



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده ریاضی و علوم کامپیوتر

پروژه  
علوم کامپیوتر

بررسی الگوریتم  $A^*$  و کاربرد آن

نگارش  
مهدی عباسعلی پور

استاد راهنما  
جناب آقای دکتر قطعی

مهرماه ۱۴۰۲

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

## چکیده

در این گزارش قصد بررسی یکی از الگوریتم های جست و جو به نام الگوریتم سرچ  $A^*$  را داریم . پس از بیان برخی از مقدمات و پیشینه ، به برخی از شیوه های بهبود این الگوریتم متناسب با فضای مسائل متفاوت می پردازیم . این مسائل غالباً مربوط به وسایل نقلیه خودران و بحث های مسیریابی آنان می باشد . تمرکز بیشتر مقاله به بررسی گستردگی ها و تنوع تغییراتی که می توان بر روی الگوریتم  $A^*$  اعمال نمود . هم چنین سوالاتی که در فایل راهنمای پروژه آورده شده است در داخل متن به آن ها پاسخ داده شده است .

## واژه های کلیدی:

هوش مصنوعی ، حمل و نقل هوشمند، مسیریابی هوشمند، بررسی الگوریتم  $A^*$

# فهرست مطالب

صفحه

عنوان

۲	۱ مقدمه
۳	۱-۱ اهمیت مسیریابی بهینه در سیستم های حمل و نقل خودران
۳	۲-۱ تاریخچه
۴	۲ الگوریتم $A^*$
۵	۱-۲ شیوه ی کار
۷	۳ دگرگونی های الگوریتم $A^*$
۸	۱-۳ ورژن ها
۹	۱-۱-۳ هندسی
۱۱	۲-۱-۳ بهبودیافته
۱۲	۳-۱-۳ پویا ساده
۱۳	۴-۱-۳ همیشه پویا
۱۴	۴ جمع بندی
۱۶	مراجع

شکل	فهرست تصاویر	صفحه
۱-۲	شبه کد مربوط به الگوریتم $A^*$	۶
۱-۳	بهینه سازی مسیر با استفاده از فیلتر $P(x,y)$	۹
۲-۳	بهینه سازی مسیر با استفاده از فیلتر $W(x,y)$	۹
۳-۳	مسیر یابی توسط الگوریتم کلاسیک $A^*$	۱۰
۴-۳	مسیر یابی توسط الگوریتم روش هندسی $A^*$	۱۰
۵-۳	شیوه کار $A^*$ بهبود یافته	۱۱
۱-۴	مقایسه ی گسترش یافته های الگوریتم $A^*$	۱۵



# فصل اول

## مقدمه

## ۱-۱ اهمیت مسیریابی بهینه در سیستم های حمل و نقل خودران

توسعه سیستم های اتومات مانند هواپیماهای بدون سرنشین، وسایل نقلیه هدایت شونده خودکار و ربات های خودکار مزایای بسیاری را برای انسان داشته اند. توسعه وسایل نقلیه خودران منجر به افزایش ایمنی جاده ها و بهبود مصرف انرژی شده است. برای خودران سازی وسایل نقلیه باید نوعی سیستم داشت تا مسیرهای خود را مطابق با محیطی که قرار است در آن حرکت کنند برنامه ریزی کند. خواسته ی ما در این گونه مسائل این است که این مسیرها تا حد امکان کوتاه باشند و وسیله نقلیه به راحتی حرکت کند و از همه مهمتر اینکه بدون مانع باشند. با این حال، تحقیق در مورد برنامه ریزی حرکتی سیستم های خودران جدید نیست و به دهه ۱۹۵۰ برمی گردد، با الگوریتم هایی مانند جستجوی عرضی و جستجوی عمقی در مرحله اولیه تحقیقات برنامه ریزی حرکتی فرموله شده است. از آن زمان تاکنون چندین پیشرفت بزرگ در توسعه الگوریتم های برنامه ریزی حرکت صورت گرفته است. [۲]

. یکی از الگوریتم های مهم برای هوشمندسازی و توانمند سازی این وسایل برای مسیریابی الگوریتم جست و جوی A\* می باشد. هدف اصلی در [۲] بر این است که انواع مختلفی از تغییرات این الگوریتم را در مسائل متفاوت بررسی نماید و آن ها را براساس زمان اجرا و برخی دیگر از ویژگی ها بررسی نماید.

## ۲-۱ تاریخچه

پیتر هارت (Peter Hart)، نیلز نیلسون (Nils Nilsson) و برترام رافائل (Bertram Raphael) از موسسه پژوهشی استنفورد (Stanford Research Institute) که اکنون با عنوان اس آر آی اینترنشنال<sup>۱</sup> فعالیت می کند، برای اولین بار، مقاله ای پیرامون الگوریتم A\* را در سال ۱۹۶۳ منتشر کردند. این الگوریتم را می توان به عنوان افزونه ای از «الگوریتم دیکسترا»<sup>۲</sup> در نظر گرفت که توسط «ادسخر دیکسترا»<sup>۳</sup> در سال ۱۹۵۹ ارائه شده است. الگوریتم A\* با بهره گیری از «الگوریتم جستجوی کاشف» (جستجوی هیوریستیک Heuristics Search) برای هدایت فرایند جستجو، به کارایی بهتری دست پیدا می کند [۱].

<sup>۱</sup>SRI International

<sup>۲</sup>Dijkstra's Algorithm

<sup>۳</sup>Edsger Dijkstra



## فصل دوم

## الگوریتم $A^*$

## ۱-۲ شیوه ی کار

کاری که الگوریتم  $A^*$  انجام می‌دهد آن است که در هر گام، گره را متناسب با مقدار  $f$  که پارامتری مساوی با مجموع دو پارامتر دیگر  $g$  و  $h$  است انتخاب می‌کند. در هر گام، گره/خانه‌ای که دارای کمترین مقدار  $f$  است را انتخاب و آن گره را پردازش می‌کند.  $g$  و  $h$  به روش ساده‌ای که در زیر بیان شده است محاسبه می‌شوند.

- $g$  هزینه حرکت از نقطه آغاز به یک مربع خاص در شبکه، با دنبال کردن مسیری که برای رسیدن به آن تولید شده است.

- $h$  هزینه تخمین زده شده برای حرکت از یک خانه داده شده در شبکه به مقصد نهایی است. از  $h$  معمولا با عنوان هیوریستیک یاد می‌شود. هیوریستیک چیزی به جز نوعی حدس هوشمندانه نیست. کاربر واقعا فاصله واقعی را تا هنگام یافتن مسیر نمی‌داند، زیرا هر مانعی (دیوار، آب و سایر موانع) ممکن است در مسیر باشد. راه‌های زیادی برای محاسبه  $h$  وجود دارد به عنوان مثال فاصله ی مستقیم اقلیدسی یک ز مواردی است که می‌تواند در ماسئل به عنوان این حدس اولیه در نظر گرفته شود.

الگوریتم  $A^*$  به نحوی الگوریتم دایکسترا است که راس ها را با اولویت  $f(n) = g(n) + h(n)$  انتخاب می‌کند.

**Algorithm 1** Classical A\* Algorithm

---

```

OpenList  $\leftarrow$  [StartingNode]
ClosedList  $\leftarrow$  []
 $g(start) = 0$ 
 $h(start) = heuristic(start, end)$ 
 $f(start) = g(start) + h(start)$ 
while OpenList  $\neq \phi$  do
   $n \leftarrow$  node with lowest f in OpenList
  if  $n = goal$  then
    return ClosedList
  else
    OpenList  $\leftarrow$  OpenList  $\setminus n$ 
    ClosedList  $\leftarrow$  ClosedList  $\cup n$ 
    for all  $n'$  (each  $n'$  is a neighbor of  $n$ ) do
       $cost_{nn'} = g(n) + distance(n, n')$ 
      if  $n' \in ClosedList$  then
        continue
      else if  $n' \in OpenList$  and  $cost_{nn'} < g(n')$  then
        OpenList  $\leftarrow$  OpenList  $\setminus n'$ 
      else if  $n' \in ClosedList$  and  $cost_{nn'} < g(n')$  then
        ClosedList  $\leftarrow$  ClosedList  $\setminus n'$ 
      else
        OpenList  $\leftarrow$  OpenList  $\cup n'$ 
         $f(n') = g(n') + h(n')$ 
      end if
    end for
  end if
end while

```

---

شکل ۲-۱: شبه کد مربوط به الگوریتم A\*

[۲]

## فصل سوم

### دگرگونی های الگوریتم $A^*$

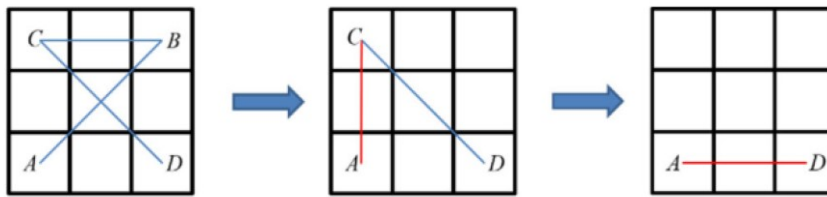
## ۱-۳ ورژن ها

این الگوریتم به شیوه های متفاوت متناسب با شرایط فضای مسئله گسترش یافته است . در ادامه به برخی از این ورژن ها می پردازیم :

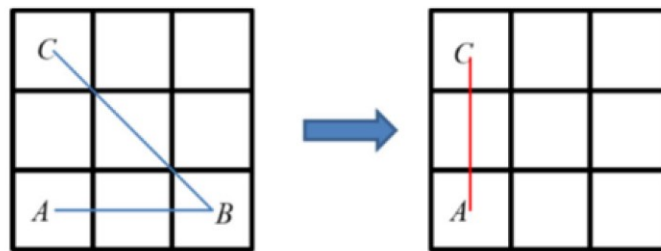
## ۱-۱-۳ هندسی

الگوریتم هندسی  $A^*$  نخستین بار در مسائل مسیریابی در محیط های بندری بیان شد؛ در این محیط ها عامل علاوه بر وظیفه ی حمل و نقل کالا، بایستی خود را در زمان مناسب به ایستگاه شارژ می رساندند. این الگوریتم اساساً برای رسیدگی به مسائلی مانند زوایای چرخش بزرگ، گره های متعددی که معمولاً در مسیرهای متقاطع وجود دارند و مسیرهای دنداناره ای که توسط الگوریتم کلاسیک  $A^*$  تولید می شوند، توسعه داده شد.

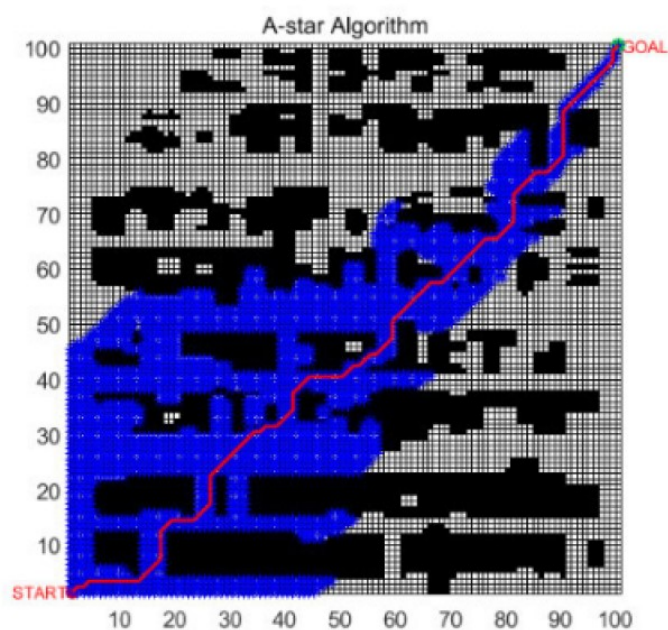
$A^*$  هندسی ابتدا یک نقشه شبکه ای از محیط ایجاد می کند و موانع غیر ضروری را از بین می برد و اشکال نامنظم را منظم می کند. پس از این، الگوریتم کلاسیک  $A^*$  برای به دست آوردن یک مسیر بدون مانع از موقعیت شروع تا موقعیت نهایی اعمال می شود. چنین مسیرهایی به عنوان لیستی از نقاط به دست می آیند. سپس الگوریتم هندسی  $A^*$  با استفاده از توابع فیلتر  $P(x, y)$  (۱-۳) و  $W(x, y)$  (۲-۳) گره های نامعتبر را از این لیست فیلتر می کند. نتایج حاصل از کار این فیلتر نشان می دهد که تعداد گره های بررسی شده ی حاصل از این ورژن نسبت به حالت کلاسیک کاهش قابل ملاحظه ای یافته است و در یک نمونه مسئله از ۲۲۴۶ به ۱۰۹ مورد رسیده است. در ۳-۳ و ۴-۳ نقاط آبی گره های بررسی شده توسط الگوریتم و نقاط قرمز مسینهایی الگوریتم می باشند.



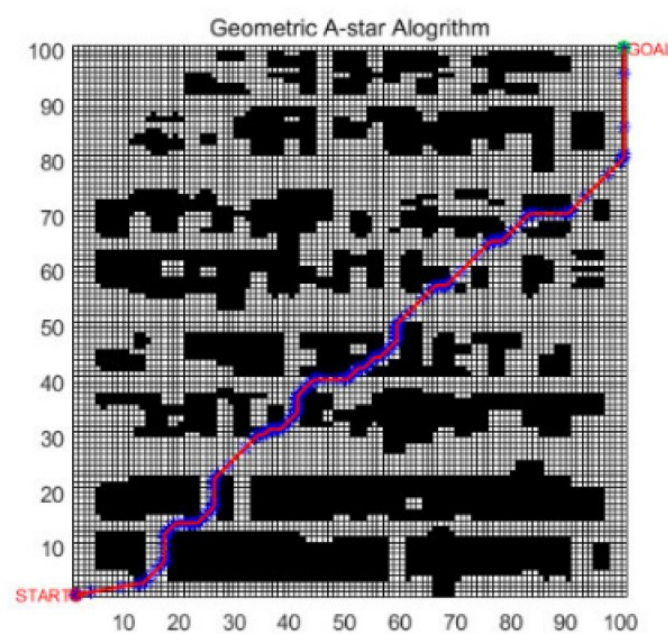
شکل ۱-۳: بهینه سازی مسیر با استفاده از فیلتر  $P(x, y)$  [۲]



شکل ۲-۳: بهینه سازی مسیر با استفاده از فیلتر  $W(x, y)$  [۲]



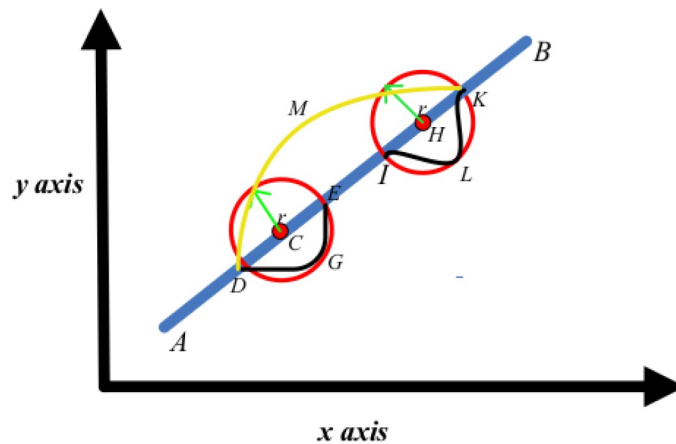
شکل ۳-۳: مسیر یابی توسط الگوریتم کلاسیک A\* [۲]



شکل ۳-۴: مسیر یابی توسط الگوریتم روش هندسی A\* [۲]

## ۳-۱-۲ بهبود یافته

این الگوریتم ابتدا هدف و موقعیت شروع را توسط یک خط مستقیم به هم متصل می کند. سپس فهرستی از مراکز، که نقاطی روی خط هستند که برخورد کردن به آن ها ممنوع می باشد را ایجاد می کند. برای هر مرکز، دایره ای به شعاع معین  $r$  در اطراف این نقاط می سازد، و سپس مختصات محل های برخورد این دایره ها را با خط پیدا می کند به عنوان نمونه شکل ۳-۵ را در نظر بگیرید. سپس با انتخاب هر یک از این جفت ها به عنوان نقطه شروع و هدف محلی، با استفاده از آستانه  $\epsilon$  بررسی می کند که آیا هدف محلی  $E$  و شروع محلی بعدی  $I$  به یکدیگر نزدیک هستند یا خیر. اگر نزدیک باشند، هدف محلی فعلی به هدف محلی بعدی تغییر می کند، یعنی برنامه ریز باید مسیری را از  $D$  به  $K$ ، همانطور که در شکل نشان داده شده است، برنامه ریزی کند. پس از انجام این کار، الگوریتم با استفاده از الگوریتم  $A^*$ ، مسیرهای موضعی را در اطراف همه جفت های  $(D,E)$  و  $(D,K)$  مسیریابی می کند. در نهایت مسیری حاصل می شود که بخشی از آن شامل خطی راست است که کواه ترین مسیر است و موانع با استفاده از الگوریتم  $A^*$  دور زده می شوند. در نتایج تجربی این الگوریتم مسیر بهتری نسبت به حالت کلاسیک پیدا کرد اما زمان بیشتری صرف شد (تقریباً ۴ برابر).



شکل ۳-۵: شیوه کار  $A^*$  بهبود یافته [۲]



## ۳-۱-۳ پویا ساده

الگوریتم  $D^* \text{ Lite}$ ، مانند الگوریتم  $A^*$  کلاسیک با شروع از یک گره سعی می کند تابه نقطه ی هدف برسد اما با این فرض که گراف بتواند تغییر کند . تشخیص یک مانع همان اثری را بر روی نمودار خواهد داشت که با حذف یال بین دو گره که مانع بین آنها تشخیص داده می شود، و این معادل تغییر وزن یال به  $\infty$  است. الگوریتم  $D^* \text{ Lite}$  چنین تغییرات وزنی را در نمودار محاسبه می کند و به طور مداوم مسیر خودروی خودران را دوباره برنامه ریزی می کند. الگوریتم  $D^* \text{ Lite}$  دو امتیاز را برای هر گره از گراف در نظر می گیرد  $G$  و  $RHS$ . که همانند  $A^*$  امتیاز  $G$  هزینه رفتن از گره شروع به گره فعلی در گراف می باشد. امتیاز

$RHS$  به صورت  $RHS(n) = \min(G(n) + C(n, n))$  تعریف می شود که در آن،  $n$  گره فعلی،  $n'$  گره قبلی است. و  $C(n', n)$  هزینه جابجایی از  $n$  به  $n'$ . برای یافتن گره بهینه بعدی استفاده می شود. اگر گره بعدی مسدود شود،  $C$  برای آن لبه روی  $\infty$  تنظیم می شود، در نتیجه،  $RHS$  نیز روی  $\infty$  تنظیم می شود. با استفاده از این امتیازها، الگوریتم مسیری را از هدف تا شروع برنامه ریزی می کند که در نهایت یک مسیر بهینه را برای دنبال کردن به دست می دهد. در صورت مواجهه با یک مانع ناشناخته قبلی، امتیاز  $RHS$  با هزینه های جدید اتصال برای گره های آسیب دیده، دوباره محاسبه می شود. نتایج نشان می دهد که  $D^* \text{ Lite}$  منجر به جست و جوی راس اضافی کمتر، گسترده شدن کمتری می شود، که ثابت می کند که  $D^* \text{ Lite}$  در واقع الگوریتم خوبی برای برنامه ریزی مجدد و حرکت در محیط های ناشناخته است.

## ۴-۱-۳ همیشه پویا

یکی از مهمترین مواردی که باید رد نظر گرفته شود محیط های پویاست . از آنجایی که فرد می خواهد کل زمان سفر به حداقل برسد، باید برنامه ریزی راه حل ها را با بیشترین سرعتی که می تواند محاسبه کند. در صورت برخورد با یک مانع جدید، زمان سفر با زمان اضافی برای محاسبه یک مسیر جدید افزایش می یابد. برای مقابله با این مشکل، الگوریتم  $ADA^*$  با ترکیب ایده های دو الگوریتم قبلاً مورد بحث،  $D^*lite$  و  $ARA^*$  توسعه یافت. مشابه  $ARA^*$ ،  $AD^*$  راه حل هایی غیر بهینه را ابتدا جست و جو می کند . تغییری که در محیط شناسایی شود (از طریق سنسور یا دوربین)، به عنوان تغییر در گرافی که مسیر در آن برنامه ریزی شده است، منعکس خواهد شد. گره های آسیب دیده در فهرست باز مانند الگوریتم  $D^*$   $Lite$  ارسال می شوند، با اولویت های آنها در میان مقدار کلید قبلی و مقدار کلید به روزرسانی می شوند . سپس گره های موجود در لیست پردازش می شوند تا زمانی که راه حل فعلی تضمین شود بنابراین  $ADA^*$ ، که یک الگوریتم برخط است، امکان برنامه ریزی مجدد سریع راه حل ها را با محدود کردن بهینه بودن آنها فراهم می کند.

## فصل چهارم

### جمع بندی

حوزه خودروهای خودران پیشرفت چشمگیری داشته است، اما هنوز هم دامنه پیشرفت زیادی در آینده دارد. [۲] در ابتدا تکنیک‌های مختلف برنامه‌ریزی حرکتی را ارائه کرد که تمرکز اصلی آن بر روی الگوریتم جستجوی  $A^*$  و تغییرات آن بود. چندین گونه از الگوریتم  $A^*$  مورد بررسی قرار گرفت و نشان داده شد که چگونه این تغییرات به حل مسائل با الگوریتم کلاسیک  $A^*$  کمک می‌کند. بسیاری از تغییرات الگوریتم  $A^*$  را می‌توان در آینده با بهینه سازی الگوریتم کلاسیک یا انواع آن توسعه داد و آزمایش کرد. یکی از این تغییرات بالقوه، اصلاح تابع اکتشافی مورد استفاده در این الگوریتم‌ها است. نویسندگان امیدوار است که این مقاله در ارائه یک نمای کلی از انواع مختلف الگوریتم  $A^*$  موفق باشد، که می‌تواند به عنوان مرجع در حین کار بر روی انواع جدید برای ارائه عملکرد بهتر نسبت به الگوریتم‌های برنامه‌ریزی حرکتی موجود استفاده شود. در روش‌های متفاوتی که بررسی شد ورژن هندسی الگوریتم  $A^*$  برای زمانی مطلوب است که نیاز به مسیری تاحد امکان صاف داریم و در این حالت الگوریتم هندسی با کاهش تعداد گره‌های گسترده شده زمان یافتن مسیر بهینه را به شدت کاهش می‌دهد. از مزایای آن می‌توان این موارد را نام برد: تعداد کمتری از گره‌ها را گسترش می‌دهد، محاسبه مسیرهای کوتاه‌تر از  $A^*$  کلاسیک، مسیرهای هموارتر و واقعی‌تر را ایجاد می‌کند، و برای وسایل نقلیه خودران مناسب است. الگوریتم  $improved A^*$  مسیر کوتاه‌تری را نسبت به حالت کلاسیک در مسئله‌ی بررسی شده در مقاله پیدا کرد اما زمانی بیشتر صرف شد. بنابراین زمان اجرا می‌توند نقطه ضعف این الگوریتم باشد. الگوریتم‌های دیگر مثل  $ARA^*$  و  $ADA^*$  زمان اجرا را بسیار پایین می‌آوردند و بسیار برای کاربردهای برخط مناسب اند. همین‌طور  $Anytime Dynamic A^* (AD^*)$  و  $ADA^*$  برای محیط‌های پویا مناسب اند. در ۴-۱ می‌توانید یک نمای کلی از نظر تعداد گره‌های گسترش یافته برای هریک از الگوریتم‌ها در یک مسئله خاص مشاهده نمایید. الگوریتم  $A^*$  پتانسیل بالایی دارد تا با سایر الگوریتم‌ها ترکیب شود و الگوریتم‌هایی با عملکرد زمانی و بهینگی بهتری پدید آورد. به عنوان مثال می‌توان تابع هیوریستیک را تغییر داد یا برای انتخاب هر گره معیارهای دیگری نیز افزود و یا ضریب  $\epsilon$  را در طول مسیر تغییر داد و با نزدیک شدن به هدف به سمت ۱ برد.

Algorithm	Number of states expanded	Final Path
$A^*$	31	Optimal
$A^*$ with $\epsilon = 2.5$	19	Optimal
$D^*$ Lite	27	Optimal
$D^*$ Lite with $\epsilon = 2.5$	13	Sub-Optimal
$ARA^*$	24	Optimal
$AD^*$	20	Optimal

شکل ۴-۱: مقایسه‌ی گسترش یافته‌های الگوریتم  $A^*$

[۲]

## مراجع

- [1] Hesaraki, Elham. A\* algorithm in a simple way.
- [2] Paliwal, Pulkit. A survey of a-star algorithm family for motion planning of autonomous vehicles. In *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pages 1–6. IEEE, 2023.