Navigáció és mozgástervezés

Algoritmusok és alkalmazásaik

Osváth Róbert

Sorbán Sámuel

Feladat

- Adottak: pálya (C), játékos, játékos ismerethalmaza, kezdőpont, célpont.
- Pálya szerkezete: akadályokkal (O) ellátott terület.
- Játékos ismerethalmaza: műveletek, melyeket a játékos végre tud hajtani.
- Cél: ismerve a játékos kezdeti pozicióját, eljutni egy kívánt végpontba.
- C – O = F szabad tér, ahol a játékos szabadon mozoghat

Algoritmusok

A megoldási módszerek:

1. Skeletonization.
2. Bounded-error planning. (Fine – motion)
3. Online algoritmusok.

1. Skeletonization

- Besűrítjük a pályát egy viszonylag egyszerűbb adatstruktúrába, csontvázba.
- Utak a csontvázon.
- Csontváz: véges csúcspontokkal rendelkező háló (gráf), gráfkereső algoritmusokat alkalmazhatunk utak keresésére a csontvázon.
- A szabad tér “minimális” leírása.

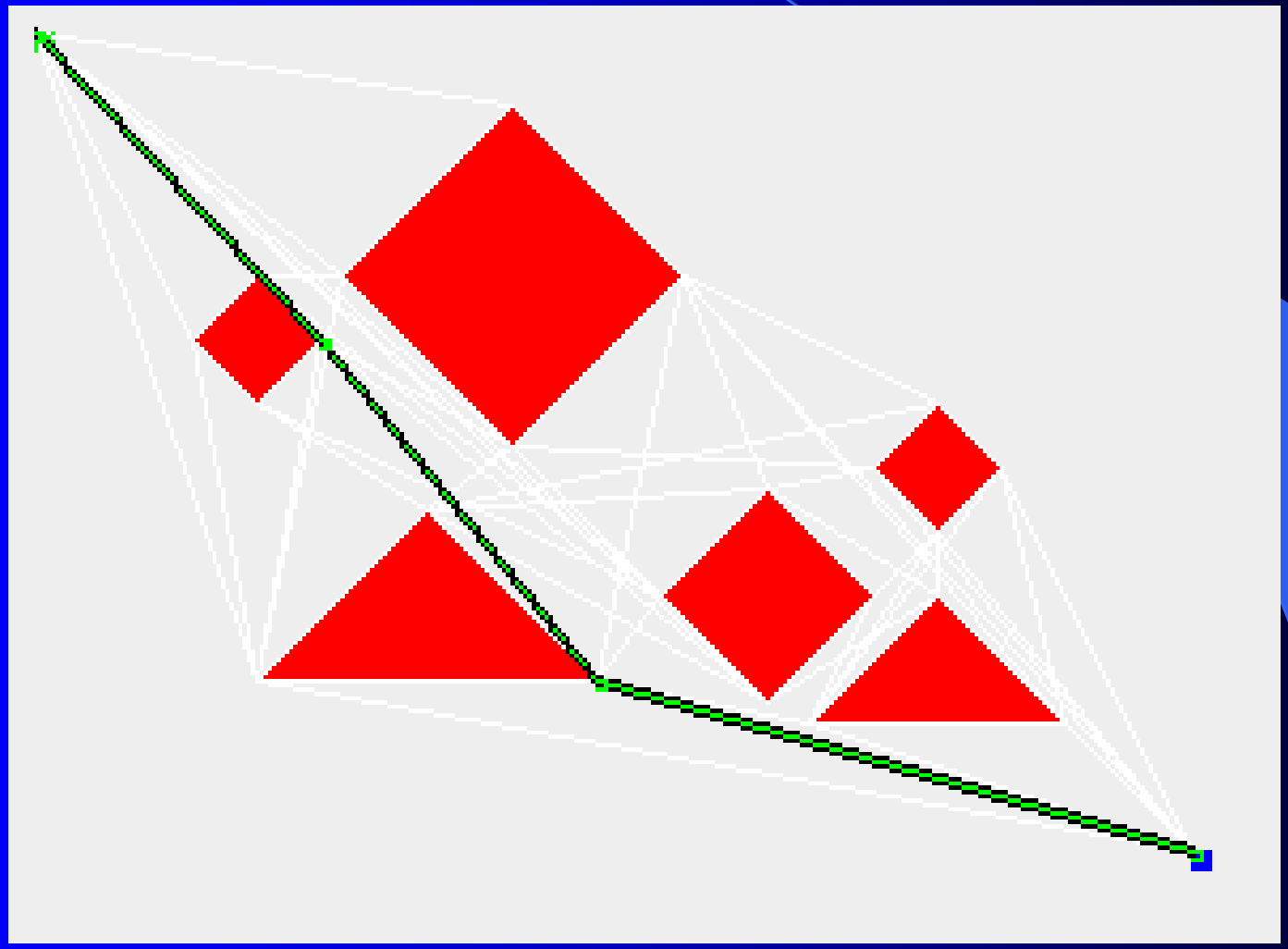
Skeletonization (2)

- Mozgás tervezés szempontjából helyes csontváz:
  1. Ha *S F* szabad térnek egy csontváza, akkor *S* –nek kell, hogy legyen egy(!) összefüggő része F minden különböző összefüggő régiójában.
  2. *F* minden *p* pontjára, egy út *p* –től a csontvázig "könnyen" megszerkeszthető kell legyen.
- (Skeletonization) Eljárások:
  1. Láthatósági gráf.
  2. Voronoi diagramm.

Skeletonization
1. Láthatósági gráf

- Láthatósági gráf egy *C* tér esetében: egymással láthatósági viszonyban levő csúcspontokat összekötő élek. (az akadályok határozzák meg)
- A gráf alkalmas (teljes) tervezésre: tartalmazza az akadályok közti legrövidebb utakat.

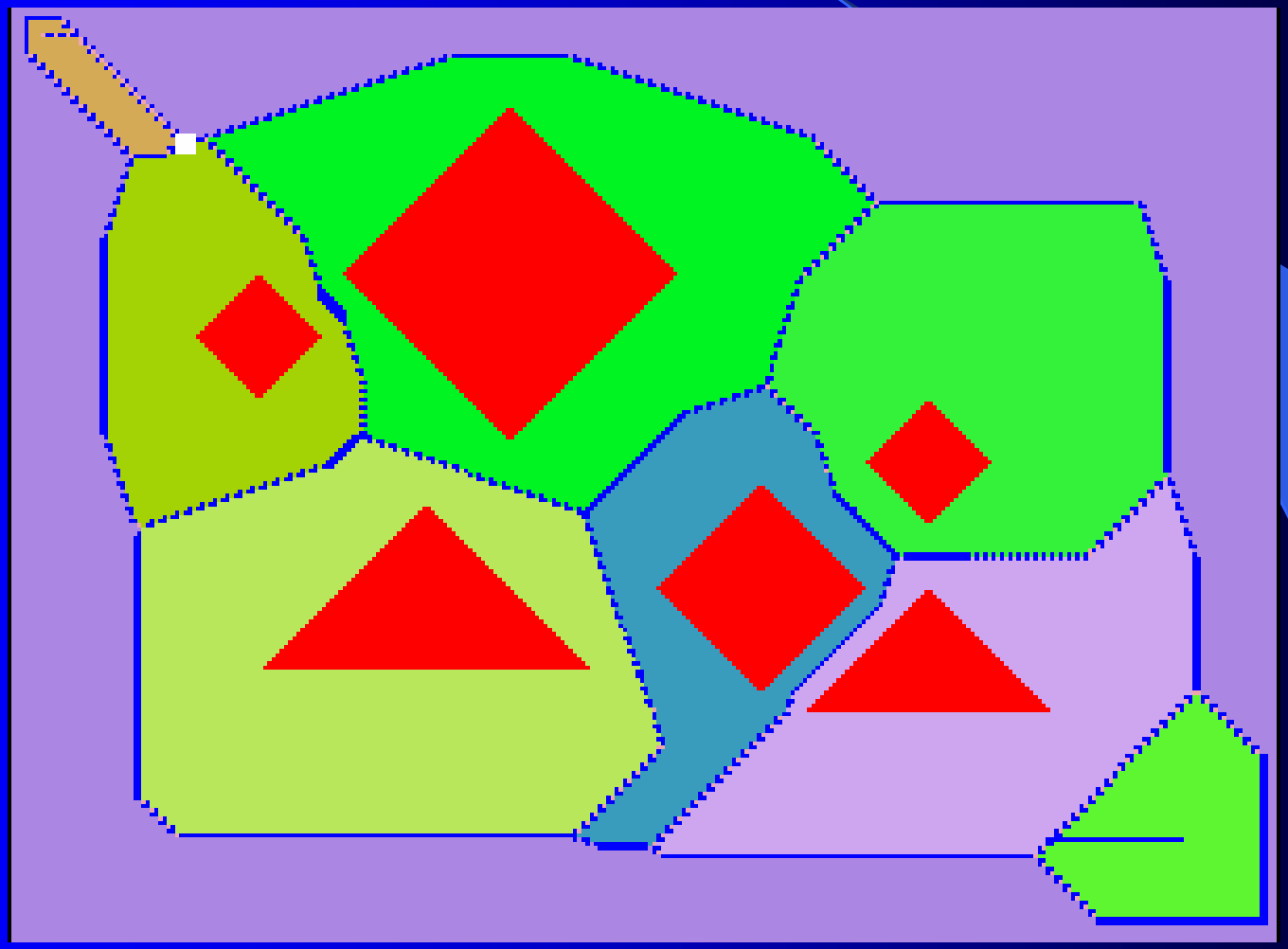
Láthatósági gráf



Skeletonization
2. Voronoi diagram

- F minden pontjára kiszámoljuk a d(p,N(p,O)) értéket, vagyis a ponthoz legközelebb eső akadálytól való távolságot.
- Ábrázoljuk ezen távolságokat mint a diagrammból kiemelkedő magasságokat.
- A terep magassága 0 közvetlenül az akadályok határánál és növekszik ha távolodunk tőlük.
- Két vagy több akadálytól egyenlő távolságra levő pontok kiemelkedő domborulatokat eredményeznek.
- Ezek alkotják a Voronoi diagramot.
- Teljes algoritmusok.
- Hátrány: legtöbb esetben nem a legrövidebb utat eredményezi.

Voronoi diagram



2. Bounded-error planning (Fine-motion)

- Apró, pontos mozgások tervezése későbbi összeillesztésre.
- Terv: felügyelt mozgások sorozata.
- Felügyelt mozgás:
  1. Mozgás parancs – engedékeny mozgás (csúszás, rugó...).
  2. Leállási feltétel – felügyelt mozgás vége.

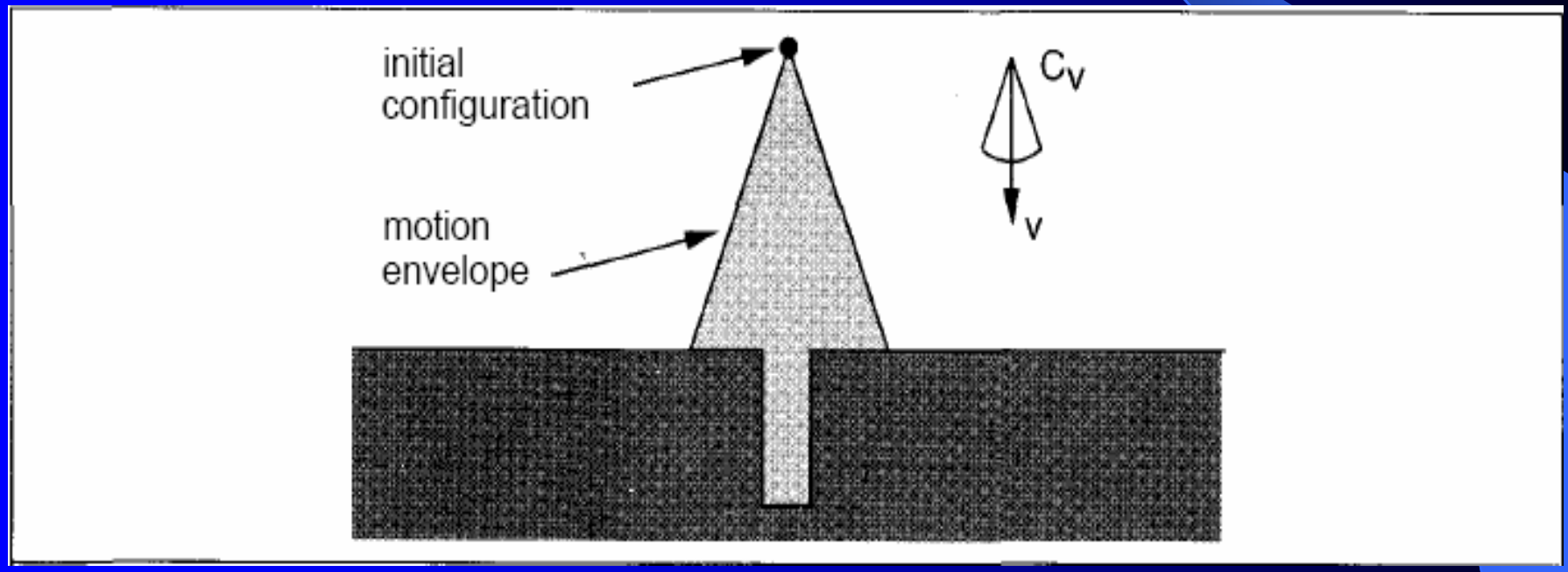
Bounded-error planning (2)

- Fine-motion tervező paraméterei: pálya leírása, leállási feltétel, engedékeny mozgás paraméterei.
- Az eredmény: több lépéses kondicionált terv vagy stratégia, mely garantáltan jó eredményhez vezet, feltéve ha létezik ilyen terv.
- A tervek a legrosszabb esetekhez igazodnak.
- Rendkivül magas összetettség.

Bounded-error planning (3)
Példa

- 2D környezet
- Téglatest alakú cölöp behelyezése egy gödörbe
- Mozgás parancsok: állandó sebességek
- Leállási feltétel: felülettel való érintkezés
- Bizonytalanság modellezés: a megadott sebesség helyett a robot mozgását a megszerkesztett kúp írja le

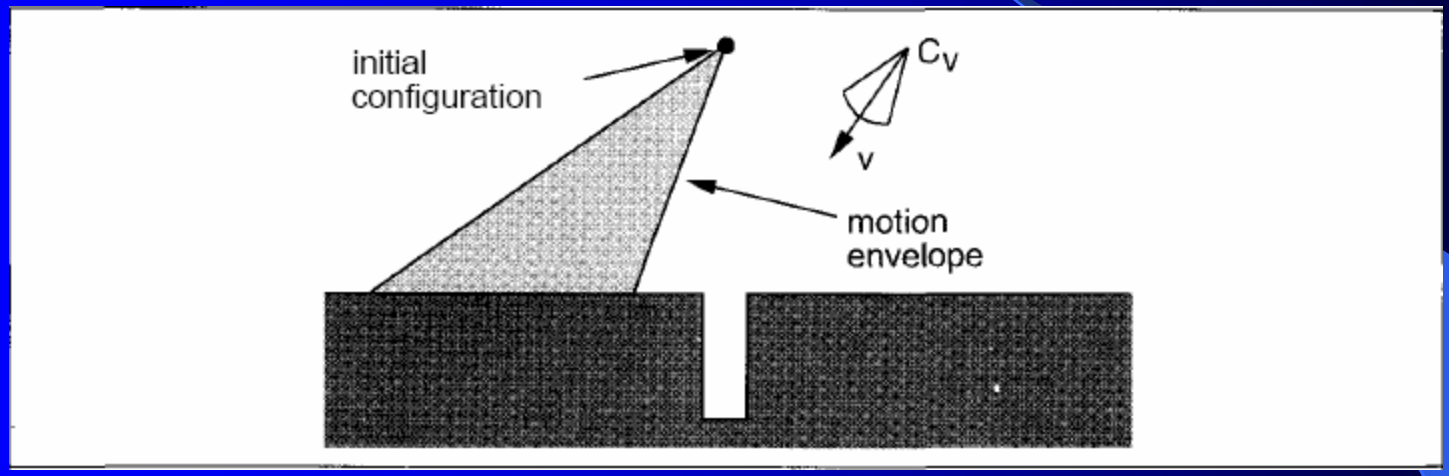
Bounded-error Planning (4)
Példa (folytatás)

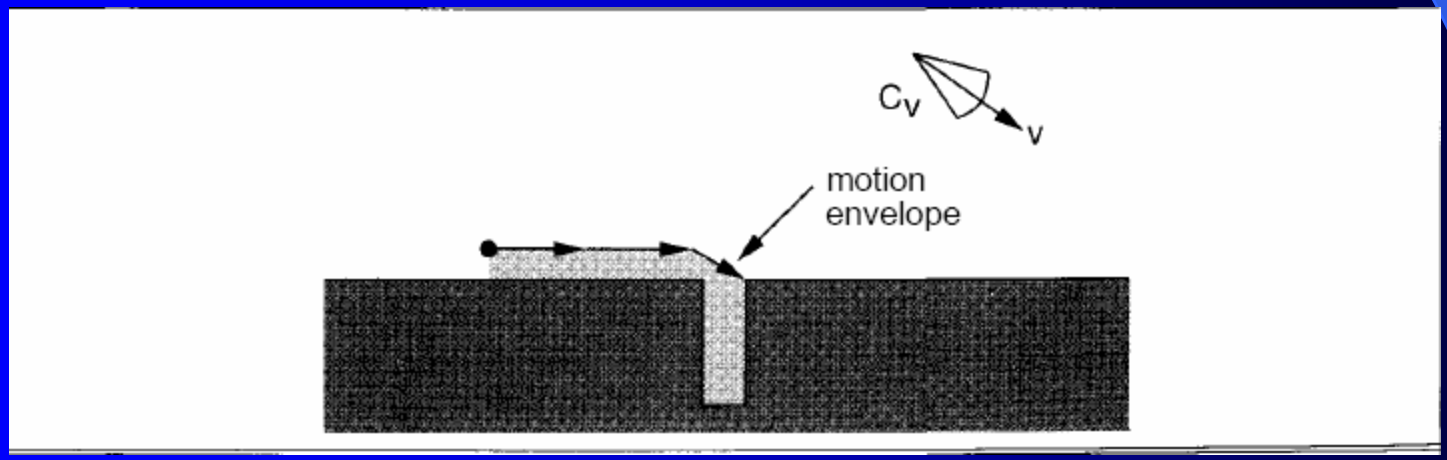


Bounded-error Planning (5)
Példa (folytatás)

- Más megközelítésben:
- Első lépésben a mozgás parancs és a leállási feltétel a robotot a gödör egyik felére landoltattja (bal)
- Következő lépésben pedig: vízszintes felülettel való érintkezéskor jobbra csúszik, a gödörhöz érve lefele csúszik, mivel a vertikális felületkere nézve a megadott sebességek lefele (jobbra) relatívak

Bounded-error Planning (6)
Példa (folytatás)





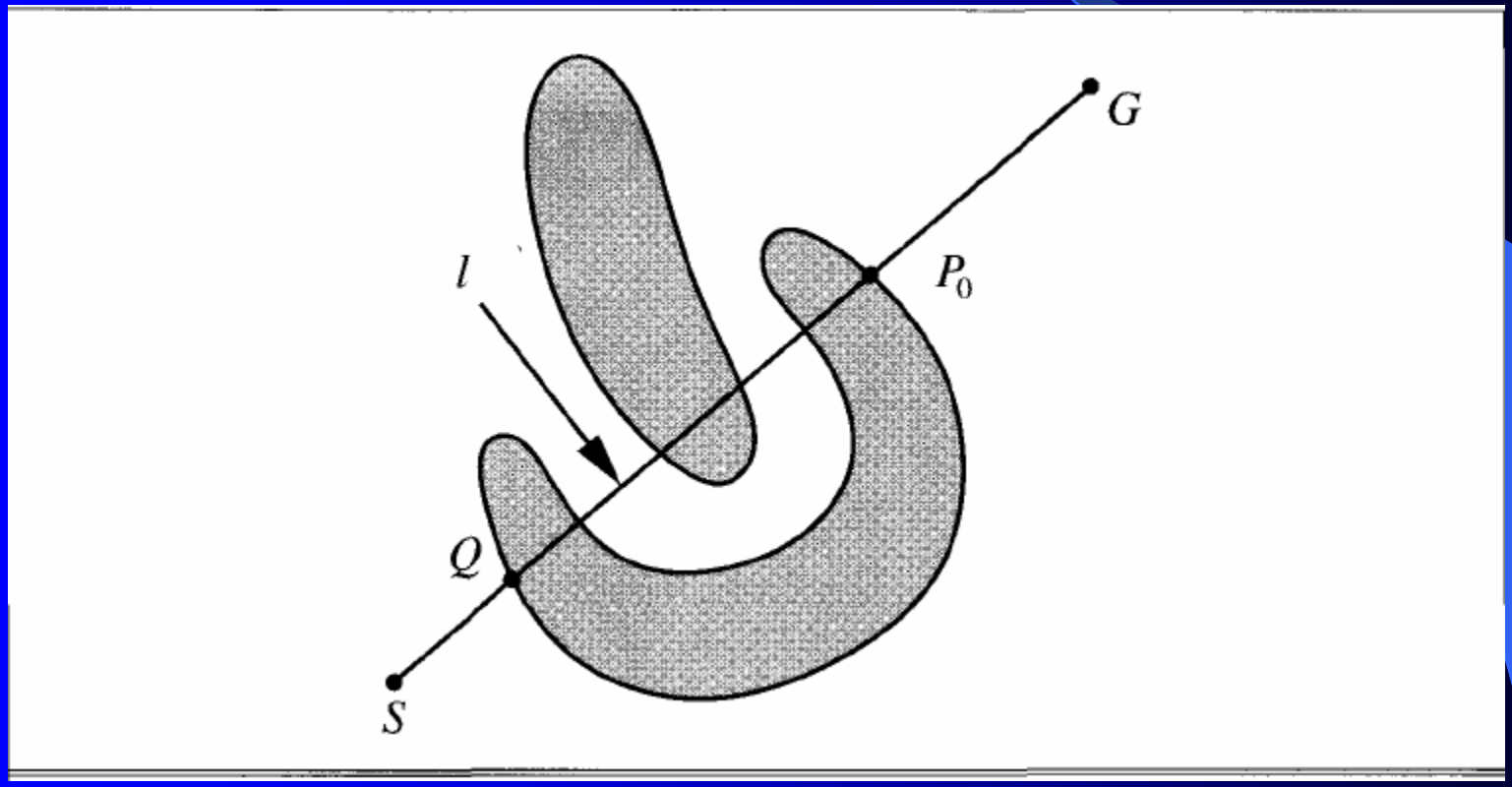
3. Online Algoritmusok

- A környezet többnyire vagy teljesen ismeretlen.
- Kondicionált terv létrehozása.
- Egyszerűségre kell törekednünk.
- Gyorsaság és teljesség.
- Szinte minden esetben hiányzik a legrövidebb út megtalálásának bizonyossága.

Online algoritmusok (2)

- Stratégia:
  1. Egyenes vonal mentén mozgunk a cél felé (L).
  2. Ha akadályhoz érkezünk megjegyezzük az aktuális pozíciónkat (Q). Megkerüljük az akadályt, óramutató járásával megegyező irányba, vissza Q-ig. Megjegyezzük azon pontokat és távolságokat, amelyek esetén átmentünk L-en. Ezek közül kiválasztjuk a célponthoz a legközelebbit (P_0)
  3. Megkerüljük az akadályt Q-ból P_0 -ba. Ismerjük a legrövidebb utat P_0 –ig. P_0 –ból ismételjük a fenti lépéseket.

Online algoritmusok (3)



Összegzés

- Skeletonization: diszkrét gráfkeresési feladat. Láthatósági gráf: 2D – ben gyors, optimális megoldás, Voronoi – akadékos implementáció.
- Fine-motion: kondicionált terv. Komplexitás exponenciálisan nő a robot szabadságfokának és a terv lépései számának függvényében.
- Online algoritmusok: mozgás alatti döntés, nem garantált siker, környezettől függ.

Kiegészítés

- Láthatósági gráf algoritmusa

```
Minden PolygonCsúcsból (A)
    Minden PolygonCsúcsba (B)
        Ha ( [A,B] szakasz nem metszi egyik Polygont sem) akkor
            elek[A,B] = 1
        Ha vége
    Minden vége
Minden vége
```

Út választása: Dijkstra vagy egyéb gráfkereső algoritmussal.

Kiegészítés

- Online algoritmus

  Robot: 3 státusz változó (követ, keres, folytat)

- Elindul a kezdőpontból **követés** státusszal (Követi a start és cél által meghatározott egyenest)
- Mindig figyeli a környező pontokat hogy akadály-e (szenzor szimulálása)
- Ha akadályt talál átlép a **keresés** státuszba
- Minden környező pontra kiszámítja az ezt a pontot körülvevő akadálypontok összegét
- Ahol az összeg minimális oda lép és folytatja a keresést
- Megjegyzi a metszéspontokat, és amikor visszaért az akadály észlelési pontjába átlép a **folytat** státuszba ami megkeresi melyik az optimálisabb irány
- Visszalép **követési** státuszba

Kiegészítés

- Voronoi roadmap

- 2 térképet használunk, HATÁR jelzi a voronoi cellákat

Elegendő számú lépésig végezd el

Minden pontra

Halmaz h = térkép[pont] szomszédjai
Ha elemei(h) == 1 akkor térkép2[pont] = h.elem()

Ha elemei(h) > 1 akkor térkép2[pont] = HATAR

Minden vége

térkép = térkép2

For vége

- Miután a térkép megvan, egy Robotot indítunk el, ami az online robothoz hasonlóan keresi az utat, de kizárólag csak a HATAR vonalon

- Be lehet bizonyítani hogy ha létezik út a a start és cél között, akkor a voronoi cella határvonalaival is el lehet oda jutni.

- a start és cél pontokat is kiterjesztjük. Ha a robot start vagy vég cellába ér, ott a legrövidebb úton mozoghat a cél fele