بنام خدا

هندسه محاسباتی
فصل سیزدهم
برنامه ریزی حرکت روبات

محمدکاظم فعال کاری

خرداد 1391

مقدمه

- هدف نهایی علم روباتیک ساخت یک روبات خودمختار است.

یعنی اینکه بگوییم چه کاری انجام دهد بدون اینکه بگوییم چگونه انجام دهد.

- در نتیجه روبات باید قدرت تصمیم گیری داشته باشد.
- یکی از مهمترین این تصمیمات، تصمیم در مورد حرکت است.
- این مسئله به مسئله برنامه ریزی حرکت روبات[1] معروف است.

1- robot motion planning

مقدمه

- لذا یک روبات برای تصمیم گیری در مورد حرکتش باید اطلاعاتی در مورد محیط خود داشته باشد.
- بنابراین در ابتدا اطلاعاتی را درمورد محیط به او میدهیم.
- اما آیا میتوان در ابتدا تمام اطلاعات مربوط به محیط را به او داد؟
  - پاسخ خیر است. چون اکثر اوقات محیط اطراف روبات پویا است.
- مثلا روباتی که در یک کارخانه در حال حرکت است.
  - محیط ایستا: دیوارها، ماشینها و تجهیزات صنعتی و ...
  - محیط پویا: افرادی که در کارخانه درحال تردد هستند.

مقدمه

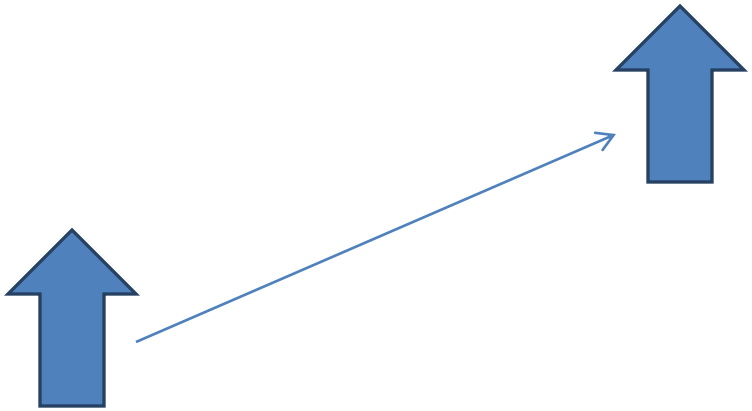
- لذا در ابتدا فقط اطلاعات مربوط به محیط ایستا را به روبات میدهیم و برای بدست آوردن سایر اطلاعات از سنسورهایش کمک میگیرد.
- به این اطلاعات ابتدایی برنامه سطح [1] میگوییم.

1- Floor plan

انواع روبات

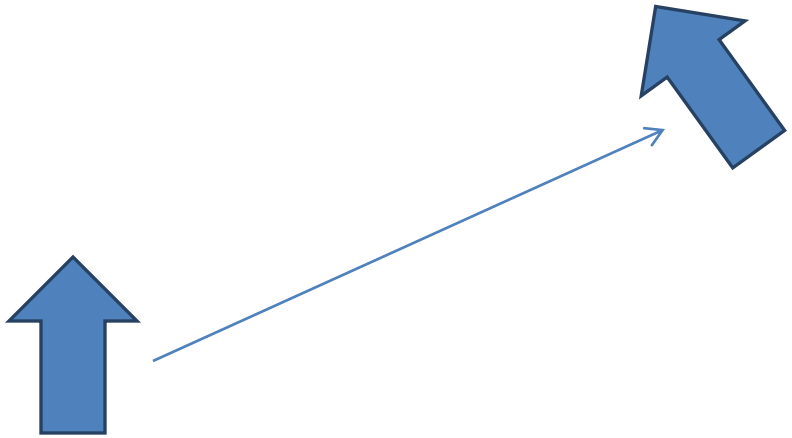
- روبات ها را از لحاظ نوع حرکت میتوان به دو نوع تقسیم کرد:

1- روبات هایی که فقط قابلیت جابجایی دارند.(translation robot)



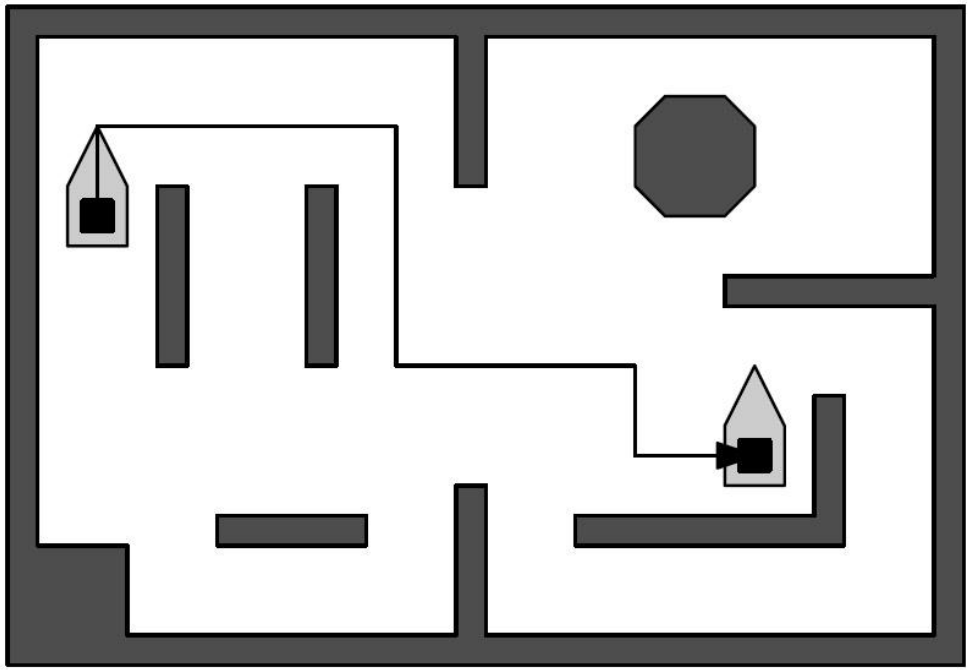
انواع روبات

2- روبات هایی که درحین حرکت چرخش هم دارند.(rotation robot)



طرح مسئله

- مسئله پیدا کردن مسیری از نقطه مبدا به نقطه مقصد بدون برخورد با محیط (موانع [1]) است.



1- obstacles

فرضیات مسئله

- برای ساده کردن مسئله، فرضیات زیر را درنظر میگیریم:
  - فضای مسئله دوبعدی است.
  - موانع و روبات هم دوبعدی هستند.
  - محیط مسئله استاتیک است.
  - روبات قابلیت دوران ندارد (فقط جابجایی است).

فضای کاری[1] و فضای پیکربندی[2]

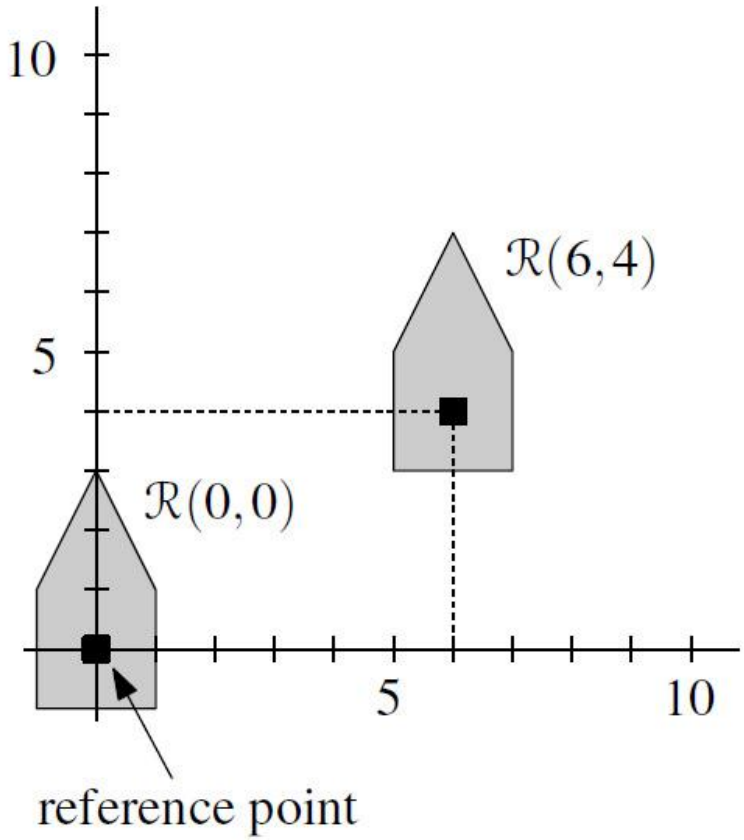
- محیطی که روبات در حال حرکت در آن است و شامل مجموعه موانع $S=\{p_1,p_2,\ldots,p_t\}$ را **فضای کاری** میگوییم.
- هر روبات در فضای کاری را با یک **نقطه مرجع** [3] و حرکت آن را با یک بردار انتقال نشان میدهیم.
- اگر روبات توانایی تغییر جهت هم داشته باشد نیاز به پارامتر دیگری بنام ϕ داریم که میزان چرخش آن را در جهت خلاف عقربه های ساعت نشان میدهد.

1- work space
2- configuration space
3- reference point

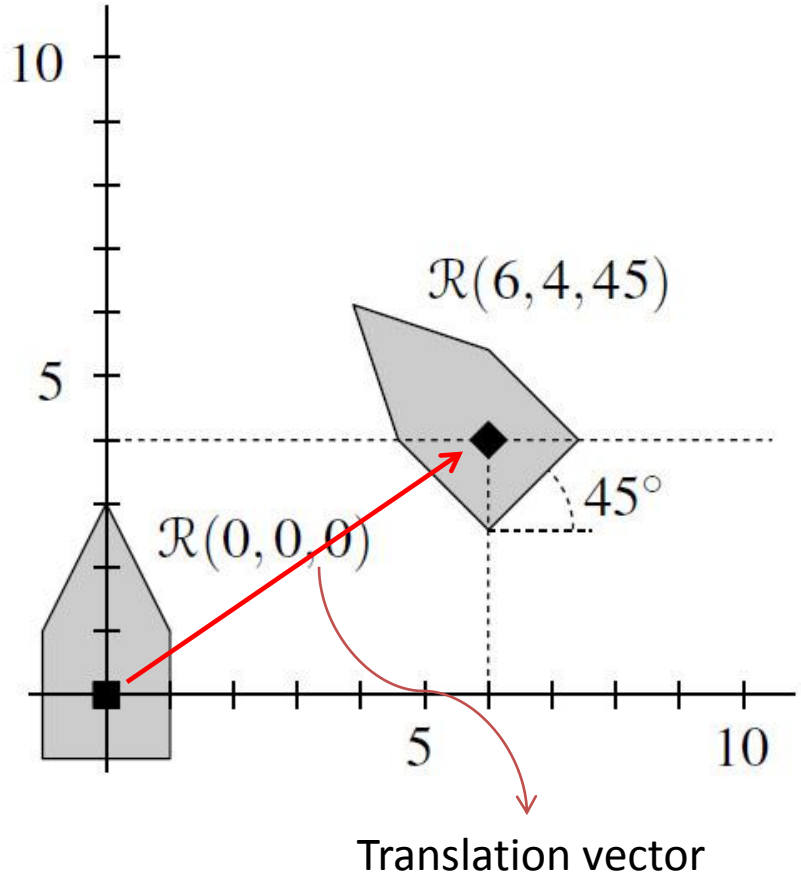
فضای کاری و فضای پیکربندی

- یک روبات جابجایی که نقطه مرجع آن در مختصات (x,y) قرار دارد را با R(x,y) نشان میدهیم.
- این امر را در رباتهای دورانی با R(x,y, ϕ) نشان میدهیم.

Translation robot



Rotation robot



درجه آزادی[1]

- در حالت کلی تعداد پارامترهای مورد نیاز برای تشخیص مکان و وضعیت یک روبات را درجه آزادی (DOF) روبات مینامیم.
- این مقدار در یک فضای دوبعدی، برای روبات انتقالی **دو** وبرای روبات دورانی **سه** است. همچنین در یک فضای سه بعدی برای روبات انتقالی **سه** و برای روبات دورانی **شش** خواهد بود.
- به درجه آزادی یک روبات فضای پارامتری آن نیز میگویند.

1- degrees of freedom

فضای پیکربندی[1]

- فضای پارامتری روبات R را فضای پیکربندی آن میگوییم و آن را با C(R) نشان میدهیم.

فضای ممنوع[1] و فضای آزاد[2]

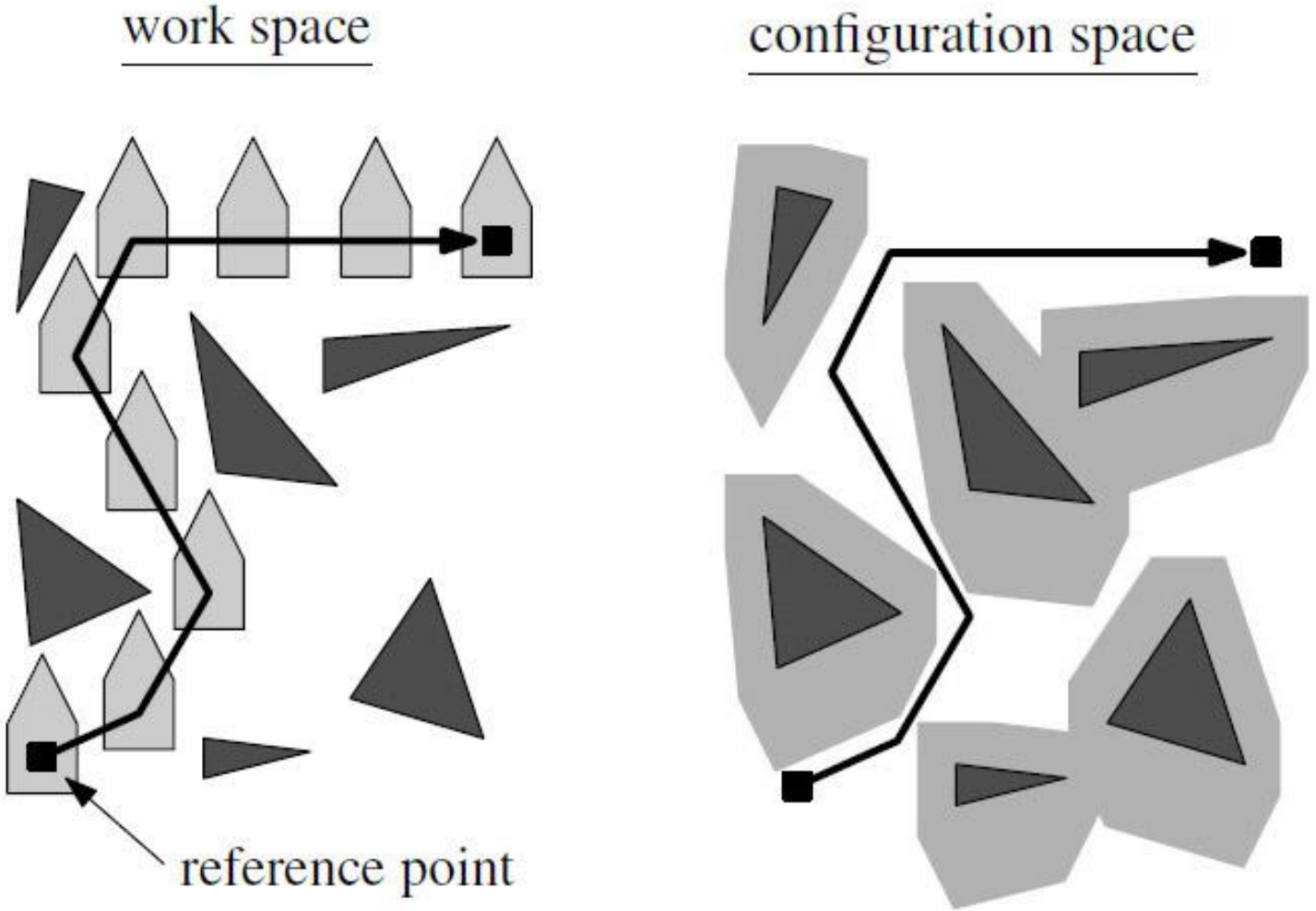
- فضای پیکربندی روبات انتقالی در فضای دو بعدی، یک صفحه دو بعدی اقلیدسی است.
- نقاطی از صفحه که در آن نقاط، روبات R با موانع موجود در S اشتراک دارد را فضای پیکربندی ممنوع یا بطور خلاصه فضای ممنوع مینامیم و آن را با $C_{forb}(R,S)$ نشان میدهیم.
- باقیمانده فضای پیکربندی که در آن روبات با هیچ مانعی اشتراک ندارد را فضای آزاد مینامیم و آن را با $C_{free}(R,S)$ نشان میدهیم.

1- forbidden space
2- free space

مسیر بدون برخورد[1]

- مسیر یک روبات در فضای کاری (محیط)، بوسیله یک منحنی، در فضای پیکربندی نگاشت میشود.
- یک مسیر بدون برخورد، مسیری در فضای آزاد روبات است.

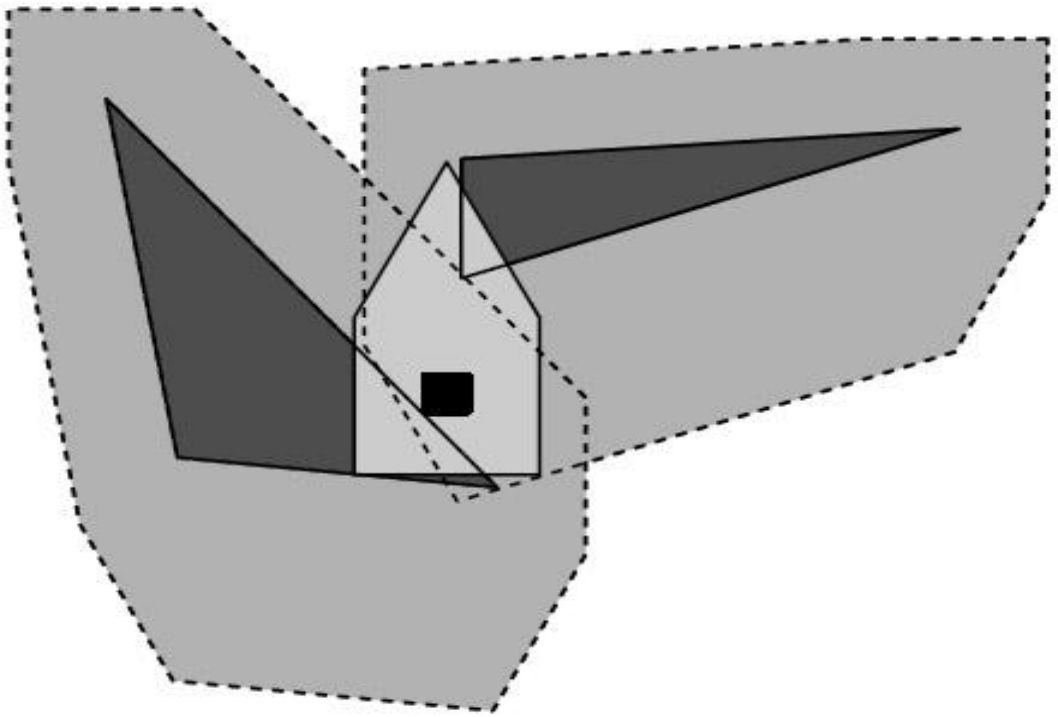
1- collision-free path


work space
configuration space
reference point

فضای پیکربندی موانع

- یک مانع p در مجموعه نقاط p در فضای پیکربندی نگاشت میشود بطوریکه R(p) با p مشترک باشد. مجموعه حاصل را فضای پیکربندی موانع نامیده و با C-obstacle نشان میدهیم.
- C-obstacle ها ممکن است حتی در زمانی که موانع در فضای کاری از هم جدا هستند، همپوشانی داشته باشند. این زمانی اتفاق می افتد که روبات در یک لحظه با چند مانع بطور همزمان برخورد داشته باشد.

- این امر در تصویر زیر قابل ملاحظه است.



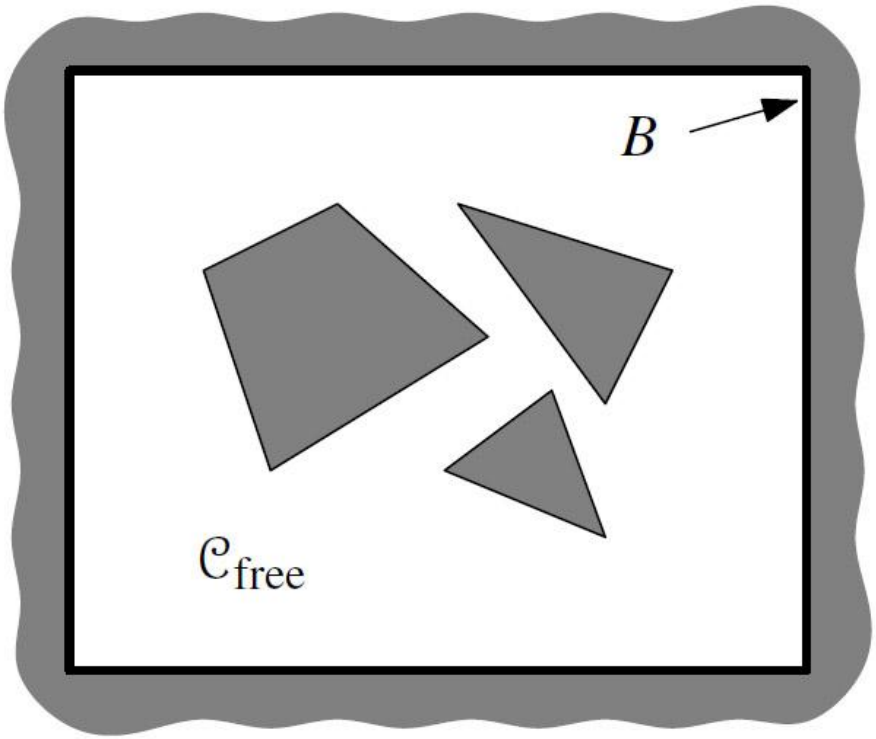
یک سوال

- سوالی که ممکن است در اینجا پرسیده شود اینست که برخورد روبات با مانع چگونه است؟ به عبارتی آیا لمس کردن مانع توسط روبات برخورد حساب میشود؟
- جواب این پرسش با این پرسش که «آیا موانع مجموعه هایی باز هستند یا بسته؟» برابر است.
- جواب: ما فرض میکنیم که موانع، مجموعه هایی باز هستند و در نتیجه با لمس کردن یک مانع توسط روبات برخوردی رخ نخواهد داد.

روبات نقطه ای

- قبل از بررسی حرکت روبات چندضلعی، ابتدا حرکت یک روبات نقطه ای را بررسی میکنیم.
- مانند حالت قبل :
  - روبات را با R نمایش می دهیم.
  - موانع را با مجموعه $S=\{p_1,...,p_t\}$ نمایش می دهیم.
  - موانع، چندضلعی های جدا از هم می باشند که مجموع رئوس آنها n است.
- توجه شود که برای یک روبات نقطه ای فضای کاری با فضای پیکربندی یکسانند و نقطه مرجع خود روبات است.

- برای ساده سازی مسئله، حرکت روبات را درون یک فضای بزرگ و بسته B که شامل همه موانع است محدود می کنیم.



- در این صورت فضای آزاد C_{free} آن قسمت از B است که توسط موانع پوشیده نشده است. به عبارتی دیگر داریم :

$$\mathcal{C}_{\text{free}} = B \setminus \bigcup_{i=1}^{t} \mathcal{P}_i.$$

- فضای آزاد ممکن است ناحیه ای همبند نباشد و دارای حفره باشد.
- هدف ما پیدا کردن ناحیه ای از فضای آزاد است که به ما اجازه می دهد به ازای هر نقطه شروع و پایان مسیری را پیدا کنیم.
- برای این کار از ترسیم ذوزنقه ای که در فصل 6 مطرح شد استفاده میکنیم.

الگوریتم محاسبه فضای آزاد

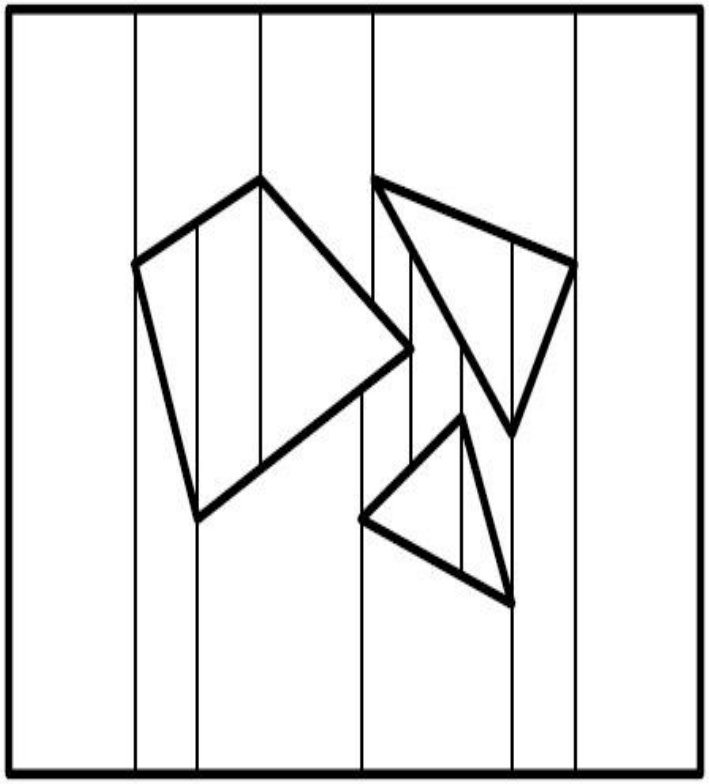
Algorithm COMPUTEFREESPACE(S)

Input. A set S of disjoint polygons.

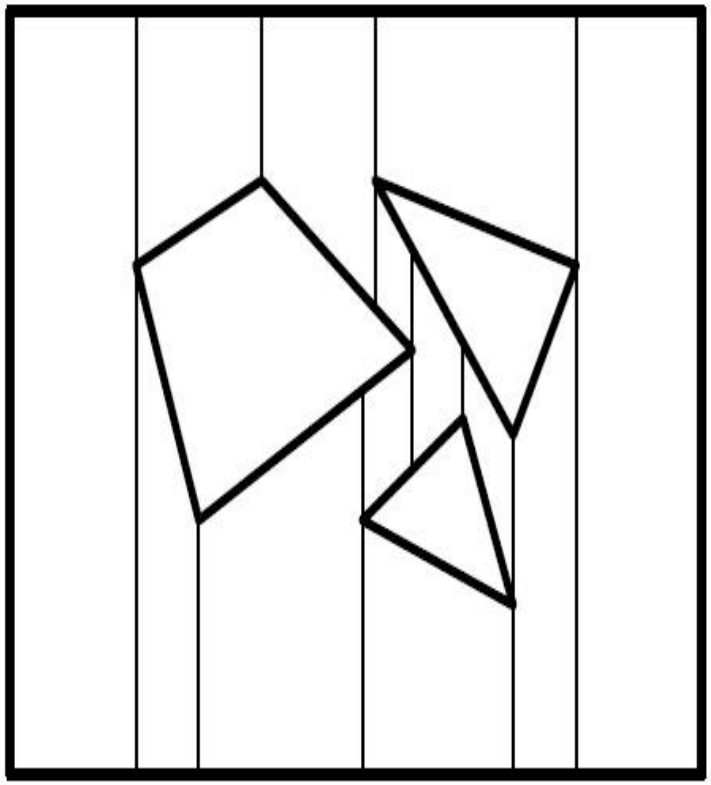
Output. A trapezoidal map of $\mathcal{C}_{\text{free}}(\mathcal{R}, S)$ for a point robot $\mathcal{R}$.

1. Let E be the set of edges of the polygons in S.
2. Compute the trapezoidal map $\mathcal{T}(E)$ with algorithm TRAPEZOIDALMAP described in Chapter 6.
3. Remove the trapezoids that lie inside one of the polygons from $\mathcal{T}(E)$ and return the resulting subdivision.

(a)



(b)



یک سوال مهم

- سوالی که در اینجا مطرح میشود اینست که «چگونه ذوزنقه های درون موانع را حذف میکنیم؟»
- بعد از اجرای الگوریتم **افراز ذوزنقه ای**، برای هر ذوزنقه میدانیم که چه یالی آن را از بالا محدود کرده است. و همچنین میدانیم که هر یال متعلق به کدام مانع است. لذا برای تصمیم گیری در مورد حذف یک یال فقط کافی است بررسی کنیم که آیا این یال ذوزنقه را از بالا محدود کرده یا از پایین.
- به عنوان مثال از راست ترین نقطه مانع به طرف چپ ترین نقطه آن حرکت میکنیم (در جهت عقربه های ساعت) و هر یالی که از بالا با یال مانع برخورد داشت را حذف می کنیم.

- این کار در زمان ثابت انجام پذیر است.
- زیرا یالهای موانع را (بصورت لیست شده) در اختیار داریم.

لم 13.1

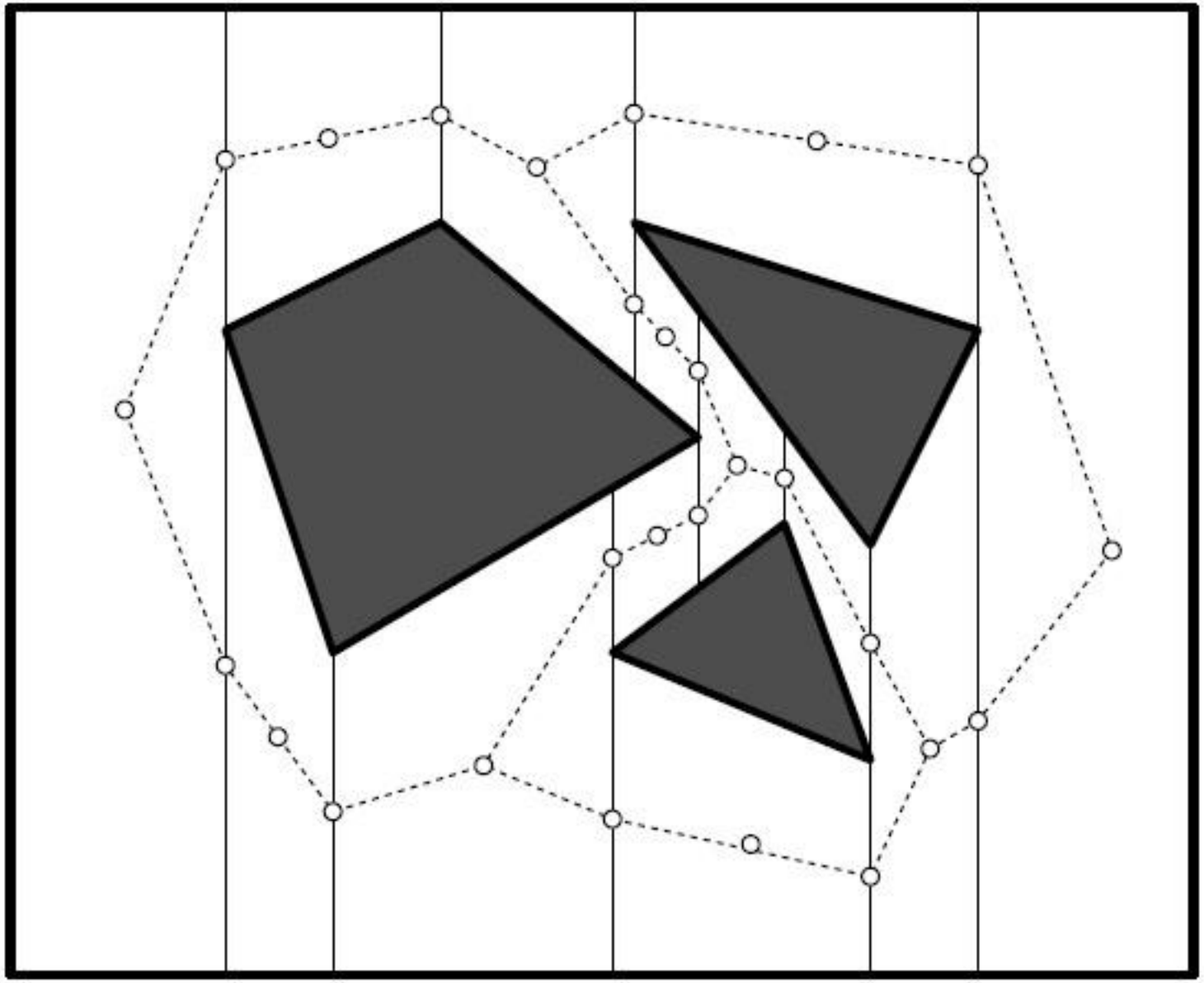
- افراز ذوزنقه ای فضای آزاد برای یک روبات نقطه ای که در میان مجموعه ای از موانع چند ضلعی جدا از هم با مجموع n یال حرکت می کند، می تواند با یک الگوریتم تصادفی در زمان مورد انتظار O(nlogn) محاسبه شود.
- در ادامه، این نقشه ذوزنقه ای را با $T(C_{free})$ نشان می دهیم.

- اما چگونه میتوانیم با استفاده از $T(C_{free})$ مسیری از نقطه شروع p_{start} به نقطه پایان p_{goal} پیدا کنیم؟
- اگر نقطه شروع و نقطه پایان در یک ذوزنقه باشند این کار به راحتی با یک حرکت مستقیم از p_{start} به p_{goal} امکان پذیر است.
- اما اگر نقطه شروع و نقطه پایان در دو ذوزنقه متفاوت باشند، مسیر بین آنها از تعدادی ذوزنقه عبور خواهد کرد و همچنین ممکن است در نقاطی گردش داشته باشد.
- لذا برای هدایت حرکت روبات در میان این ذوزنقه ها، ما نقشه راه را میسازیم.

نقشه راه[1]

- نقشه راه یک **گراف مسطح** است که در فضای آزاد ساخته شده و آن را با g_{road} نشان میدهیم.
- بجز در قسمتهای ابتدایی و انتهایی، مسیر نقشه راه را دنبال خواهد کرد.
- نحوه ساخت نقشه راه:
  - یک نقطه در مرکز هر ذوزنقه قرار میدهیم.
  - یک نقطه در میانه هر گسترش عمودی قرار می دهیم.
  - بین هر نقطه مرکزی ذوزنقه و نقاط روی مرزهای همان ذوزنقه یک یال رسم میکنیم.
- نقشه راه را میتوان در زمان $O(n)$ با پیمایش DCEL مربوط به $T(C_{free})$ ساخت.

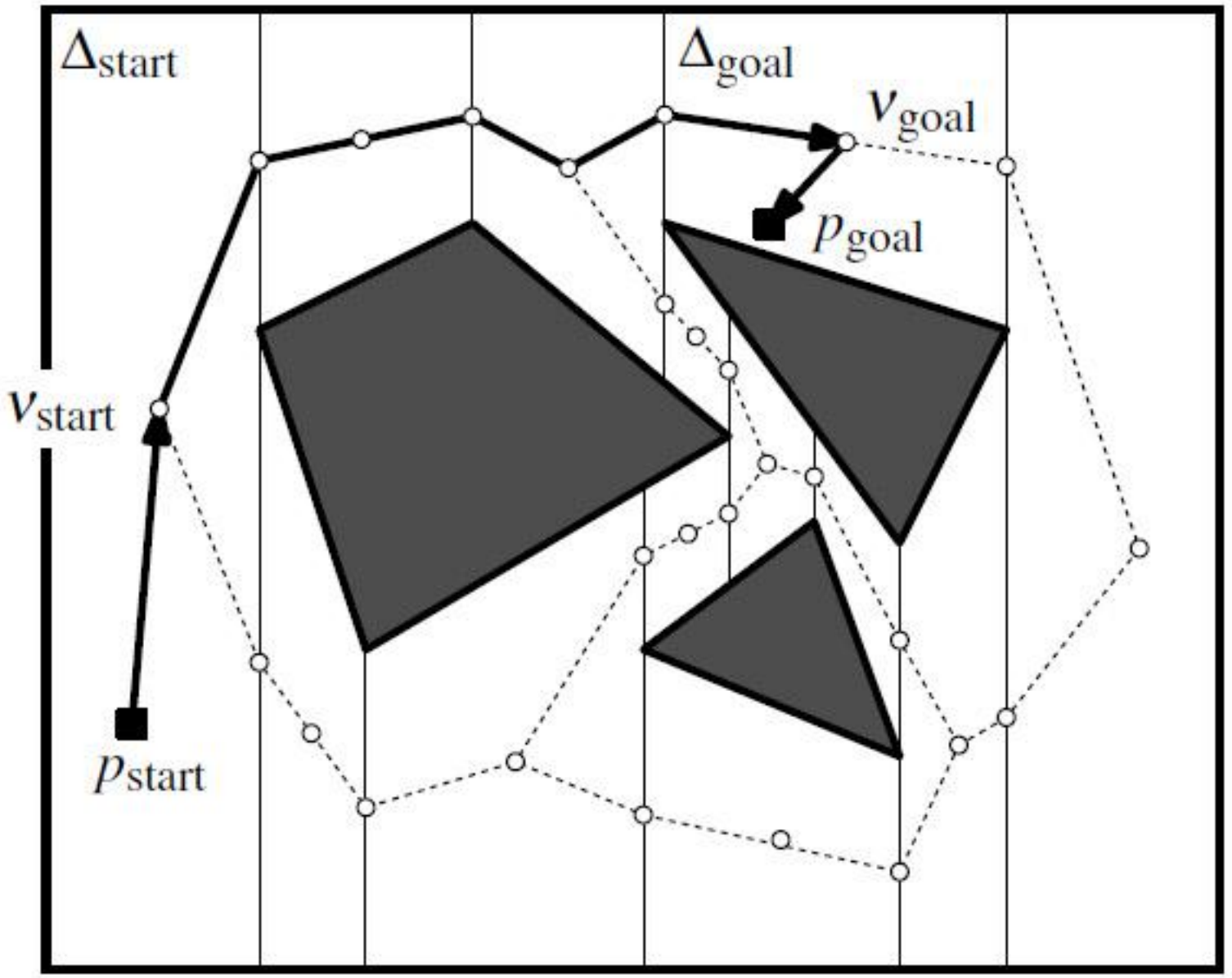
1- road map



برنامه ریزی حرکت با استفاده از نقشه راه

- اکنون میتوانیم از نقشه راه و افراز ذوزنقه ای برای طراحی حرکت روبات از نقطه شروع به نقطه انتها استفاده کنیم.
- برای این کار ابتدا ذوزنقه های Δ_{start} و Δ_{goal} که شامل این نقاط میباشند را مشخص میکنیم. اکنون دو حالت داریم:
  - $\Delta_{start} = \Delta_{goal}$
  - $\Delta_{start} \neq \Delta_{goal}$

- مسیری که در حالت دوم طی میشود شامل سه قسمت است:
  - خط مستقیمی که از p_{start} به v_{start} رسم می شود.
  - مسیر بین v_{start} و v_{goal} که شامل یالهایی از نقشه راه است.
  - خط مستقیمی که از v_{goal} به p_{goal} رسم میشود.


Δ_{start}
Δ_{goal}
v_{goal}
p_{goal}
v_{start}
p_{start}

الگوریتم محاسبه مسیر

Algorithm COMPUTEPATH($\mathcal{T}(\mathcal{C}_{\text{free}})$, $\mathcal{G}_{\text{road}}$, p_{start}, p_{goal})

Input. The trapezoidal map $\mathcal{T}(\mathcal{C}_{\text{free}})$ of the free space, the road map $\mathcal{G}_{\text{road}}$, a start position p_{start}, and goal position p_{goal}.

Output. A path from p_{start} to p_{goal} if it exists. If a path does not exist, this fact is reported.

1. Find the trapezoid Δ_{start} containing p_{start} and the trapezoid Δ_{goal} containing p_{goal}.
2. **if** Δ_{start} or Δ_{goal} does not exist
3. **then** Report that the start or goal position is in the forbidden space.
4. **else** Let v_{start} be the node of $\mathcal{G}_{\text{road}}$ in the center of Δ_{start}.
5. Let v_{goal} be the node of $\mathcal{G}_{\text{road}}$ in the center of Δ_{goal}.
6. Compute a path in $\mathcal{G}_{\text{road}}$ from v_{start} to v_{goal} using breadth-first search.
7. **if** there is no such path
8. **then** Report that there is no path from p_{start} to p_{goal}.
9. **else** Report the path consisting of a straight-line motion from p_{start} to v_{start}, the path found in $\mathcal{G}_{\text{road}}$, and a straight-line motion from v_{goal} to p_{goal}.

بررسی درستی الگوریتم

- برای بررسی دستی الگوریتم باید به دو سوال زیر پاسخ دهیم:
  - آیا مسیرهایی که گزارش میشوند همیشه بدون برخوردند؟
  - آیا در صورت وجود مسیر بدون برخورد، حتما آن مسیر گزارش خواهد شد؟
- در پاسخ به سوال اول میتوان گفت: چون مسیر گزارش شده شامل خطوط درونی ذوزنقه هاست و ذوزنقه ها نیز در فضای آزاد ساخته شده اند، لذا مسیر گزارش شده حتما بدون برخورد است.

- برای پاسخ به سوال دوم فرض میکنیم که یک مسیر بدون برخورد از نقطه شروع به نقطه پایان وجود داشته باشد. بدیهی است که نقطه شروع و نقطه پایان باید در ذوزنقه هایی در فضای آزاد باشند. مسیر بین نقطه شروع تا نقطه پایان از دنباله ذوزنقه های $\Delta_1,\ldots,\Delta_k$ میگذرد، که با استفاده از تعریف داریم:

$$\Delta_1 = \Delta_{start}$$

و

$$\Delta_k = \Delta_{goal}$$

فرض کنید v_i نقطه مرکزی Δ_i باشد. اگر مسیر از Δ_i به Δ_{i+1} برود در اینصورت Δ_i و Δ_{i+1} مجاورند و درنتیجه یک مسیر شامل دو کمان از Δ_i به Δ_{i+1} وجود دارد. در نتیجه یک مسیر از Δ_1 به Δ_k وجود دارد.

توجه

- الگوریتم breadth – first مسیرهای متفاوتی را از v_{start} به v_{goal} پیدا خواهد کرد.

بررسی زمان مصرفی الگوریتم

- پیدا کردن ذوزنقه های شامل نقاط شروع و پایان در زمان O(logn) انجام پذیر است. (فصل ششم)
- زمان breadth – first search نیز از اندازه g_{road} خطی است.
- g_{road} نیز به ازای هر ذوزنقه یک نقطه و برای هر گسترش عمودی آن نیز یک نقطه دارد که مجموع همه ذوزنقه ها و گسترش های عمودی از تعداد رئوس موانع خطی است.
- لذا زمان breadth – first search نیز خطی است.
- زمان گزارش مسیر نیز O(n) است.
- در نتیجه قضیه زیر را داریم:

قضیه 13.2

- فرض کنید R روبات نقطه ای باشد که در مجموعه S از موانع چند ضلعی با مجموع یالهای n در حرکت است. میتوانیم S را در زمان O(nlogn) پیش پردازش کنیم بطوریکه می توان بین هر دو نقطه شروع و پایان، مسیر بدون برخوردی را (در صورت وجود) در زمان O(n) محاسبه کرد.

با تشکر از توجه شما