دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

درس هوش مصنوعی و کارگاه

الگوریتم A-Star

نگارش

فاطمه گل محمدی

۴۰۱۱۳۴۲۶

استاد اول

دکتر مهدی قطعی

استاد دوم

بهنام یوسفی مهر

مهر ۱۴۰۳

چکیده

این مقاله الگوریتم جدیدی را برای مسیریابی وسایل نقلیه خودران زمینی بر اساس مقاله ای با همین نام توسط شنک ارک ارائه میدهد.

در این مقاله یک اســـتاندارد ارزیابی برای اندازه‌گیری عملکرد الگوریتم‌های مختلف و انتخاب پارامترهای مناسب برای الگوریتم پیشنهادی معرفی می‌شود. از یک راهنما استفاده می‌شود تا تابع هدایتی بهبود پیدا کند و نقاط ضـــعف الگوریتم A-Star را برطرف کند . همین طور برای بهبود عملکرد اجتناب از موانع، از نقاط کلیدی اطراف مانع استفاده می‌شود که باعث میشود سریع تر از الگوریتم سنتی عمل کند.

درنهایت الگوریتم های Greedy, UCS , A-Star , BFS , DFS تعریف شـــده و پیاده ســـازی آنها را بررسی کردیم.

واژه های کلیدی:

وسایل نقلیه خودران زمینی ، الگوریتم A-Star بهبود یافته ، مسیریابی ، نقاط کلیدی

فهرست مطالب

صفحه

چکیده ..... أ

۱. فصل اول مقدمه ۳

۱-۱- روش های برنامه ریزی مسیر ۴

۱-۲- الگوریتم ای - استار ۵

۲. فصل دوم بهبود های پیشنهادی ۷

۲-۱-معرفی استاندارد ارزیابی جهانی ۹

۲-۲- الگوریتم های بهبود یافته ۱۰

۳. فصل سوم تحلیل نتایج آزمایش‌های انجام شده ۱۳

۳-۱- آزمایش در محیط‌های شهری ۱۴

۳-۲-آزمایش در سناریوهای پیچیده و طولانی‌مدت ۱۵

۳-۳- مقایسه با الگوریتم‌های دیگر ۱۶

۴. فصل چهارم گزارش کار ۱۸

۵. فصل پنجم جمع‌بندی و نتیجه‌گیری ۲۲

منابع و مراجع ۲۴

۱. فصل اول

مقدمه

مقدمه

ابتدا راجب اهمیت وسایل نقلیه خودران زمینی صحبت میکنیم.

وسایل نقلیه خودران زمینی (Autonomous Land Vehicles یا ALVs) یکی از مهم‌ترین پیشرفت‌های فناوری در زمینه رباتیک و هوش مصنوعی اند. این وسایل نقلیه از مجموعه‌ای از سیستم‌های پیچیده تشکیل شده‌اند که در هماهنگی کامل با یکدیگر کار می‌کنند تا وسیله نقلیه بتواند بدون دخالت انسان حرکت کند. این وسیله متشکل است از بخش‌هایی مانند درک محیط، برنامه‌ریزی مسیر، کنترل وسیله نقلیه، موقعیت‌یابی و غیره. مهمترین بخش آن سیستم برنامه‌ریزی مسیر است، زیرا بر اساس اطلاعات محیط رادرک و عملیات حرکتی وسیله نقلیه را فراهم میکند.

درک محیط:

این بخش از سیستم‌های حسگر مانند دوربین‌ها، لیدار و رادار استفاده می‌کند تا نقشه‌ای از محیط پیرامون وسیله نقلیه ایجاد کند و اطلاعات مربوط به موانع، جاده‌ها و علائم راهنمایی و رانندگی و هرچه در محیط هست را جمع‌آوری کند.

برنامه‌ریزی مسیر:

این بخش از سیستم مسیری را طراحی می‌کند که وسیله نقلیه بتواند به بهترین شکل ممکن از نقطه شروع به نقطه مقصد حرکت کند. این مسیر طوری طراحی میشود که وسیله نقلیه بتواند از موانع اجتناب کند، زمان و انرژی بهینه‌ای مصرف کند و همچنین حرکتی ایمن و روان داشته باشد.

کنترل وسیله نقلیه:

این بخش از سیستم مسئول اجرای دستورات گفته شده از بخش برنامه‌ریزی مسیر است و با سرعت، شتاب، ترمز و جهت حرکت وسیله نقلیه کار دارد.

۱-۱- روش های برنامه ریزی مسیر

روش‌های مختلفی برای برنامه‌ریزی مسیر وسایل نقلیه خودران در طول سال‌ها توسعه یافته‌اند. این روش‌ها را می‌توان به چهار دسته کلی تقسیم کرد:

الگوریتم‌های جستجوی گراف:

این الگوریتم‌ها از گراف‌ها برای نمایش محیط و جستجوی بهترین مسیر استفاده می‌کنند. مثال‌هایی از این دسته شامل الگوریتم دایجسترا و A-Star هستند. این الگوریتم‌ها به دلیل کارایی و سادگی، به طور گسترده‌ای در کاربردهای واقعی استفاده می‌شوند.

الگوریتم‌های نمونه‌برداری:

این الگوریتم‌ها با نمونه‌برداری تصادفی از فضا ، تلاش می‌کنند که مسیری بهینه را پیدا کنند. الگوریتم RRT یکی از معروف‌ترین الگوریتم‌های این دسته است.مسیرهای تولیدشده توسط این الگوریتم معمولاً بهینه نیستند.

الگوریتم‌های درون‌یابی:

این الگوریتم‌ها از روش‌های ریاضی برای تخمین مسیرهای پیوسته استفاده می‌کنند. این روش‌ها زمانی که نیاز به تولید مسیرهای روان و بدون تغییرات ناگهانی باشد استفاده میشوند.

الگوریتم‌های بهینه‌سازی عددی:

این الگوریتم‌ها از روش‌های بهینه‌سازی عددی برای یافتن مسیرهای بهینه در محیط‌های پیچیده استفاده می‌کنند. این روش‌ها پیچیدگی محاسباتی بالایی دارند و به زمان بیشتری برای پردازش نیاز دارند.

۲-۱- الگوریتم ای - استار

یکی از پرکاربردترین الگوریتم‌ها در زمینه برنامه‌ریزی مسیر، الگوریتم A-Star است. این الگوریتم از یک تابع هدایتی (Heuristic Function) برای تسریع در جستجوی گره‌ها و پیدا کردن مسیر بهینه استفاده می‌کند.

تابع هیوریستیک :

$F = G + H$ که G به معنای تابع هزینه و $H = dis(C,T)$ که C موقعیت الان رو نشون میده و T موقعیت مقصد را نشان میدهد.

A-Star به دلیل سادگی، کارایی و سرعت بالاتری که نسبت به الگوریتم دایجسترا دارد، به طور گسترده‌ای مورد استفاده قرار گرفته است.

اما این الگوریتم نقاط ضعفی نیز دارد. یکی از مشکلات اصلی آن این است که در شرایطی که وسیله نقلیه باید چرخش‌های تند انجام دهد یا از موانع زیادی عبور کند، عملکرد آن بهینه نیست. این موضوع می‌تواند منجر به تولید مسیرهایی شود که ایمنی کمتری دارند و همچنین زمان و انرژی بیشتری مصرف می‌کنند.

مشکلات الگوریتم A-Star کلاسیک :

ناتوانی در چرخش‌های تند و پیچ‌ها:

این الگوریتم در پیدا کردن مسیرهای دقیق در زمانی است که وسیله نقلیه باید از پیچ‌های تند عبور کند ضعیف عمل میکند. این ضعف به دلیل نحوه عملکرد تابع هدایتی الگوریتم است. تابع هدایتی استاندارد در A-Star بهینه‌ترین مسیر را از نظر فاصله به سمت هدف هدایت می‌کند، اما در مواقعی که نیاز به تغییر مسیر سریع باشد (مانند پیچ‌ها)، نمی‌تواند به درستی مسیر را تنظیم کند.

در برخی سناریوها، حتی زمانی که مانعی وجود ندارد، A-Star کلاسیک مسیرهایی را تولید می‌کند که دقیق و مناسب نیستند. این مشکل به ویژه در جاده‌های پیچ‌دار و مسیرهایی با زوایای تند مشاهده می‌شود. در این شرایط، وسیله نقلیه ممکن است نتواند به درستی از مسیر خارج شده یا با موانع برخورد کند.

انتخاب پارامترهای کلیدی:

یکی دیگر از چالش‌های A-Star کلاسیک، انتخاب پارامترهای کلیدی مانند اندازه گام جستجو و زاویه چرخش است. اگر این پارامترها به درستی انتخاب نشوند، ممکن است مسیرهای ناکارآمد و طولانی تولید شوند.

افزایش زمان محاسباتی و تعداد گره‌های توسعه یافته:

A-Star کلاسیک به دلیل نیاز به توسعه تعداد زیادی از گره‌ها در طول جستجو، معمولاً زمان محاسباتی بالایی دارد. انتخاب طول گام‌های مختلف می‌تواند منجر به نتایج متفاوتی شود. تعداد گره های کم باعث کاهش دقت میشود و همین طور تعداد گره های زیاد باعث افزایش زمان محاسباتی میشود.

اگر اندازه گام جستجو بیش از حد کوچک باشد، تعداد گره‌های توسعه‌یافته به شدت افزایش می‌یابد و این موضوع منجر به افزایش زمان محاسباتی و کاهش کارایی می‌شود. این امر درمحیط‌های پیچیده و بزرگ که نیاز به جستجوی گسترده دارند، بیشتر مشاهده می‌شود.

۲. فصل دوم

بهبود های پیشنهادی

بهبود های پیشنهادی

همانطور که گفتیم این مقاله به مرور چهار دسته اصلی از الگوریتم‌های برنامه‌ریزی مسیر برای (ALV) می‌پردازد. این دسته‌ها شامل الگوریتم‌های جستجوی گراف (مانند A-Star و دایجسترا)، الگوریتم‌های نمونه‌برداری (مانند RRT)، الگوریتم‌های درون یابی (مانند اسپلاین‌ها)، و الگوریتم‌های بهینه‌سازی عددی است. الگوریتم A-Star و نسخه‌های مختلف آن، به دلیل قابلیت کاربرد در وسایل نقلیه خودمختار، به طور گسترده مورد استفاده قرار گرفته‌اند. با این حال، نسخه‌های کلاسیک A-Star مشکلاتی از قبیل ناتوانی در چرخش در پیچ‌ها و دشواری در انتخاب پارامترهای مناسب را دارند. بهبودهایی مانند Hybrid A-Star و Autoware A-Star مورد استفاده قرار گرفته‌اند، اما همچنان مشکلاتی در تشخیص و اجتناب از موانع در شرایط پیچیده وجود دارد.

الگوریتم‌های نمونه‌برداری مانند RRT برای برنامه‌ریزی سریع مسیر استفاده می‌شوند، اما نتایج حاصل معمولاً بهینه نیستند که برای بهبود این مشکل، نسخه RRT* معرفی شده است. تحقیقات جدیدتری نیز برای دنبال کردن مسیرها و جلوگیری از برخورد با موانع ایجادشده.

۱-۲- معرفی استاندارد ارزیابی جهانی

الگوریتم برنامه‌ریزی مسیر باید ایمنی، راحتی و بهینه‌سازی انرژی را فراهم کند. این بخش یک استاندارد ارزیابی را معرفی می‌کند که به کمک آن می‌توانیم عملکرد الگوریتم‌های مختلف برنامه‌ریزی مسیر را به صورت کمی اندازه‌گیری کنیم.

سرعت و فاصله مجازی:

در این روش به هر قسمت از مسیر با توجه به شرایط جاده (مانند پیچ‌ها) یک سرعت مجازی داده می‌شود. سرعت مجازی برای خطوط مستقیم ۱ و برای پیچ‌ها و بخش‌های پیچیده کمتر نظر گرفته است و شبیه‌سازی می‌کند چگونه وسیله نقلیه در واقعیت سرعت خود را کاهش می‌دهد.

زمان هزینه:

معیار کلیدی برای ارزیابی یک مسیر، کل زمانی است که وسیله نقلیه برای طی کردن مسیر بر اساس سرعت‌های مجازی می‌گیرد. هرچه این زمان کمتر باشد، عملکرد الگوریتم برنامه‌ریزی بهتر است.

طول‌های مختلف گام می‌تواند به طور چشمگیری بر کیفیت مسیر و زمان محاسبه تأثیر بگذارد. اگر کم باشد میتواند دقیق ولی کند باشد و اگر زیاد باشد کیفیت کم میشود. بر اساس آزمایش‌ها، گام جستجو با طول ۴ بهترین تعادل بین زمان محاسبه و کیفیت مسیر را دارد.

این روش به یافتن پارامترهای بهینه برای الگوریتم A-Star کمک می‌کند.

۲-۲- الگوریتم های بهبود یافته

الگوریتم A-Star مبتنی بر راهنما (Guideline-Based A-Star):

این بخش نشان می‌دهد که چگونه الگوریتم کلاسیک A-Star اصلاح شده است. این الگوریتم با استفاده از یک راهنما(برنامه یا انسان) بهبودهایی در تابع هدایتی ارائه می‌دهد. راهنما مسیرهای بهینه‌تری را برای پیچ‌های تند تولید میکند. الگوریتم مبتنی بر راهنما در پیچ ها عملکرد بهتری دارد و مسیرهای دقیق‌تری تولید می‌کند.

مشکل الگوریتم کلاسیک در پیچیدن در پیچ‌ها (تند پیچیدن که ایمن و واقع‌گرایانه نیست) و نیت های انسانی (ماندن در خطوط یا پیروی از یک منحنی) است.

برای رفع این مشکلات راهنما ایجاد میشود.. راهنما به عنوان یک مسیر مرجع عمل می‌کند که نیت راننده را بیان می‌کند و به وسیله نقلیه کمک می‌کند تا در مسیری واقعی‌تر و ایمن‌تر حرکت کند.

در این الگوریتم، تابع هیوریستیک F(i) برای هر نقطه i به دو بخش تقسیم می‌شود:

H1(i) و H2(i) که اولی به معنای فاصله نقطه کنونی تا نزدیکترین نقطه راهنما است و دومی فاصله نقطه روی راهنما تا مقصد است.

$$F(i) = G(i) + H1(i)*a1 + H2(i)*a2$$

این تابع جدید نه تنها تلاش می‌کند تا فاصله تا هدف را به حداقل برساند، بلکه اطمینان حاصل می‌کند که وسیله نقلیه به راهنما نزدیک بماند و رفتاری شبیه به یک راننده انسانی داشته باشد.

در موقعیت‌هایی که جاده دارای پیچ‌های تند است، الگوریتم کلاسیک نمی‌تواند به‌خوبی مسیر را شناسایی کند، اما با استفاده از راهنما، وسیله نقلیه قادر است مسیرهای دقیق‌تر و بهتری را برای چرخش‌ها پیدا کند.

الگوریتم مبتنی بر راهنما با اضافه کردن یک مسیر پیشنهادی به عنوان راهنما، مشکلات رایج A-Star کلاسیک در مسیرهای پیچیده و غیرمستقیم را برطرف می‌کند. این الگوریتم به‌ویژه در شرایطی که نیاز به پیروی دقیق از مسیرهای خاص یا اجتناب از پیچ‌های تند و موانع وجود دارد، بسیار مفید است.

الگوریتم A-Star مبتنی بر نقاط کلیدی (Key-Point Based A-Star):

در حالی که الگوریتم A-Star مبتنی بر راهنما عملکرد مسیر را در بسیاری از موارد بهبود می‌بخشد، اما زمانی که راهنما از روی موانع عبور می‌کند، با چالش‌هایی مواجه می‌شود.

برای بهبود عملکرد اجتناب از موانع، از نقاط کلیدی در اطراف موانع استفاده شده است. این نقاط به الگوریتم کمک می‌کنند تا از موانع زودتر اجتناب کند و از ایجاد مسیرهای نامطلوب جلوگیری کند. نقاط کلیدی به عنوان نقاط هدایتی در مسیر عمل می‌کنند و به الگوریتم کمک می‌کنند تا مسیرهایی امن‌تر و بهینه‌تر تولید کند. این الگوریتم چند مرحله دارد:

اول فضای جستجو، راهنما، نقطه شروع، نقطه هدف، و موانع در صحنه تعریف می‌شوند.

سپس، الگوریتم بررسی می‌کند که آیا روی مسیر راهنما موانعی وجود دارد یا نه. اگر هیچ مانعی وجود نداشته باشد، الگوریتم به صورت عادی به کار خود ادامه می‌دهد. اما اگر موانعی در مسیر راهنما قرار داشته باشند، الگوریتم به مرحله بعد می‌رود.

در این مرحله، الگوریتم نقاط کلیدی را در اطراف موانعی که روی راهنما قرار دارند، شناسایی می‌کند. این نقاط کلیدی به نحوی انتخاب می‌شوند که وسیله نقلیه را به دور از موانع هدایت کنند. برای هر مانع، دو نقطه کلیدی در دو سمت مانع انتخاب می‌شود که به وسیله نقلیه کمک می‌کنند تا به طور ایمن از مانع عبور کند.

پس از یافتن نقاط کلیدی، الگوریتم خط های جدیدی را که از نقاط کلیدی عبور می‌کنند، تولید می‌کند. این خطوط مسیرهایی هستند که جایگزین مسیر راهنمای اصلی اند و مسیرهایی هستند که از موانع فاصله دارند.

الگوریتم مسیرهای جدید را که از نقاط کلیدی میگذرد، بررسی می‌کند و مسیری را که هیچ مانعی روی آن قرار ندارد و زمان عبور از آن بهینه است را انتخاب میکند و ادامه مسیر را مانند الگوریتم A-Star مبتنی بر راهنما طی میکند.

این الگوریتم به طور مؤثری موانع را دور می‌زند و زمان مسیر نشان می‌دهد که عملکرد بهتری داشته و وسیله نقلیه را ایمن‌تر نگه می‌دارد.

الگوریتم A-Star مبتنی بر گام‌های متغیر (Variable-Step A-Star):

الگوریتم A-Star مبتنی بر گام متغیر اندازه گام را بر اساس محیط به صورت پویا تنظیم می‌کند.

در الگوریتم کلاسیک و نسخه‌های بهبود یافته آن برای کاهش زمان محاسبات و بهبود کارایی الگوریتم، از گام‌های ثابت استفاده می‌شود. ولی در این الگوریتم در نواحی باز از گام‌های بزرگ‌تر استفاده میشود تا سریع‌تر حرکت کند و در نواحی با موانع گام‌ها کوچک‌تر می‌شوند تا با دقت بیشتری موانع را دور بزند. این باعث کاهش تعداد گره‌ها و زمان محاسبات کل الگوریتم می‌شود.

مراحل این الگوریتم به شکل زیر است :

ابتدا حداکثر و حداقل گام‌های جستجو تنظیم می‌شوند.

در هر مرحله از جستجو، الگوریتم فاصله وسیله نقلیه تا موانع یا نواحی پیچیده را پیدا می‌کند و اگر فاصله زیاد باشد (منطقه باز)، گام بزرگ‌تر انتخاب می‌شود و اگر فاصله کم باشد (منطقه پیچیده)، گام کوچک‌تر می‌شود. الگوریتم با استفاده از گام‌های متغیر به جستجو ادامه می‌دهد.

نتایج نشان داده که زمان محاسبات کاهش یافته دقت بیشتر شده و گام‌های متغیر باعث شده‌اند که وسیله نقلیه سریع‌تر به هدف برسد، بدون اینکه از دقت مسیر کاسته شود.

۳. فصل سوم

تحلیل نتایج آزمایش‌های انجام شده

تحلیل نتایج آزمایش‌های انجام شده

برای ارزیابی عملکرد الگوریتم بهبود یافته A-Star (شـــامل بهبودهای گام متغیر و نقاط کلیدی) در محیط‌های مختلف، مجموعه‌ای از آزمایش‌ها انجام شده. این الگوریتم جدید با الگوریتم کلاسیک مقایسه شـــده تا عملکرد آن در زمینه‌هایی مانند کیفیت مســـیر، کارایی و ایمنی در ســـناریوهای مختلف دنیای واقعی مشاهده شود.

در میدان باز با موانع ثابت به تحلیل هر الگوریتم پرداختیم در این میدان رفتار انها موقع چرخش و مواجهه با موانع بررسی شد :

الگوریتم کلاسیک مسیرهای نادرستی تولید کرد، به ویژه در نزدیکی پیچ‌های تند، جایی که نتوانست به درستی جاده را دنبال کند.

الگوریتم پیشنهادی مسیرهای صاف‌تر و واقع‌گرایانه‌تری تولید کرد که مسیر را دنبال کرده و موانع را به طور مؤثری دور زد.

الگوریتم گام متغیر در هر دو حالت موانع S شـــکل و نقطه چرخش عملکرد بهتری داشـــت و مســـیری طبیعی‌تر را حفظ کرد.

۱-۳- آزمایش در محیط‌های شهری

در این بخش به بررســـی عملکرد الگوریتم‌های مختلف در محیط‌های شـــهری ســـاختاریافته و جاده‌های پیچ‌دار پرداختیم. نتایج آزمایش‌ها نشـــان داده اســـت که الگوریتم بهبود یافته در مقایســـه با کلاســـیک، عملکرد بهتری از نظر کیفیت مسیر و زمان محاسباتی داشته است.

منطقه آزمایش خیابانی بود که ســناریوی معمولی یک شـهر را شـبیه‌ســازی می‌کرد. موانع مختلفی مانند خودروها، درختان و عابران پیاده در این محیط وجود دارد که وســـیله نقلیه باید از آن‌ها اجتناب کند. در این آزمایش، مجموعه موانع در خیابان قرار داده شـدند تا توانایی هر الگوریتم در اجتناب از موانع آزمایش شود.

الگوریتم کلاسیک A-Star با مشکلات قابل توجهی مواجه شد:

در یکی از تست‌ها، این الگوریتم از مسیر میانبر غیرمجاز در خیابان عبور کرد که نباید وارد آن می‌شد (به دلیل تشخیص نادرست موانع).

این الگوریتم اغلب نتوانست مرزهای جاده را به درستی تشخیص دهد، که باعث ایجاد مسیرهای خطرناک یا غیرواقعی شد.

الگوریتم پیشنهادی وضعیت را به مراتب بهتر مدیریت کرد:

این الگوریتم به طور موفقیت‌آمیز مسیر مورد نظر را دنبال کرد، حتی زمانی که لبه جاده به درستی تشخیص داده نمی‌شد و موانع را با ایمنی دور زد.

الگوریتم پیشنهادی عملکرد خوبی داشت و مسیر نزدیک‌تر به راهنما را نسبت به الگوریتم کلاسیک دنبال کرد و وقتی موانع در وسط خیابان قرار داده شدند، الگوریتم پیشنهادی توانایی خود را در دور زدن آن‌ها به طور دقیق نشان داد، بدون اینکه از مسیر مورد نظر خیلی منحرف شود.

۳-۲ آزمایش در سناریوهای پیچیده و طولانی‌مدت

در این بخش پایداری و قابلیت اطمینان الگوریتم پیشنهادی را در مسافت‌های طولانی آزمایش شد. این الگوریتم در محیط‌های واقعی مختلفی مانند دانشگاه‌ها، روستاها و مناطق مسکونی به کار برده شد. این محیط‌ها شامل چالش‌های شهری معمولی مانند عابران پیاده، خودروهای در حال حرکت و موانعی هستند که همیشه به خوبی تشخیص داده نمی‌شوند.

در هر دو حالت دانشگاه و پارک الگوریتم در وسیله نقلیه خودکار برای چندین بار عبور از این مناطق آزمایش شد.

الگوریتم پیشنهادی در این مسافت‌های طولانی عملکرد بسیار خوبی داشت. وسیله نقلیه به طور مداوم مسیرراهنما را با کمترین انحراف دنبال کرد. حتی در حضور عناصر پویا مانند عابران پیاده و خودروهای در حال حرکت، الگوریتم به طور ایمن از آن‌ها اجتناب کرد و مسیری دقیق را حفظ کرد.

در محیط‌های واقعی که موانع ممکن است به درستی توسط سیستم‌های درک شناسایی نشوند، الگوریتم جدید پایداری بیشتری نشان داد و توانست مسیرهای ایمن‌تر و دقیق‌تری را ارائه دهد.

۳-۳- مقایسه با الگوریتم‌های دیگر

مقاله نتایج حاصل از الگوریتم بهبود یافته را با نسخه‌های کلاسیک A-Star و برخی دیگر از الگوریتم‌های پیشرفته مقایسه کرده است. این مقایسه‌ها نشان داد که الگوریتم بهبود یافته در شرایط واقعی و محیط‌های پیچیده عملکرد بهتری داشته است.

الگوریتم مبتنی بر راهنما در مقایسه با A-Star کلاسیک:

در الگوریتم کلاسیکA-Star ، تنها از فاصله اقلیدسی به عنوان تابع هیوریستیک استفاده می‌شود. این الگوریتم در صحنه‌هایی که وسیله نقلیه نیاز به عبور از پیچ‌ها یا موانع دارد، عملکرد مطلوبی ندارد؛ زیرا تنها به هدف نهایی توجه می‌کند و راه مناسبی برای چرخش‌ها یا اجتناب از موانع ارائه نمی‌دهد.

ولی دیگری از یک راهنما یا مسیر پیشنهادی که توسط انسان یا برنامه‌ریزی کلی تعیین شده استفاده می‌کند. این راهنما به الگوریتم کمک می‌کند تا توجه بیشتری در مسیرهای غیرمستقیم داشته باشد. تابع هیوریستیک این الگوریتم شامل دو بخش است: یکی فاصله تا راهنما و دیگری فاصله از راهنما تا هدف.

الگوریتم مبتنی بر نقاط کلیدی در مقایسه با A-Star کلاسیک:

در A-Star کلاسیک، وقتی موانعی در مسیر قرار دارند، الگوریتم بدون توجه به شرایط پیرامون، از نزدیک‌ترین مسیر برای اجتناب از مانع استفاده می‌کند. این ممکن است باعث شود که وسیله نقلیه با تأخیر واکنش نشان دهد و باعث افزایش زمان محاسبات و کاهش دقت در مسیرهای واقعی شود.

الگوریتم مبتنی بر نقاط کلیدی برای بهبود این مشکل، از نقاط کلیدی اطراف موانع استفاده می‌کند. این نقاط کلیدی به عنوان راهنمایی برای تغییر مسیر قبل از رسیدن به مانع عمل می‌کنند، به طوری که وسیله نقلیه زودتر از مانع فاصله می‌گیرد و به راحتی از آن اجتناب می‌کند.

این الگوریتم می‌تولند مسیرهایی را برنامه‌ریزی کند که کمتر به مانع نزدیک شوند و در نتیجه خطر برخورد را کاهش دهد.

الگوریتم مبتنی بر گام‌های متغیر درمقایسه با A-Star کلاسیک:

در A-Star کلاسیک، گام‌های جستجو ثابت اند. این یعنی الگوریتم به همان اندازه در مناطق باز و بدون مانع جستجو می‌کند که در مناطق پیچیده و پرمانع جستجو میکند. این می‌تواند باعث افزایش زمان محاسبات در مناطقی شود که نیاز به دقت بالا ندارند.

ولی الگوریتم مبتنی بر گام‌های متغیر به جای استفاده از گام‌های ثابت، گام‌ها را بر اساس شرایط محیطی تنظیم می‌کند. در مناطق باز و بدون مانع، گام‌ها بزرگ‌تر می‌شوند تا سریع‌تر مسیر پیدا شود، و در مناطق نزدیک به موانع یا پیچ‌های تند، گام‌ها کوچک‌تر می‌شوند تا دقت بیشتری به‌کار رود و در نتیجه دقت و کارایی همزمان بالا میرود.

این تغییرات و بهبودها، الگوریتم‌های جدید را برای استفاده مناسب‌تر می‌کند و در نهایت به عملکرد بهتر و کارآمدتر در مسیرهای ناهموار یا پرمانع منجر می‌شود.

۴. فصل چهارم

گزارش کار

گزارش کار

در این بخش هر کدام از الگوریتم های DFS,BFS,UCS,GREEDY,A-STAR تعریف شـــده و هر کدام را توضیح میدهیم و پیاده سازی انها را بررسی میکنیم.

الگوریتم DFS :

الگوریتم جستجوی عمق‌اول (DFS - Depth First Search) یک الگوریتم جستجو در ساختارهای گراف است. DFS از (stack) استفاده می‌کند تا از یک گره شروع کند و تا حد ممکن به عمق گراف برود بعد به عقب بازگردد و بقیه مسیر را بررسی میکند.

برای پیاده سازی باید بدونیم این الگوریتم از یک گره مشخص شروع میکند و انرا visited قرار میدهد :

```
Visited = set()
```

بعد برای هر فرزند گره به ترتیب از یکی از آن‌ها به عمق بیشتری می‌رود. اگر فرزند نداشت به گره قبلی برمیگردد.وقتی همه گره‌های یک مسیر بررسی شدند، به گره‌های قبلی بازمی‌گردد تا مسیرهای دیگر را جستجو کند. وقتی تمام گره‌ها بازدید شدند یا به هدف رسیدیم، الگوریتم به پایان می‌رسد.

وقتی فرزند را دید انرا و خواست انرا در visited بگذارد: stack.append(son)

الگوریتم BFS :

الگوریتم جستجوی اول سطح (BFS - Breadth-First Search) این الگوریتم، ابتدا تمام رئوس یا گره‌های فرزند یک گره مشخص را بازدید میکند و سپس به سراغ گره‌های فرزند آن‌ها می‌رود..

الگوریتم از گره مبداشـروع می‌شـود و گره‌ی مبدأ را در صـف قرار می‌دهد. گره‌ای که در صـف قرار دارد بازدید می‌شـود. بعد همه گره‌های فرزند آن که هنوز بازدید نشـدن به صـف اضـافه می‌شـوند. گره‌ای که بازدید می‌شود از صف خارج میشود وقتی صف خالی شود تموم میشود.

در مسئله تام و جری بهترین حالت را به ما میدهد این الگوریتم برای گراف های بدون وزن بسیار مناسب است، زیرا به طور خودکار کوتاه‌ترین مسیر (از لحاظ تعداد یال‌ها) را پیدا می‌کند.

الگوریتم UCS :

الگوریتم جستجوی هزینه یکنواخت (UCS - Uniform Cost Search) نوعی الگوریتم جستجو است که برای یافتن کم‌هزینه‌ترین مسیر (کوتاه‌ترین مسیر بر اساس وزن یال‌ها) در گراف‌هایی با وزن‌های متفاوت به کار می‌رود. UCS مشابه الگوریتم BFS است، با این تفاوت که در BFS گراف‌ها بدون وزن هستند و به تعداد گام‌ها توجه می‌شود ولی در UCS به هزینه مسیر توجه می‌شود.

الگوریتم از یک گره شروع می‌شود. به جای صف عادی که در BFS استفاده می‌شود، UCS از یک صف اولویت‌دار استفاده می‌کند. در این صف، گره‌هایی که کمترین هزینه رسیدن را دارند، اولویت بیشتری دارند. گره مبدأ آن را در صف قرار می‌دهد. در هر مرحله، گره‌ای که کمترین هزینه رسیدن را دارد از صف برداشته می‌شود و بررسی می‌شود.برای هرکدام تمام گره‌های فرزندها چک می‌شوند. اگر هزینه رسیدن به گره فرزند کمتر از هزینه قبلی باشد گره در صف با هزینه جدید قرار می‌گیرد. وقتی تمام میشود که کم‌هزینه‌ترین مسیر به آن پیدا شود.

برخلاف الگوریتم‌هایی مانند A-Star که از تابع هیوریستیک برای پیش‌بینی استفاده می‌کنند، UCS کاملاً بر اساس هزینه‌های واقعی یال‌ها عمل می‌کند و هیچ‌گونه پیش‌بینی انجام نمی‌دهد.

الگوریتم A* :

الگوریتم A-Star یکی از الگوریتم‌ها برای پیدا کردن کوتاه‌ترین مسیر در گراف است. این الگوریتم از ترکیبی از (UCS) و(Heuristic) استفاده می‌کند تا مسیر را با در نظر گرفتن هزینه و تخمین فاصله تا هدف پیدا کند.

تابع هیوریستیک :

$F = G + H$ که G به معنای تابع هزینه و Hفاصله از مقصد است.

A* در مسائل مربوط به جستجوی مسیر در گراف‌ها و شبکه‌ها کاربرد زیادی دارد، مانند پیدا کردن کوتاه‌ترین مسیر در نقشه‌ها یا در بازی‌ها و سیستم‌های هوش مصنوعی برای پیدا کردن بهترین مسیر.

الگوریتم A* یکی از محبوب‌ترین و پرکاربردترین الگوریتم‌ها در زمینه بهینه‌سازی مسیر است، زیرا ترکیبی از دقت و سرعت را ارائه می‌دهد.

در این مسئله ما میتوانیم از فاصله منهتنی یا اقلیدسی استفاده کنیم :

منهتن در این مسئله :

```
def heuristic(a, b):
    a =  (x1, y1)
    b = (x2, y2)
    return abs(x1 - x2) + abs(y1 - y2)
```

فاصله اقلیدسی در این مسئله :

```
def Heuristic(self, tomPos, jerryPos):
    dx = abs(tomPos[0] - jerryPos[0])
    dy = abs(tomPos[1] - jerryPos[1])
    return math.sqrt(dx**2+dy**2)
```

الگوریتم Greedy:

الگوریتم حریصانه (Greedy Algorithm) یکی از روش‌های ساده در حل مسائل بهینه‌سازی است. این الگوریتم در هر گام، تصمیمی میگیرد که به نظر می‌رسد در آن لحظه بهترین یا بهینه‌ترین گزینه است.

الگوریتم از یک وضعیت اولیه شروع می‌کند. در هر مرحله، از میان گزینه‌های موجود، گزینه‌ای که بهترین نتیجه را در آن لحظه دارد را انتخاب می‌کند. این انتخاب‌ها تا زمانی که تمام گزینه‌ها بررسی شوند، ادامه پیدا می‌کنند.

۵. فصل پنجم

جمع‌بندی و نتیجه‌گیری

جمع‌بندی و نتیجه‌گیری

این مقاله یک الگوریتم برنامه ریزی مسیر مبتنی بر A-Star بهبود یافته برای ALV ها را ارائه می دهد. برنامه ریزی حرکت در جاده برای ALV ها چالش های مختلفی دارد و الگوریتم سنتی A-Star برای این امر کافی و مناسب نیست.

این مقاله چهار بهبود اصلی برای الگوریتم A-Star کلاسیک ارائه داده است که عبارت اند از معرفی استاندارد ارزیابی، استفاده از راهنما، استفاده از نقاط کلیدی و استفاده از گام‌های متغیر.

نتایج آزمایشات نشان داده که الگوریتم جدید کارایی بیشتری از نظر زمان محاسباتی و تعداد گره‌های توسعه یافته دارد و کیفیت مسیرهای تولید شده نیز در آن بهبود یافته است. در مقایسه با روش‌های پیشرفته فعلی، این الگوریتم با بهبودهای مختلف، پایداری و کارایی بیشتری دارد و مناسبترین گزینه برای وسایل نقلیه خودران زمینی است.

منابع و مراجع

References

1. Shang E, An X, Wu T, et al. Lidar based negative obstacle detection for field autonomous land vehicles. J Field Robot 2016; 33(5): 591–C617.

2. Kim J, Jo K, Lim W, et al. A probabilistic optimization approach for motion planning of autonomous vehicles. Proc Instit Mech Eng D J Automob Eng 2018; 232(5): 632–650.

3. Gonz´alez D, P´erez J, Milan´es V, et al. A review of motion planning techniques for automated vehicles. IEEE Trans Intell Transp Syst 2015; 17(4): 1135–1145.

4. Montemerlo M, Becker J, Bhat S, et al. Junior: The Stanford entry in the urban challenge. J Field Robot 2008; 25(9): 569–597.

5. Dolgov D, Thrun S, Montemerlo M, et al. Path planning for autonomous vehicles in unknown semi-structured environments. Int J Robot Res 2010; 29(5): 485–501.

6. Brezak M and Petrovic´ I. Real-time approximation of clothoids with bounded error for path planning applications. IEEE Trans Robot 2013; 30(2): 507–515.

7. Gu T and Dolan JM. On-road motion planning for autonomous vehicles. In: International conference on intelligent robotics and applications. Montreal, QC, Canada, 3–5 October 2012, pp. 588–597. Berlin: Springer.

8. Kammel S, Ziegler J, Pitzer B, et al. Team AnnieWAY's autonomous system for the 2007 Darpa Urban Challenge. J Field Robot 2008; 25(9): 615–639.

9. Ducho˘n F, Babinec A, Kajan M, et al. Path planning with modified a star algorithm for a mobile robot. Proc Eng 2014; 96: 59–69.

10. Hart PE, Nilsson NJ, and Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 1968; 4(2): 100–107.

11. Ferguson D, Howard TM, and Likhachev M. Motion planning

12. Autoware: open-source software for urban autonomous driving, 2018-2-14. https://github.com/CPFL/Autoware/wiki.

13. LaValle SM and Kuffner JJ Jr. Randomized kinodynamic planning. Int J Robot Res 2001; 20(5): 378–400.

14. Karaman S, Walter MR, Perez A, et al. Anytime motion planning using the RRT. In: 2011 IEEE international conference on robotics and automation. Shanghai, China, 9–13 May 2011, IEEE, pp. 1478–1483.

15. Chebly A. Trajectory planning and tracking for autonomous vehicles navigation. PhD Thesis, Universit´e de Technologie de Compi`egne, 2017.

16. Kanayama YJ and Hartman BI. Smooth local planning for autonomous vehicles. Int J Robot Res 1999; 16(3): 273–284.

17. Delingette H, Hebert M, and Ikeuchi K. Trajectory generation with curvature constraint based on energy minimization. In: IEEE/RSJ international conference on intelligent robots and systems. Osaka, Japan, 3–5 November 1991, IEEE, pp. 123–128.

18. Lau B, Sprunk C, and Burgard W. Kinodynamic motion planning for mobile robots using splines. In: IEEE/RSJ international conference on intelligent robot and systems. St. Louis, MO, USA, 10–15 October 2009, IEEE, pp. 2427–2433.

19. Kato S, Tokunaga S, Maruyama Y, et al. Autoware on board: enabling autonomous vehicles with embedded systems. In: 2018 ACM/IEEE 9th international conference on cyberphysical systems (ICCPS). Porto, Portugal, 1–13 April 2018, IEEE, pp. 287–296.

20. Li B, Du H, Li W, et al. Dynamically integrated spatiotemporal-based trajectory planning and control for autonomous vehicles. IET Intell Transp Syst 2018; 12(10): 1271–1282.

21. Liu Y and Cui D. Path tracking control for inverse vehicle handling dynamics. Int J Veh Safe 2019; 11(2): 120