

Robotika: Plánování pohybu

Vladimír Petrík

vladimir.petrik@cvut.cz

9. 11. 2026

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

Motivace

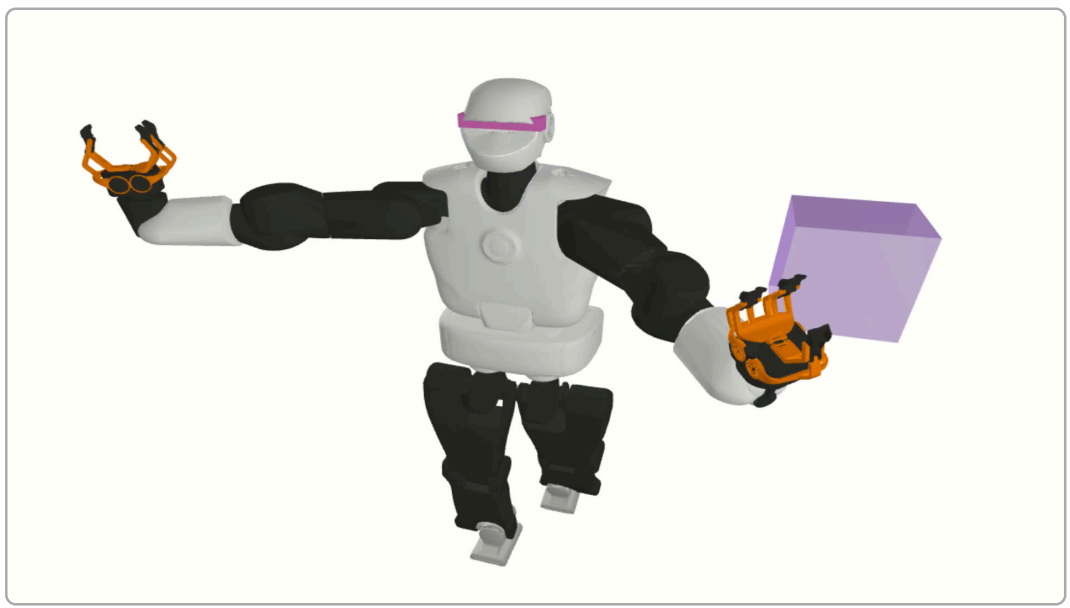
- Představte si úlohu, kde máte pomocí robota přenést plný hrnek z jednoho místa na druhé
- Jak byste to řešili pomocí metod, které znáte?
  - spojit úsečkou počáteční a koncový bod – interpolace polohy
  - zafixovat orientaci hrnku
  - výpočet inverzní kinematiky
- Možné problémy
  - inverzní kinematika spočítá několik řešení
  - inverzní kinematika nemusí najít žádné řešení
  - překážka v přímé cestě

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

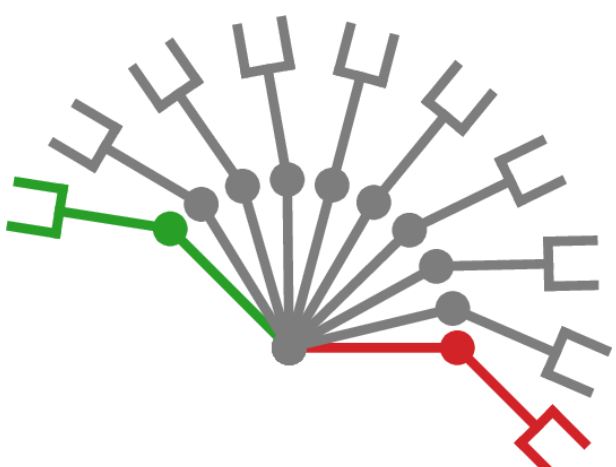
Cíl dnešní přednášky

- 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

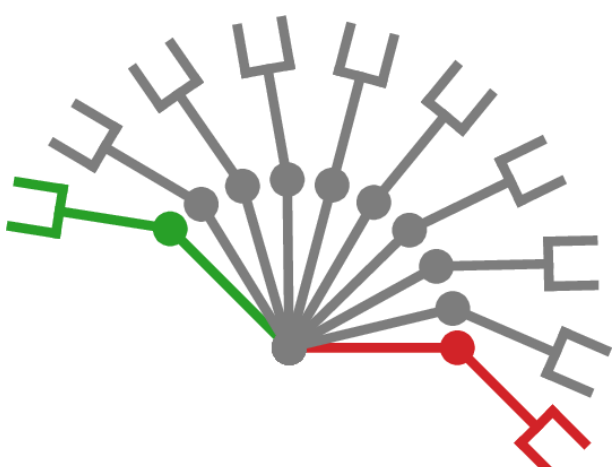
- Plánování pohybu pro robota z daného startu do daného cíle
  - vyhýbání překážkám
  - respektování omezení kloubů
  - respektování maximální rychlosti, sil/momentů
  - ostatní omezení daného prostředí (např. nevylít vodu z hrnku $\rightarrow$ omezení orientace)



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

Formulace problému

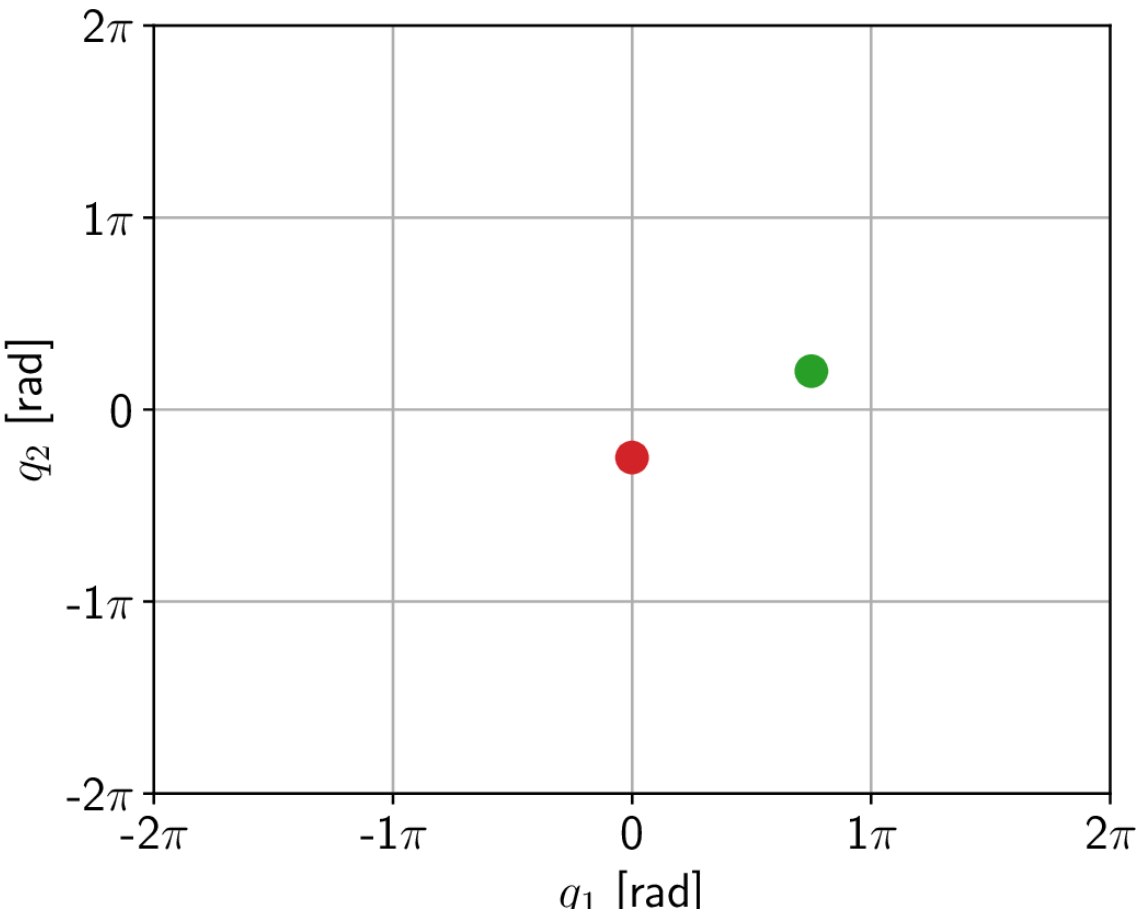
- Plánování pohybu pro robota z daného startu do daného cíle
  - vyhýbání překážkám
  - respektování omezení kloubů
  - respektování maximální rychlosti, sil/momentů
  - ostatní omezení daného prostředí (např. nevylít vodu z hrnku $\rightarrow$ omezení orientace)
- Plánování probíhá v konfiguračním prostoru robota
  - konfigurační prostor $\mathcal{C}$ (Configuration space, C-space)
  - konfigurace robota $q \in \mathcal{C}$
  - start i cíl je definován v konfiguračním prostoru, $(\boldsymbol{q}_s, \boldsymbol{q}_g) \in \mathcal{C} \times \mathcal{C}$

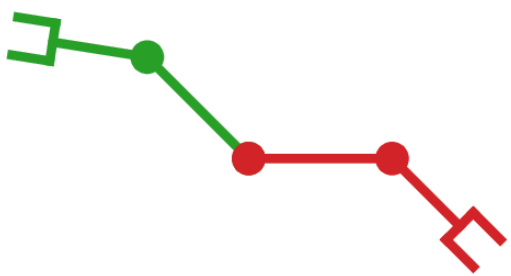


Existují *task-space* plánovače, ale nebudeme se jimi dnes zabývat.

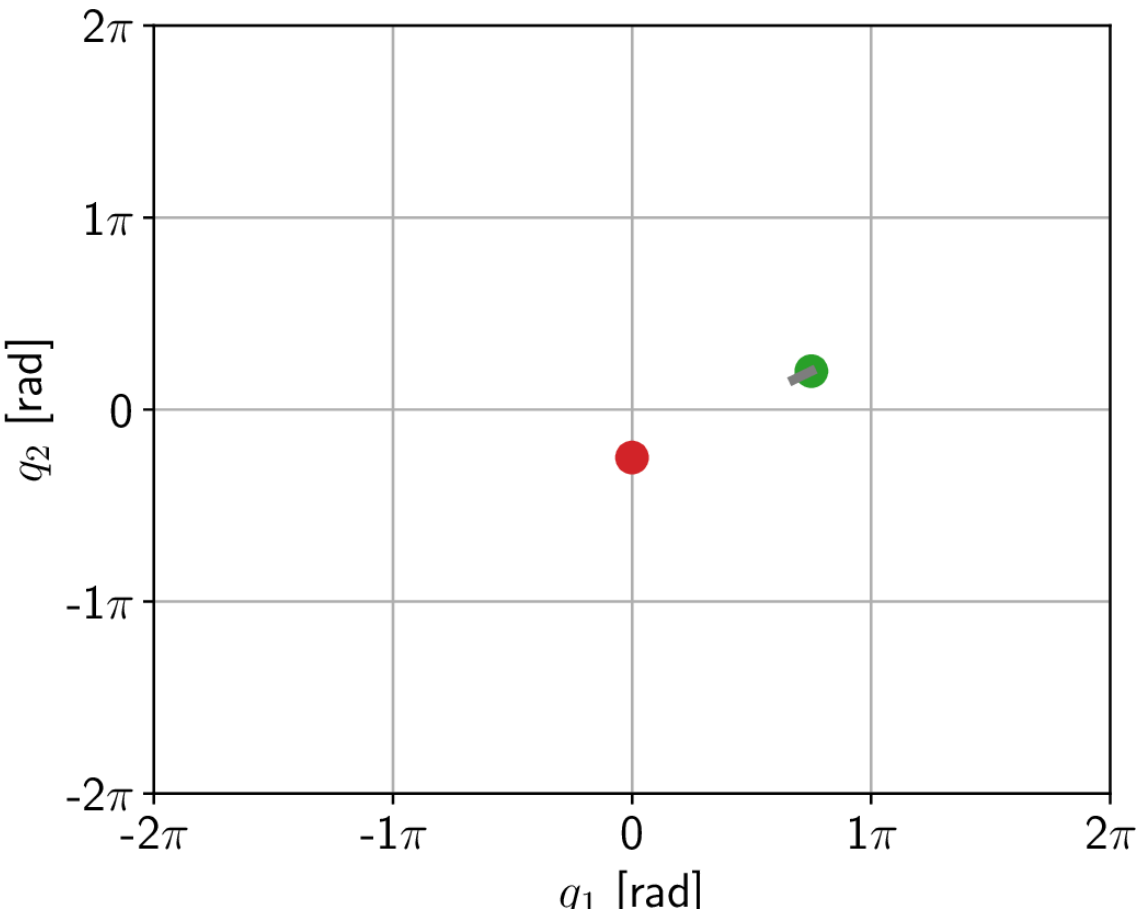
Konfigurační prostor

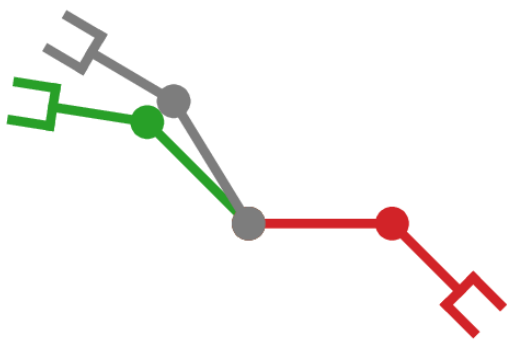
Konfigurační prostor – interpolace



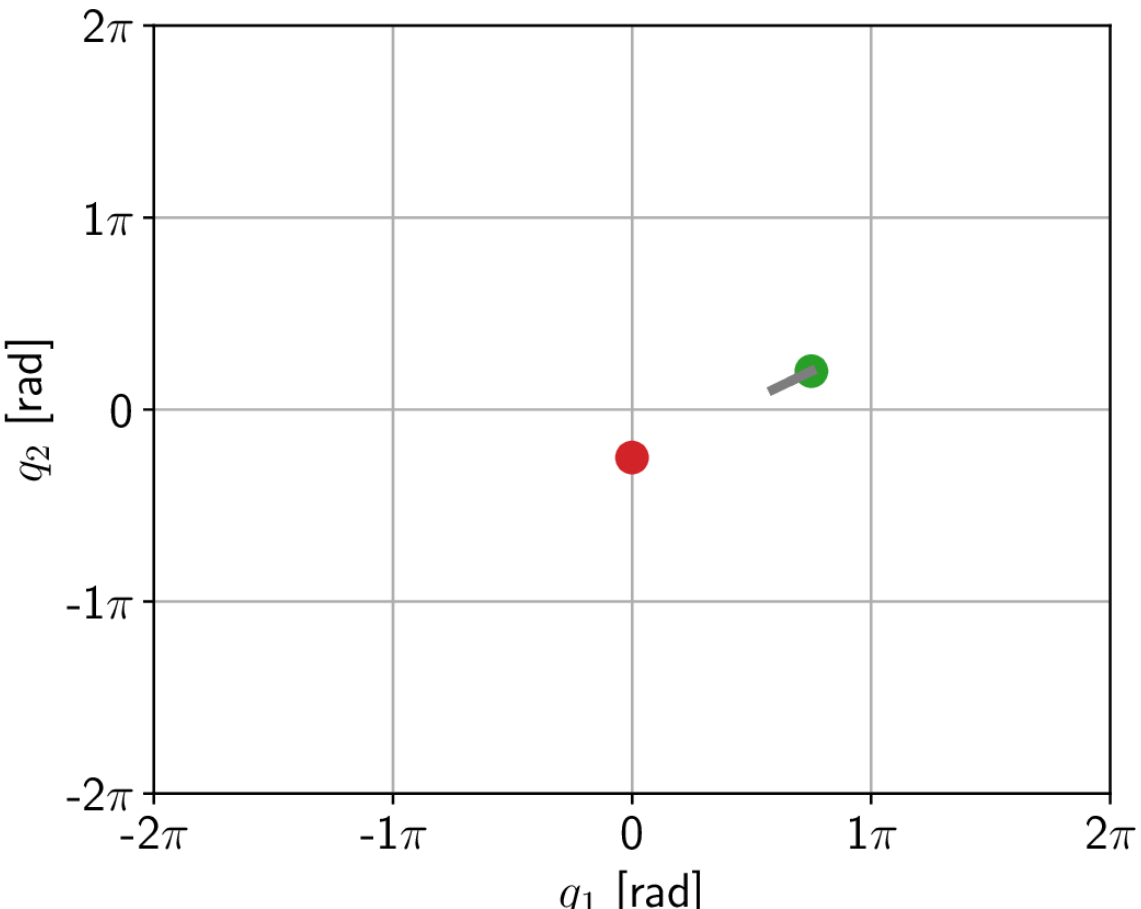


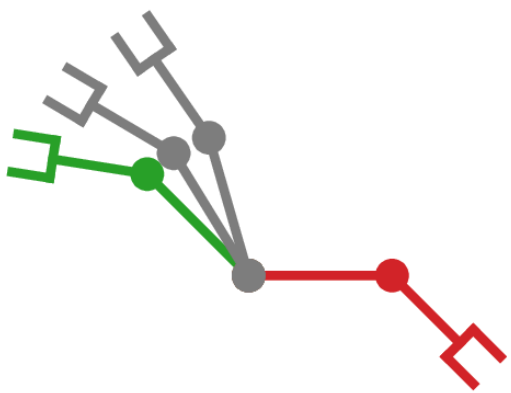
Konfigurační prostor – interpolace



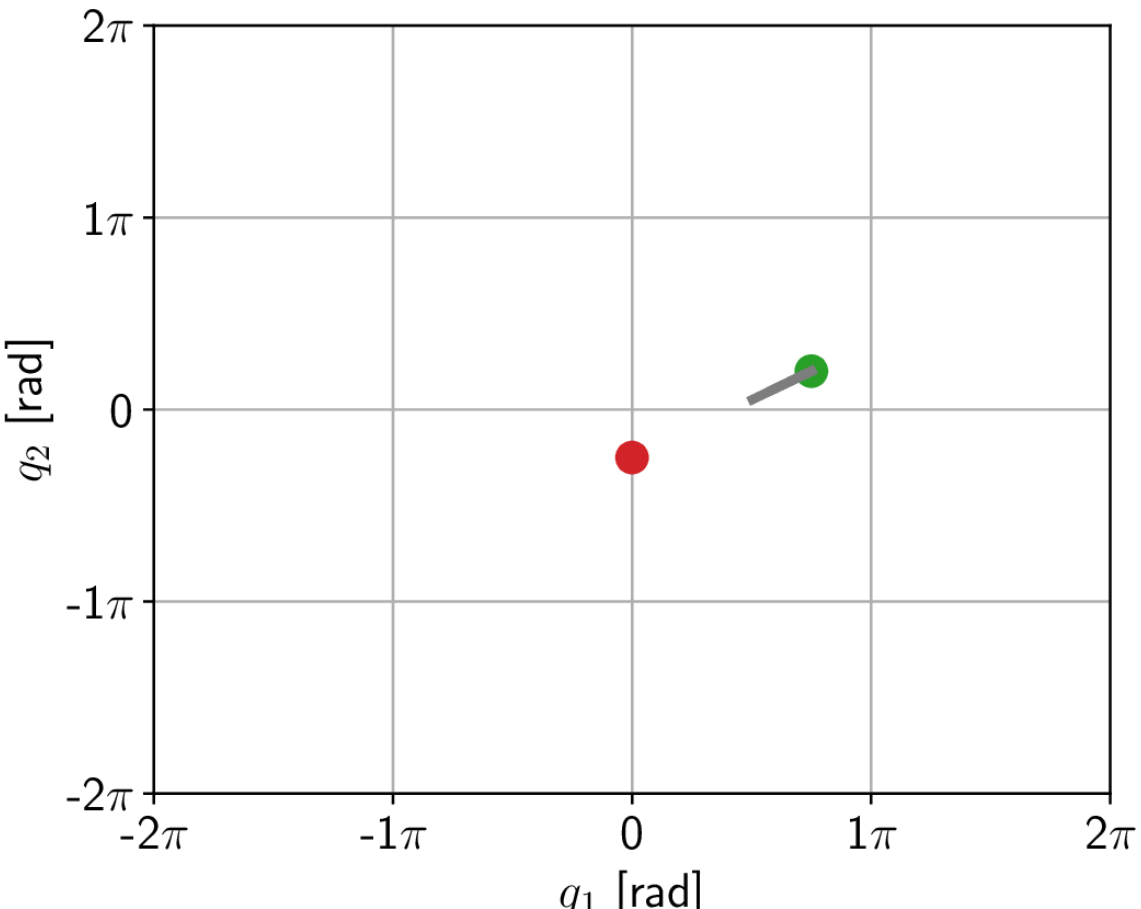


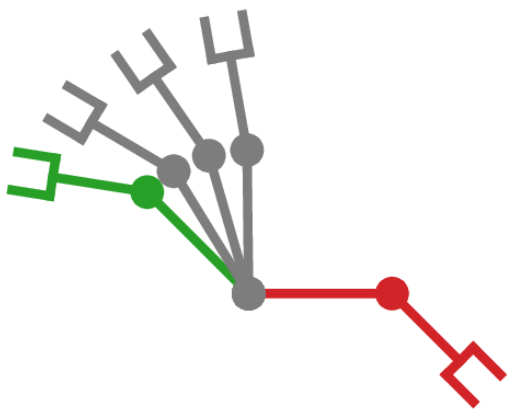
Konfigurační prostor – interpolace



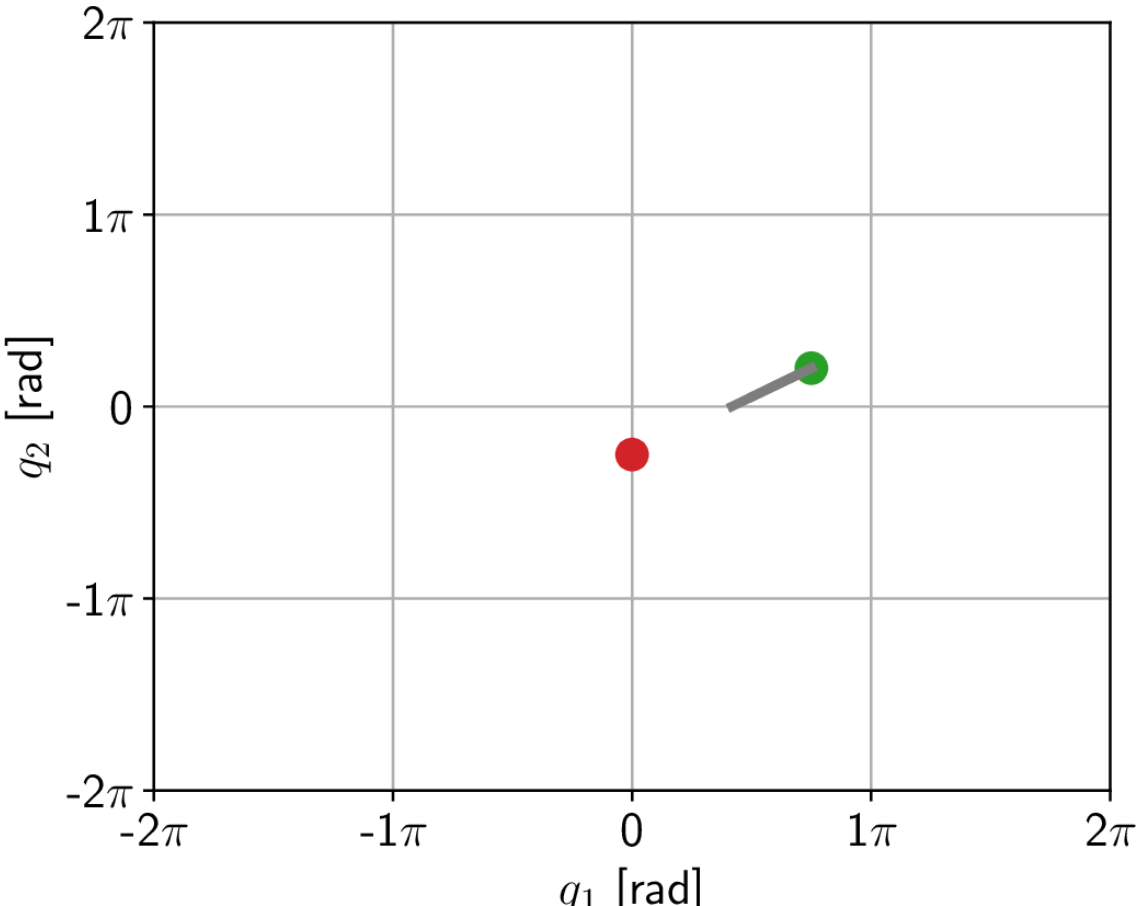


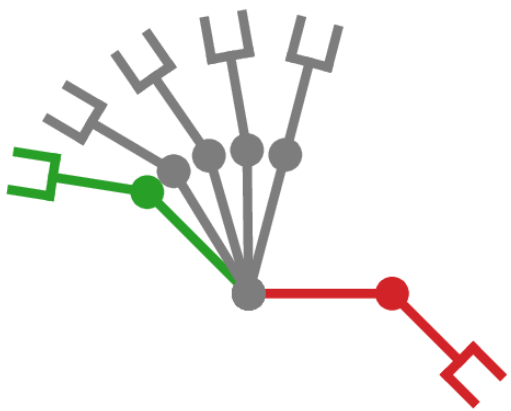
Konfigurační prostor – interpolace



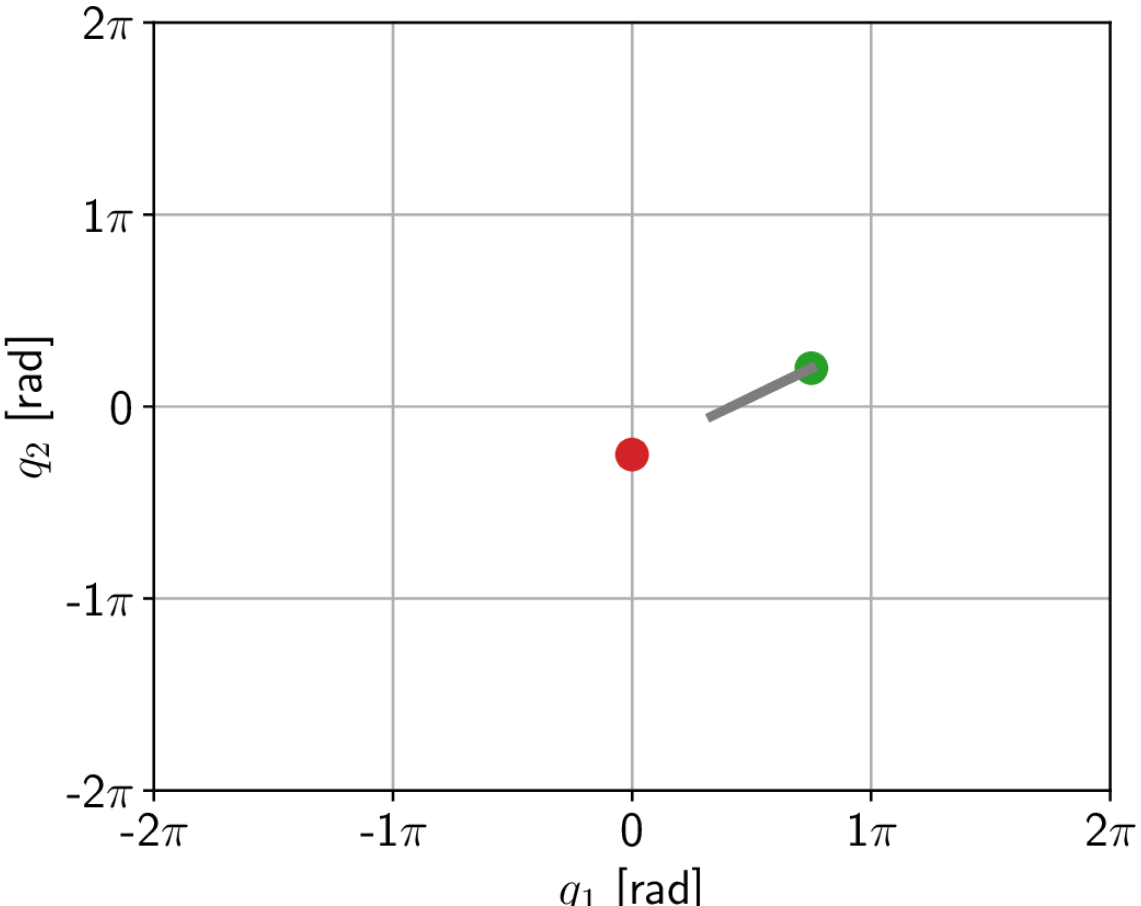


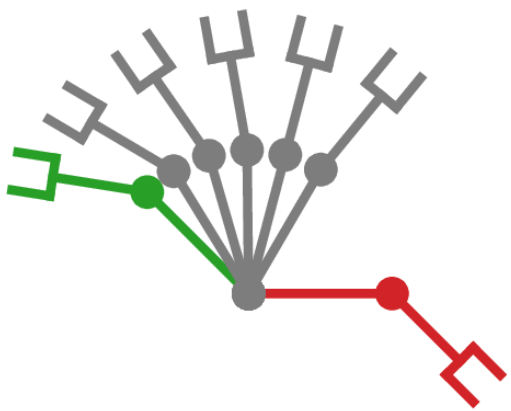
Konfigurační prostor – interpolace



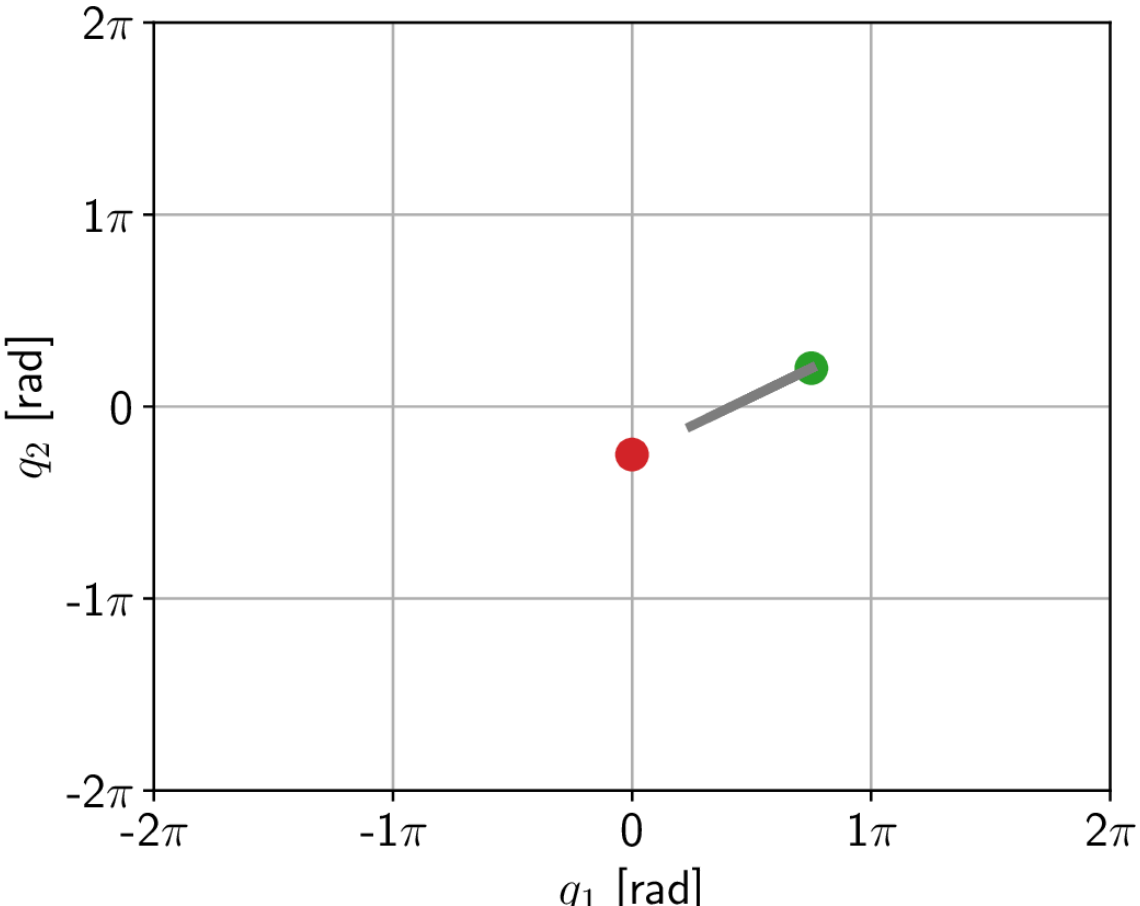


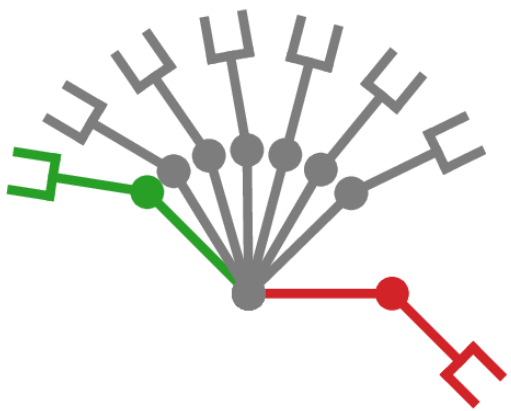
Konfigurační prostor – interpolace



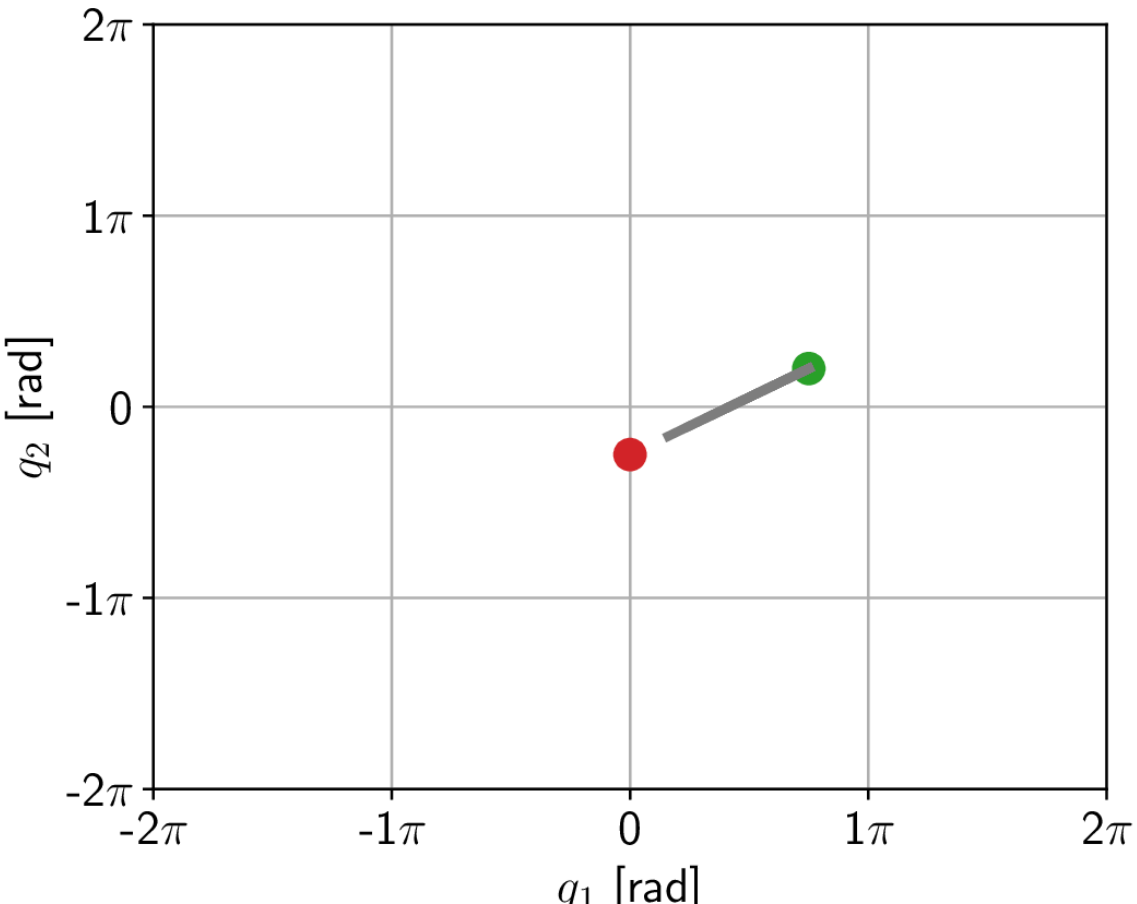


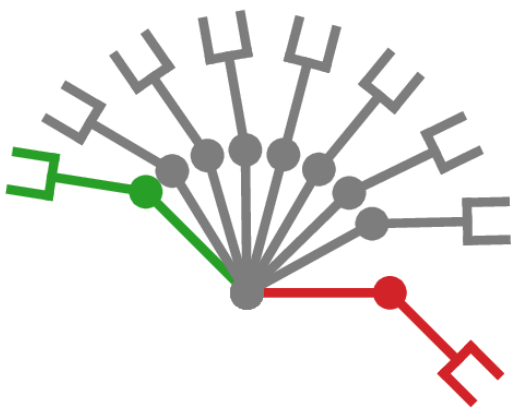
Konfigurační prostor – interpolace



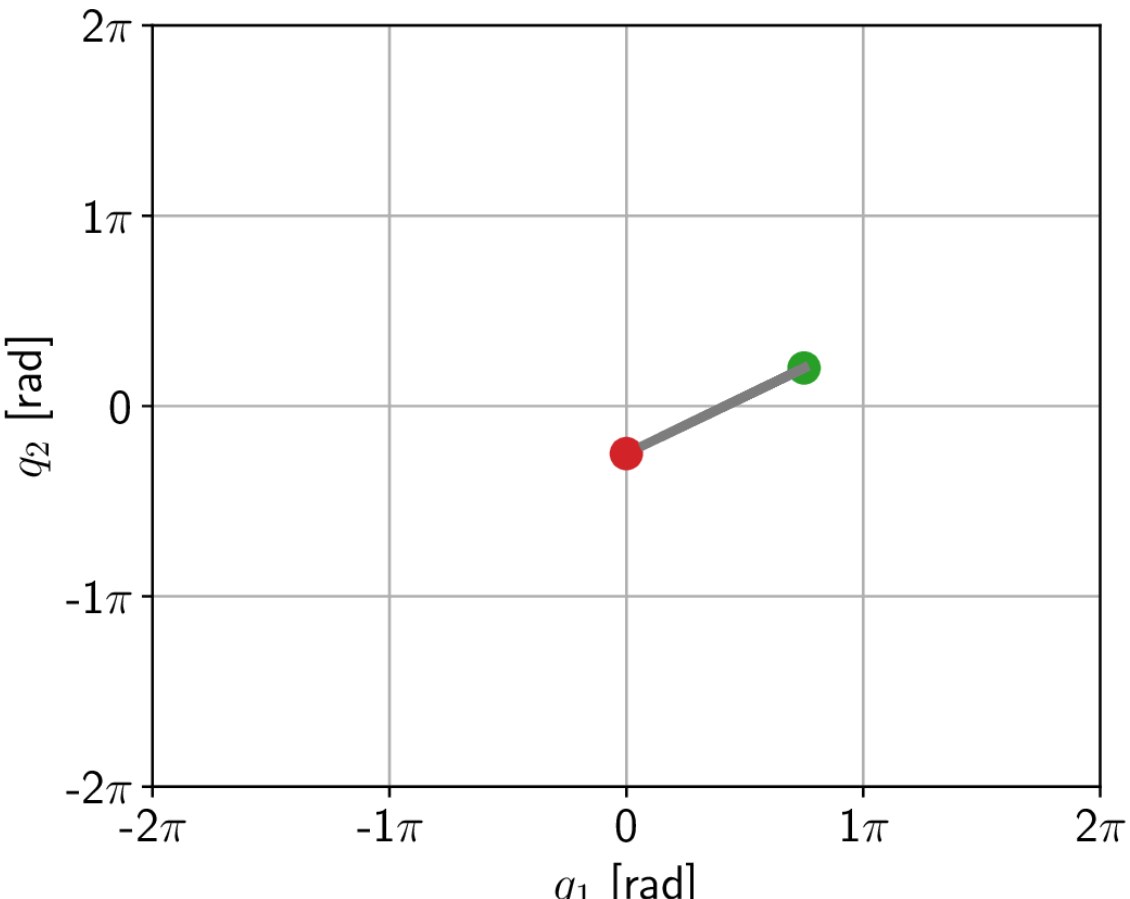


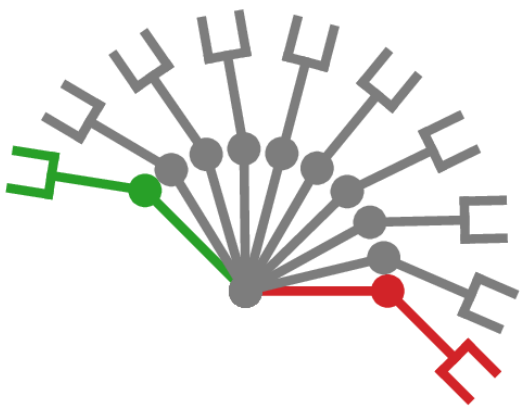
Konfigurační prostor – interpolace



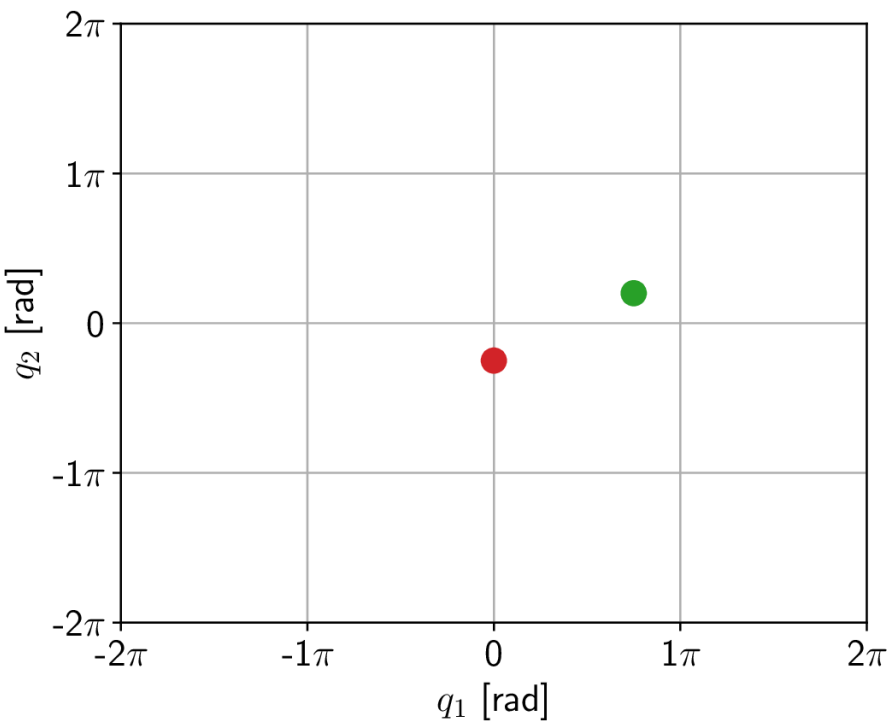


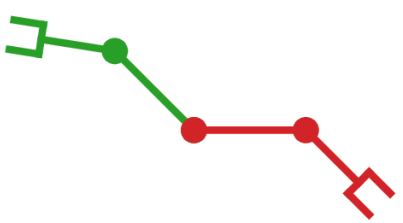
Konfigurační prostor – interpolace





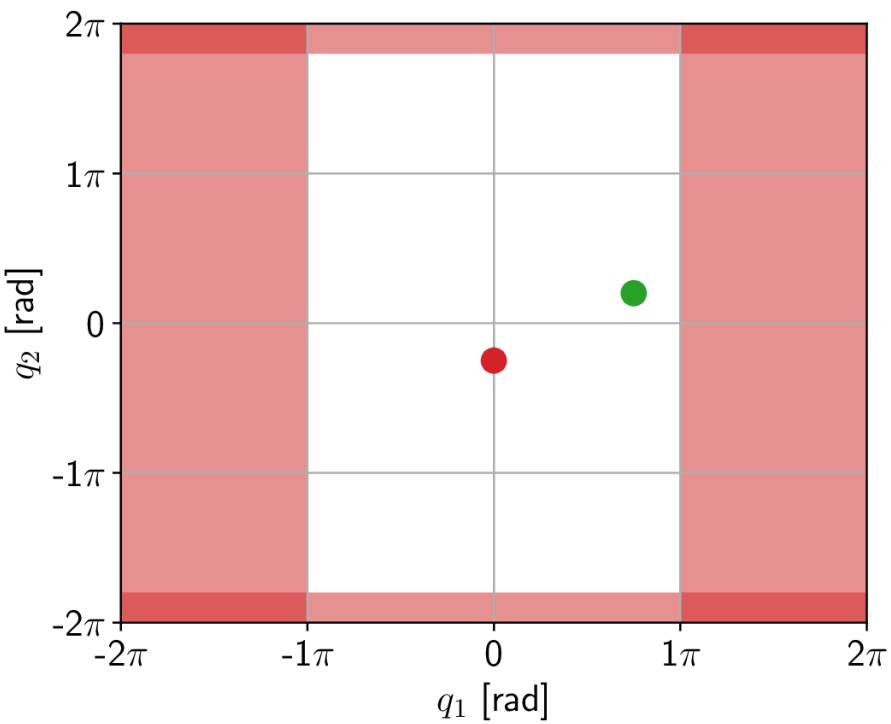
Vliv omezení na konfigurační prostor

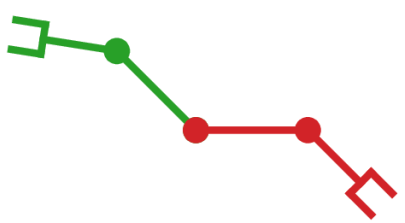




Vliv omezení na konfigurační prostor

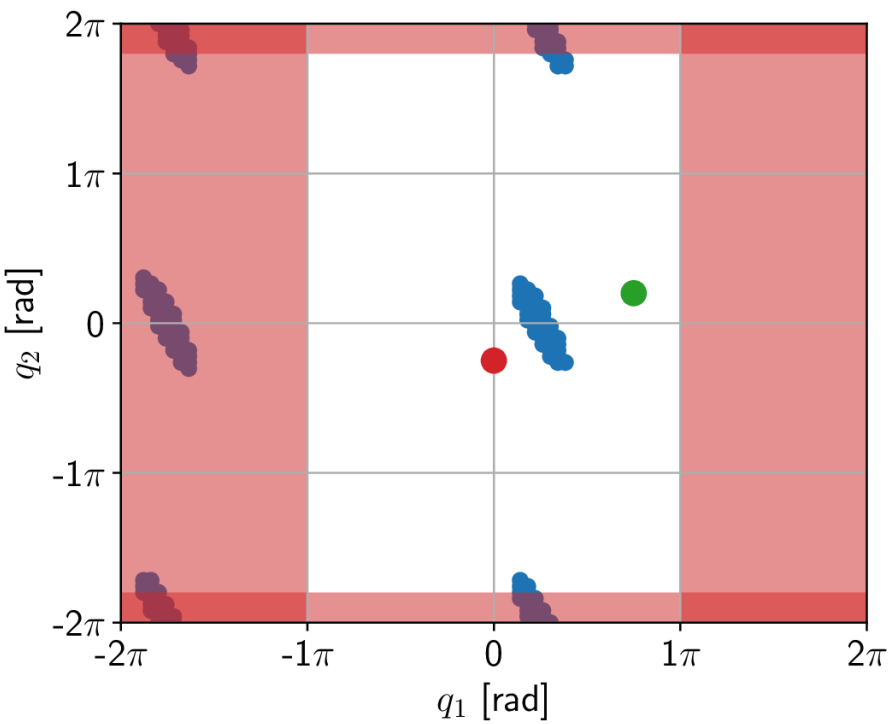
- Limity kloubů

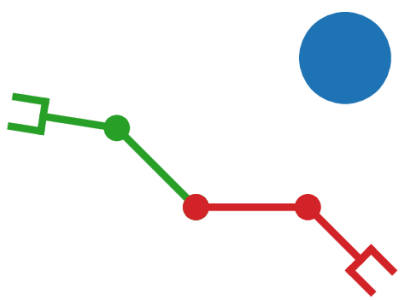




Vliv omezení na konfigurační prostor

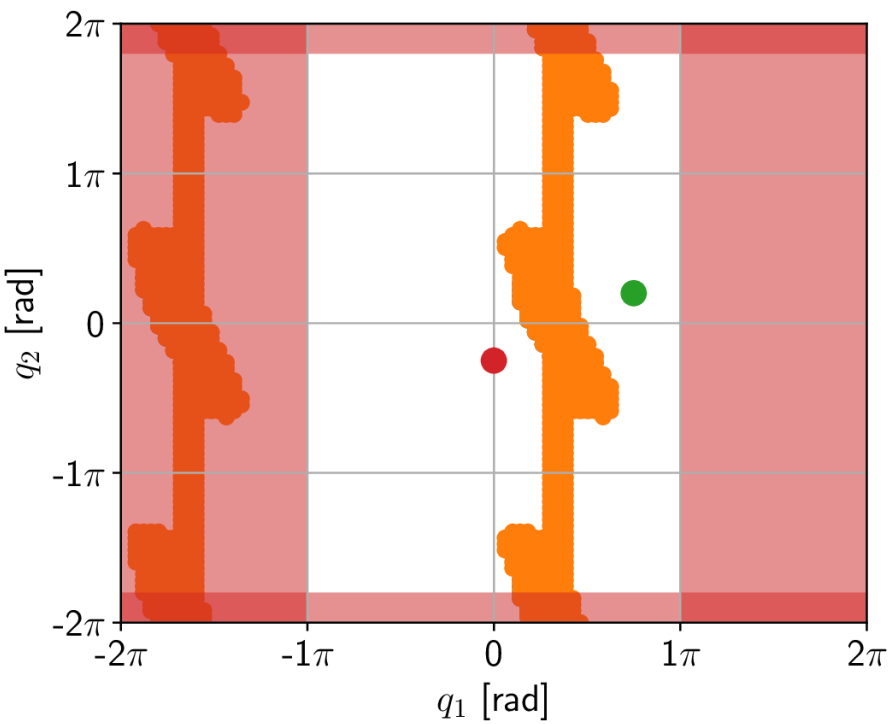
- Limity kloubů
- Překážky prostředí (pozn. tvar chapadla ignorován)

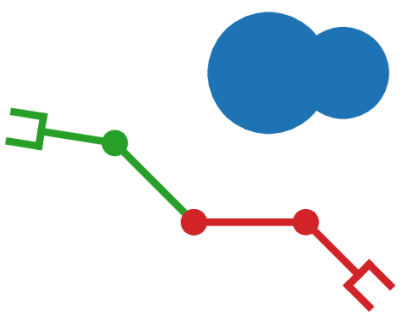




Vliv omezení na konfigurační prostor

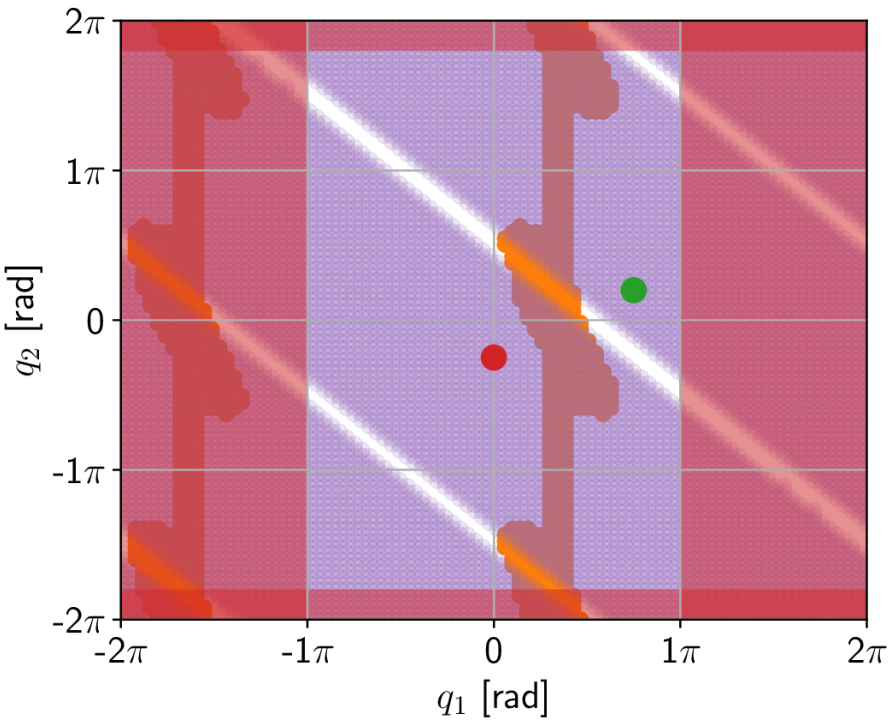
- Limity kloubů
- Překážky prostředí (pozn. tvar chapadla ignorován)

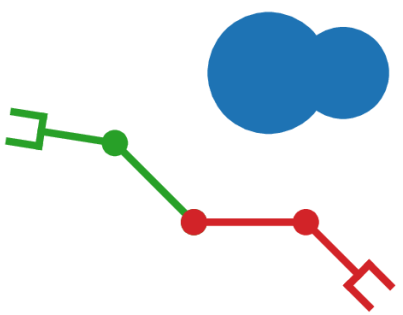




Vliv omezení na konfigurační prostor

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



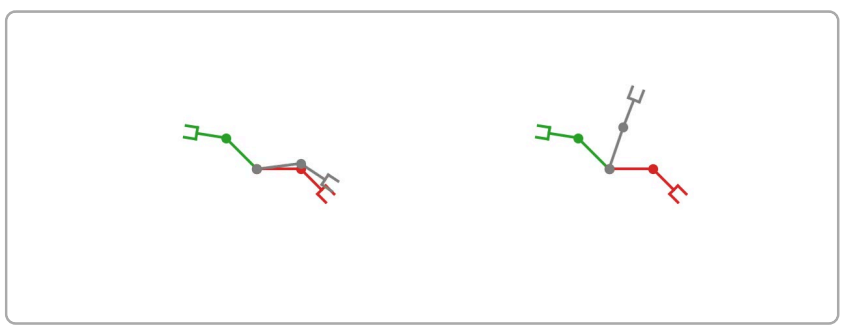


Konfigurační prostor – závěr

- Konfigurační prostor rozdělujeme na:
  - prostor přípustných konfigurací $\mathcal{C}_{\text{free}}$
  - prostor nepřípustných konfigurací $\mathcal{C}_{\text{occ}}$
  - $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{occ}}$
- 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 $\times$ trajektorie

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



Můžeme plánovat rovnou trajektorii?

- Definice problému zahrnuje rovnici pohybu systému
- Najdi kontrolní signál v čase $\boldsymbol{u}(t)$ tak, aby:
- $q(0) = \boldsymbol{q}_s, \quad q(T) = \boldsymbol{q}_g, \quad \dot{q} = f(q(t), \boldsymbol{u}(t))$
- Komplikovanější úloha než hledání cesty
- Kdy je preferovaná?
  - jsou poháněné jenom některé klouby $\rightarrow$ 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

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

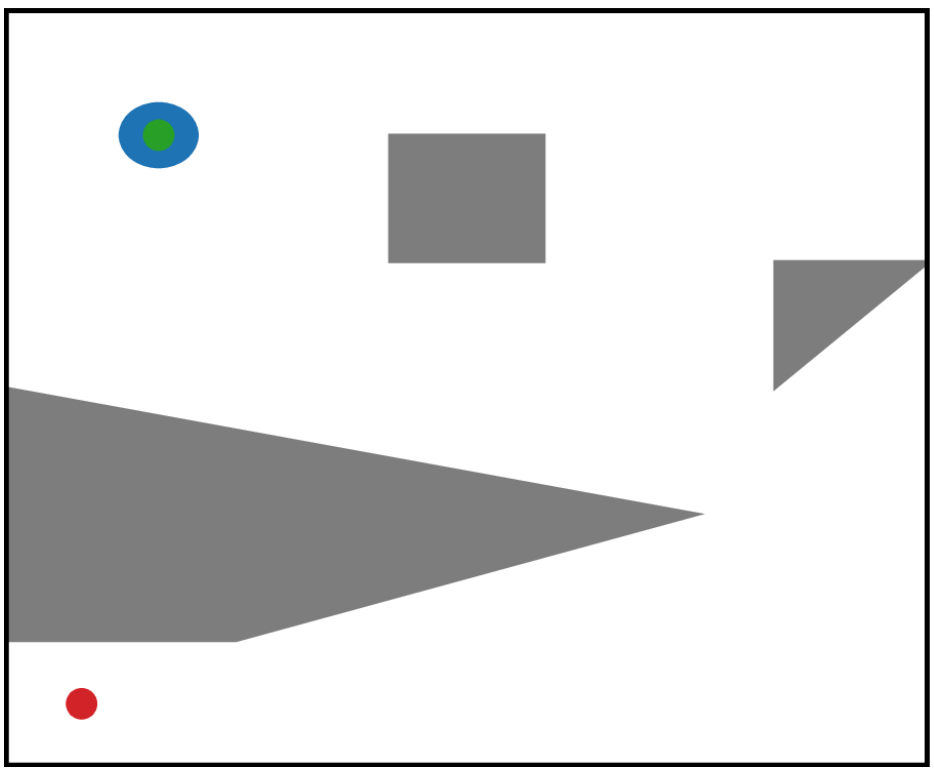
Plánování cesty v mobilní robotice

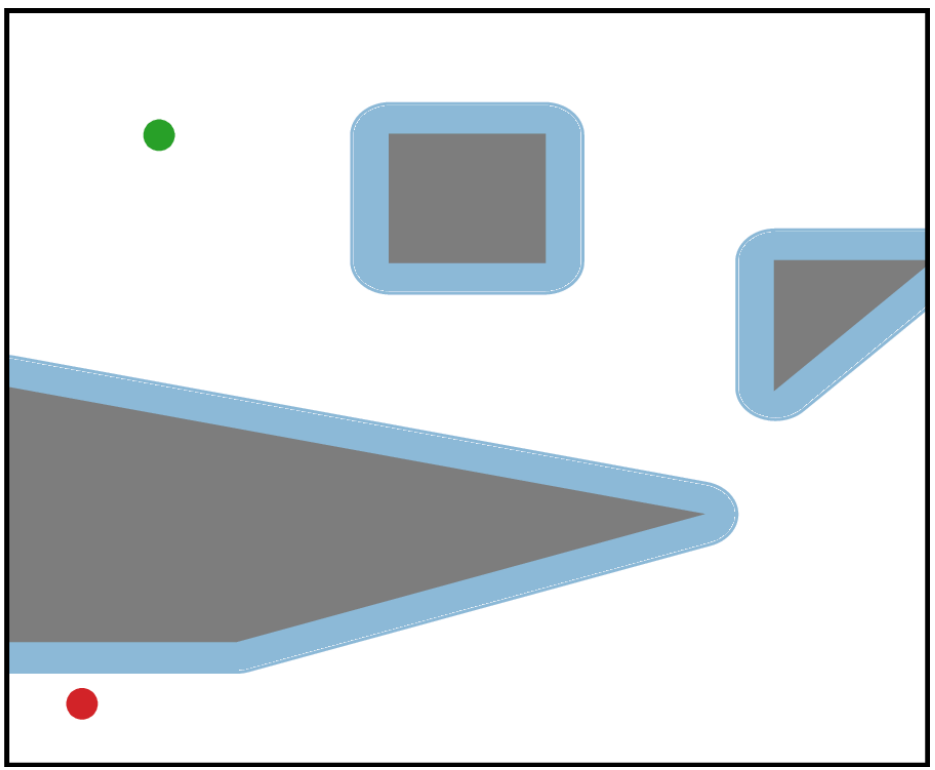
- Jak plánování funguje v ROS: MoveBase
- Robot se pohybuje v rovině $(v_x, v_\theta)^\top$
- Známe:
  - rozměry robota (půdorys)
  - mapu prostředí
  - pozici robota v mapě $(x, y, \theta)^\top$
  - cílovou pozici v mapě
- Řešení navigace do zadaného cíle v ROS: MoveBase:
  - globální plánovač cesty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

Plánování cesty v mobilní robotice

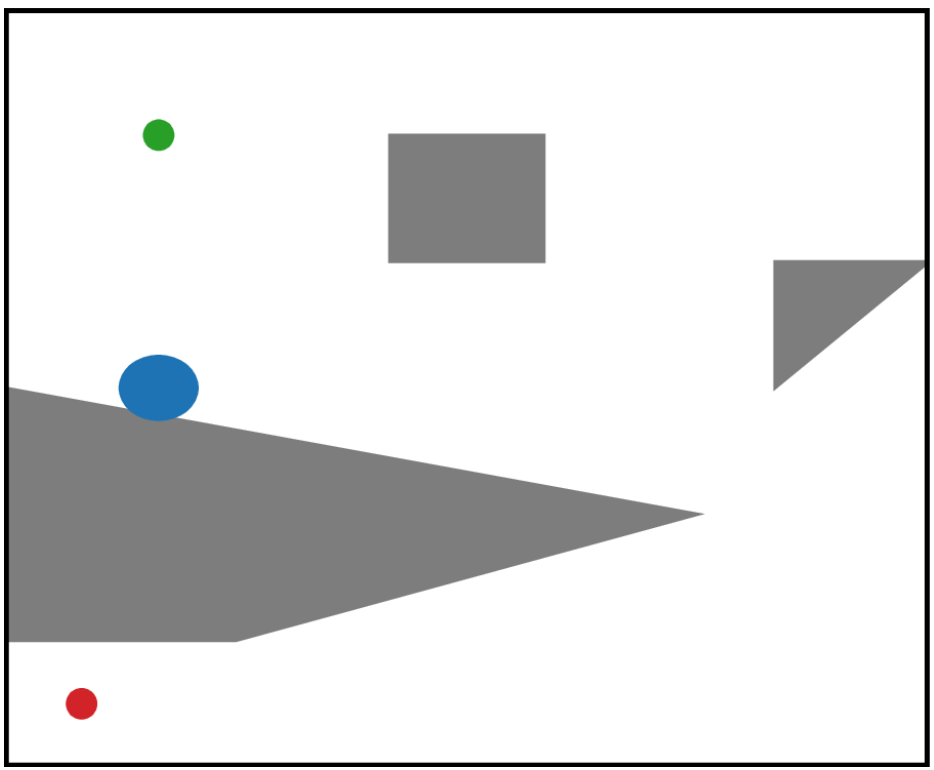
- Jak plánování funguje v ROS: MoveBase
- Robot se pohybuje v rovině $(v_x, v_\theta)^\top$
- Známe:
  - rozměry robota (půdorys)
  - mapu prostředí
  - pozici robota v mapě $(x, y, \theta)^\top$
  - cílovou pozici v mapě
- Řešení navigace do zadaného cíle v ROS: MoveBase:
  - **globální plánovač cesty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

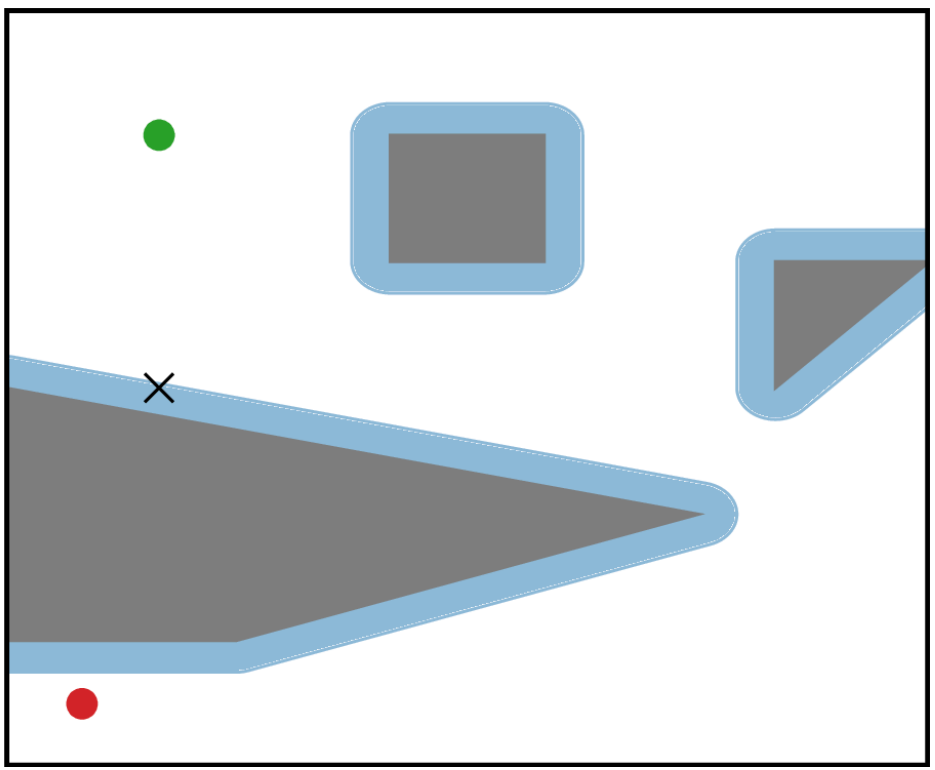
Jak vypadá mapa a konfigurační prostor





Jak vypadá mapa a konfigurační prostor

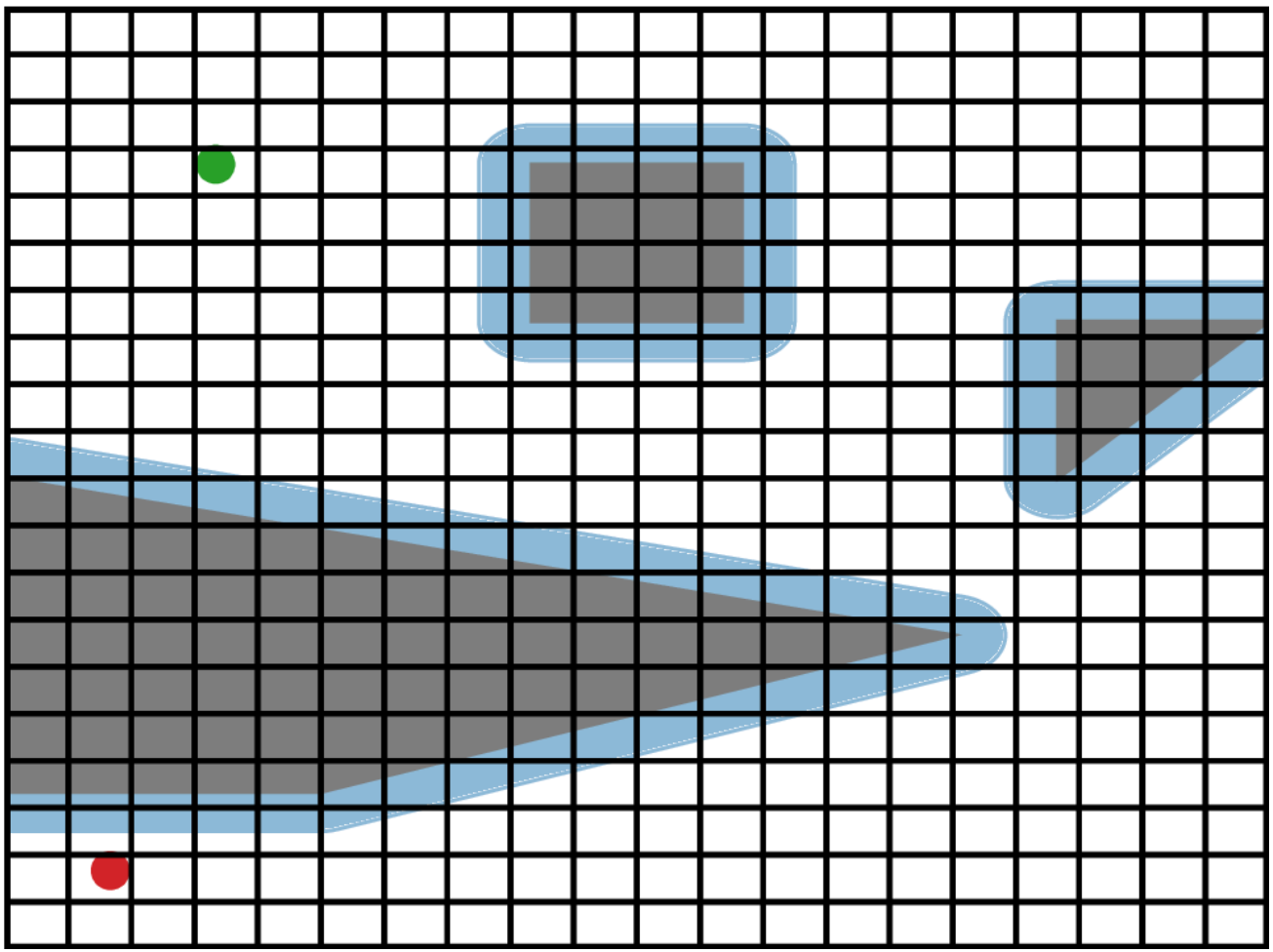




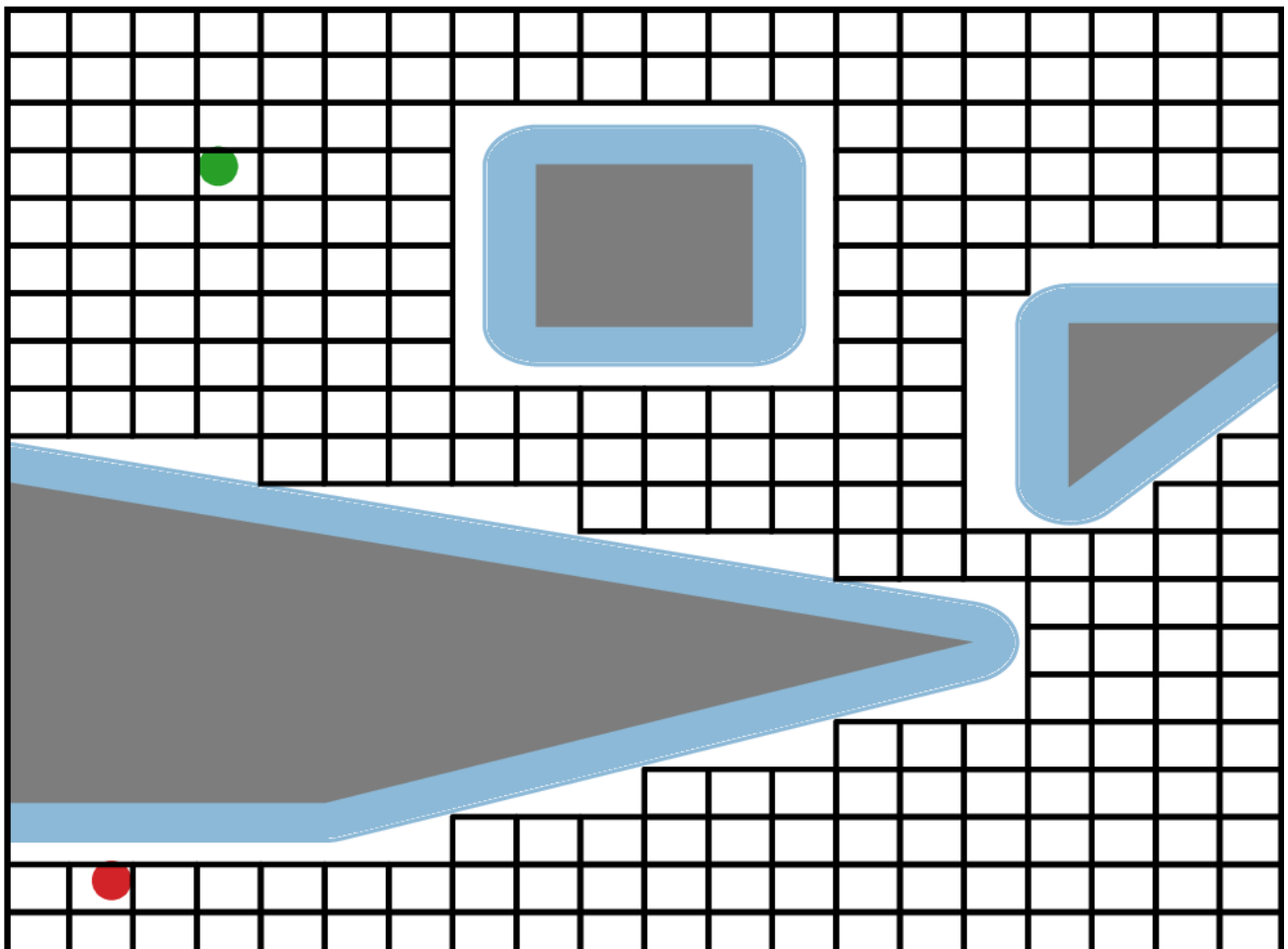
Plánovače #1: plánování v diskrétní mřížce

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

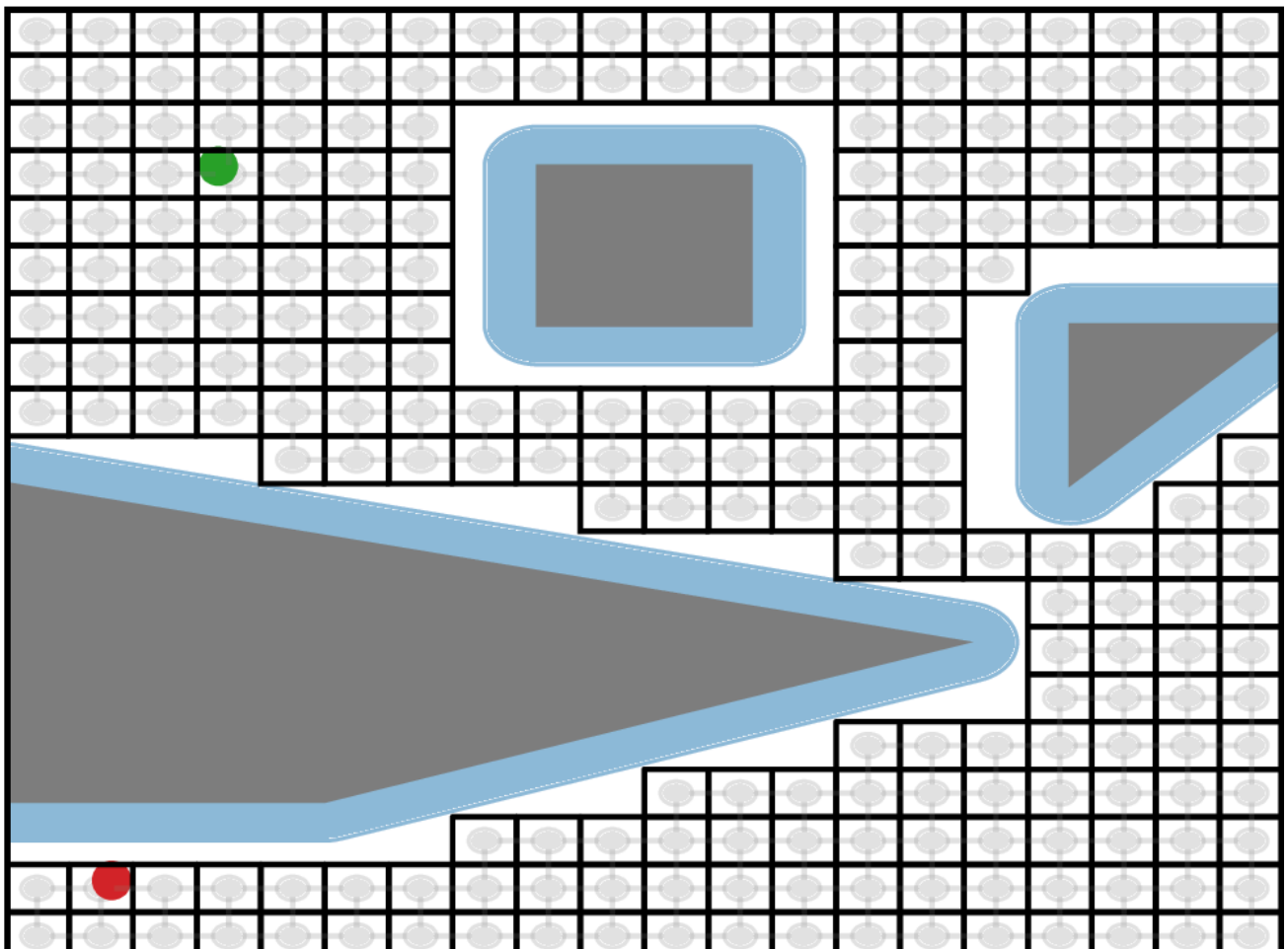
Plánovače #1: plánování v diskrétní mřížce



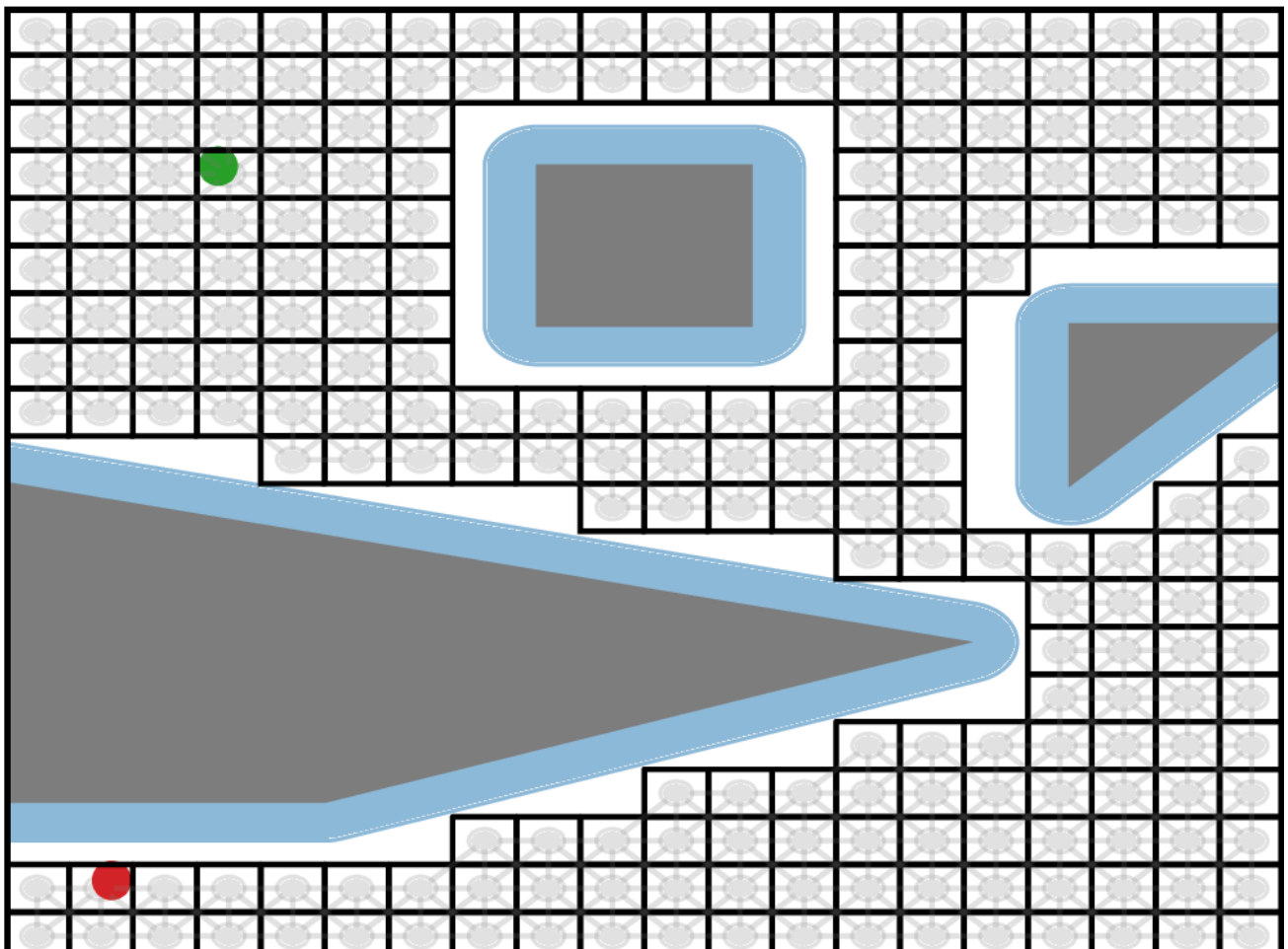
Plánovače #1: plánování v diskrétní mřížce



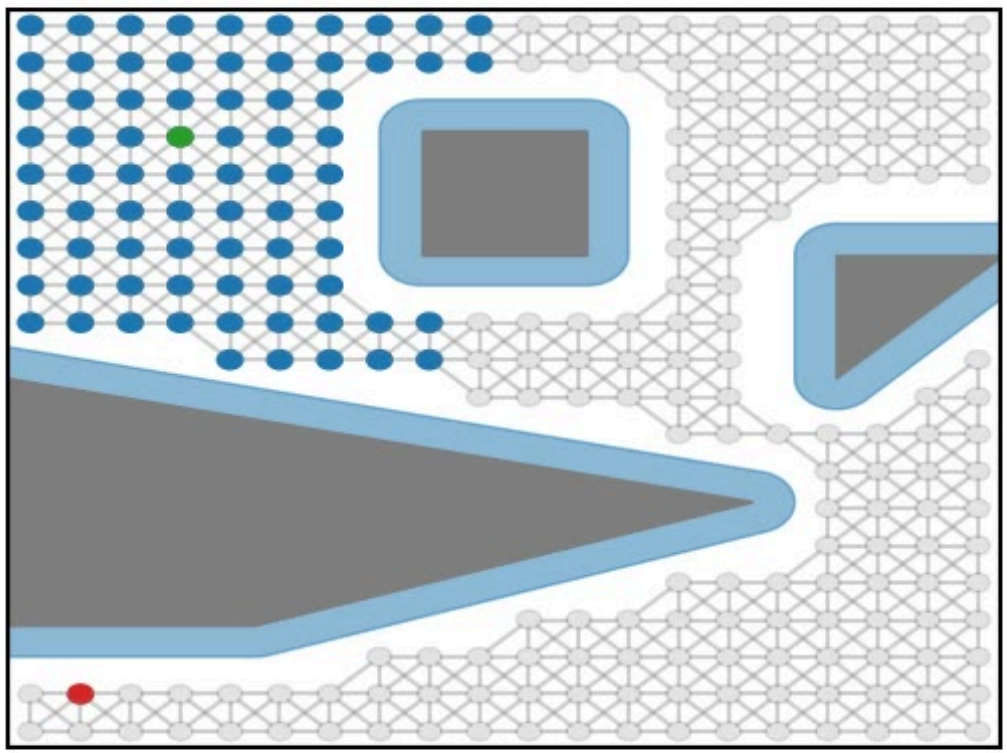
Plánovače #1: plánování v diskrétní mřížce



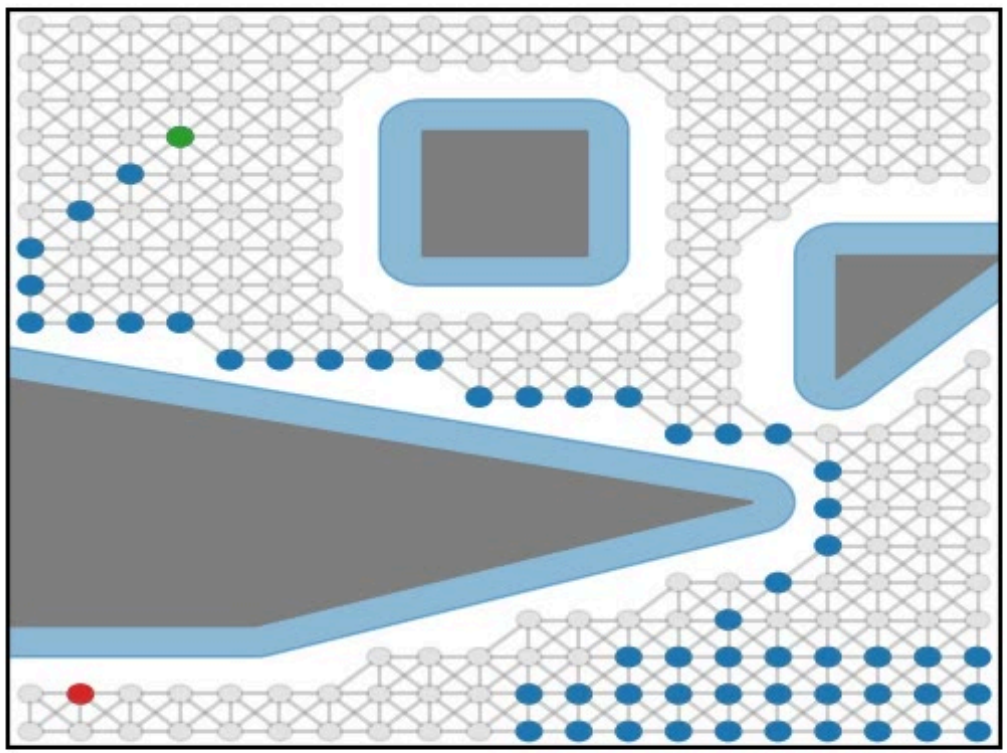
Plánovače #1: plánování v diskrétní mřížce



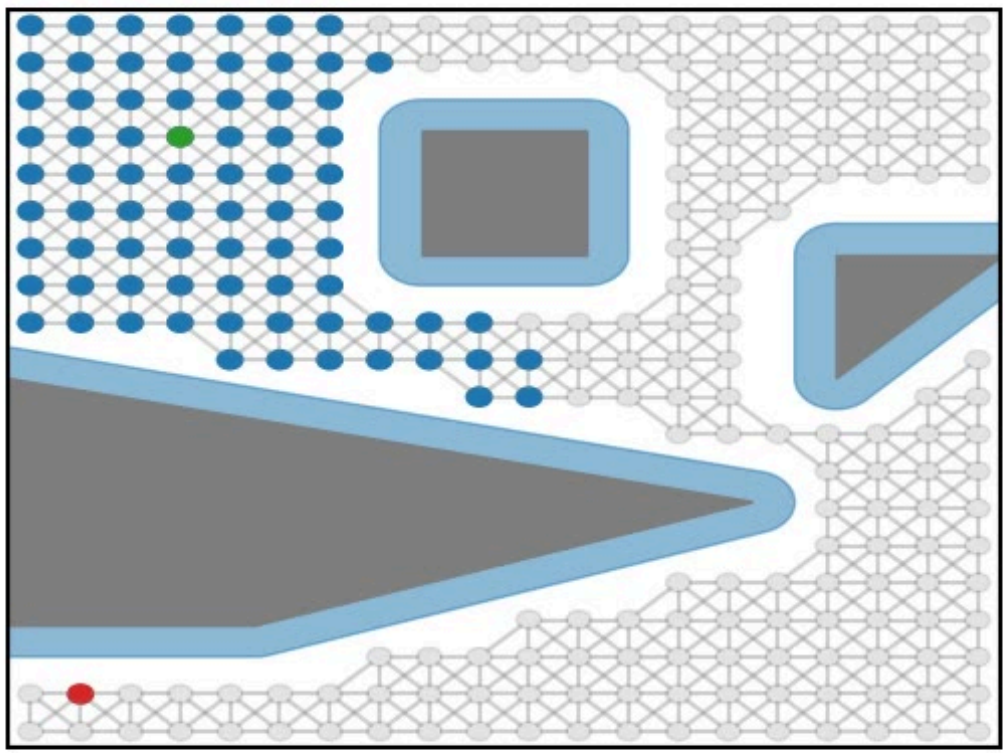
Prohledávání grafu: do šířky



Prohledávání grafu: do hloubky



Prohledávání grafu: informované prohledávání A*

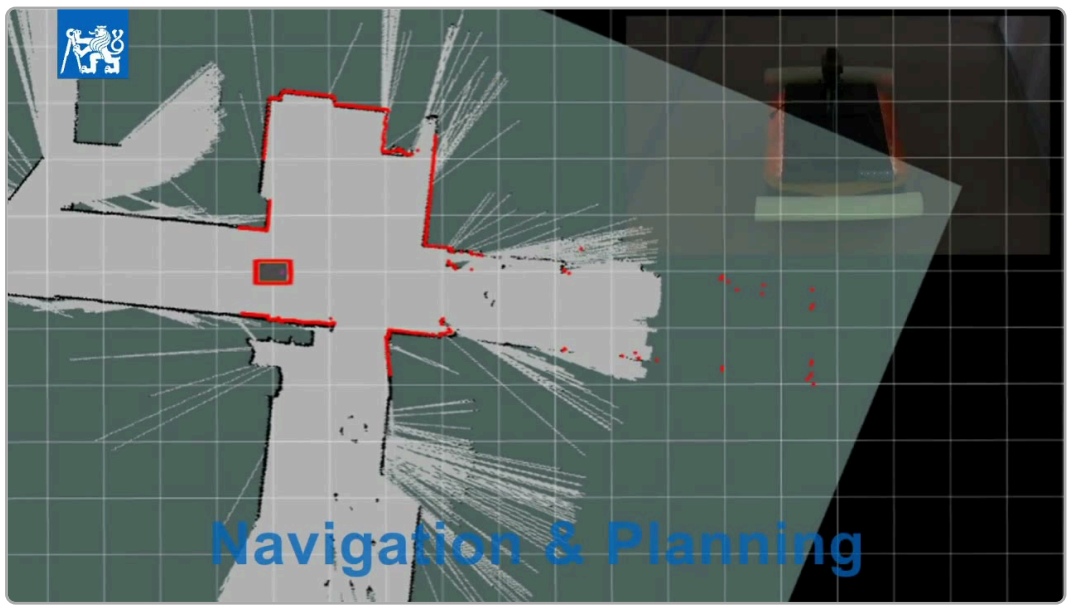


Prohledávání grafu

- 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 ho zrychluje
  - úplné, optimální

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 ho zrychluje
  - úplné, optimální

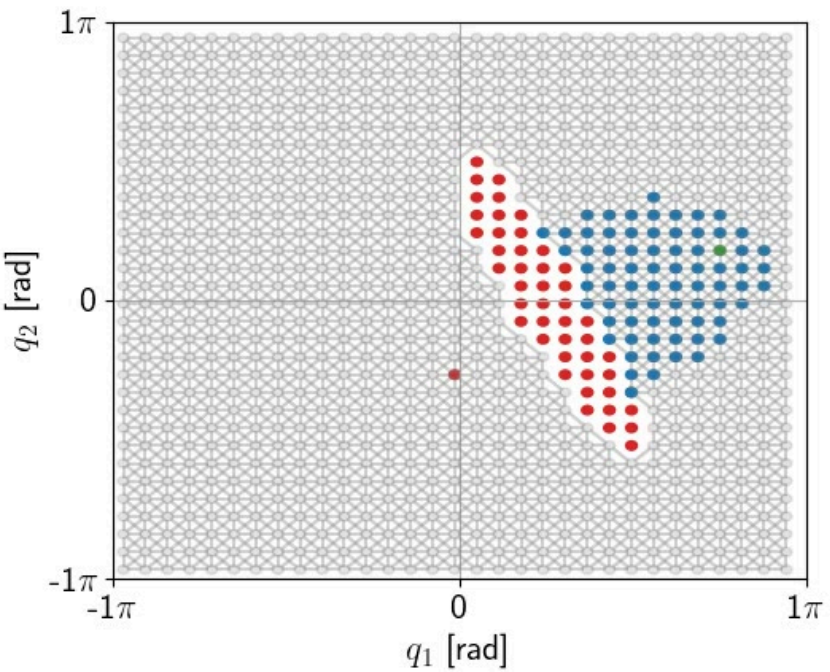


Prohledávání grafu

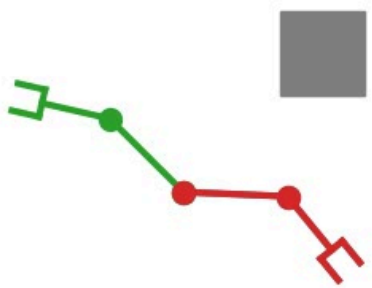
- Neinformované do šířky
  - úplné – najde řešení, pokud existuje
  - optimální
- Neinformované do hloubky
  - úplné jenom v konečných grafech
  - neoptimální
- Informované A*
  - používá heuristiku pro vedení prohledávání, a tím ho zrychluje
  - úplné, optimální
  - můžeme použít A* pro robotický manipulátor?

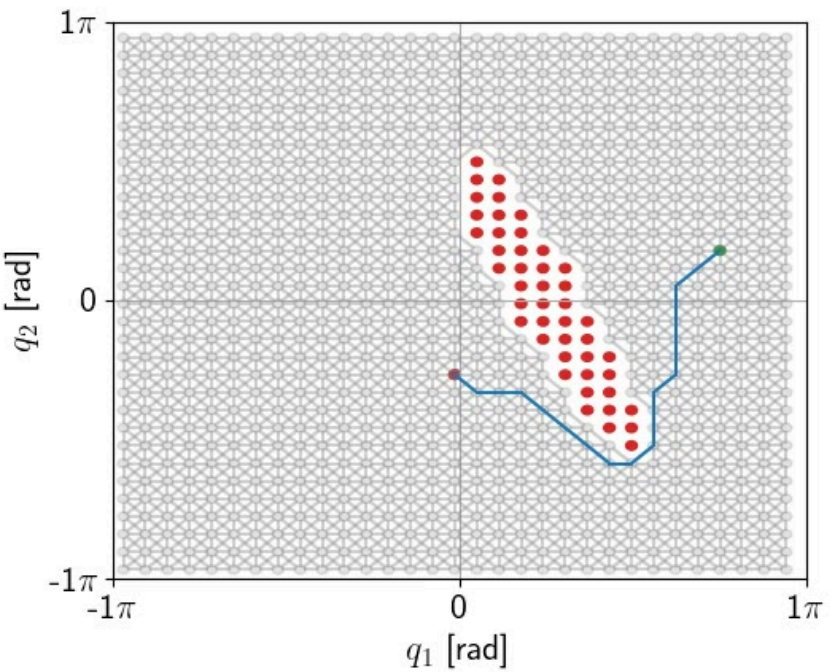
A* pro 2D manipulátor





A* pro 2D manipulátor



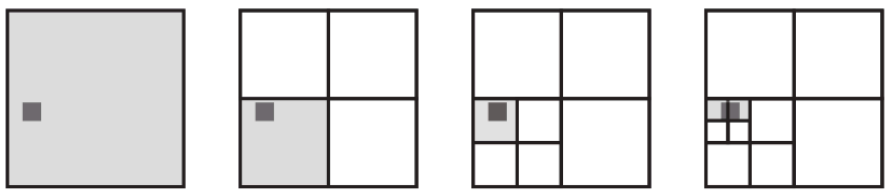


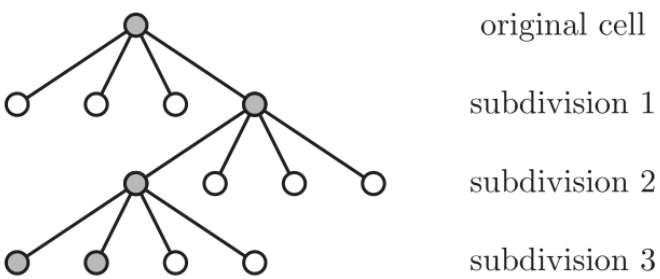
Jak velká mřížka je potřeba?

- Příklad mobilního robota: $20^2 = 400$
- Příklad manipulátoru: $30^2 = 900$
- Komplexnější manipulátor: $30^6 = 729\,000\,000$
- Humanoid: $30^{20} = 3,5 \cdot 10^{29}$
- Úplnost algoritmů závisí na rozlišení mřížky – *resolution complete*

Jak velká mřížka je potřeba?

- Příklad mobilního robota: $20^2 = 400$
- Příklad manipulátoru: $30^2 = 900$
- 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 rozlišením (obrázek z Lynch & Park)
- Diskretizovat vysoce dimenzionální prostor ale není snadné ani efektivní – používají se jiné alternativy plánování





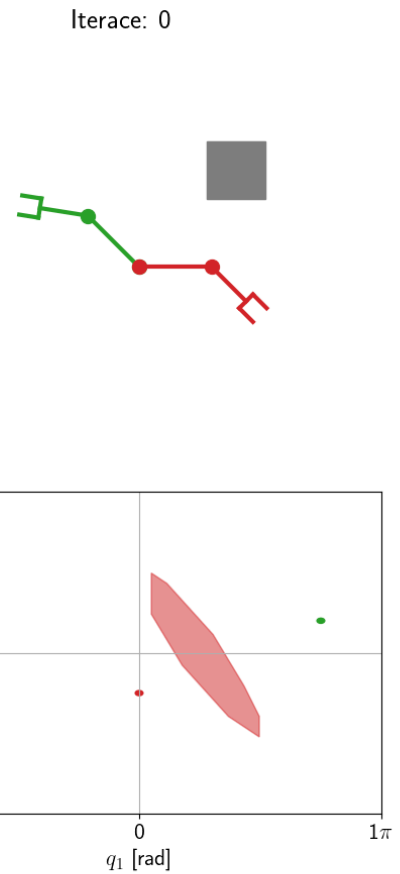
Plánování cesty v manipulátorové robotice

Plánovače #2: metody náhodného vzorkování

- Založeny na
  - náhodném vybírání vzorků z $\mathcal{C}$
  - funkci, která určí, jestli je vzorek bezkolizní
  - lokálním plánovači, 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)

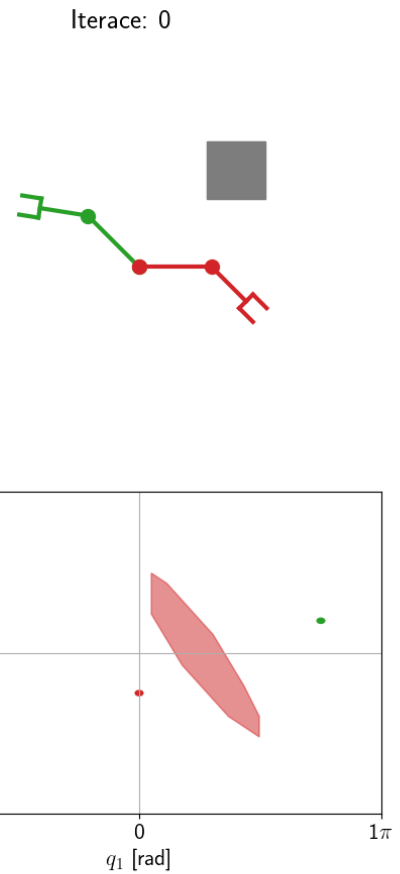
Náhodné stromy – naivní přístup

1: Vstup: start q_s, cíl q_g, maximální délka hrany Δq



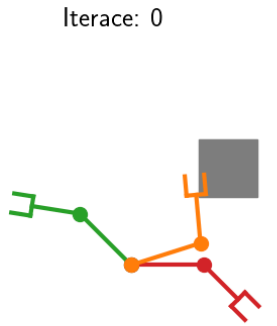
Náhodné stromy – naivní přístup

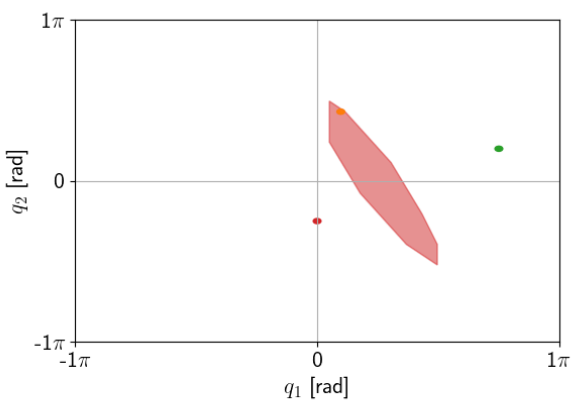
1: Vstup: start q_s, cíl q_g, maximální délka hrany Δq
2: $\mathcal{V} \leftarrow \{q_s\}, \quad \mathcal{E} \leftarrow \{\}$



Náhodné stromy – naivní přístup

1: Vstup: start q_s, cíl q_g, maximální délka hrany Δq
2: $\mathcal{V} \leftarrow \{q_s\}, \quad \mathcal{E} \leftarrow \{\}$
3: **for** $k = 1$ **to** M **do**
4: $\quad q_{\text{rand}} \sim \mathcal{C}$

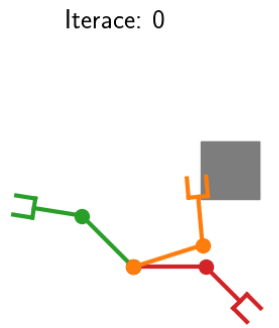


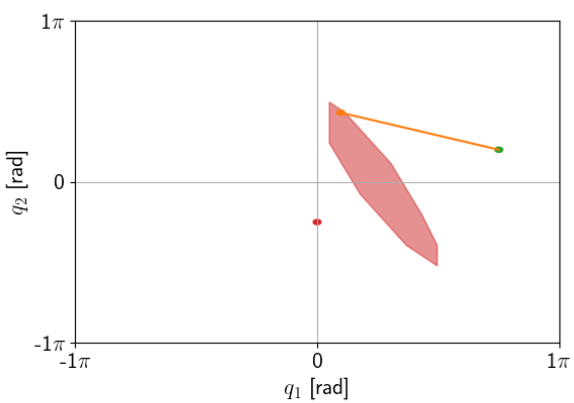


14: **end for**

Náhodné stromy – naivní přístup

1: Vstup: start q_s, cíl q_g, maximální délka hrany Δq
2: $\mathcal{V} \leftarrow \{q_s\}, \quad \mathcal{E} \leftarrow \{\}$
3: **for** $k = 1$ **to** M **do**
4: $\quad q_{\text{rand}} \sim \mathcal{C}$
5: $\quad q_{\text{tree}} \sim \mathcal{V}$

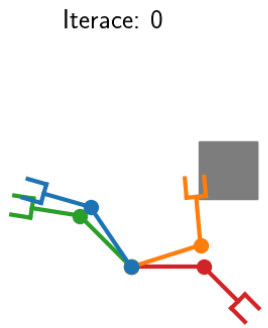


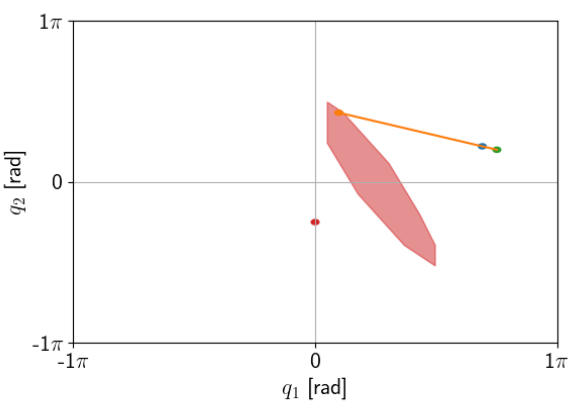


14: **end for**

Náhodné stromy – naivní přístup

1: Vstup: start q_s, cíl q_g, maximální délka hrany Δq
2: $\mathcal{V} \leftarrow \{q_s\}, \quad \mathcal{E} \leftarrow \{\}$
3: **for** $k = 1$ **to** M **do**
4: $\quad q_{\text{rand}} \sim \mathcal{C}$
5: $\quad q_{\text{tree}} \sim \mathcal{V}$
6: $\quad q_{\text{new}} \leftarrow f_{\text{plan}}(q_{\text{tree}}, q_{\text{rand}}, \Delta q)$



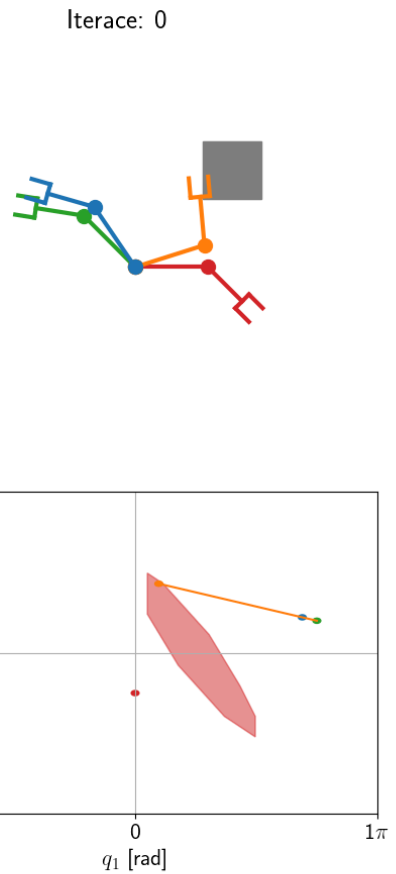


14: **end for**

Náhodné stromy – naivní přístup

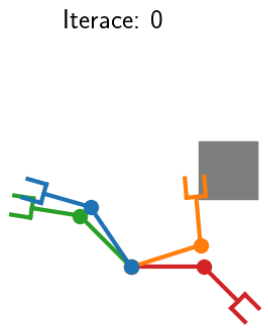
1: Vstup: start q_s, cíl q_g, maximální délka hrany Δq
2: $\mathcal{V} \leftarrow \{q_s\}, \quad \mathcal{E} \leftarrow \{\}$
3: **for** $k = 1$ **to** M **do**
4: $\quad q_{\text{rand}} \sim \mathcal{C}$
5: $\quad q_{\text{tree}} \sim \mathcal{V}$
6: $\quad q_{\text{new}} \leftarrow f_{\text{plan}}(q_{\text{tree}},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

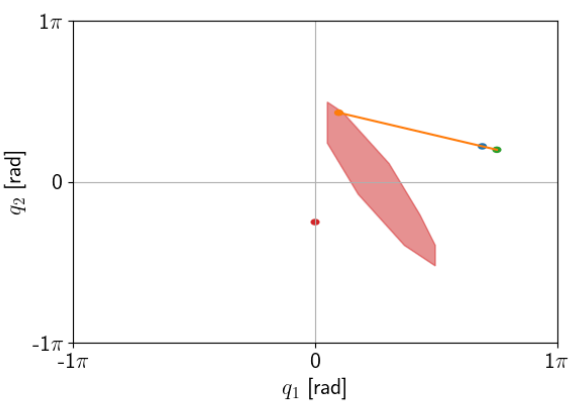
14: **end for**



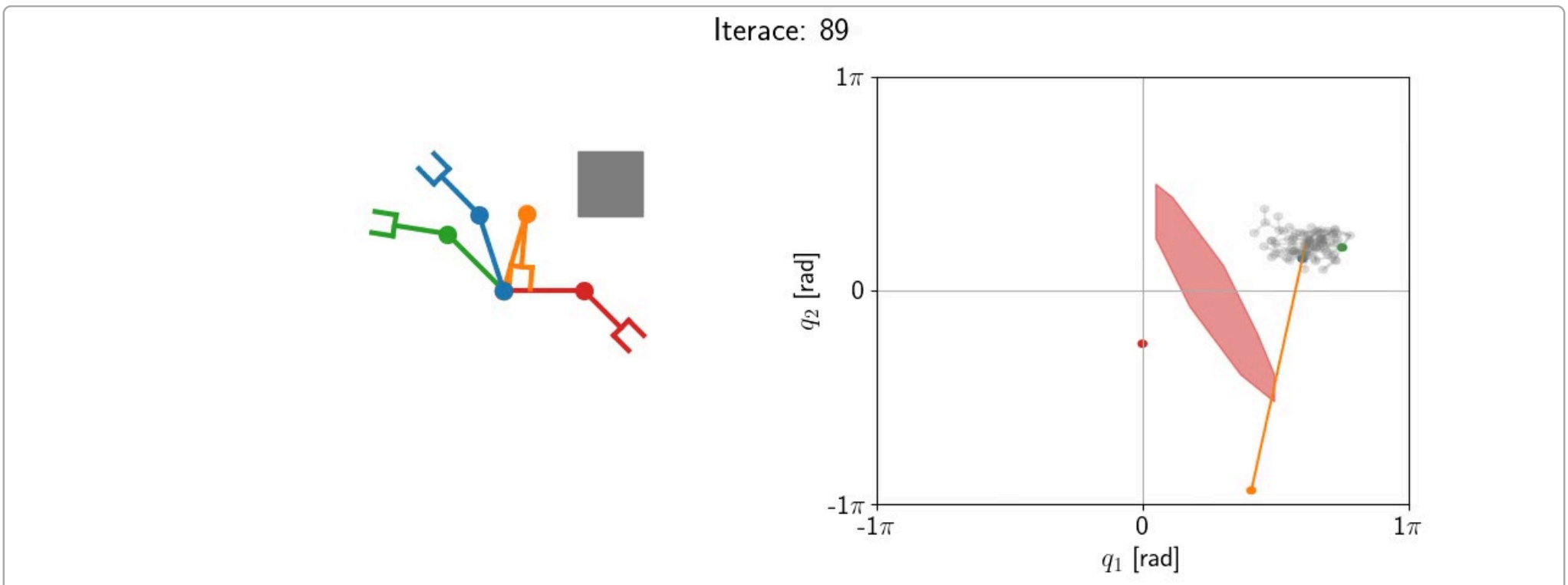
Náhodné stromy – naivní přístup

Vstup: start q_s, cíl q_g, maximální délka hrany Δq
$\mathcal{V} \leftarrow \{q_s\}, \quad \mathcal{E} \leftarrow \{\}$
for $k = 1$ **to** M **do**
    $q_{\text{rand}} \sim \mathcal{C}$
    $q_{\text{tree}} \sim \mathcal{V}$
    $q_{\text{new}} \leftarrow f_{\text{plan}}(q_{\text{tree}}, q_{\text{rand}}, \Delta q)$
    if $q_{\text{new}} \in \mathcal{C}_{\text{free}}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{q_{\text{new}}\}, \quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(q_{\text{tree}}, q_{\text{new}})\}$
    end if
    if $\|q_{\text{new}} - q_g\| < \varepsilon$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{q_g\}, \quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(q_{\text{new}}, q_g)\}$
        break
    end if
end for





Náhodné stromy



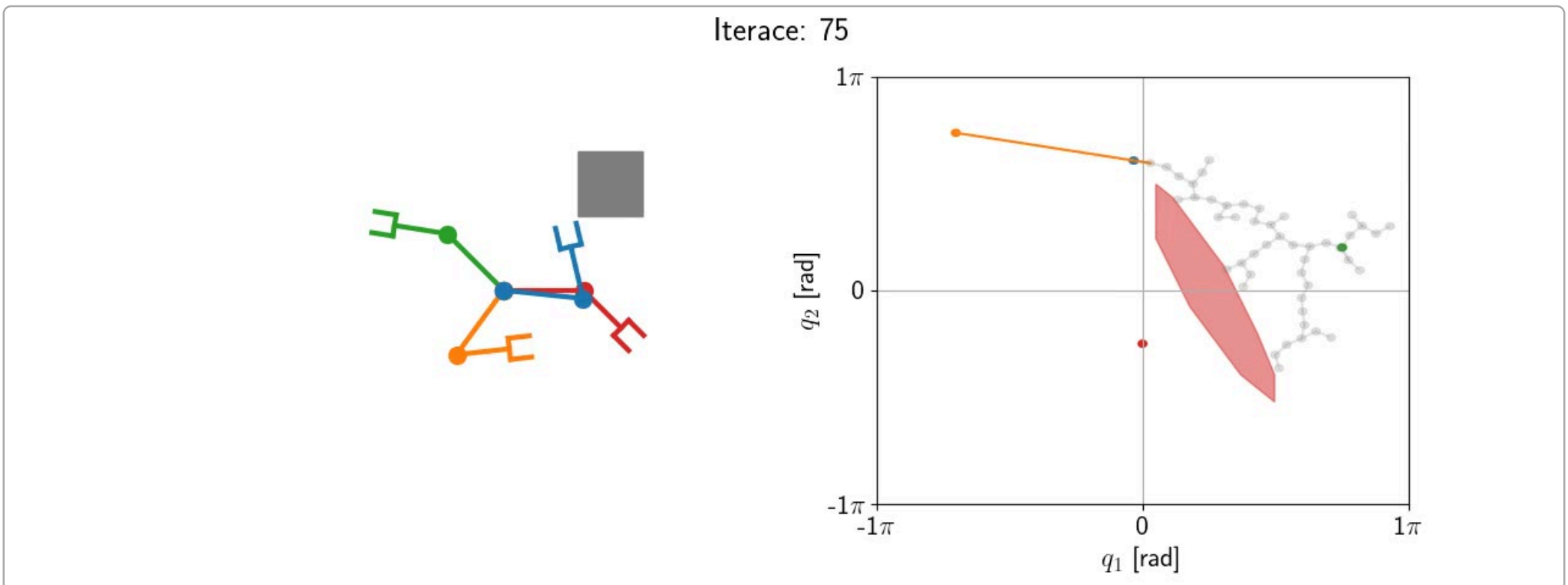
Rychle rostoucí stromy (RRT)

- Náhodné vybírání vrcholu stromu nepodporuje exploraci
- RRT (rapidly-exploring random tree) navrženy pro exploraci
- Pro výpočet q_{new} je náhodně vybraná konfigurace „spojena“ s nejbližším vrcholem stromu

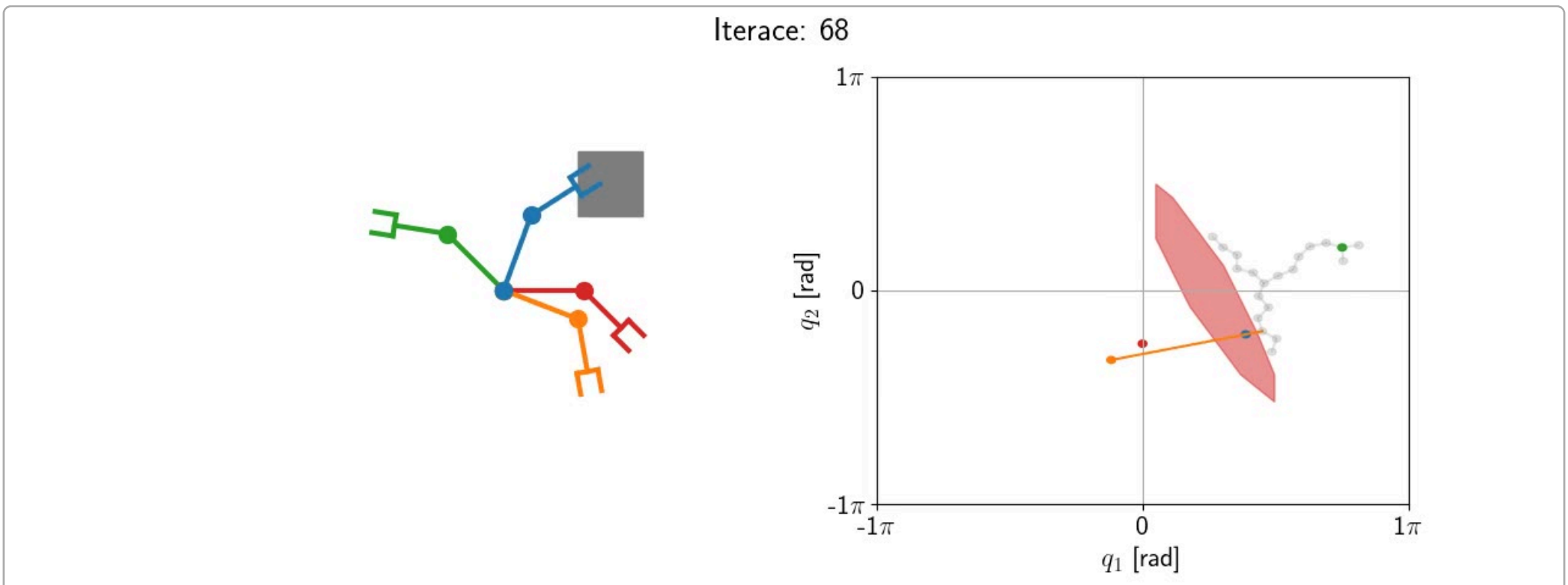
$$q_{\text{tree}} = \arg\min_{q \in \mathcal{V}} \|q - q_{\text{rand}}\| \qquad \cancel{q_{\text{tree}} \sim \mathcal{V}}$$

LaValle and Kuffner, *Randomized Kinodynamic Planning*, IJRR 2001

RRT



RRT – vzorkovací bias



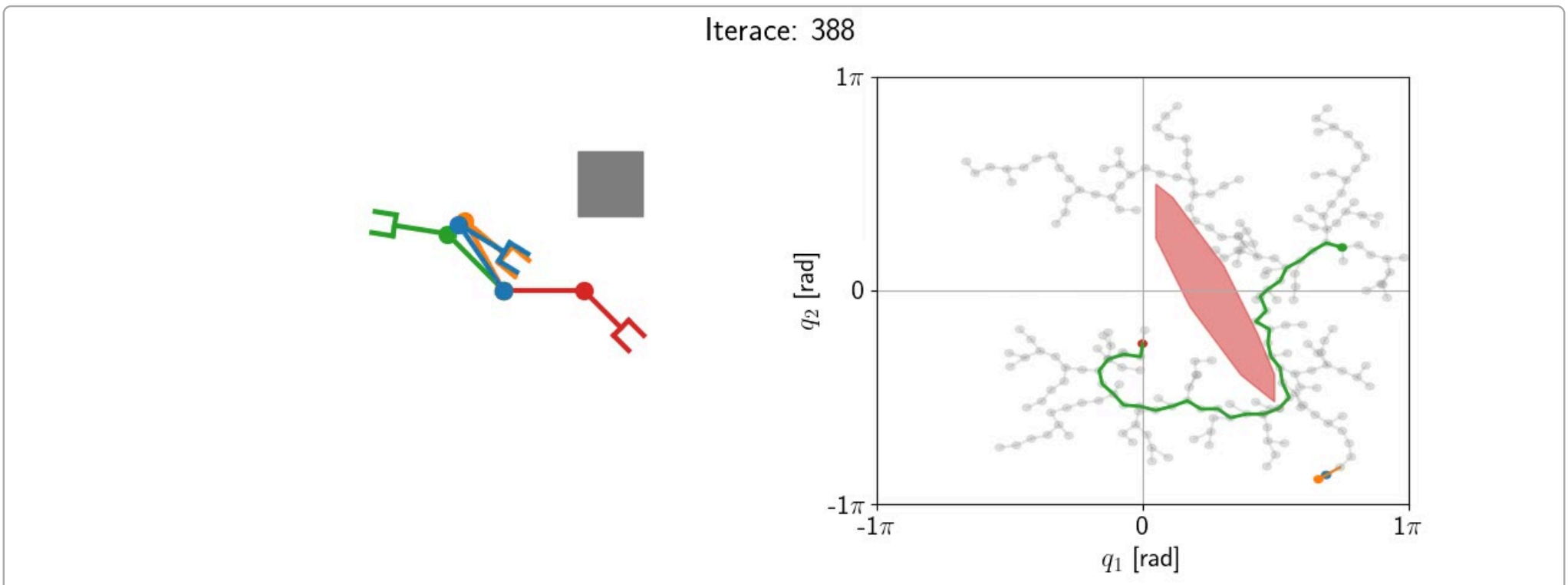
RRT

- Výsledná cesta není optimální – často nám ale stačí alespoň nějaké řešení
- RRT je efektivní i pro velké dimenze
- Vhodným vzorkováním lze zvýšit efektivitu prohledávání (304 vs 257 iterací)
- 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*

RRT*

- 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

RRT*

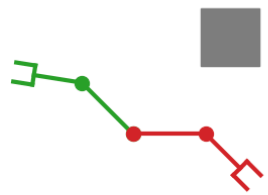


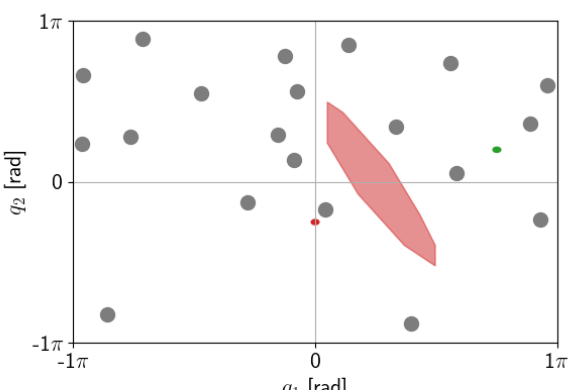
Pravděpodobnostní cestovní mapy – PRM

- Vytvoří neorientovaný graf pro pokrytí $\mathcal{C}_{\text{free}}$ 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

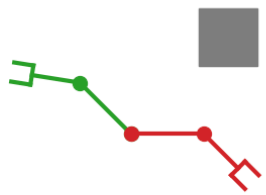
1: $\mathcal{V} \leftarrow \{q_i \mid q_i \sim \mathcal{C}_{\text{free}}\}_{i=0}^{M}$

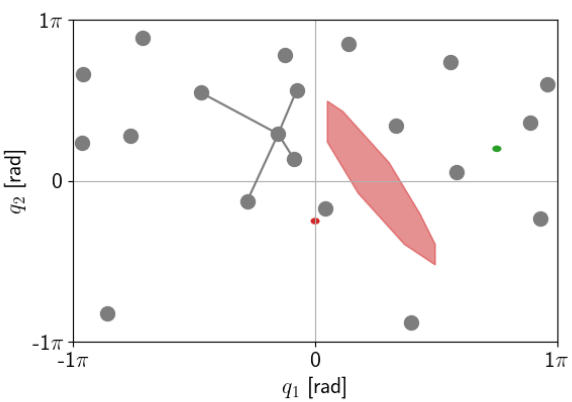




PRM – vytvoření grafu

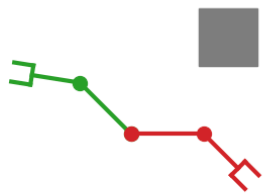
1: $\mathcal{V} \leftarrow \{q_i \mid q_i \sim \mathcal{C}_{\text{free}}\}_{i=0}^{M}$
2: $\mathcal{E} \leftarrow \{\}$
3: **for** $q_i \in \mathcal{V}$ **do**
4: $\quad \mathcal{N}(q_i) \leftarrow k$ nejbližších sousedů vrcholu q_i
5: $\quad$ **for** $q_j \in \mathcal{N}(q_i)$ **do**
6: $\quad\quad$ **if** $\text{plan}(q_i, q_j) \in \mathcal{C}_{\text{free}} \wedge (q_i, q_j) \notin \mathcal{E}$ **then**
7: $\quad\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(q_i, q_j)\}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**

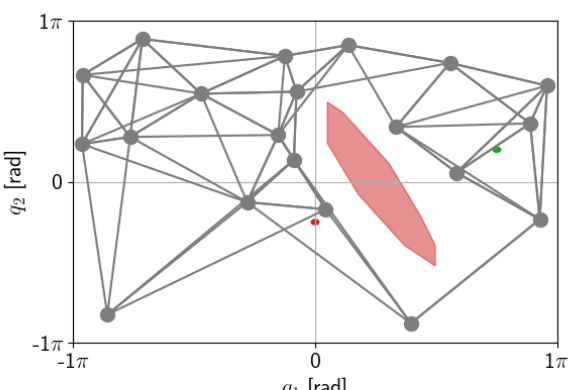




PRM – vytvoření grafu

1: $\mathcal{V} \leftarrow \{q_i \mid q_i \sim \mathcal{C}_{\text{free}}\}_{i=0}^{M}$
2: $\mathcal{E} \leftarrow \{\}$
3: **for** $q_i \in \mathcal{V}$ **do**
4: $\quad \mathcal{N}(q_i) \leftarrow k$ nejbližších sousedů vrcholu q_i
5: $\quad$ **for** $q_j \in \mathcal{N}(q_i)$ **do**
6: $\quad\quad$ **if** $\text{plan}(q_i, q_j) \in \mathcal{C}_{\text{free}} \wedge (q_i, q_j) \notin \mathcal{E}$ **then**
7: $\quad\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(q_i, q_j)\}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**

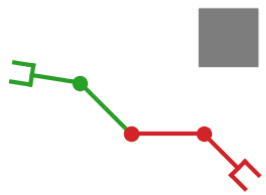


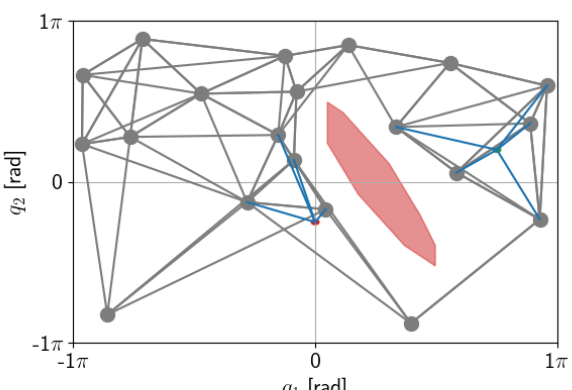


PRM – vytvoření grafu

1: $\mathcal{V} \leftarrow \{q_i \mid q_i \sim \mathcal{C}_{\text{free}}\}_{i=0}^{M}$
2: $\mathcal{E} \leftarrow \{\}$
3: **for** $q_i \in \mathcal{V}$ **do**
4: $\quad \mathcal{N}(q_i) \leftarrow k$ nejbližších sousedů vrcholu q_i
5: $\quad$ **for** $q_j \in \mathcal{N}(q_i)$ **do**
6: $\quad\quad$ **if** $\text{plan}(q_i, q_j) \in \mathcal{C}_{\text{free}} \wedge (q_i, q_j) \notin \mathcal{E}$ **then**
7: $\quad\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(q_i, q_j)\}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**

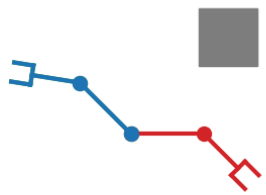
- Při plánování se do grafu vloží q_s a q_g

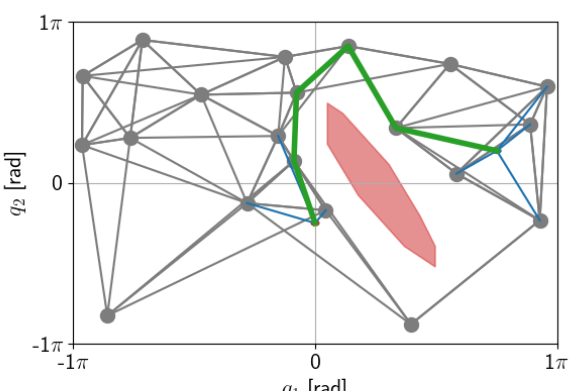




PRM – vytvoření grafu

1: $\mathcal{V} \leftarrow \{q_i \mid q_i \sim \mathcal{C}_{\text{free}}\}_{i=0}^{M}$
2: $\mathcal{E} \leftarrow \{\}$
3: **for** $q_i \in \mathcal{V}$ **do**
4: $\quad \mathcal{N}(q_i) \leftarrow k$ nejbližších sousedů vrcholu q_i
5: $\quad$ **for** $q_j \in \mathcal{N}(q_i)$ **do**
6: $\quad\quad$ **if** $\text{plan}(q_i, q_j) \in \mathcal{C}_{\text{free}} \wedge (q_i, q_j) \notin \mathcal{E}$ **then**
7: $\quad\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(q_i, q_j)\}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**

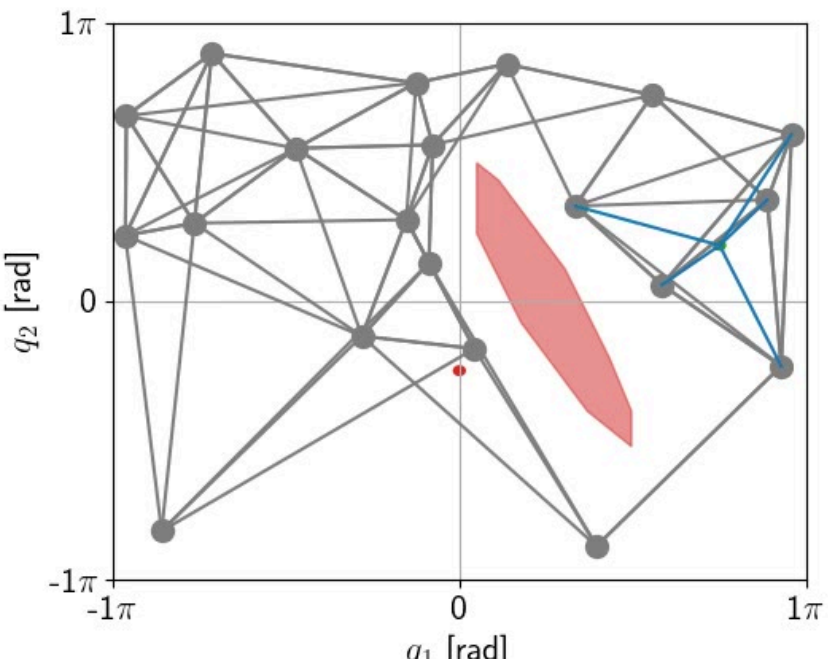




- Při plánování se do grafu vloží q_s a q_g
- Pomocí prohledávání grafu se nalezne cesta

PRM





Jak by plánování vypadalo pro reálný manipulátor?

Plánování 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)

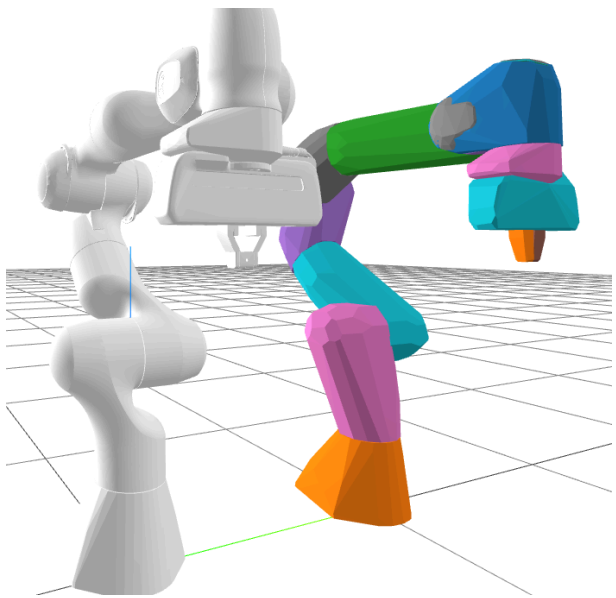


Výpočet kolize

- Hledáme polohu a orientaci každého ramena
  - výpočet přímé kinematické úlohy
  - počítáme kolizi každého ramena zvlášť

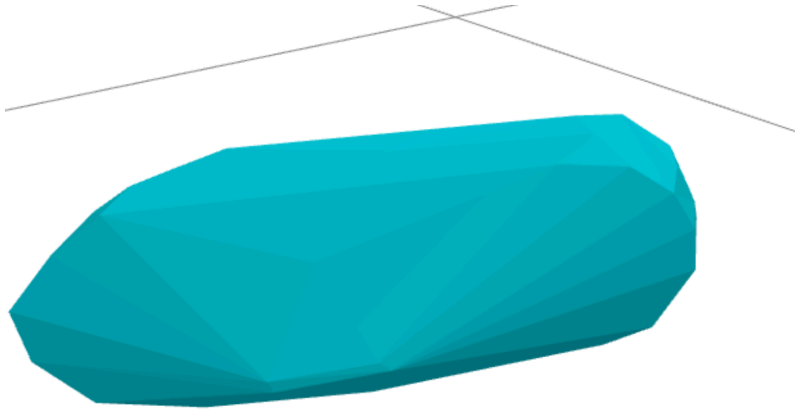
Výpočet kolize

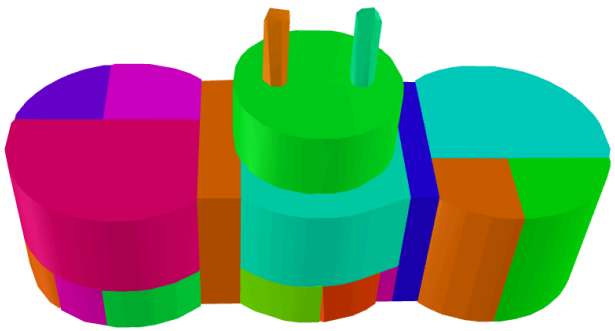
- Hledáme polohu a orientaci každého ramena
  - výpočet přímé kinematické úlohy
  - počítáme kolizi každého ramena zvlášť
- 3D modely bývají velmi detailní (a nekonvexní)
  - pro urychlení výpočtu se spočítá konvexní aproximace
  - konvexní obálka nebo konvexní dekompozice
  - konvexní dekompozice se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



Ukázka konvexní dekompozice objektů

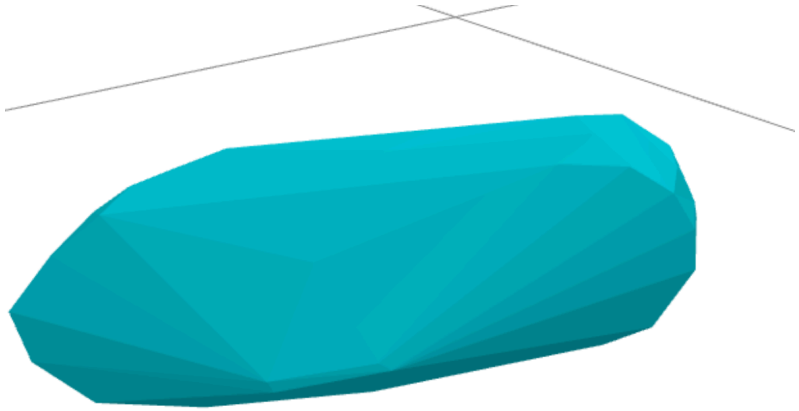


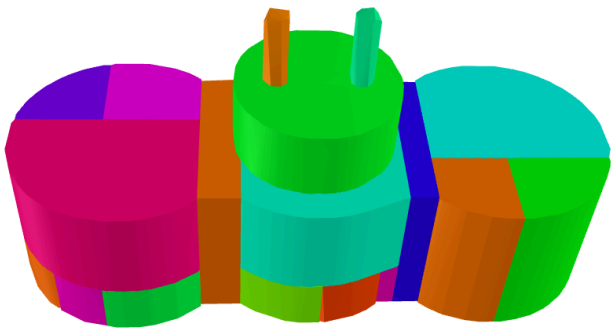




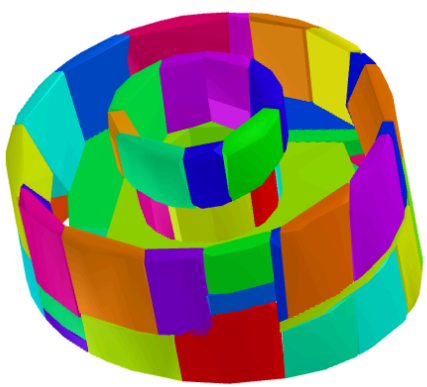
Ukázka konvexní dekompozice objektů

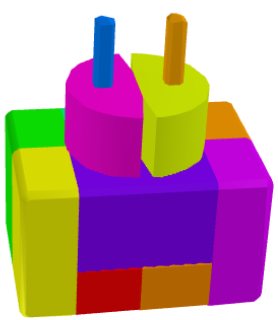












Kolize mezi rameny



- Jsou ramena na obrázku v kolizi?

Kolize mezi rameny

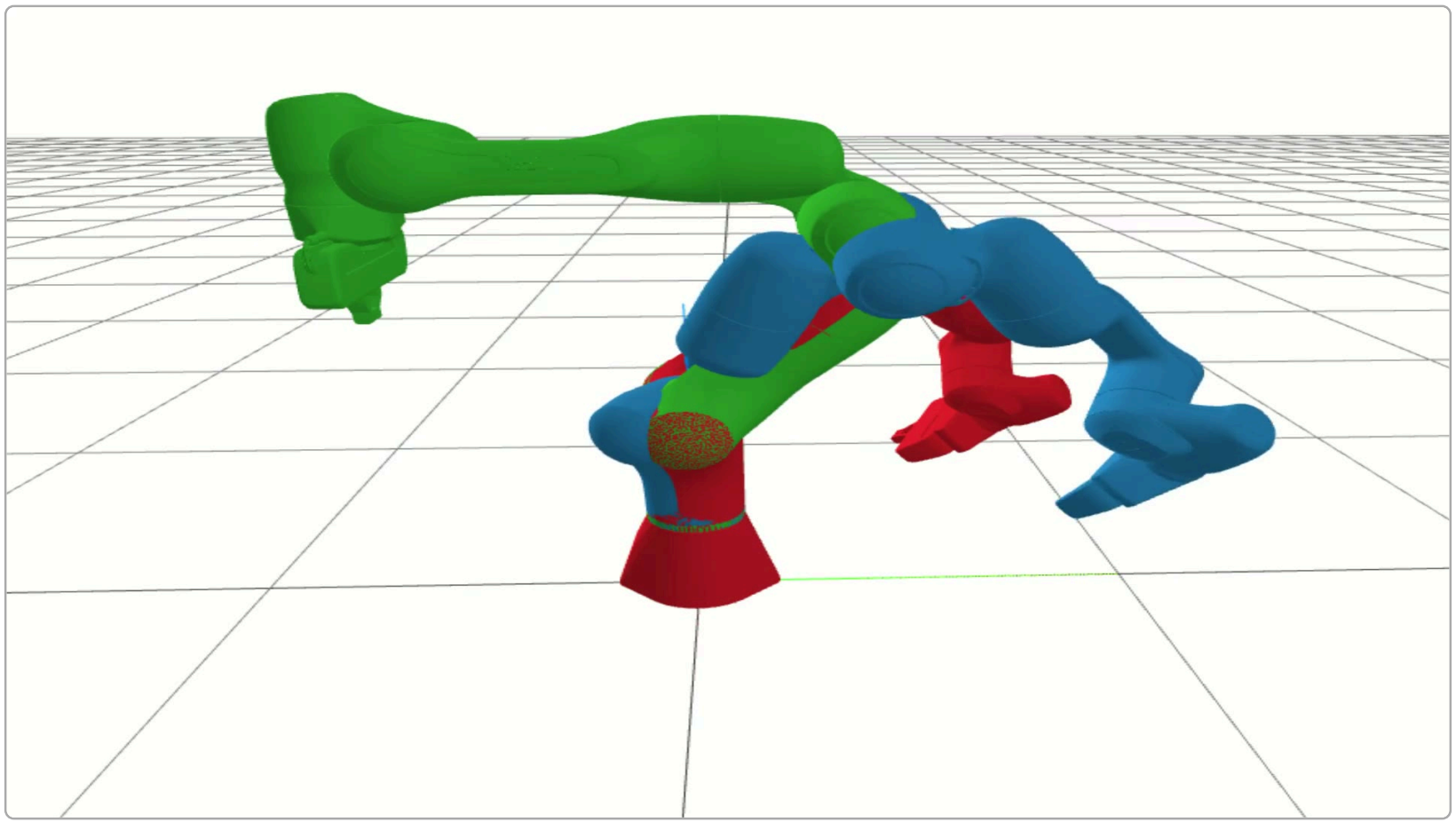


- Jsou ramena na obrázku v kolizi?
- Výpočet kolize je časově náročný (zvlášť pro objekty, které jsou téměř v kontaktu)
- Snažíme se počet kolizních párů zredukovat
- V praxi vypínáme kolize mezi objekty, které:
  - jsou pořád v kolizi, např. sousední ramena
  - nemohou být nikdy v kolizi, např. oddělené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

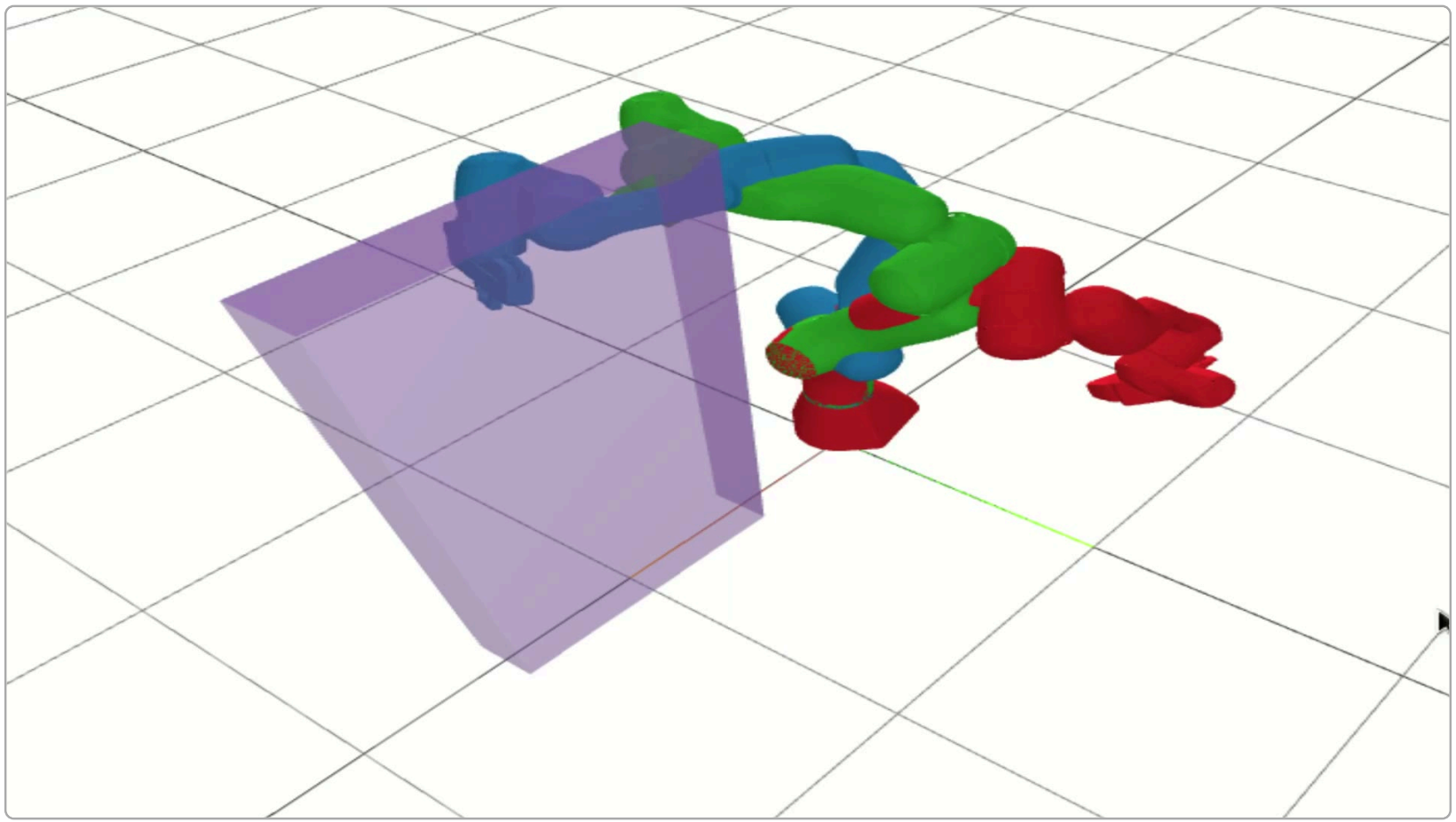
Detekce kolizí pro robota – závěr

- 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

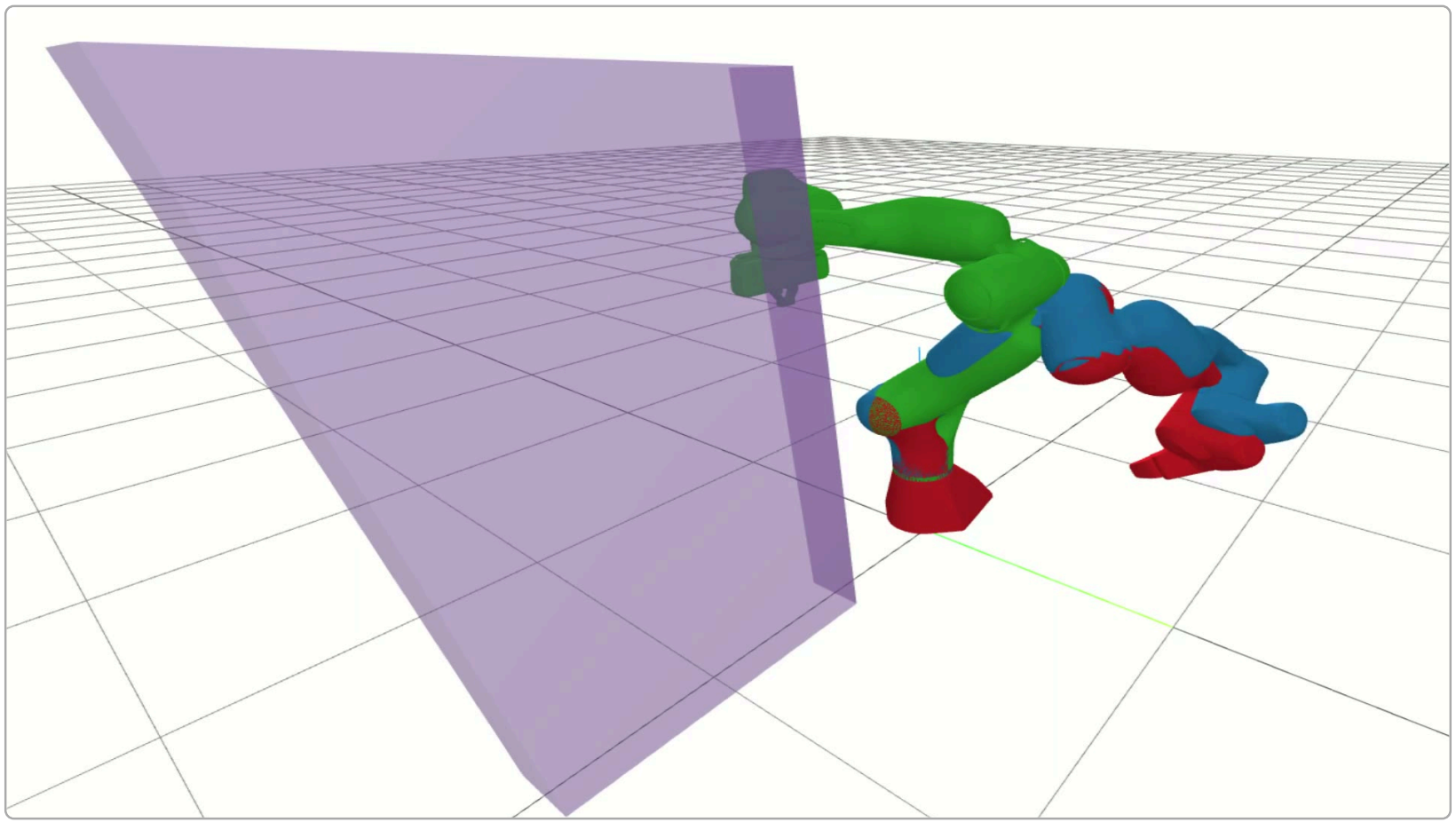
Naplánovaný pohyb



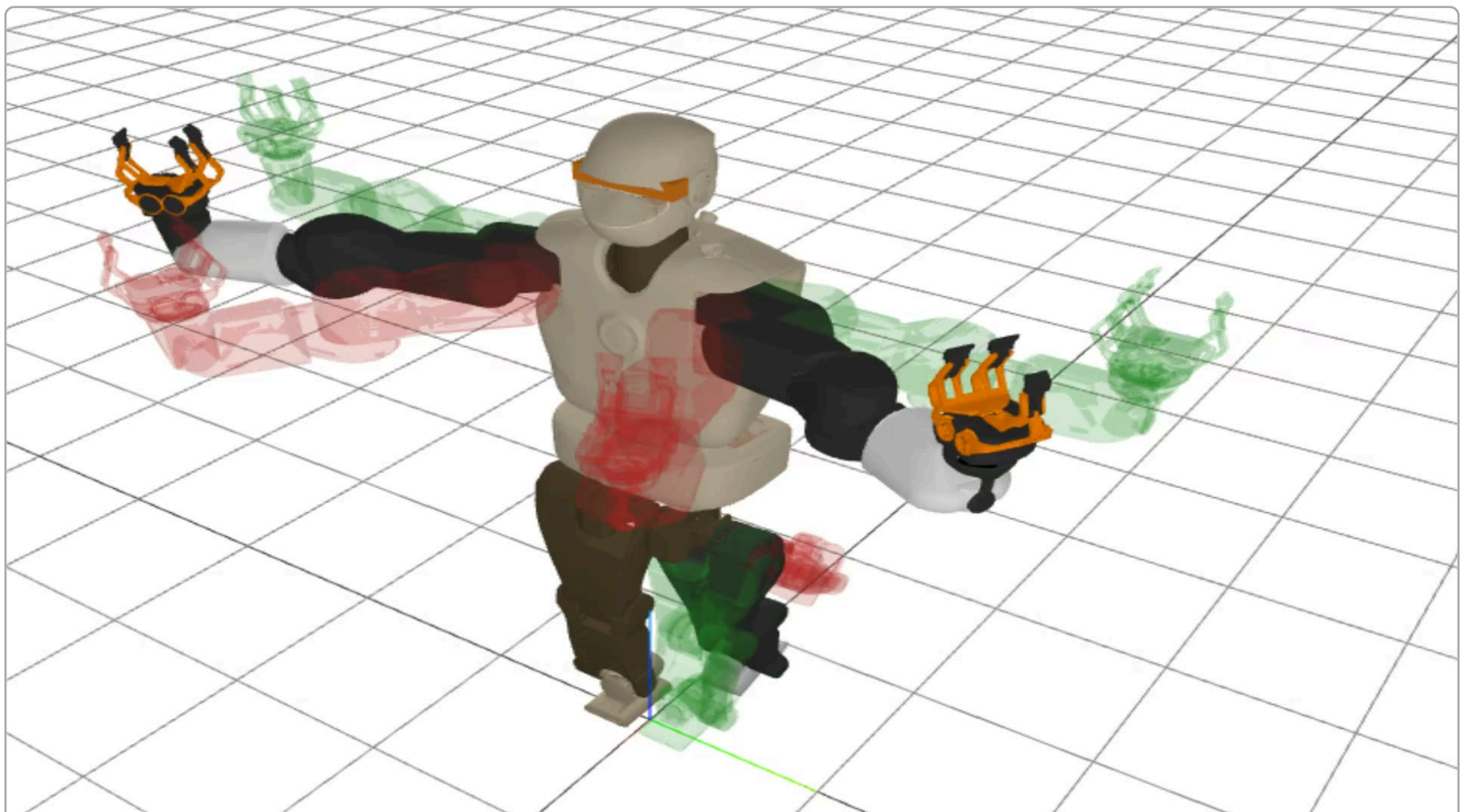
Naplánovaný pohyb RRT – 3132 iterací



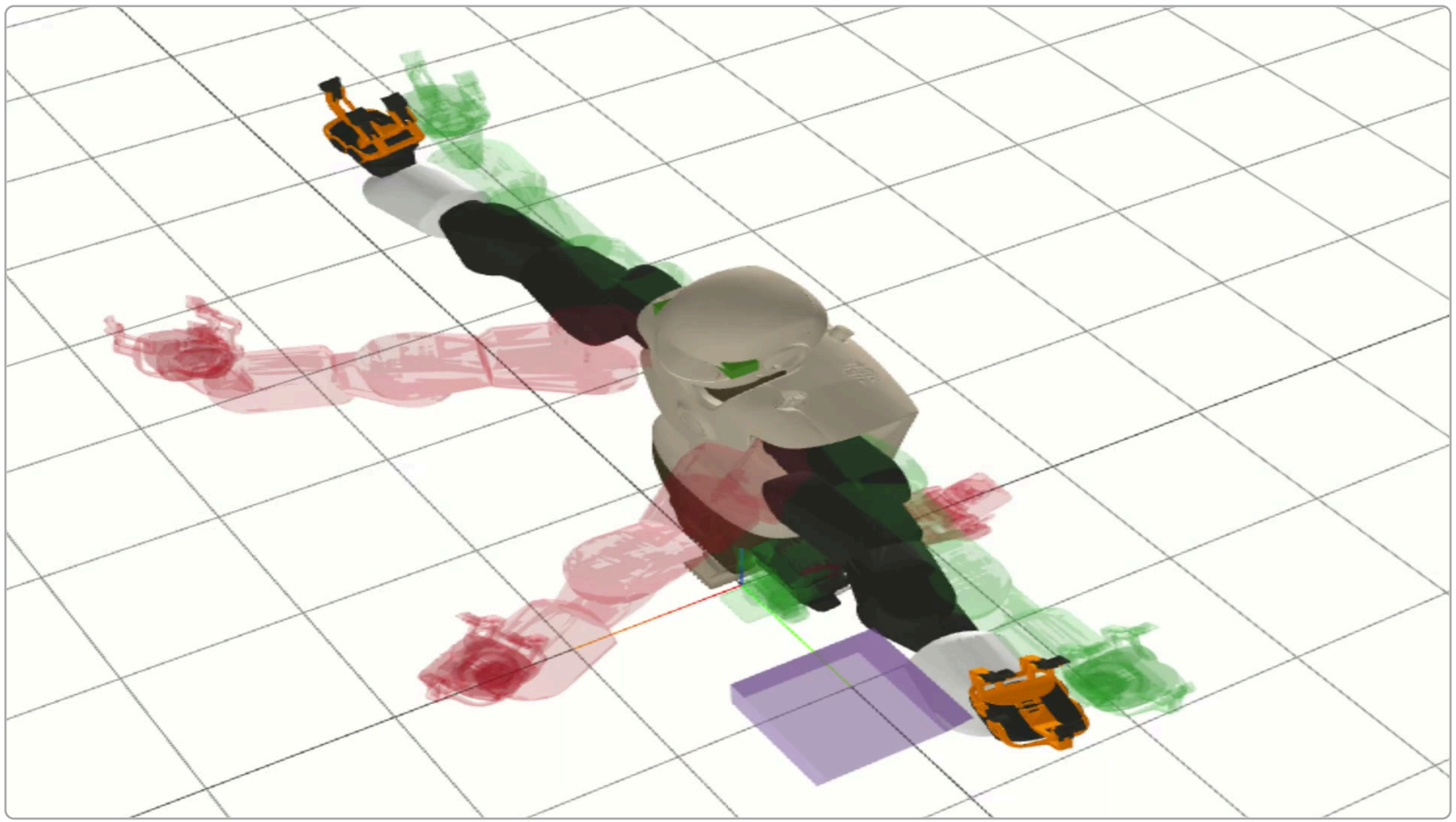
Naplánovaný pohyb RRT* – 10000 iterací



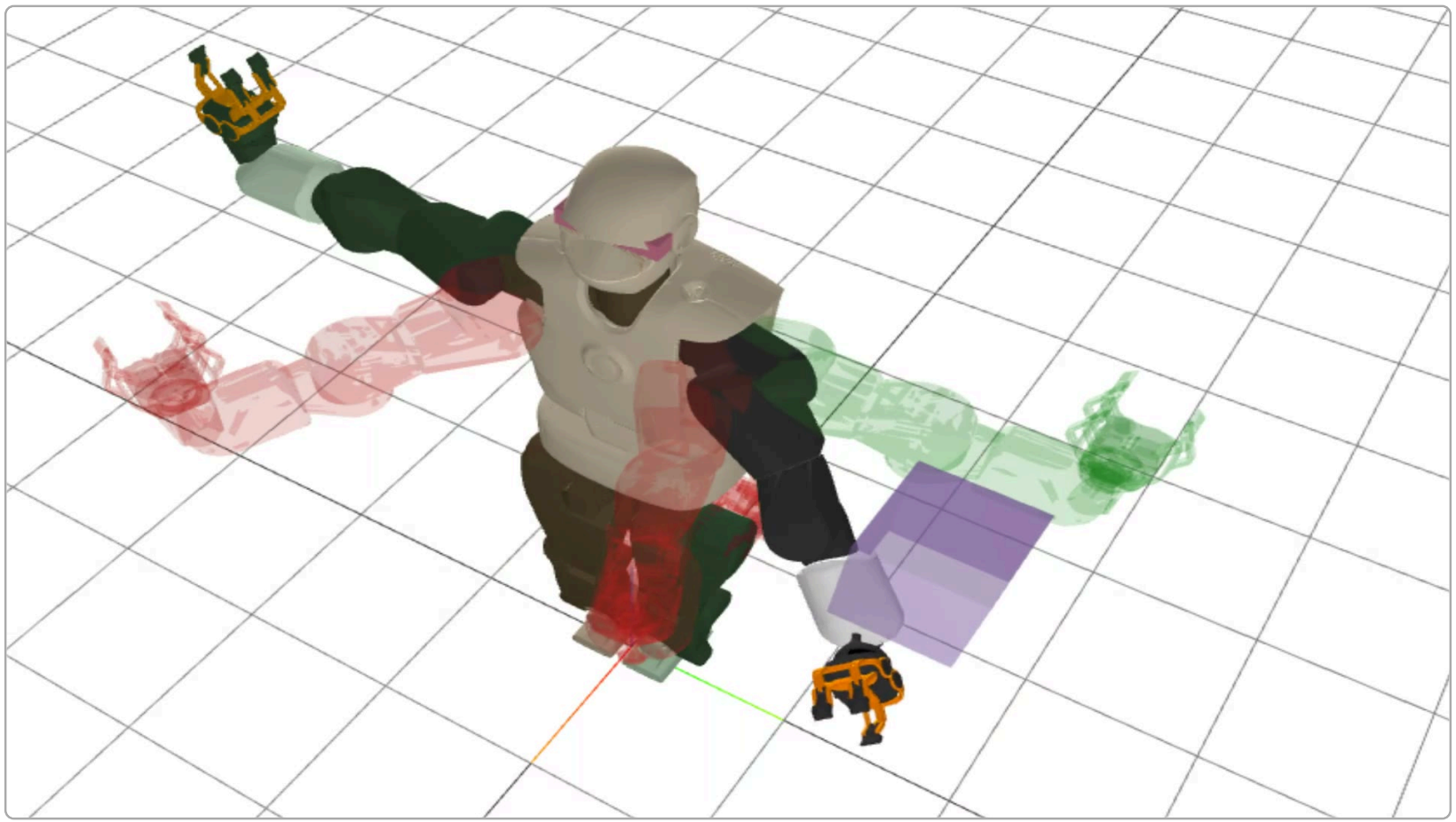
Naplánovaný pohyb Talos



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

Cvičení

- Tento týden (10.–11. 11.): inverzní kinematika, hw05 a opt_hw05 odevzdat do 22. 11.
  - numerická IK v $SE(2)$
  - analytická IK v $SE(2)$ pro manipulátor RRR
  - analytická IK v $SE(2)$ pro manipulátor PRR (dobrovolné)
- Příští týden (17.–18. 11.): individuální práce na semestrálním projektu, konzultace
- Za dva týdny (24.–25. 11.): plánování pohybu, opt_hw06 odevzdat do 6. 12.
  - implementace PRM
  - domácí úkol: implementace RRT a random shortcut (dobrovolné)