دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده ریاضی و علوم کامپیوتر

تمرین کارشناسی
هوش مصنوعی و کارگاه

گزارش ۲: مقاله A Survey of A-Star Algorithm Family for Motion Planning of Autonomous Vehicles

نگارش
سارینا حشمتی
۴۰۰۱۳۰۴۳

استاد ۱
دکتر مهدی قطعی

استاد ۲
دکتر بهنام یوسفی مهر

آبان ۱۴۰۲

چکیده

در این گزارش بـه بررسـی و خلاصـه نویسـی مفهـومی مقالـه‌ی A Survey of A-Star Algorithm Family for Motion Planning of Autonomous Vehicles می پردازیم و در آن به اهمیت بـالای مسأله‌ی motion planning میپردازیم و در ادامه انواع مختلف الگوریتم *A را، که یک الگوریتم مبتنی بر جست و جوی گراف کارآمد میباشد، بررسی میکنیم و با یکدیگر مقایسه و به نقاط ضعف و قـوت آنهـا میپردازیم. نسخه‌های مختلف الگوریتم *A که به آنها در این مقاله اشاره میشود شامل :

Geometric A* Algorithm

Improved A* Algorithm

Anytime Repairing A* Algorithm

Dynamic A* Algorithm

Anytime Dynamic Algorithm

می باشد. در هریک از این موارد یک یا چند ویژگی مورد نظر هدف قرار داده شده‌اند و با ایجاد تغییراتـی در الگوریتم بسیار کارای کلاسیک، کارایی آنها در مسائل هدفشان حتی بهتر شده که ایـن باعـث میشـود پاسخ‌های ما به مسأله‌ی بسیار مهم motion planning بهتر و بهتر بشوند.

صفحه

فهرست مطالب

فصل اول
مقدمه

مقدمه

در مسائل مربوط به خودروهای خودران یکی از مهم‌ترین موارد برای اینکه خودرو توانایی عملکرد بدون هیچ گونه دخالت انسانی را داشته باشد، مسأله‌ی برنامه ریزی حرکت (motion planning) میباشد. برنامه‌ریزی حرکت یا motion planning فرآیند پیدا کردن یک مسیر بهینه از مبدأ به مقصد با در نظر گرفتن محدودیت‌ها و موانع میباشد. این فرآیند شامل تحلیل محیط، شناخت موانع و مسیرهای ممکن، تصمیم‌گیری در مورد مسیر بهتر و برنامه‌ریزی برای جزئیات حرکت است. همچنین، عواملی مانند موانع، محدودیت‌ها، اولویت‌ها و نیازهای خاص در برنامه‌ریزی حرکت باید در نظر گرفته شوند.

در این مسائل یک نقشه، یک نقطه‌ی شروع و یک نقطه‌ی پایان و یک تابع (یا تابع‌هایی) برای محاسبه‌ی هزینه‌ی هر بخش از مسیر انتخابی داده میشود، برای مثال مواردی مانند موانع ثابت یا پویا در محیط یا محدودیت‌های حرکتی به دلایل مختلف و یا محدودیت‌های زمانی در این توابع گنجانده میشوند تا با توجه به آنها مسأله‌ی motion planning بهینه‌ترین مسیر را پیدا کند.

در طی سالها با استفاده از الگوریتم‌های متعدد تلاش شده تا این مسآله حل بشود و هرکدام با توجه به نقاط قوت و ضعفی که داشتند نتایج مختلفی داشتند و همچنین در طی سالهای متوالی تلاش شده تا این الگوریتم‌ها به گونه‌ای ارتقا پیدا کنند که نقاط ضعف آنها بهتر شده و نقاط قوت آنها تقویت یابد.

یکی از این الگوریتم‌ها الگوریتم A* میباشد که در طی سالها انواع متفاوتی از آن ارائه شده که هرکدام با ایجاد با تغییراتی و اضافه کردن ویژگی‌هایی باعث بهتر شدن عملکرد این الگوریتم در شرایط مختلف و با اهداف متفاوت شده‌اند.

در این گزارش به بررسی برخی از این موارد میپردازیم.

فصل دوم
مسأله‌ای که قصد حلش را داریم.

مسأله‌ی برنامه‌ریزی برای حرکت (motion planning problem)

برنامه‌ریزی حرکت یک مسئله اساسی در رباتیک و سیستم‌های خودران است. در این برنامه‌ریزی، مسیر قابل انجام برای یک ربات یا یک عامل برای حرکت از یک حالت اولیه به یک حالت هدف مورد نظر، با رعایت موانع و در نظر گرفتن محدودیت‌های مختلف تعیین می‌شود.

هدف برنامه‌ریزی حرکت تولید یک توالی حالت‌ها است که حالت اولیه و حالت هدف را به هم متصل کند و در عین حال شرایط خاصی را ارضا کند، مانند بهینگی، ایمنی یا کارایی. حالت‌ها معمولاً موقعیت، جهت و متغیرهای مرتبط دیگر خودرو را نشان می‌دهند.

این مسئله با تعیین حالت اولیه، حالت هدف، توانایی‌های حرکتی خودرو و هر محدودیت یا الزام مرتبط دیگر تعریف می‌شود. محیط معمولاً به عنوان یک شبکه 2D یا 3D نشان داده می‌شود که در آن موانع به عنوان سلول‌های اشغال مشخص می‌شوند.

برنامه‌ریزی حرکت یک مسئله چالش‌برانگیز است به دلیل پیچیدگی زیاد و تنوع بالای محیط‌ها و نیاز به توازن بین کارآیی، بهینگی و ایمنی. روش‌ها و الگوریتم‌های مختلفی برای حل جوانب مختلف برنامه‌ریزی حرکت توسعه یافته است و این باعث می‌شود که برنامه‌ریزی حرکت یک حوزه تحقیقاتی پویا و فعال در سیستم‌های خودران باشد.

2-1- چرا به دنبال حل این مسأله هستیم؟

حل مسأله‌ی برنامه‌ریزی حرکت (motin planning) برای خودروهای خودران بسیار ضروری و حایز اهمیت میباشد، در ادامه به شماری از دلایل برای این اهمیت اشاره میکنیم:

1. ایمنی: خودروهای خودران در محیط‌های پویا و پیش‌بینی‌نشده عمل می‌کنند و باید به طور ایمن در میان پیاده‌روها، خودروهای دیگر و موانع مختلف دیگر حرکت کنند. الگوریتم‌های برنامه‌ریزی حرکت کمک می‌کنند تا خودرو تصمیمات ایمنی بگیرد و با تولید مسیرهای بدون تصادف، از برخورد با موانع جلوگیری کند.

2. کارآیی: الگوریتم‌های برنامه‌ریزی حرکت مسیر خودرو را بهینه می‌کنند تا حرکتی کارآمد و روان (smooth) داشته باشد. آنها عواملی مانند شرایط ترافیک، قوانین جاده، فیزیک خودرو و مصرف انرژی را

در نظر می‌گیرند تا مسیری که بین نقطه شروع و مقصد بهترین است را برنامه‌ریزی کنند. این کار بهبود در مصرف بهینه‌ی سوخت و کاهش زمان سفر را به همراه دارد.

3. حرکات پیچیده: خودروهای خودران ممکن است نیاز به انجام حرکات پیچیده‌ای مانند تغییر لاین، ورود به بزرگراه، پارک و سبقت داشته باشند. الگوریتم‌های برنامه‌ریزی حرکت به خودرو امکان می‌دهند این حرکات را به دقت و با ایمنی برنامه‌ریزی و اجرا کنند و با در نظر گرفتن محیط اطراف و شرایط ترافیک، عمل کنند.

4. قابلیت سازگاری: خودروهای خودران در زمان واقعی (real time) فعالیت می‌کنند و با عدم قطعیت‌های مختلفی مواجه می‌شوند، از جمله تغییرات در شرایط جاده، الگوهای ترافیک و موانع غیرمنتظره. الگوریتم‌های برنامه‌ریزی حرکت قابلیت سازگاری و بازطراحی مسیر خودرو را در حین حرکت فراهم می‌کنند.

بطور کلی، برنامه‌ریزی حرکت برای خودروهای خودران برای مسیریابی ایمن، کارآمد و هوشمند در محیط‌های پیچیده و پویا بسیار حائز اهمیت است و این خودروها را قادر می‌سازد حمل و نقل ایمن‌تر و کارآمدتر را به ارمغان بیاورند.

2-2- تاریخچه‌ی پاسخ‌ها!

توسعه برنامه‌ریزی حرکت در طول سالها تحولات قابل توجهی را پشت سر گذاشته است. در ادامه، یک بررسی مختصر از تکامل برنامه‌ریزی حرکت آورده شده است:

۱. رویکردهای اولیه: در مراحل اولیه، الگوریتم‌های برنامه‌ریزی حرکت عمدتاً بر سناریوهای ساده با محیط‌های از پیش‌تعیین شده و ساده شده تمرکز داشتند. این الگوریتم‌ها از روش‌هایی مانند potential fields، roadmaps و grid-based methods برای برنامه‌ریزی مسیر استفاده می‌کردند. با این حال آنها در مواجهه با محیط‌های پیچیده و پویا، محدودیت‌هایی داشتند.

۲. رویکردهای مبتنی بر نمونه‌برداری (sample-based approaches): رویکردهای مبتنی بر نمونه‌برداری به دلیل توانایی مدیریت فضاهای با بُعد بالا، مورد توجه قرار گرفتند. الگوریتم‌هایی مانند PRM، RRT و نسخه‌های مشتق شده از آنها مثالهایی برای این دسته از الگوریتم‌ها هستند. برای مثال RRTها را بررسی میکنیم:

Rapidly-Exploring Random Trees: RRTها که در اوایل دهه ۲۰۰۰ معرفی شدند، با بررسی بهینه‌ی محیط، برنامه‌ریزی حرکت را تحول بخشیدند. RRTها مبتنی بر ایجاد سریع ساختاری شبیه به

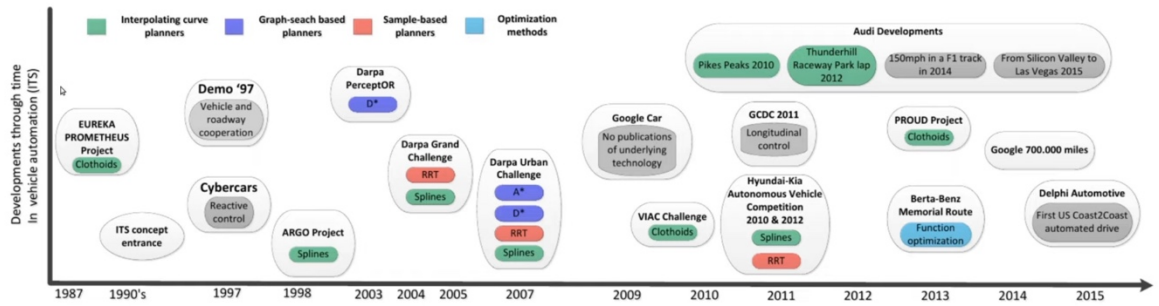
درخت از موقعیت ابتدایی و در جهت هدف تمرکز داشتند که امکان کاوش سریع فضای جستجو را فراهم می‌کردند.

۳. رویکردهای مبتنی بر بهینه‌سازی: در سالهای اخیر، رویکردهای مبتنی بر بهینه‌سازی در برنامه‌ریزی حرکت توجه بیشتری به خود جلب کرده‌اند. این رویکردها برنامه‌ریزی حرکت را به عنوان یک مسئله بهینه‌سازی مطرح می‌کنند و عواملی مانند روان بودن مسیر، کارآیی انرژی و اجتناب از موانع را در نظر می‌گیرند. تکنیک‌های بهینه‌سازی برای تولید برنامه‌های حرکت بهینه یا نزدیک به بهینه استفاده میشوند.

۴. رویکردهای مبتنی بر جستجوی گراف: دسته‌ای از الگوریتم‌های استفاده شده در برنامه‌ریزی حرکتی هستند که از تکنیک‌های جستجوی گراف برای پیدا کردن مسیرهای قابل اجرا برای سیستم‌های خودران در یک محیط استفاده می‌کنند. این برنامه‌ریزها یک نمایش از محیط را به شکل یک گراف دارند که در آن رأس‌ها حالت‌های خودرو و یالها انتقالهای قابل اجرا را بین حالت‌ها را نشان می دهند.

برنامه‌ریزهای مبتنی بر جستجوی گراف از الگوریتم‌های مختلف جستجوی گراف برای کاوش گراف و پیدا کردن مسیری از حالت اولیه تا حالت هدف استفاده می‌کنند. الگوریتم‌های جستجوی گراف رایجی که در برنامه‌ریزی حرکتی استفاده می‌شوند عبارتند از BFS، DFS، A* و Dijkstra، که در این مقاله به بررسی A*ها میپردازیم.

در تصویر زیر یک نمودار جالب از رویکردهای مختلف برای حل مسأله‌ی برنامه ریزی حرکت ارائه شده، البته این تصویر رویکردها را تا سال ۲۰۱۵ پوشش میدهد.



فصل سوم
راه‌حل‌هایی که با آن قصد حل مسأله را داریم.

الگوریتم A*

الگوریتم A*، یک روش مبتنی بر جستجوی گراف است که مسیری را از حالت ابتدایی بـه حالـت هـدف پیدا میکند به طوریکه حداقل هزینه را در یک محیط معین متحمل شود.

در روش جست و جوی A*، در هر رأس گراف هزینه‌ی f(n) برای تمام رأس‌های مجاور محاسبه میشود؛ داریم: $f(n) = g(n) + h(n)$

در این معادله دو جز اصلی داریم، g(n) و f(n). G(n) هزینه‌ی رسیدن از نقطه‌ی ابتدایی به رأس (حالت) n با استفاده از مسیر بهینه‌ی ایجاد شده تا n است. در مقابل، h(n) که همان heuristic میباشد در واقـع هزینه‌ی احتمالی (تخمین زده شده) برای رفتن از نقطه‌ی n به حالت نهایی (هـدف) اسـت. در الگـوریتم A* رأس بعدی مسیر با انتخاب رأس مجاوری انتخاب میشود کـه f(n) کمتـری دارد. بـرای محاسـبه‌ی h(n) روش‌های مختلفی وجود دارد (مانند فاصله‌ی منهتنی یا فاصله‌ی اقلیدسی) ولی به طور معمول برای محاسبه‌ی h(n) از فاصله‌ی اقلیدسی استفاده میشود.

Geometric A* Algorithm -3-2

این الگوریتم نسخه‌ای از الگوریتم کلاسیک است که به طور خاص برای حل مشـکلاتی ماننـد زاویـه‌های بـزرگ چـرخش و مسـیرهای پـر از رئـوس (ماننـد مسـیرهای ضـربدری (cross path) و مسـیرهای sawtooth) که معمولا توسط الگوریتم کلاسیک تولید میشوند طراحی شده است.

به طور کلی اکثر الگوریتم‌های مسیریابی مسـیرهایی بـا خطهـای زیـاد (polyline) تولیـد میکننـد کـه مشکلاتی مانند غیر رواں بودن حرکت خودرو و تغییر سـرعت در هنگـام چـرخش را بـه همـراه دارد، ایـن الگوریتم در جهت حل این مشکلات تلاش میکند.

در این الگوریتم ابتدا یک نقشه‌ی شبکه‌ای از نقشه‌ی واقعی تولید میکنیم که در آن موانع بی تأثیر حـذف شده‌اند و شکل‌های غیرعادی، عادی‌تر (ساده‌تر) شده‌اند. بعد از ایـن روی ایـن نقشـه‌ی جدیـد الگـوریتم کلاسیک A* اجرا میشود تا مسیری بهینه و بدون مانع از حالت ابتدایی به حالت نهایی ایجـاد شـود. ایـن مسیر را به عنوان یک لیست از رئوس ذخیره میکنیم، سپس با استفاده از دو تـابع P(x, y) و W(x, y) بـه ترتیب مسیرهای ضربدری و مسیرهای sawtooth را مانند تصویر زیر بهینه میکنند.

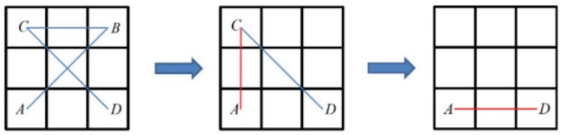


Fig. 2. Cross Path Optimization, using the filter function $P(x, y)$ [7]

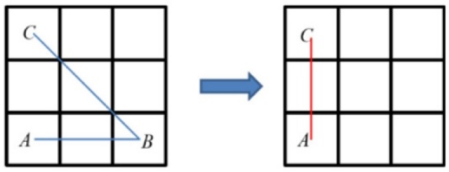


Fig. 3. Sawtooth Path Optimization, using the filter function $W(x, y)$ [7]

در نهایت مسیر ایجاد شده را به مسیرهای B-spline تبدیل میکند که بسیار روانتر از مسیر واقعی تولید شده هستند.

همانطور که در جدول زیر مشخص است این الگوریتم به نسبت الگوریتم کلاسیک کارایی بسیار بهتری دارد، زمان کلی اجرا شدن مسیر کلی طی شده، تعداد چرخش‌ها و به خصوص تعداد رئوسی که در الگوریتم بررسی شدند کمتر شده و این نشان از کارایی بالای این الگوریتم در مقایسه با الگوریتم قبلی دارد.

TABLE I
PERFORMANCE COMPARISON OF A* AND GEOMETRIC A* ALGORITHMS
ENVIRONMENT USED IS SAME AS THAT IN FIG. 4 AND 5 [7]

Path Parameters	A*	Geometric A*
Run time (s)	316.334	295.142
Nodes	131	109
Turns	36	27
Max Turning Angle	45^o	45^o
Expansion Nodes	2246	109
Total Distance (m)	158.167	147.571

در این الگوریتم چون روی نقشه‌ی بسیار ساده‌تر و بهینه‌تری A* را اجرا کرده‌ایم و بعد از آن هم در دو مرحله برای بهینه‌تر کردن و قابل اجراتر (روانتر کردن مسیر) کردن مسیر نهایی تلاش کردیم کارایی الگوریتم بسیار پیشرفت کرد و به طور واضح در خودروهای خودران استفاده از این الگوریتم به دلیل بهینگی بهترو مسیر روانتر نسبت به الگوریتم A* کلاسیک ارجحیت دارد.

برای مثال در عواملی که در محیط‌های ثابت ولی پیچیده از نظر توپولوژی فضایی استفاده میشوند، استفاده از این الگوریتم میتواند گزینه‌ی خوبی باشد چرا که با روش‌هایی که برای بهینه کردن و قابل اجرا کردن مسیر ارائه کرده میتواند مسیرهای بسیار مناسبی را در زمان قابل قبول ارائه کند.

البته این به این معنی نیست که این الگوریتم بهترین روش ممکن است، برای مثال در این الگوریتم روشی برای مقابله با موانع پویا و غیر منتظره ارایه نشده که این خود میتواند جهتی برای بهبود این الگوریتم باشد.

Improved A* Algorithm -3-2

این الگوریتم نسخه‌ای از الگوریتم کلاسیک است که برای کوتاه‌تر و سرراست‌تر کردن مسیر نهایی ایجاد شده.

در این الگوریتم ابتدا یک خط صاف میکشیم که حالت اولیه را به حالت هدف وصل میکند، سپس یک مجموعه از نقاط را در نظر میگیریم که شامل نقاطی از روی خط هستند که مانع میباشند و امکان عبور از آنها نیست. سپس دایره‌هایی به شعاع r و به مرکز این نقاط رسم میکند و نقاطی که هر دایره با خط رسم شده برخورد دارد را به صورت pair ذخیره میکند، هرکدام از این pairها در واقع local initial state و local goal state هستند و باید در نهایت برای هرکدام، مانند حالت کلی، مسیر بهینه پیدا کند. سپس با استفاده از یک مقدار اپسیلون از پیش تعیین شده فاصله‌ی بین این نقاط برخورد (فاصله‌ی بین local goal رأس قبلی و local initial بعدی) را بررسی میکند. اگر این فاصله‌ها از اپسیلون کوچکتر باشند الگوریتم انگار آن دو مانع کوچک را یک مانع بزرگ در نظر میگیرد و سعی میکند مسیری بهینه از local initial state رأس قبلی و local goal state رأس بعدی تولید کند.

در جدول زیر مقایسه‌ی این الگوریتم با الگوریتم A* کلاسیک آمده است، همان طور که مشاهده میکنید با وجود اینکه طول مسیر طی شده کمتر شده ولی زمان اجرای الگوریتم (به دلیل اجرای مداوم و دوباره‌ی آن برای هر حالت جدید ایجاد شده بخاطر موانع روی مسیر) به شدت افزایش یافته.

TABLE II
PERFORMANCE COMPARISON OF A* AND IMPROVED A* ALGORITHM
ENVIRONMENT USED IS SAME AS THAT IN FIG. 7 [8]

Path Parameters	A*	Improved A*
Run time (s)	10.72	41.28
Path Length (m)	28.63	27.75

این الگوریتم در محیط‌هایی که موانع زیادی دارند کارایی بسیار بدی خواهد داشت چون زمان اجرای آن بشدت افزایش می یابد، همچنین مانند الگوریتم‌های قبلی روشی برای مقابله با موانع ناگهانی و غیر منتظره ارائه نمیکند. این دو نقطه ضعف میتوانند جهت‌های خوبی برای بهتر کردن این الگوریتم باشند.

البته در محیط‌هایی که موانع بسیار پراکنده و ثابت هستند و زمان اجرا اولویت چندانی برای ما ندارد ولی در مقابل صاف تر و سر راست تر بودن مسیر برایمان اهمیت بیشتری دارد این الگوریتم میتواند گزینه‌ی بسیار خوبی باشد.

Anytime Repairing A* Algorithm (ARA*) -2-3

این الگوریتم نسخه‌ای از الگوریتم کلاسیک است که برای کوتاه‌تر شدن زمان محاسبه‌ی مسیر بهینه و در نهایت کوتاه‌تر شدن زمان سفر ارائه شده است چرا که در مثالهای واقعی ممکن است زمان در دسترس برای محاسبه‌ی این مسیرهای بدون مانع و بهینه بسیار محدود باشد و با استفاده از روش‌های ارائه شده در بسیاری از موارد ممکن است محاسبه‌ی مسیر بهینه در زمان در دسترس بسیار سخت و حتی غیرممکن باشد.

در این الگوریتم با استفاده ایجاد ابتدایی یک مسأله‌ی بسیار ساده شده از مسأله‌ی (نقشه‌ی) اصلی و حل آن یک پاسخ suboptimal به سرعت پیدا میکنیم و حرکت را با همان شروع میکنیم و به صورت همزمان در جهت بهتر و دقیق تر شدن پاسخ تلاش میکنیم به این شکل که به زیر مسأله شرایط حذف شده را اضافه میکنیم و جواب بهینه‌ی جدید را با استفاده از آنها بدست میاوریم. انقدر به این کار ادامه میدهیم تا زمان در دسترسمان برابر صفر شود.

در این الگوریتم علاوه بر نکات گفته شده، برای هرچه سریعتر شدن زمان اجرا، از نتایج بدست آمده در مراحل قبلی استفاده میکند که این باعث جلوگیری از محاسبه‌ی دوباره‌ی حالت‌هایی میشود که قبلا به درستی محاسبه شده‌اند. این الگوریتم از weighted heuristic استفاده میکنه که باعث میشه پیش بینی‌های ما تاثیر بیشتری روی نتیجه داشته باشند. در این معادله اگر $e = 1$ به معادله‌ی الگوریتم کلاسیک میرسیم. معادله:

$$f(n) = g(n) + e * h(n)$$

در این معادله به ازای $e > 1$ مسأله جواب suboptimal خواهد داشت.

این روش کارایی فوقالعاده خوبی دارد مخصوصا برای خودروها و محیط‌هایی که بعدهای زیادی را باید در نظر بگیرند و در نتیجه محاسبه کنند، دلیل آن هم این است که اگر چه مسیر ممکن است کوتاه‌ترین حالت ممکن نباشد ولی چون محاسبات بسیار زودتر نتیجه میدهند این خودروها بسیار سریع در مسیر تقریبا درست شروع به حرکت میکنند و چون مدام مسیر مورد نظر خود را بهتر میکند این خودروها به مرور مسیری که در آن حرکت میکنند بهینه‌تر و بهینه‌تر میشود و انگار با این روش داریم از زمان انجام محاسباتمان استفاده میکنیم تا زمان کلی اجرای الگوریتم و رسیدن به هدفمان کمتر شود.

البته این الگوریتم همچنان جای پیشرفت دارد چرا که مانند الگوریتم‌های قبلی دارای این ضعف است که در مقابله با موانع پویا و غیرمنتظره روشی بهینه ارائه نکرده است.

Dynamic A* Lite Algorithm (D* Lite) -4-2

این الگوریتم نسخه‌ای از الگوریتم کلاسیک است که برای در نظر گرفتن و مقابله موانعی که ممکن است در حین حرکت در مسیر اتفاق بیفتند ایجاد شده است. اتفاق افتادن یک مانع در حین راه مانند وجود نداشتن یال در آن نقطه است؛ وزن یال مورد نظر بی نهایت میشود. این الگوریتم برای این تغییرات وزنی در گراف آمادگی دارد و با توجه به آنها مدام دوباره مسیر برنامه ریزی میکند.

در این الگوریتم برای هر رأس دو امتیاز در نظر گرفته میشود، G و RHS. امتیاز G که همان G در الگوریتم کلاسیک است، ولی RHS به شکل زیر تعریف میشود:

$$RHS(n) = \min(G(n') + C(n', n))$$

در این معادله n رأس حاضر است و 'n رأس قبلی n میباشد. همچنین C(n', n) هزینه‌ی رفتن از 'n به n است. اگر رأس بعدی مانع داشته باشد C آن برابر بینهایت میشود. با استفاده از این امتیازها الگوریتم یک مسیر از هدف به حالت ابتدایی تنظیم میکند که این مسیر بهینه‌ی مورد نظر را به طور متناطر به ما میدهد.

این الگوریتم برای حرکت در محیط‌های بسیار پویا و غیر قابل پیش بینی و یا ناشناخته بسیار گزینه‌ی مناسبی است چرا که به نسبت الگوریتم اصلی و سایر الگوریتم‌های مطرح شده این ویژگی را دارد که به خوبی با موانع پویا و ناگهانی مقابله کند (که این ویژگی در محیط‌های واقعی بسیار ضروری میباشد)

البته این الگوریتم همچنان جای بهتر شدن دارد، چرا که زمان اجرای آن به دلیل محاسبات مداومی که در مواجهه با هر مانع جدید بالاست و اگر کمتر شود کارایی آن بسیار بالاتر نیز میرود.

Anytime Dynamic A* (AD*) -1-2

این الگوریتم نسخه‌ای از الگوریتم کلاسیک است که با ترکیب دو الگوریتم قبلی مطرح شده به دست آمده، در این الگوریتم مقابله‌ی بهینه با محیط‌های پویا که شامل تغییرات ناگهانی و موانع غیرمنتظره هستند در نظر گرفته شده و علاوه بر آن برای کمتر شدن زمان کلی سفر، از ایده‌ی روش ARA* استفاده شده و با استفاده از زیر مسائل ساده شده و دقیق‌تر کردن آنها به صورت تدریجی (تغییر مقدار e در معادله‌ی مطرح شده در روش ARA*) مسیر را بهتر میکند. پس با ترکیب ایده‌ی این دو الگوریتم الگوریتمی بدست می آید که برنامه ریزی دوباره را براساس موانع جدید به صورت سریع انجام میدهد.

این الگوریتم برای حرکت در محیط‌های بسیار پویا و غیر قابل پیش بینی و یا ناشناخته که در آنها کوتاه بودن زمان کلی سفر برای ما اهمیت دارد، گزینه‌ی خیلی مناسبی است چرا که به نسبت الگوریتم اصلی و سایر الگوریتم‌های مطرح شده این ویژگی را دارد که به خوبی با موانع پویا و ناگهانی مقابله کند (که این ویژگی در محیط‌های واقعی بسیار ضروری میباشد) و بخاطر ویژگی Anytime بودنی که دارد زمان کلی سفر را آنچنان زیاد نمیکند.

البته این الگوریتم همچنان جای بهتر شدن دارد، برای مثال میتواند ایده‌ی geometric A* را هم با آن ترکیب کرد و مسیرهای نهایی کوتاه‌تر و روانتری را تولید کرد.

نتیجه گیری

هدف اصلی این مقاله ذکر، بررسی و مقایسه‌ی انواع مختلف روش‌های مطرح شده برای الگوریتم A* بود که روشی مبتنی بر جست و جوی گراف و بسیار کارآمد در زمینه‌ی خودروهای خودران میباشد. کـه ایـن کارآمدی بالا باعث شده توجه و اهمیت زیادی به این الگوریتم معطوف بشود (چرا که همانطور که پیشتر اشاره شد مسأله‌ی خودروهای خودران بسیار مسأله‌ی پر اهمیتی است و این روش، روشی کارآمد برای این مسأله‌ی پر اهمیت میباشد پس طبیعتا پرداختن به آن از اهمیت بالایی برخوردار است.) و در طـی سـالها نسخه‌های بهتر شده‌ای از آن ارائه شود که در ایـن مقالـه بـه بررسـی برخـی از آنهـا پرداختـه شـده بـود. خلاصه‌ی موارد مطرح شده در جدول زیر به خوبی به نمایش درآمده است.

TABLE IV
A SUMMARY OF ALGORITHMS PRESENTED IN THE PAPER

Algorithm	Advantages	Time Performance
Geometric A*	Expands lesser number of nodes Computes shorter paths than classical A* Generates smoother and more realistic paths well-suited for autonomous vehicles	Faster than classical A*
Improved A*	Computes shorter paths than classical A*	Slower than classical A*
ARA*	Anytime Planner, with significantly higher efficiency in planning in cases as presented in [10]	Faster than classical A*
D* Lite	Allows for re-planning	Faster than the classical D* (dynamic A*) Algorithm
AD*	Anytime planner and allows re-planning	Faster than D* Lite Algorithm

منابع و مراجع

A Survey of A-Star Algorithm Family for Motion Planning of Autonomous Vehicles [1]

https://www.youtube.com/watch?v=iTG7NjQu0Qs [2]

https://www.youtube.com/watch?v=PSX18U1fYEY&t=540s [3]

https://www.youtube.com/watch?v=KHAu5A_flcQ [4]

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Mathematics & Computer Science

BSc report

A Survey of A-Star Algorithm Family for Motion Planning of Autonomous Vehicles

By
Sarina Heshmati

Supervisor 1
Dr. Mehdi Ghatee

Supervisor 2
Dr. Behnam Yousefimehr

October 2023