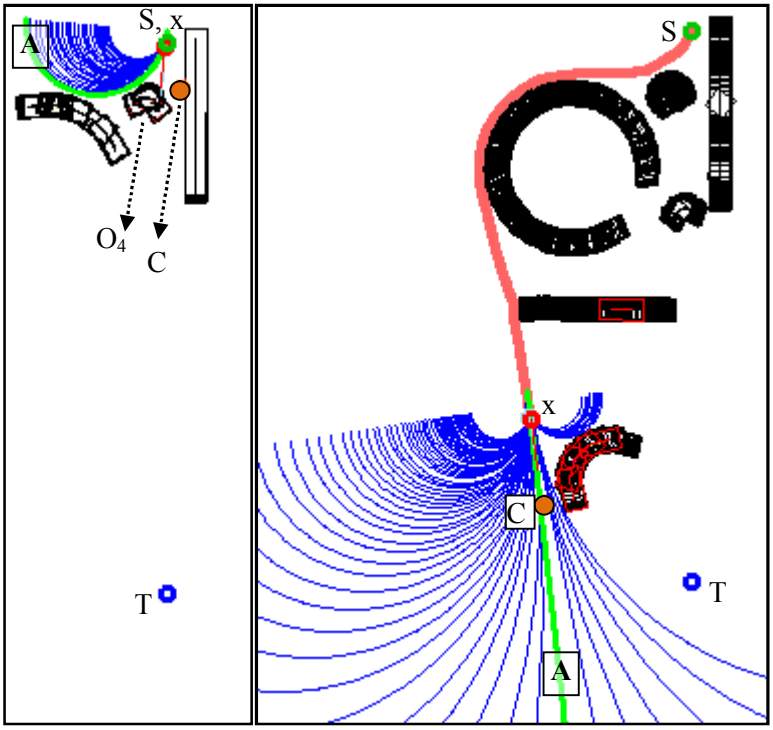
شکل ۵- ۱۲-ربات به علت وجود مانع O_1 نتوانسته است بخشی از مانع O_2 به نام UR که در سمت چپش قرار دارد را مشاهده کند و زمانی قادر به مشاهده‌ی آن شده است که فرصت کافی برای دور زدن و حرکت به سوی SO_2 ندارد و به O_2 برخورد کرده است.

ادامه‌ی سناریوی قسمت a از شکل ۵-۱۳ در قسمت b از شکل مذکور نشان داده شده است. همان طور که مشاهده می‌شود نقطه‌ی C در جایی قرار دارد که کمان مجاز A از آن عبور می‌کند. این نشان می‌دهد که هرگاه تصمیم گرفته شده توسط تنژنت‌باگ قابل اجرا باشد، فاز دوم پس‌پردازش به آن احترام می‌گذارد و آن را تغییر نمی-دهد. در نهایت در شکل ۵-۱۴ ربات را مشاهده می‌کنید که با بهره‌گیری از روش ارائه شده به هدف (T) رسیده است.

۵-۳ پیاده‌سازی عملی

پیاده‌سازی عملی سیستم مذکور بر روی رباتی با ساختار آکرمن مجهز به حسگرهای انکودر، IMU و پویش‌گر لیزری انجام شده است. در قسمت عملی پایان‌نامه برای تأمین توان پردازشی ربات مذکور از یک دستگاه

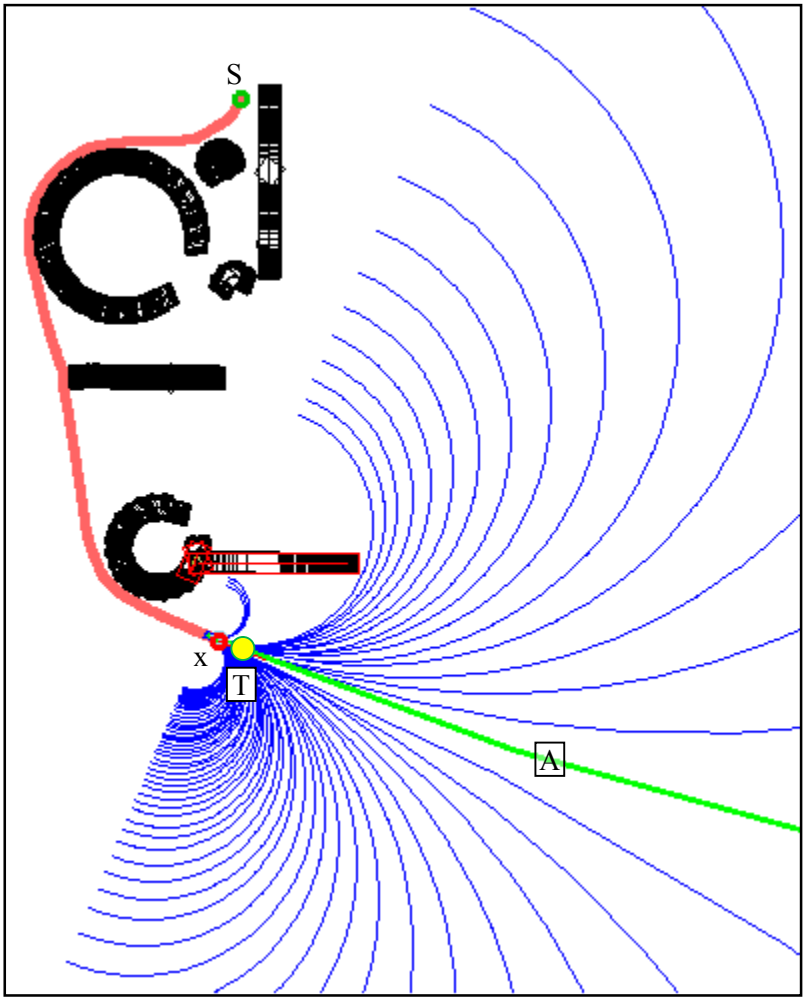
لپ‌تاپ با سیتم عامل لینوکس توزیع اوبونتو ۱۱.۱۰ بهره گرفته می‌شود. کدنویسی انجام شده برای ربات، به زبان C و با استفاده از محیط توسعه‌ی یکپارچه‌ی Eclipse می‌باشد. پلتفورم این ربات خودرویی الکتریکی است که با اعمال تغییراتی در آن، امکان نصب حسگرهای انکودر، IMU و پویش‌گر لیزری روی آن فراهم شده است. آماده‌سازی این ربات توسط تیم رباتیک آزمایشگاه هوش مصنوعی دانشکده کامپیوتر دانشگاه صنعتی اصفهان انجام شده است و راه‌اندازی حسگرهای آن بخشی از قسمت عملی این پایان‌نامه را تشکیل می‌دهد.



شکل ۵- ۱۳-نحوه‌ی رفتار روش ارائه شده در برابر سناریوی شکل ۵-۱۲: در قسمت a نقطه‌ی C توسط تئزنت‌باگ محاسبه شده است اما کمانی که از C عبور کند وجود ندارد به همین دلیل، نزدیک‌ترین کمان (به نام A) به C انتخاب شده است. در قسمت b کمانی که از C عبور کند موجود است که برای حرکت ربات انتخاب شده است.

برای بررسی صحت کارائی روش ارائه شده در این پایان‌نامه، با بهره‌گیری از ربات مذکور تست‌های عملی بسیاری انجام شده است که در شکل ۵-۱۵ نقشه‌ی ترسیم شده حین اجرای برخی از آن‌ها نشان داده شده است. در قسمت a از شکل مذکور ربات در حال انجام رفتار حرکت-به-سوی-هدف، با دو مانع به نام‌های O_1 و O_2 برخورد کرده است که با موفقیت از آن‌ها عبور کرده است. دو مانع مذکور، دو تیرک می‌باشند که علی‌رغم باریک بودنشان از دید ربات مخفی نمانده‌اند. طول مسیر طی شده توسط ربات در این تست، حدود ۲۲ متر می‌باشد. در قسمت b از شکل ۵-۱۵ ربات مسیری به طول حدوداً ۶۰ متر را طی کرده است. در این تست نیز مانند تست قسمت a، بررسی صحت رفتار حرکت-به-سوی-هدف مد نظر بوده است. این بار دو مانعی که بر سر راه ربات قرار داده شده‌اند، دو خودروی سواری می‌باشند که در مقایسه با موانع O_1 و O_2 از قسمت a، اندازه‌ی بزرگ‌تری دارند. همان طور که

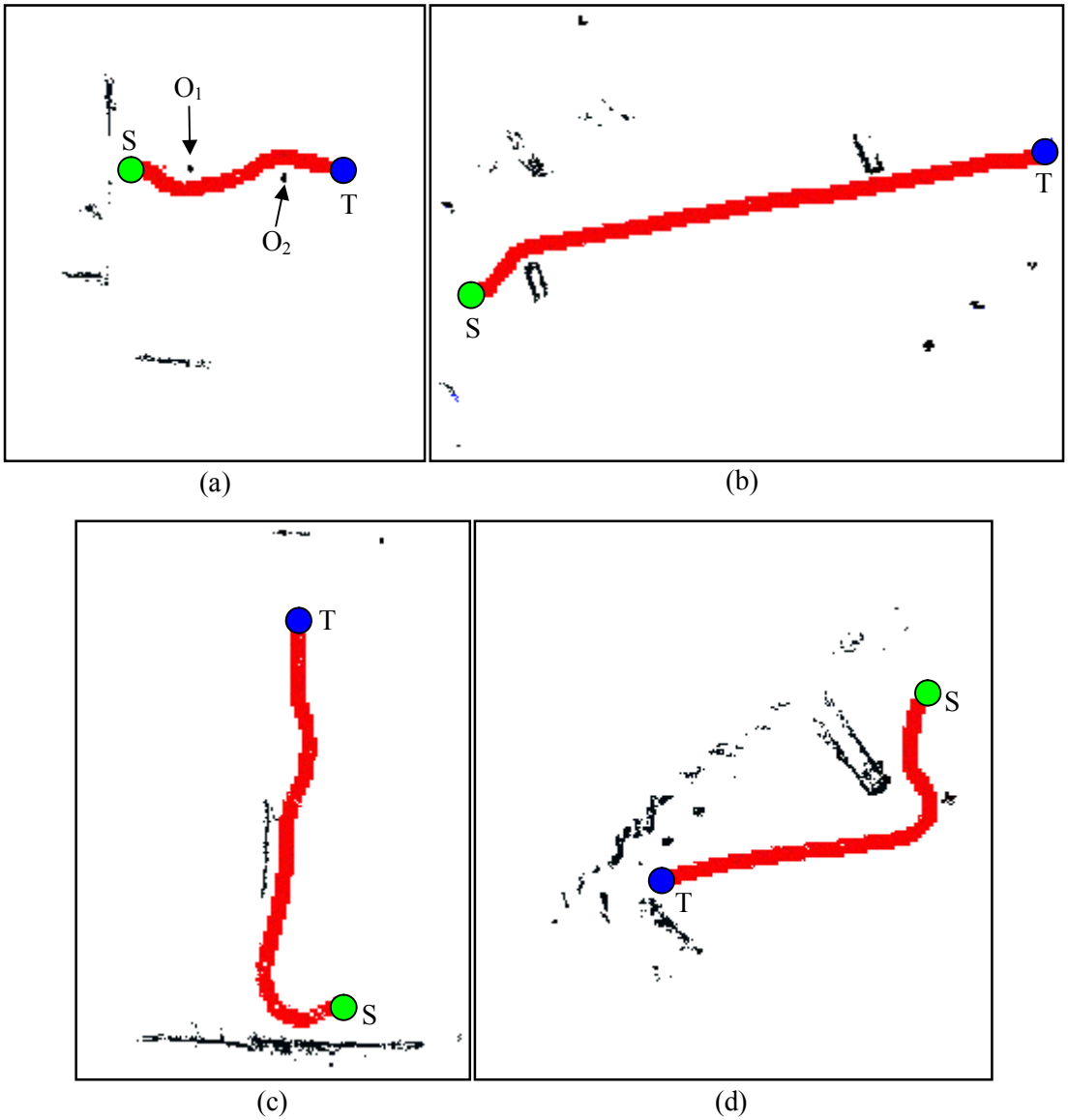
مشاهده می‌شود ربات توانسته است این موانع را نیز پشت سر بگذارد. قسمت c از شکل ۵-۱۵ نیز یکی دیگر از تست-های گرفته شده می‌باشد. در قسمت d ربات با مانعی بزرگ (یک دستگاه اتوبوس) مواجه شده است اما همان طور که مشاهده می‌شود با موفقیت آن را دور زده است.



شکل ۵- ۱۴-فاز پس‌پردازش روش ارائه شده، با توجه به موانع متورم شده و محدودیت‌های حرکتی ربات، آن را از مسیری ایمن به هدف رسانده است.

مکان تست‌هایی که در قسمت‌های a، b، c و d از شکل ۵-۱۵ ارائه شده است، پارکینگ دانشگاه می‌باشد. در قسمت e از شکل ۵-۱۵ رفتار دنبال کردن مرز مانع به آزمایش گذاشته شده است. برای گرفتن این تست، ربات به یک زمین تنیس انتقال داده شد. زیرا که دور زمین مذکور دیوارهایی وجود داشت که برای پویش‌گر لیزری قابل رویت بود. همان طور که در شکل مشاهده می‌شود، نقطه‌ی هدف خارج از محوطه‌ی قابل دسترس ربات داده شده است و در نتیجه ربات یک دور کامل مرز دیوار زمین تنیس را دنبال کرده و به نقطه‌ی شروع بازگشته است. از آنجا که ربات نتوانسته راه خروجی پیدا کند و یک دور کامل زده است، با پیغام "هدف غیرقابل دسترس می‌باشد"، متوقف شده است. لازم به ذکر است که به علت خطای مکان‌یابی، زمانی که ربات فکر می‌کند به نقطه‌ی شروع رسیده است، در واقع حدود ۴ متر با آن اختلاف دارد. این خطا در مکان‌یابی موجب شده است که دیوارهای W1 و

W2 بر روی یکدیگر منطبق نشوند علی‌رغم این که مربوط به یک مانع در محیط واقعی می‌باشند.



شکل ۵- ۱۵- نقشه‌های ترسیم شده با استفاده از داده‌های پویش‌گر لیزری در تست‌های عملی که با ربات انجام شده است.

٤-٥ نتیجه‌گیری

در این فصل نتایج حاصل از شبیه‌سازی و پیاده‌سازی عملی روش ارائه شده بررسی گردید و تأثیر فازهای پس‌پردازش خروجی الگوریتم تنژنت‌باگ مورد تحلیل قرار گرفت. نتایج عملی به دست آمده بر روی یک ربات آکرمن، صحت کارایی روش ارائه شده را در محیطی واقعی نشان می‌دهد. مشاهده شد که خطای روش مکان‌یابی مورد استفاده در مسیری به طول ۲۲۰ متر حدود ۴ متر می‌باشد. لازم به یادآوری است که خطای مذکور افزایشی می‌باشد و با طولانی‌تر شدن مسیر، افزایش پیدا خواهد کرد. که در کارهای آینده باید برای بهبود آن تدبیری اندیشیده شود.

فصل ششم

نتیجه‌گیری و جمع‌بندی

٦-١ مقدمه

در این پایان‌نامه مسئله‌ی ناوبری در محیط ناشناخته‌ی خارج از جاده برای یک ربات آکرمن مورد بررسی قرار گرفت. پایه‌ی سیستم ارائه شده، الگوریتم تنژنت‌باگ بود که محدودیت‌های حرکتی و اندازه‌ی ربات را در تصمیم‌گیری‌های خود در نظر نمی‌گیرد. در این پایان‌نامه سعی شد تا با بازبینی و انجام اصلاحاتی، تنژنت‌باگ را برای هدایت ربات با معماری آکرمن قابل استفاده کرد. در این راستا از مفاهیم روش‌های میدان پتانسیل و VFH+ استفاده شد. با شبیه‌سازی در نرم‌افزار Webots صحت روش ارائه شده مورد بررسی قرار گرفت و در نهایت کارایی روش ارائه شده در سناریوهای عملی نیز محک زده شد.

٦-٢ نقاط قوت روش ارائه شده

از جمله نقاط قوت روش ارائه شده می‌توان به مواردی از قبیل: ١- در نظر گرفتن اندازه‌ی ربات برای بررسی امکان عبور از بین موانع ٢-پایداری بیشتر در حین رفتار حرکت-به-سوی-هدف ٣- امکان‌سنجی مسیر تعیین شده توسط تنژنت‌باگ و اصلاح آن در صورت نیاز ۴-بازبینی رفتار دنبال-کردن-مرز-مانع اشاره کرد. مورد اوّل با اضافه کردن فاز ادغام از به رویه‌ی ساخت گراف مماس محلی انجام شده است. مورد دوم با اصلاح فاصله‌ی مکاشفه‌ای الگوریتم تنژنت‌باگ و مورد سوم در فاز پس‌پردازش تصمیمات تنژنت‌باگ محقق شده است. یادآوری می‌شودکه

پس‌پردازش تصمیمات تنژنت‌باگ زمانی ضروری است که ربات دارای محدودیت‌های غیر-هولونومیکی (مانند ربات آکرمن) باشد و در هر لحظه توانایی حرکت در هر جهت دلخواه را نداشته باشد.

٦-٣ کاستی‌های روش ارائه شده

یادآوری می‌شود که دقت مکان‌یابی برای موفقیت ربات در رسیدن به هدف بسیار حائز اهمّیت می‌باشد. به این ترتیب مهم‌ترین نقطه ضعف روش ارائه شده مکانیزم مکان‌یابی آن می‌باشد. اگر به خاطر داشته باشید در فصل چهارم ذکر شد که مکانیزم مکان‌یابی مورد استفاده در این پایان‌نامه صرفاً بر اساس dead-reckoning و با استفاده از حسگرهای انکودر و IMU می‌باشد. استفاده از dead-reckoning محض برای مکان‌یابی، موجب می‌شود تا به مرور باور ربات در مورد مکانش، از موقعیت حقیقی آن فاصله بگیرد. این اختلاف بین مکان حقیقی ربات با باور آن از مکانش مطلوب نمی‌باشد زیرا که اگر ربات نداند کجا قرار دارد، نمی‌تواند به خوبی ناوبری انجام دهد و به هدف برسد. البته روش dead-reckoning در مسیرهای نسبتاً کوچک (زیر ۲۰۰متر) مشکل جدی ایجاد نخواهد کرد به خصوص که تنژنت‌باگ الگوریتمی حسگر-مبنا می‌باشد و به صورت محلی تصمیم می‌گیرد. لازم به یادآوری است که انتخاب dead-reckoning به عنوان مکانیزم مکان‌یابی دلایلی داشته است که در ادامه توضیح داده می‌شوند.

٦-٣-١ عدم استفاده از GPS

در این پایان‌نامه هدف استفاده از GPS برای اصلاح باور ربات در مورد مکانش نبوده است. به علاوه باید توجه شود که دقت حسگر مذکور در مکان‌هایی که تعداد ماهواره‌ی کافی مشاهده نمی‌کند به شدت کاهش پیدا می‌کند و در ضمن به طور کل دقت این حسگر بهتر از ۷ تا ۸ متر نمی‌باشد که برای مکان‌یابی ربات خیلی مناسب نیست. البته دقت DGPS در حد چند سانتی‌متر می‌باشد ولی برای استفاده از آن همواره باید داده‌های DGPS با توجه به یک ایستگاه مرجع روی کره‌ی زمین اصلاح شوند. که فاصله‌ی ایستگاه مذکور تا ربات، نمی‌تواند از یک حد مشخص بیشتر باشد.

٦-٣-٢ عدم اطمینان از چگال بودن موانع قابل رویت محیط

ابتدا منظور از چگال بودن موانع و قابل رویت بودن را بیان می‌کنیم. منظور از چگال بودن موانع محیط این است که بخش قابل توجهی از ۱۰۸۰ پرتوی پویش‌گر لیزری به موانع برخورد کند. به عنوان مثال محیط‌های درون‌در، جزء محیط‌های چگال محسوب می‌شوند زیرا که اکثر پرتوهای پویش‌گر به اشیاء و دیوارهای موجود در محیط برخورد می‌کند؛ اما یک محیط برون‌در با سطحی تقریباً صاف (مانند دشت) از نظر وجود موانع، چگالی پایینی دارد.

یادآوری می‌شود که پویش‌گر لیزری مورد استفاده در این پایان‌نامه از نوع دو بعدی می‌باشد که به صورت موازی با زمین روی ربات نصب شده است. با توجه به این نکته، منظور از قابل رویت بودن موانع این است که ارتفاع

آن‌ها از سطح زمین در حدی باشد که پرتوهای پویش‌گر لیزری به آن‌ها برخورد کند.

اگر محیطی که ربات در آن مشغول ناوبری است از نظر وجود موانع قابل رویت چگال باشد، یکی از روش-های بهبود مکان‌یابی تطبیق الگوی موجود در داده‌های پویش‌گر لیزری در گام فعلی با الگوی داده‌های پویش‌گر لیزری در گام قبلی می‌باشد [۵۷]. این روش سعی می‌کند ماتریس انتقال و دورانی محاسبه کند که اگر بر موقعیت ربات در زمان t-1 اعمال شود، موقعیت ربات در زمان t را با دقت مناسب تخمین بزند. این روش زمانی نتایج قابل قبول تولید می‌کند که به اندازه‌ی کافی در محیط، مانع موجود باشد تا فرآیند تطبیق الگوی داده‌های پویش‌گر به خوبی صورت گیرد. حال باید توجه شود که محیط مورد نظر این پایان‌نامه لزوماً از نظر وجود موانع، چگال نمی‌باشد و به همین دلیل در این پایان‌نامه از این روش برای بهبود مکان‌یابی استفاده نشده است. لازم به یادآوری است که اگر استفاده از حسگر GPS بلامانع باشد، می‌توان از روش مذکور برای تخمین مکان ربات در قسمت‌هایی که موانع زیادی وجود دارد استفاده کرد و در جاهایی که موانع به اندازه‌ی کافی موجود نیست، از حسگر GPS برای مکان‌یابی بهره برد.

٦-٤ پیشنهادات برای کارهای آینده

مهم‌ترین پیشنهادهایی که برای ادامه‌ی کار بر روی موضوع این پایان‌نامه می‌توان ذکر کرد، در زیر آورده شده است:

۱- بهبود روش مکان‌یابی: برای این کار می‌توان از تکنیک‌های بینایی ماشین برای تخمین بهتر مکان ربات استفاده کرد. در این زمینه می‌توان از ایده‌ای که در [۵۸] ارائه شده است، کمک گرفت. به علاوه می‌توان بر روی فرمان ربات انکودری نصب کرد و با استفاده از آن تخمینی از زاویه فرمان به دست آورد و به این ترتیب روابط سینماتیکی با توجه به زاویه فرمان برای تخمین مکان ربات در نظر گرفت تا دقت dead-reckoning بهبود داده شود.

۲- بهبود قابلیت ربات در درک موانع موجود در محیط: از آنجا که لزوماً تمام موانع موجود در محیط دارای ارتفاع کافی برای قرار گرفتن در دید پویش‌گر لیزری نصب شده (به صورت موازی با زمین) بر روی ربات نیستند، برای بهبود درک موانع باید از پویش‌گر لیزری سه بعدی و یا پویش‌گر لیزری دو بعدی که به صورت مایل به سمت زمین نصب شده است، استفاده کرد. تا به این ترتیب ناهمواری‌های زمین مانند درّه‌ها که ممکن است برای ربات مخاطره انگیز باشند تشخیص داده شود.

۳- قابلیت درک و اجتناب از موانع متحرک: از آنجا که سیستم ناوبری ارائه شده در این پایان‌نامه بر پایه‌ی تئزنت‌باگ می‌باشد، قابلیت درک و اجتناب از موانع متحرک را ندارد که می‌توان در آینده روی روشی

برای تشخیص و اجتناب از موانع متحرک تحقیقاتی انجام داد.

۴- حرکت ربات با سرعت متغیر: در این پایان‌نامه فرض شده است که سرعت ربات همواره ثابت می‌باشد در حالی که می‌توان سرعت ربات را با توجه به میزان فضای آزادی که در پیرامون ربات قرار دارد، به صورت هوشمندانه تغییر داد. برای این کار ربات می‌تواند در جاهایی که فضای آزاد زیادی وجود دارد با سرعت بیشتر و در جاهایی که فضای آزاد کم‌تری موجود است محتاطانه‌تر حرکت کند و به این ترتیب زمان کم-تری برای رسیدن به هدف صرف شود.

۵- اصلاح فاصله مکاشفه‌ای ارائه شده: رابطه‌ای که در این پایان‌نامه برای فاصله‌ی مکاشفه‌ای رفتار حرکت-به-سوی-هدف ارائه شد، در برخی موارد موجب افزایش طول مسیر طی شده می‌شود. این مسئله را می‌توان با ترکیب فاصله‌ی مکاشفه‌ای ارائه شده و فاصله‌ی مکاشفه‌ای تنژنت‌باگ رفع کرد. به این صورت که در حالت عادی ربات از فاصله‌ی مکاشفه‌ای تنژنت‌باگ برای دور زدن مانع مسدودکننده استفاده کند و هر گاه سوئیچ کردن بین نقاط انتهایی مانع مسدودکننده شروع شد، از فاصله‌ی مکاشفه‌ای ارائه شده بهره گیرد. به این ترتیب هر کجا ممکن باشد، با استفاده از فاصله‌ی مکاشفه‌ای تنژنت‌باگ، مسیر بهینه‌ی محلی انتخاب می‌شود و هر کجا خطر سوئیچ کردن بین نقاط انتهایی مانع وجود داشته باشد، از فاصله مکاشفه‌ای ارائه شده برای تأمین ایمنی ربات استفاده می‌شود.

مراجع

[1] Hart, P. E., Nilsson, N. J., Raphael, B., "A formal basis for the heuristic determination of minimum cost paths", *IEEE Transactions on Systems Science and Cybernetics,* Vol. 4, No. 2, pp. 100-107, July 1968.

[2] Stentz, A., "Optimal and efficient path planning for partially-known environments", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 3310-3317, San Diego, CA, May 1994.

[3] Laubach, S. L., Burdick, J. W., "An autonomous sensor-based path planner for planetary Microrovers", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 347-354, Detroit, MI, May 1999.

[4] Kamon, I., Rimon, E., Rivlin, E., "TangentBug: a range-sensor-based navigation algorithm", *International Journal of Robotics Research,* Vol. 17, No. 9, pp. 934-953, September 1998.

[5] Holland, O., "*The Grey Walter online archive",* [online] http://www.ias.uwe.ac.uk/Robots/gwonline/gwonline.html, (December 13th 2013).

[6] "*The Hopkins Beast",* [online] http://www.frc.ri.cmu.edu/~hpm/talks/revo.slides/1960.html, (December 13th 2013).

[7] Wilber, B. M., A Shakey primer, Stanford Research Institute, Stanford University, Technical Report, Menlo Park, CA, 1972.

[8] Earnest, L., "*Stanford Cart",* [online] 2012, http://www.stanford.edu/~learnest/cart.htm, (December 13th 2013).

[9] "*Luna exploring the moon",* [online] http://www.zarya.info/Diaries/Luna/Luna17.php, (December 13th 2013).

[10] Hess, S. L., Henry, R. M., Leovy, C. B., Ryan, J. A., Tillman, J. E., "Meteorological results from the surface of Mars: Viking 1 and 2", *Journal of Geophysical Research,* Vol. 82, No. 28, pp. 4559-4574, September 1977.

[11] "*Prof. Schmidhuber's highlights of robot car history",* [online] http://www.idsia.ch/~juergen/robotcars.html, (December 13th 2013).

[12] Keirsey, D. M., Mitchell, J. S., Payton, D. W., Tseng, D. Y., Wong, V. S., Autonomous Land Vehicle (ALV) planning and navigation system, Hughes Research Labs, Malibu, CA, Technical Report No. 1, 1986.

[13] Nock, L., *The robot the life story of a technology*. Greenwood Press, USA, 2007.

[14] "*Khepera II",* [online] http://www.k-team.com/mobile-robotics-products/khepera-ii, (December 13th 2013).

[15] Krotkov, E., Bares, J., Katragadda, L., Simmons, R., W.L., W., "Lunar rover technology demonstrations with Dante and Ratler", Proceedings of the *International Symposium on Artificial Intelligence, Robotics, and Automation for Space*, Vol. pp. 113-116, October 1994.

[16] Bares, J., Wettergreen, D., "Dante II: technical description, results and lessons learned", *International Journal of Robotics Research,* Vol. 18, No. 7, pp. 621-649, July 1999.

[17] Dickmanns, E. D., *Dynamic Vision for Perception and Control of Motion*. Springer, London, England, 2007.

[18] "*The Mercedes-Benz VITA-2 twin prototype went by itself from Munich to Copenhagen",* [online] http://www.autoevolution.com/news-image/a-short-history-of-mercedes-benz-autonomous-driving-technology-68148-8.html, (December 13th 2013).

[19] Pomerleau, D., "Defense and civilian applications of the ALVINN robot driving system", presented at the Government Microcircuit Applications Conference, pp. 358-362, Sandiego, CA, November 1994.

[20] "*Autonomous research robot",* [online] http://en.wikipedia.org/wiki/Autonomous_research_robot, (December 13th 2013).

[21] Mishkin, A. H., Morrison, J. C., Nguyen, T. T., Stone, H. W., Cooper, B. K., Wilcox, B. H., "Experiences with operations and autonomy of the Mars pathfinder microrover",

presented at the IEEE Aerospace Conference, pp. 337-351, Snowmass at Aspen, CO,March 1998.

[22] Yamauchi, B., "PackBot: A Versatile Platform for Military Robotics", Proceedings of the *SPIE*, Vol. 5422, pp. 228-237, September 2004.

[23] "*Swarm-bots: Swarms of self-assembling artifacts",* [online] http://www.swarm-bots.org/, (December 13th 2013).

[24] "*DARPA Grand Challenge (2004)",* [online] http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2004) (December 13th 2013).

[25] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L., Koelen, C. H., Markey, C., Rummel, C., Niekerk, J. V., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P., "Stanley: the robot that won the DARPA grand challenge", *Journal of Field Robotics,* Vol. 23, No. 9, pp. 661-692, 2006.

[26] "*DARPA Grand Challenge (2007)",* [online] http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007), (December 13th 2013).

[27] "*MAGIC 2010: Super-smart robots wanted for international challenge",* [online] http://www.dsto.defence.gov.au/MAGIC2010/, (December 13th 2013).

[28] Daniel, K., Nash, A., Koenig, S., Felner, A., "Theta*: any-angle path planning on grids", *Journal of Artificial Intelligence Research,* Vol. 39, No. pp. 533-579, 2010.

[29] Simmons, R., "The curvature-velocity method for local obstacle avoidance", Proceedings of the *International Conference on Robotics and Automation*, Vol. 4, pp. 3375-3382, Minneapolis, MN, April 1996.

[30] Fox, D., Burgard, W., Thrun, S., "The dynamic window approach to collision avoidance", *IEEE Robotics & Automation Magazine,* Vol. 4, No. 1, pp. 23-33, March 1997.

[31] Brock, O., Khatib, O., "High-speed navigation using the global dynamic window approach", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 341 - 346, Detroit, MI, May 1999.

[32] Latombe, J. C., Barraquand, J., "Robot motion planning: a distributed presentation approach", *International Journal of Robotics Research,* Vol. 10, No. 6, pp. 628–649, December 1991.

[33] Schlegel, C., "Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot", Proceedings of the *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 594 - 599, Victoria, BC, October 1998.

[34] Minguez, J., Montano, L., "Nearness diagram navigation (ND): a new real time collision avoidance approach", Proceedings of the *IEEE/RSJ International Conference on Intelligent Robots and Systems* Vol. 3, pp. 2094 - 2100, Takamatsu, October 2000.

[35] Borenstein, J., Koren, Y., "The vector field histogram-fast obstacle avoidance for mobile robots", *IEEE Journal of Robotics and Automation,* Vol. 7, No. 3, pp. 278-288, June 1991.

[36] Minguez, J., Montano, L., Simeon, T., Alami, R., "Global nearness diagram navigation (GND)", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 33-39, 2001.

[37] Lumelski, V. J., Stepanov, A. A., "Dynamic path planning for a mobile automaton with limited information on the environment", *IEEE Transactions on Automatic Control,* Vol. AC-31, No. 11, pp. 1058-1063, 1986.

[38] Siegwart, R., Nourbakhsh, I. R., Scaramuzza, D., *Introduction to autonomous mobile robots*, second ed., MIT Press, London England, 2011.

[39] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S., *Principles of robot motion theory, algorithms, and implementation*. MIT Press, London, England, 2005.

[40] Antich, J., Ortiz, A., Minguez, J., Bug2+: details and formal proofs, University of the Balearic Islands, Technical Report A-1, 2009.

[41] Antich, J., Ortiz, A., Minguez, J., "A Bug-inspired algorithm for efficient anytime path planning", presented at the IEEE/RSJ International Conference on Intelligent Robots and

Systems, pp. 5407-5413, St. Louis, MO,October 2009.
[42] Gabriely, Y., Rimon, E., "CBUG: a quadratically competitive mobile robot navigation algorithm", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. pp. 2014-2019, 2005.
[43] Magid, E., Rivlin, E., "CautiousBug: a competitive algorithm for sensory-based robot navigation", Proceedings of the *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 2757-2762, Sendai, Japan, October 2004.
[44] Taylor, K., LaValle, S. M., "I-Bug: an intensity-based Bug algorithm", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. pp. 3981-3986, Kobe, May 2009.
[45] Kim, D. H., Lee, J. Y., Han, C., "Sensor-based motion planning for a car-like robot based on Bug family algorithms", *World Academy of Science, Engineering and Technology,* Vol. 6, No. 11, pp. 1540-1547, 2012.
[46] Conlter, R. C., Implementation of the pure pursuit path tracking algorithm, The Robotics Institute, Carnegie Mellon University, Technical Report, January 1992.
[47] Zhu, Y., Zhang, T., Song, J., Li, X., "A new bug-type navigation algorithm for mobile robots in unknown environments containing moving obstacles", *Industrial Robot: An International Journal, Emerald,* Vol. 39, No. 1, pp. 27-39, 2012.
[48] Liu, H., Arimoto, S., "Path planning using a tangent graph for mobile robots among polygonal and curved obstacles", *International Journal of Robotics Research, Sage,* Vol. 11, No. 4, pp. 376-382, August 1992.
[49] "*Star domain",* [online] http://en.wikipedia.org/wiki/Star_domain, (December 13th 2013).
[50] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots", *International Journal of Robotics Research, Sage,* Vol. 5, No. 1, pp. 90-98, March 1986.
[51] Borenstein, J., Koren, Y., "Real-time obstacle avoidance for fast mobile robots", *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 19, No. 5, pp. 1179-1187, September-October 1989.
[52] Elfes, A., "Sonar-based real-world mapping and navigation", *RA-3,* Vol. 3, No. 249-265, 1987.
[53] Moravec, H. P., Elfes, A., "High resolution maps from wide angle sonar", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 116-121, March 1985.
[54] Moravec, H. P., "Sensor fusion in certainty grids for mobile robots", *AI Magazine,* Vol. 9, No. 2, pp. 61-74, 1988.
[55] Ulrich, I., Borenstein, J., "VFH+: reliable obstacle avoidance for fast mobile robots", Proceedings of the *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1572-1577, Leuven, May 1998.
[56] Pavlidis, T., Horowitz, S. L., "Segmentation of plane curves", *IEEE Transactions on Computers,* Vol. C-23, No. 8, pp. 860-870, August 1974.
[57] Siddiqui, A., Hellström, T., Laser-based localization of vehicles and robots in natural and unstructured environments, Department of Computing Science, Umea University, Technical Report, Umea, Sweden, September 2005.
[58] Gonzaleza, R., Rodrigueza, F., Guzmana, J. L., Pradaliera, C., Siegwart, R., "Combined visual odometry and visual compass for off-road mobile robots localization", *Robotica,* Vol. 30, No. 6, pp. 865-878, 2012.

Autonomous Navigation in Unknown Off-road Environment based on Family of Bug Algorithms

Sayed Navid Hoseini Izadi

navid.hoseini@ec.iut.ac.ir

Date of Submission: 2014/1/13

Department of Electrical and Computer Engineering

Isfahan University of Technology, Isfahan 84156-83111, Iran

Degree: M.Sc. Language: Farsi

Supervisor: MaziarPalhang, palhang@cc.iut.ac.ir

Abstract

Mobile robots applications are growing everyday. Some of their applications include exploring unknown planets like Mars, finding woundeds in an earthquake and helping them and etc. The most important characteristic of mobile robots is their ability to navigate through the environment autonomously in order to reach the goal point.

The type of operational environment of a mobile robot can be divided to two broad classes namely 1.known and 2.unknown. If the environment is known in advance, the robot can make use of the environment map and a global path planning algorithm in order to find the optimal path to the goal. But knowing the environment in advance is not always possible. On the other hand, as the environment size gets bigger, the amount of memory and processing power needed to store and update (if necessary) its map grows as well. That is why sensor-based path planners are attractive. These planners assume no prior knowledge of the environment and rely solely on their sensory data to navigate toward the goal. Since the robot sensors range is limited, the sensor-based planners have to compute the path to the goal incrementally. To this end, at each time step the robot senses its surrounding environment locally. Using these sensory data, the sensor-based path planner determines the appropriate action such that the robot moves toward the goal while it avoids the environment obstacles. The nice property of sensor-based path planners is that they converge to the goal although the environment is not known. Bug algorithms are a famous group of sensor-based path planners since they have acceptable performance despite their simple structure.

The goal of this thesis is the modification of TangentBug algorithm so that it can be used on Ackerman mobile robots. The robot environment is assumed to be unknown and offroad. By offroad, we mean that no prior assumptions on the environment characteristic (like having roads or any guiding signs) are made. The proposed navigation method of this thesis is based on TangentBug algorithm (one of the most famous Bug algorithms). Despite its robustness, TangentBug has some shortcomings for example it assumes that the robot is a point (with zero width and length); also it ignores the maneuver limitations of the robot. In this thesis, the original TangentBug algorithm is modified so that it can be used as the navigation algorithm on a real car-like robot. To this end, potential field obstacle avoidance method is integrated with TangentBug and the kinematic equations of the car-like robot are taken into account by utilizing a method similar to VFH+ obstacle avoidance algorithm. To evaluate the proposed method, it is compared with the original TangentBug in simulated environment. In addition to the simulation, the proposed approach is implemented and tested on a real robot.

Keywords:

Autonomous navigation, Unknown environment, Obstacle avoidance, Sensor-based