

# Úvod do mobilní robotiky — AIL028

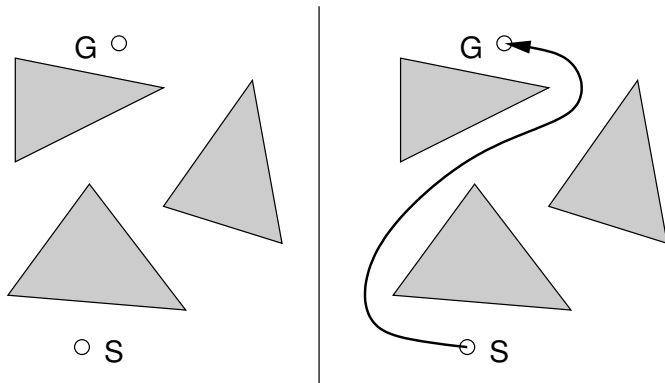
Martin Dlouhý a Zbyněk Winkler

md at robotika.cz, zbynek.winkler at mff.cuni.cz  
<http://robotika.cz/guide/umor07/cs>

27. listopadu 2007

- 1 Úvod
  - Mapa světa
  - Exaktní plánování
- 2 Bodový robot
  - Graf viditelnosti
  - Voronoi diagramy
  - Lichoběžníková dekompozice
- 3 Konvexní robot
  - Plánování s otáčením

# Mapa světa - příklad



- Malinkatý robot pohybující se po podlaze
- Úkolem je nalézt cestu ze startu do cíle a s ničím se nesrazit

# Exaktní plánování

- nalezne cestu vždy, pokud existuje
  - v opačném případě ověří, že neexistuje
- nejsou to aproximace
- jsou použitelné/pochopitelné pouze pro jednodušší případy
- jsou elegantní a efektivní

# Dělení plánovacích algoritmů

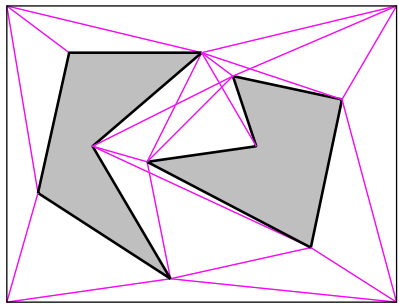
- podle přesnosti
  - exaktní
  - aproximační
- podle popisu prostředí
  - mapa cest
  - dělení na jednoduché části
- podle tvaru robota
  - bodový
  - kruhový
  - konvexní
  - obecný
- podle stupňů volnosti robota
  - posun
  - posun s otáčením
  - non-holonomic, např. autíčko

# Plánování pro bodového / kruhového robota

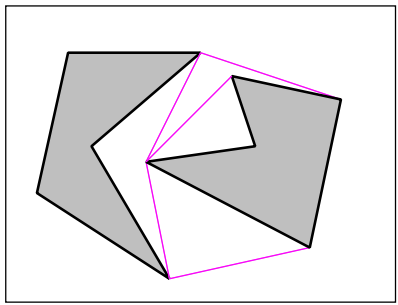
- svět  $\mathcal{W} = \mathbb{R}^2$
- překážky  $\approx$  polygony  $\mathcal{O}$
- volný prostor  $\approx \mathcal{W} \setminus \mathcal{O}$  (free space)
- v nejjednodušším případě je robot bezrozměrný bod
  - některé algoritmy lze zobecnit pro kružnici „nafouknutím“ překážek

# Graf viditelnosti

- dva vrcholy jsou spojeny hranou, pokud jejich spojnice neprochází žádnou z překážek
- optimalizace — pouze hrany, které mohou „jít za roh“
  - hrana prodloužená o  $\epsilon$  nesmí zasahovat do překážky
- naivní algoritmus  $O(n^3)$ , sweep-line (koště) lepší
- nejkratší cesta nalezená v grafu je zároveň nejkratší možná pro původní problém

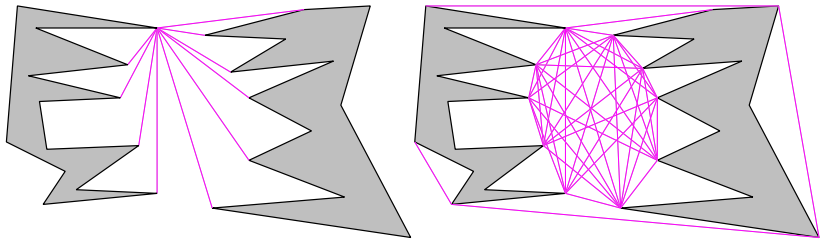


(a)



(b)



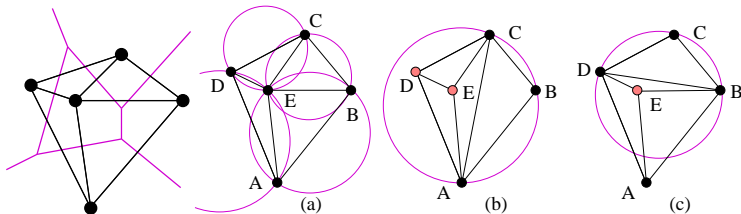


# Obr v lese

- obr (robot) = kružnice, stromy (překážky) = body
- obr je v lese, jak se dostane ven?
- kolik může přibrat (zhubnout)?

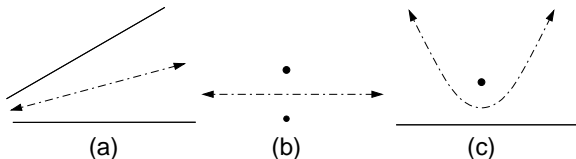
# Voronoi diagramy

- stromy se stanou řídicími body a rozdělí prostor do  $n$  oblastí
  - každá oblast odpovídá právě jednomu řídicímu bodu
  - body z oblasti mají nejbližší právě ke svému řídicímu bodu
- hranice oblastí — Voronoi hrany
  - stejná vzdálenost ke dvěma nejbližším překážkám
- při pohybu po hranách si udržujeme maximální možnou vzdálenost od překážek
- průchodnost hrany — vzdálenost řídicích bodů



# Algoritmus pro cestu z lesa

- přejdi na síť
  - např. směrem od nejbližší překážky
- hledej cestu v grafu
  - pro nalezení „nejširší“ cesty vybíráme vždy hranu s největší průchodností
- rozšíření pro polygony
  - kromě řídicích bodů i řídicí úsečky
  - hrany mohou být i kusy parabol



# Algoritmy pro konstrukci Voronoi diagramu

- nejlepší algoritmus  $O(n \log n)$ 
  - rozděl & panuj
  - Fortune's line sweep
- další vesměs  $O(n^2)$ 
  - inkrementální
  - konverze z obecné triangulace

# Lichoběžníková dekompozice

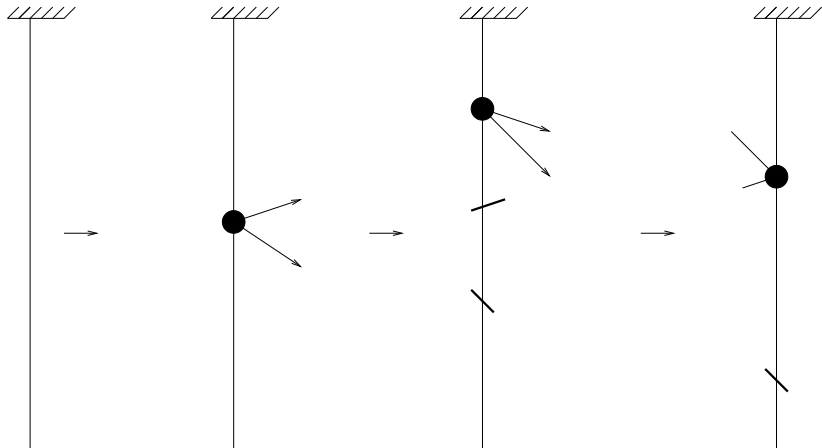
- rozdělit prostor na jednoduché buňky
- plánování uvnitř buňky je jednoduché
- cesta, je posloupnost buněk taková, že platí
  - první buňka obsahuje start
  - poslední buňka obsahuje cíl
  - buňky v posloupnosti za sebou sdílí společnou hranu
- pro buňky lichoběžníkového tvaru to umíme v čase  $O(n \log n)$

## Algoritmus zametací přímky

- setřídít všechny vrcholy podle souřadnice  $x$
- zametáme zleva doprava *koštětem*
- koště má svůj *stav* — utříděný seznam úseček podle  $y$ -ové osy, které zrovna protíná.
- stav se mění pouze ve vrcholech (*událost*)
- při každé události je *stav* aktualizován — nalezení levého/pravého konce hrany koresponduje přidání/smazání hrany
- při průchodu sestavujeme graf reprezentující sousednost buněk

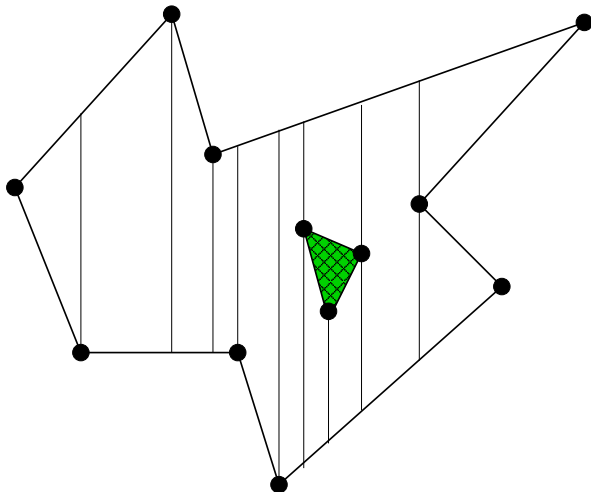
Složitost  $O(n \log n)$  — třídění  $O(n \log n)$ , jedna aktualizace *stavu* koštěte  $O(\log n)$ , počet aktualizací (bez průsečíků)  $O(n)$

# Průběžné změny koště





# Výsledná dekompozice



# Konvexní robot

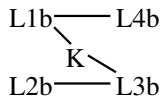
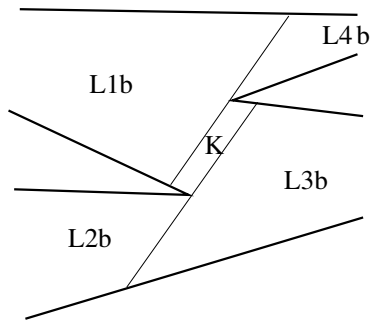
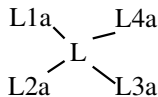
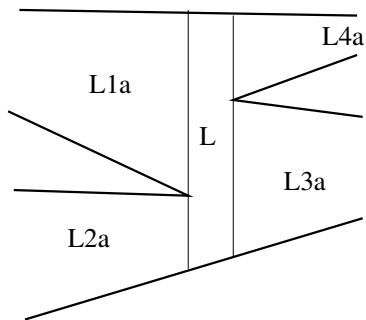
- co když robot není bezrozměrný bod ani kružnice?
- pro konvexního robota bez otáčení taky umíme nafouknout překážky

# Plánování s otáčením

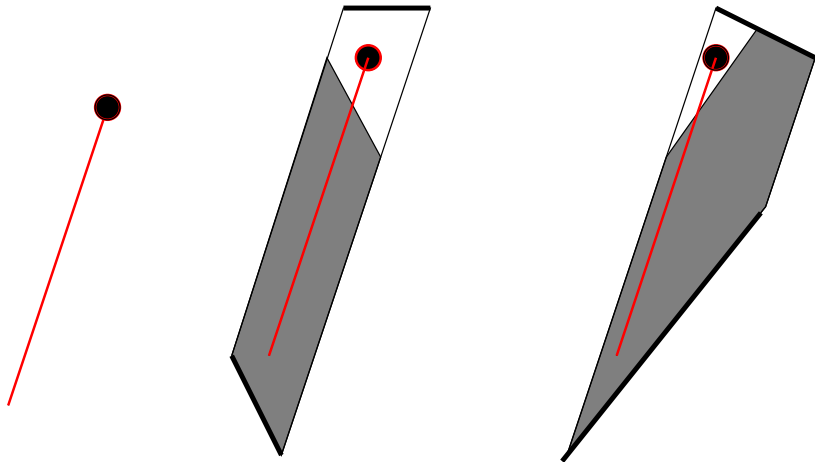
## problém žebříku

- robot reprezentován úsečkou
- pro úsečku v jedné poloze lze použít lichoběžníkovou dekompozici spojenou s nafouknutím překážek
- v dalším kroku zkusíme úsečku pootočit
- topologický graf zůstává stejný
- po jisté době ale ke změně dojde (nějaký lichobežník zanikne, změní sousedy atp.) — je možno spočítat
- pro  $360^\circ$  získáme složitější graf, ale s žebříkem můžeme i otáčet

# Změna topologického grafu při otáčení



# Ořez lichoběžníkových buněk



- Pravděpodobnostní plánování
  - jak naplánovat cestu pro auto bez couvání?
  - a co couvání traktoru s několika přívěsy?
  - nebo piáno v krápníkové jeskyni (3D nekonvexní objekt v 3D prostředí s otáčením)?