



Robotika: Plánování pohybu

Vladimír Petřík

vladimir.petrik@cvut.cz

06.11.2023

Motivace

- ▶ Představte si úlohu, kde máte pomocí robota přenést plný hrnek z jednoho místa na druhé
- ▶ Jak byste to řešili pomocí metod, které znáte?
 - ▶ Spojit úsečkou počáteční a koncový bod - interpolace polohy
 - ▶ Zafixovat orientaci hrnku
 - ▶ Výpočet inverzní kinematiky
- ▶ Možné problémy
 - ▶ Inverzní kinematika spočítá několik řešení
 - ▶ Inverzní kinematika nemusí najít žádné řešení
 - ▶ Překážka v přímé cestě
- ▶ Definovat přípustné natočení a použít plánování



Cíl dnešní přednášky

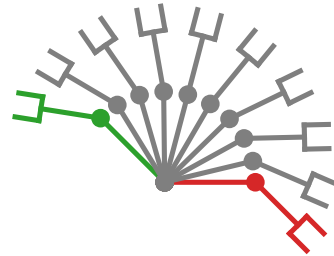


- ▶ Plánování pohybu
 - ▶ Co je to plánování?
 - ▶ Plánování v mobilní robotice
 - ▶ Plánování pro 2D manipulátor
 - ▶ Plánování pro reálný 7D manipulátor
- ▶ Založeno na 10. kapitole Lynch&Park: Modern Robotics



Formulace problému

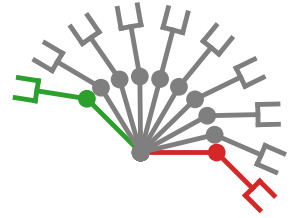
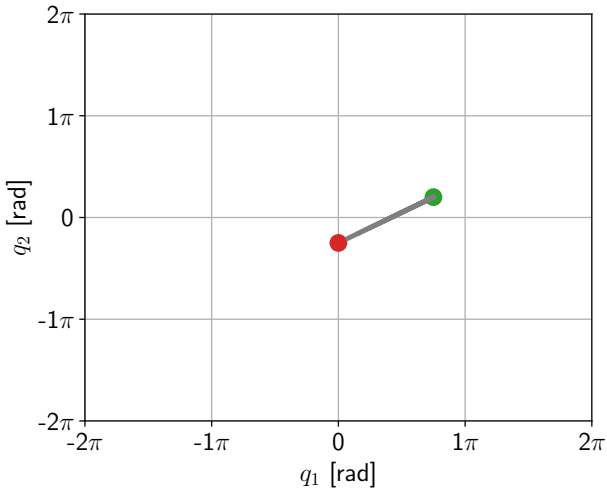
- ▶ Plánování pohybu pro robota z daného startu do daného cíle
 - ▶ Vyhýbání překážkám
 - ▶ Respektování omezení kloubů
 - ▶ Respektování maximální rychlosti, sil/momentů
 - ▶ Ostatní omezení daného prostředí (např. nevylít vodu z hrnku → omezení orientace)
- ▶ Plánování probíhá v konfiguračním prostoru robota¹
 - ▶ Konfigurační prostor \mathcal{C} (Configuration space, C-space)
 - ▶ Konfigurace robota $\mathbf{q} \in \mathcal{C}$
 - ▶ Start i Cíl je definován v konfiguračním prostoru
($\mathbf{q}_s, \mathbf{q}_g$) $\in \mathcal{C} \times \mathcal{C}$



¹Existují *task-space* plánovače, ale nebudeme se jimi dnes zabývat.

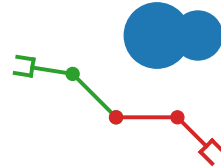
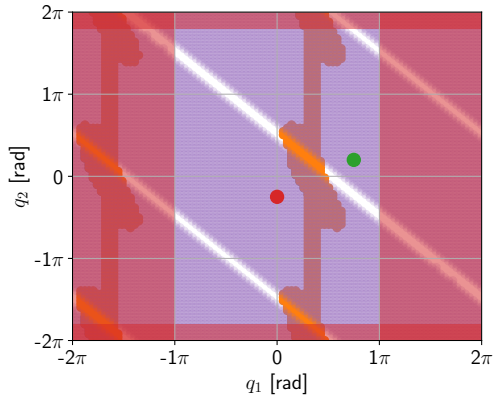


Konfigurační prostor - interpolace



Vliv omezení na konfigurační prostor

- ▶ Limity kloubů
- ▶ Překážky prostředí (pozn. tvar chapadla ignorován)
- ▶ Omezení orientace chapadla ($80^\circ < \theta < 100^\circ$)



Konfigurační prostor - závěr

- ▶ Konfigurační prostor rozdělujeme na:
 - ▶ Prostor přípustných konfigurací $\mathcal{C}_{\text{free}}$
 - ▶ Prostor nepřípustných konfigurací \mathcal{C}_{occ}
 - ▶ $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{occ}}$
- ▶ Konfigurační prostor je (většinou) těžké popsat explicitně
- ▶ I když Start i Cíl leží v $\mathcal{C}_{\text{free}}$, nemusí existovat pohyb, který je spojuje



Co je to pohyb? Cesta × Trajektorie

- ▶ Cesta (path)
 - ▶ Geometrický popis (sekvence konfigurací)
 - ▶ Bez časových značek, dynamiky a omezení na kontrolní signál
 - ▶ Najdi $\mathbf{q}(s) \in \mathcal{C}_{\text{free}}, s \in [0, 1]$ tak, aby:
 - ▶ $\mathbf{q}(0) = \mathbf{q}_s, \quad \mathbf{q}(1) = \mathbf{q}_g$
 - ▶ Hlavním předpokladem je, že následným zpracováním je možné získat trajektorii
- ▶ Trajektorie (trajectory)
 - ▶ Popisuje pozici robota v čase
 - ▶ $\mathbf{q}(t) \in \mathcal{C}_{\text{free}}, t \in [0, T]$



Můžeme plánovat rovnou trajektorii?

- ▶ Definice problému zahrnuje rovnici pohybu systému
- ▶ Najdi kontrolní signál v čase $u(t)$ tak, aby:
- ▶ $q(0) = q_s, \quad q(T) = q_g, \quad \dot{q} = f(q(t), u(t))$
- ▶ Komplikovanější úloha než hledání cesty
- ▶ Kdy je preferovaná?
 - ▶ jsou poháněné jenom některé klouby - nemusí být možné sledovat naplánovanou cestu
- ▶ V robotickém plánování pohybu se typicky naplánuje cesta a z ní se spočítá trajektorie



Plánování cesty v mobilní robotice

- ▶ Jak plánování funguje v ROS: MoveBase
- ▶ Robot se pohybuje v rovině $(v_x, v_\theta)^\top$
- ▶ Známe:
 - ▶ rozměry robota (půdorys)
 - ▶ mapu prostředí
 - ▶ pozici robota v mapě $(x, y, \theta)^\top$
 - ▶ cílovou pozici v mapě
- ▶ Řešení navigace do zadaného cíle v ROS: MoveBase:
 - ▶ **Globální plánovač cesty ve 2D** Globální plánovač cesty ve 2D
 - ▶ Dopočítání orientace (robot směřuje dopředu podél cesty - proč?)
 - ▶ Lokální plánovač (vyhýbání dynamickým překážkám, používá vstup z lidaru)
 - ▶ Poslání příkazu robotovi

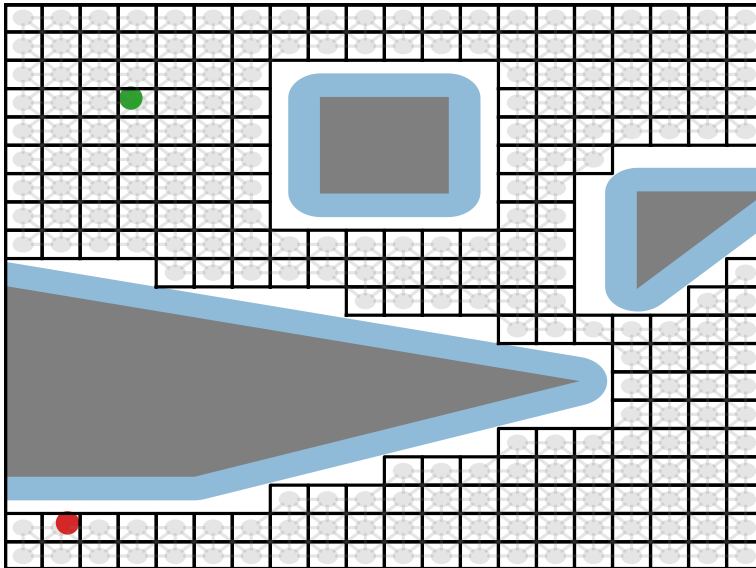


Plánovače #1: Plánování v diskrétní mřížce

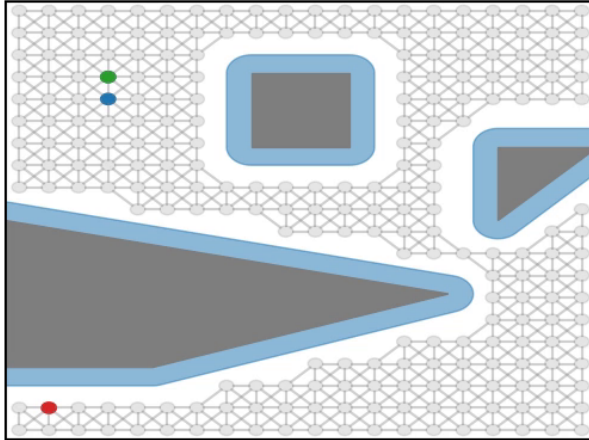
- ▶ Diskretizovat konfigurační prostor (mapa je často diskretizovaná na vstupu)
- ▶ Nejjednodušší způsob diskretizace je použít mřížku
- ▶ Cílem je reprezentovat volný konfigurační prostor pomocí grafu



Plánovače #1: Plánování v diskretní mřížce



Prohledávání grafu: Informované prohledávání A*



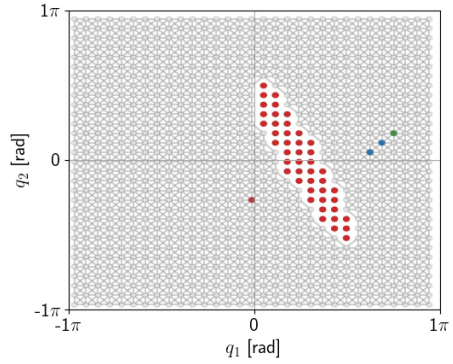
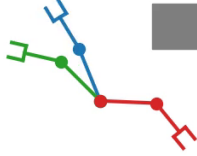
Prohledávání grafu



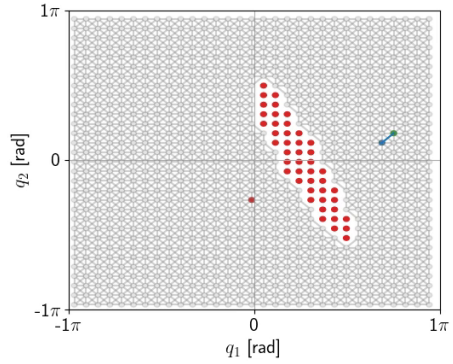
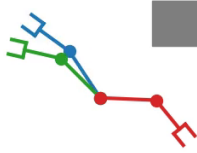
- ▶ Neinformované do šířky
 - ▶ úplné - najde řešení pokud existuje
 - ▶ optimální
- ▶ Neinformované do hloubky
 - ▶ úplné jenom v konečných grafech
 - ▶ neoptimální
- ▶ Informované A*
 - ▶ používá heuristiku pro vedení prohledávání a tím zrychluje prohledávání
 - ▶ úplné, optimální
 - ▶ můžeme použít A* pro robotický manipulátor?



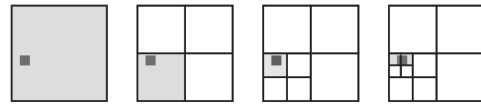
A* pro 2D manipulátor



A* pro 2D manipulátor



Jak velká mřížka je potřeba?



- ▶ Příklad mobilního robota $20^2 = 400$
- ▶ Příklad manipulátora $30^2 = 900$
- ▶ Komplexnější manipulátor: $30^6 = 729.000.000$
- ▶ Humanoid: $30^{20} = 3,5 \cdot 10^{29}$
- ▶ Úplnost algoritmů závisí na rozlišení mřížky -
- *resolution complete*
- ▶ Existují mřížky s variabilním rozlišením²
- ▶ Diskretizovat vysoce dimenzionální prostor ale není snadné ani efektivní - používají se jiné alternativy plánování.



²Obrázek z Lynch&Park: Modern Robotics



Plánovače #2: Metody založené na náhodném vzorkování

Sampling Methods

- ▶ Založena na
 - ▶ náhodném vybírání vzorků z \mathcal{C}
 - ▶ funkci, která určí, jestli je vzorek bezkolizní
 - ▶ lokální plánovač, který nalezne cestu k náhodnému vzorku
- ▶ Vytváříme graf náhodně \times vytvoření grafu pomocí diskretizace
- ▶ Méně výpočetně náročné než mřížkové metody
- ▶ Pravděpodobnostně úplné - pravděpodobnost, že najdeme řešení, se blíží 1 s počtem vzorků blížícím se nekonečnu
- ▶ Náhodné stromy (random trees - RT) a pravděpodobnostní cestovní mapy (probabilistic roadmap - PRM)

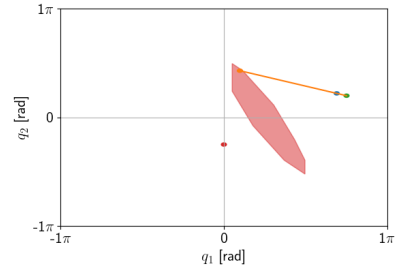
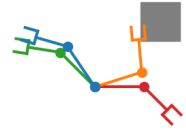


Náhodné stromy - naivní přístup

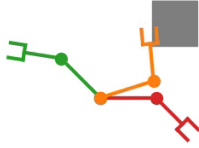
```

1: Vstup: start  $q_s$ , cíl  $q_g$ , maximální délka hrany  $\Delta q$ 
2:  $\mathcal{V} \leftarrow \{q_s\}$     $\mathcal{E} = \{\}$ 
3: for  $k = 1$  to  $M$  do
4:    $q_{\text{rand}} \sim \mathcal{C}$ 
5:    $q_{\text{tree}} \sim \mathcal{V}$ 
6:    $q_{\text{new}} \leftarrow f_{\text{plan}}(q_{\text{tree}}, q_{\text{rand}}, \Delta q)$ 
7:   if  $q_{\text{new}} \in \mathcal{C}_{\text{free}}$  then
8:      $\mathcal{V} \leftarrow \mathcal{V} \cup \{q_{\text{new}}\}$     $\mathcal{E} \leftarrow \mathcal{E} \cup \{(q_{\text{tree}}, q_{\text{new}})\}$ 
9:   end if
10:  if  $\|q_{\text{new}} - q_g\| < \epsilon$  then
11:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{q_g\}$     $\mathcal{E} \leftarrow \mathcal{E} \cup \{(q_{\text{new}}, q_g)\}$ 
12:    break
13:  end if
14: end for

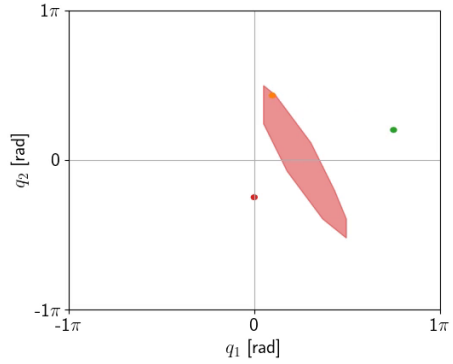
```



Náhodné stromy



Iterace: 0



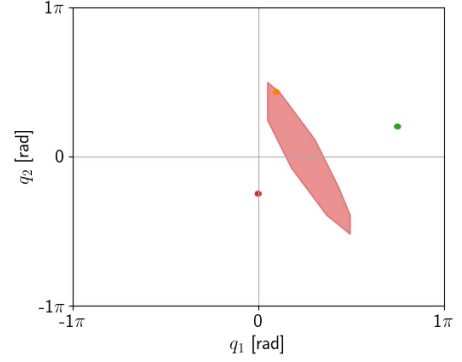
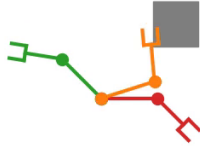
- ▶ Náhodné vybírání vrcholu stromu nepodporuje exploraci
- ▶ RRT³ navrženy pro exploraci
- ▶ Pro výpočet q_{new} je náhodně vybraná konfigurace "spojena" s nejbližším vrcholem stromu

$$q_{\text{tree}} = \arg \min_{q \in \mathcal{V}} \|q - q_{\text{rand}}\| \quad q_{\text{tree}} \sim \mathcal{V}$$

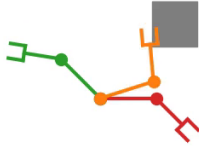
³LaValle et. al., Randomized Kinodynamic Planning, IJRR 2001



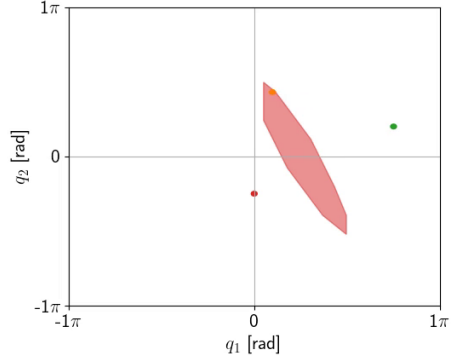
Iterace: 0



RRT - vzorkovací bias



Iterace: 0



- ▶ Výsledná cesta není optimální - často nám ale stačí alespoň nějaké řešení
- ▶ RRT je efektivní i pro velké dimenze
- ▶ Vhodným vzorkováním lze zvýšit efektivita prohledávání (304×257)
- ▶ Existuje spousta modifikací RRT algoritmu
 - ▶ RRT*, BiDirectional RRT, RRG, PDRRTs, RRT*-Smart, A*-RRT, RRT*FN, RRT*-AR, RT-RRT*, RRT#, Theta*-RRT, RRT* FND, RRT-GPU, RRdT*

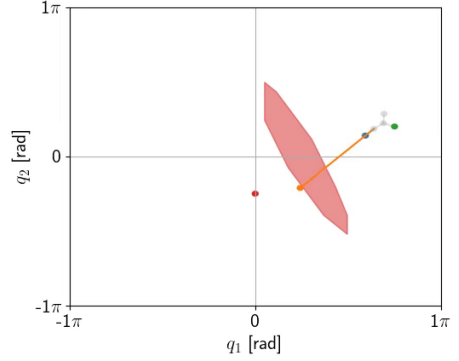


- ▶ Cestu vylepšuje pomocí dalších vzorků
- ▶ Je nepatrně pomalejší než RRT (dodatečné operace pro správu stromu)
- ▶ Je potřeba definovat ukončovací kritérium
 - ▶ maximální počet vzorků
 - ▶ čas plánování





Iterace: 4



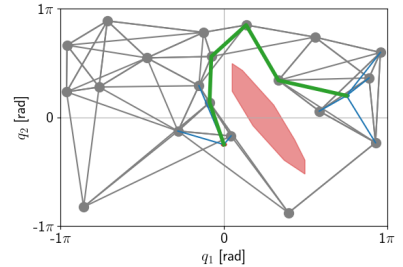
Pravděpodobnostní cestovní mapy - PRM

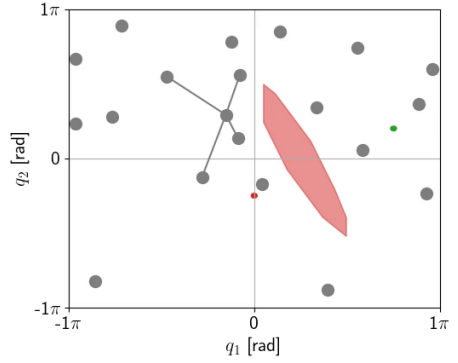
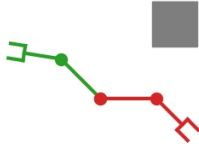
- ▶ Vytvoří neorientovaný graf pro pokrytí C_{free} před hledáním plánu
- ▶ Graf se pak prohledá pomocí A^*
- ▶ PRM vs RRT:
 - ▶ Graf nemusí být strom
 - ▶ Vytvořený graf se dá použít opakovaně
 - ▶ Multi-query vs Single-query



PRM vytvoření grafu

- 1: $\mathcal{V} \leftarrow \{q_i | q_i \sim \mathcal{C}_{\text{free}}\}_{i=0}^M$
 - 2: $\mathcal{E} \leftarrow \{\}$
 - 3: **for** $q_i \in \mathcal{V}$ **do**
 - 4: $\mathcal{N}(q_i) \leftarrow k$ nejblížešších sousedů vrcholu q_i
 - 5: **for** $q_j \in \mathcal{N}(q_i)$ **do**
 - 6: **if** $\text{plan}(q_i, q_j) \in \mathcal{C}_{\text{free}} \wedge (q_i, q_j) \notin \mathcal{E}$ **then**
 - 7: $\mathcal{E} \leftarrow \mathcal{E} \cup \{(q_i, q_j)\}$
 - 8: **end if**
 - 9: **end for**
 - 10: **end for**
- ▶ Při plánování se do grafu vloží q_s a q_g
 - ▶ Pomocí prohledávání grafu se nalezne cesta

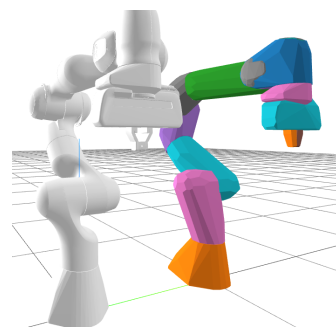




Jak by plánování vypadalo pro reálný manipulátor?

- ▶ Franka Emika Panda Robot
- ▶ Výpočet kolize je komplikovanější ve 3D
 - ▶ Vstup: q a kinematický model včetně 3D modelů jednotlivých ramen
 - ▶ Výstup: Je konfigurace v kolizi?
- ▶ Jinak lze RRT/PRM aplikovat beze změny
- ▶ Prohledávaný prostor je vícedimenzionální (7 DoF + 1 DoF)

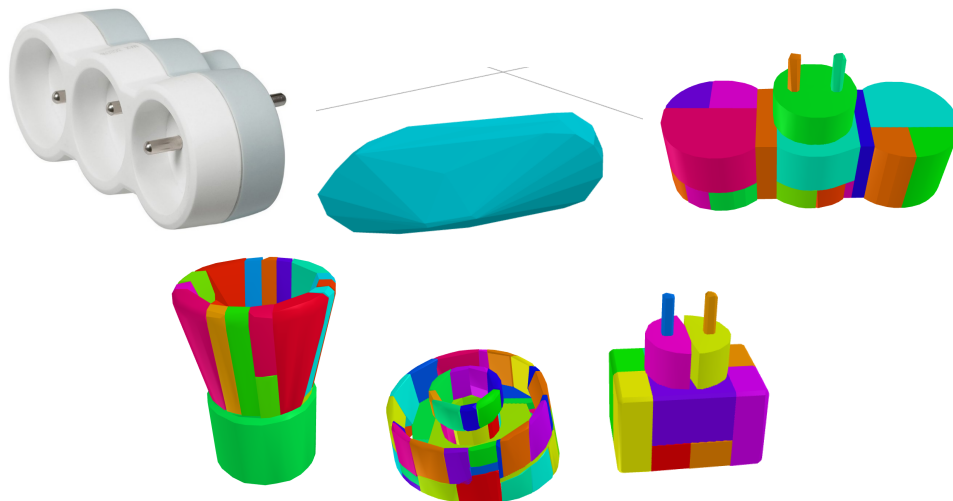




- ▶ Hledáme polohu a orientaci každého ramena
 - ▶ Výpočet přímé kinematické úlohy
 - ▶ Počítáme kolizi každého ramena zvlášť
- ▶ 3D modely bývají velmi detailní (a nekonvexní)
 - ▶ Pro urychlení výpočtu se spočítá konvexní aproximace
 - ▶ Konvexní obálka nebo konvexní dekompozice
 - ▶ Konvexní dekompozice se spočítá i pro objekty scény
- ▶ Bezkoliznost dvou konvexních objektů lze ověřit např. pomocí algoritmu GJK



Ukázka konvexní dekompozice objektů



Kolize mezi rameny



- ▶ Jsou ramena na obrázku v kolizi?
- ▶ Výpočet kolize je časově náročný (zvláště pro objekty, které jsou téměř v kontaktu)
- ▶ Snažíme se počet kolizních párů zredukovat
- ▶ V praxi vypínáme kolize mezi objekty, které:
 - ▶ jsou pořád v kolizi, např. sousední ramena
 - ▶ nemohou být nikdy v kolizi, např. oddělené statické kolizní objekty
 - ▶ ROS: MoveIt! - detekuje tyto kolizní páry při vytváření konfiguračních souborů

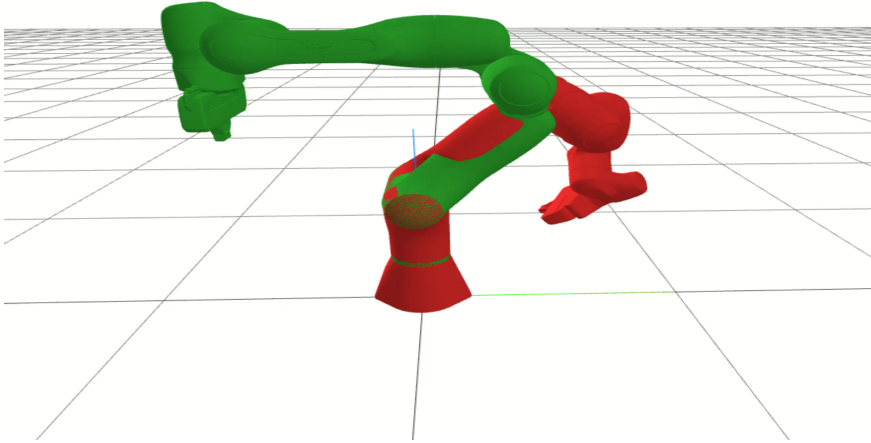


Detekce kolizí pro robota - závěr

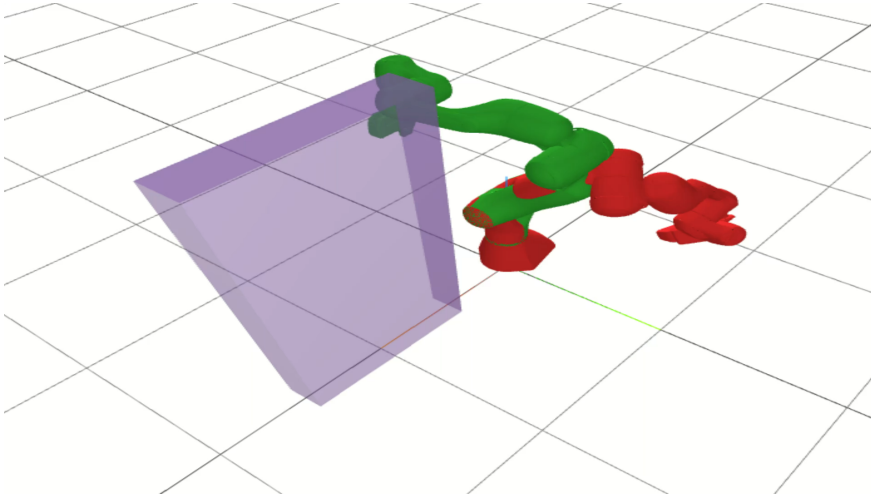
- ▶ Přípravná fáze
 - ▶ Spočítat konvexní obaly
 - ▶ Definovat kolizní páry
- ▶ Výpočetní fáze
 - ▶ Pomocí přímé kinematiky spočítáme polohu a orientaci každého ramena
 - ▶ Pro každý kolizní pár spočítáme bezkoliznost
 - ▶ Pokud je kterýkoliv pár v kolizi, konfigurace je nepřípustná



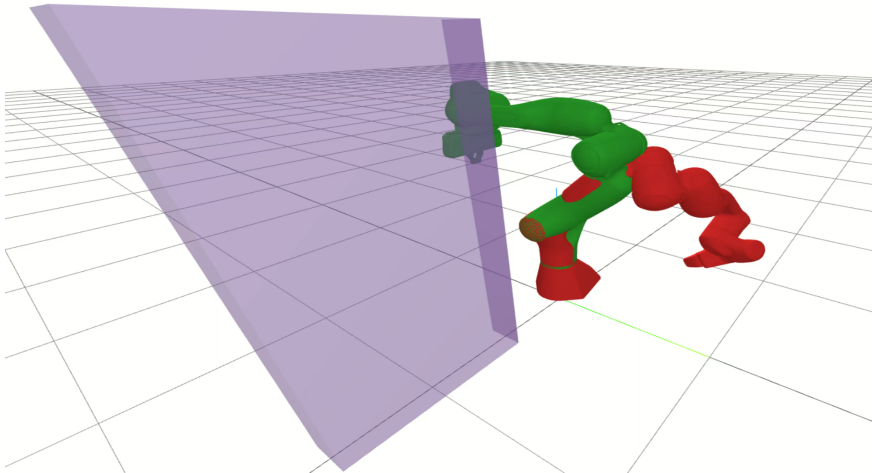
Naplánovaný pohyb



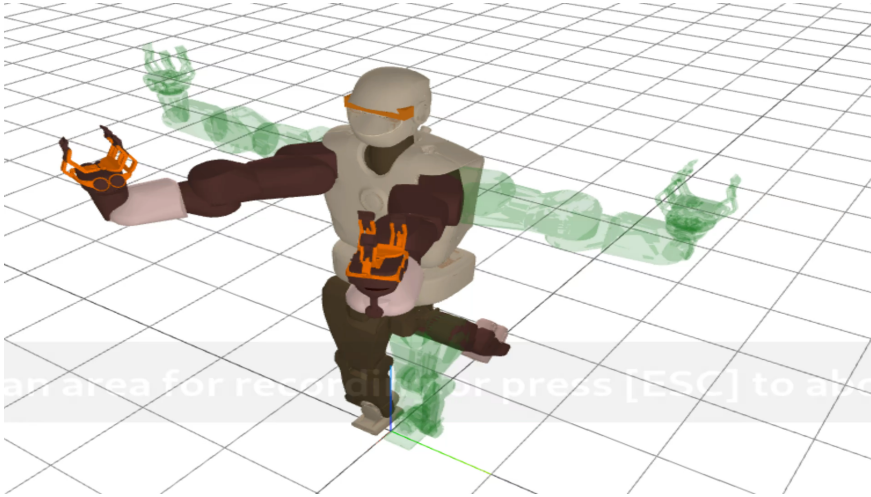
Naplánovaný pohyb RRT - 3132 iterací



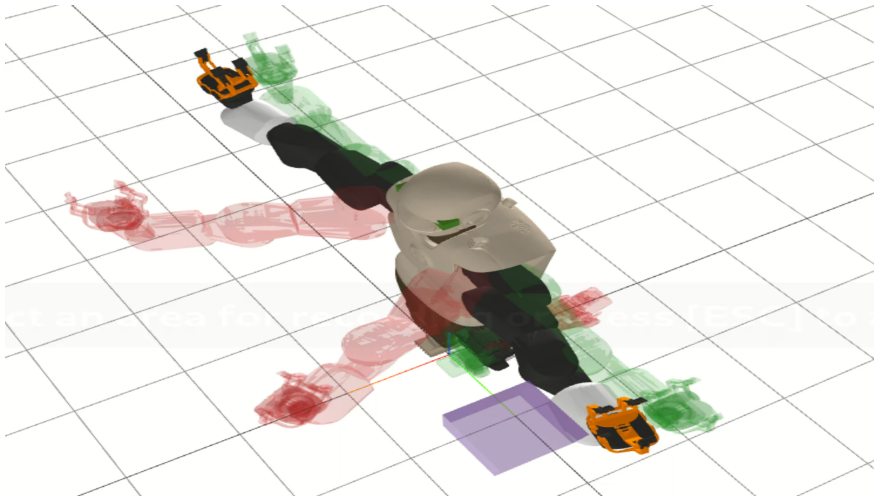
Naplánovaný pohyb RRT* - 10000 iterací



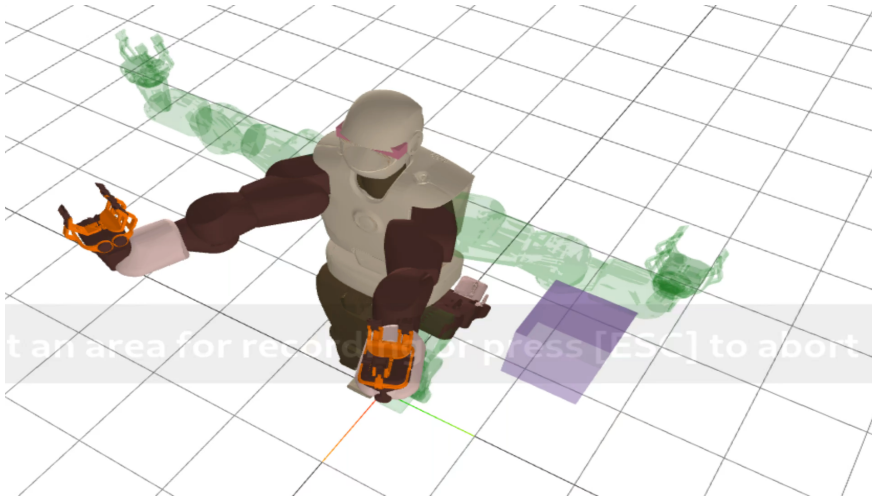
Naplánovaný pohyb Talos



Naplánovaný pohyb



Naplánovaný pohyb



Zkracování cesty: metoda náhodného zkracování

- ▶ Random shortcut
- ▶ Zkracování cesty pomocí náhodných vzorků a výpočtu nejkratší cesty
- ▶ Algoritmus:
 - ▶ Vyber náhodný bod q_1 na cestě
 - ▶ Vyber náhodný bod q_2 na cestě
 - ▶ Pokud je interpolace mezi q_1 a q_2 bezkolizní, nahraď cestu mezi q_1 a q_2 přímkou v konfiguračním prostoru



Shrnutí

- ▶ Co je to konfigurační prostor
- ▶ Rozdíl mezi cestou a trajektorií
- ▶ Co je to plánování cesty
- ▶ Rozdíl mezi plánovacími metodami:
 - ▶ Diskretizovaná mřížka
 - ▶ RRT
 - ▶ RRT*
 - ▶ PRM
- ▶ Co chybělo:
 - ▶ Plánovačů existuje spousta
 - ▶ Optimalizační metody plánování



- ▶ Implementace PRM
- ▶ Domácí úkol: implementace RRT a Random shortcut

