Plánovanie hladkých a bezpečných pohybov ramena pomocou učenia posilňovaním

Autor: Bc. Tomáš Janeta
Školiteľ: prof. Ing. Igor Farkaš, Dr.
Oponent: Ing. Michal Dobiš, PhD.

8. júna 2023

Úvod

- Problém hľadania cesty pre robotického agenta je dobre známy a existuje preň množstvo algoritmov
- Učenie posilňovaním - oblasť strojového učenia, skúmajúca algoritmy na hľadanie optimálnych stratégií pre agenta v neznámom prostredí.
- V práci sa zaoberáme problémom hľadania cesty pre robotickú ruku a skúmame možnosti využitia techník učenia s posilňovaním pri tomto probléme.

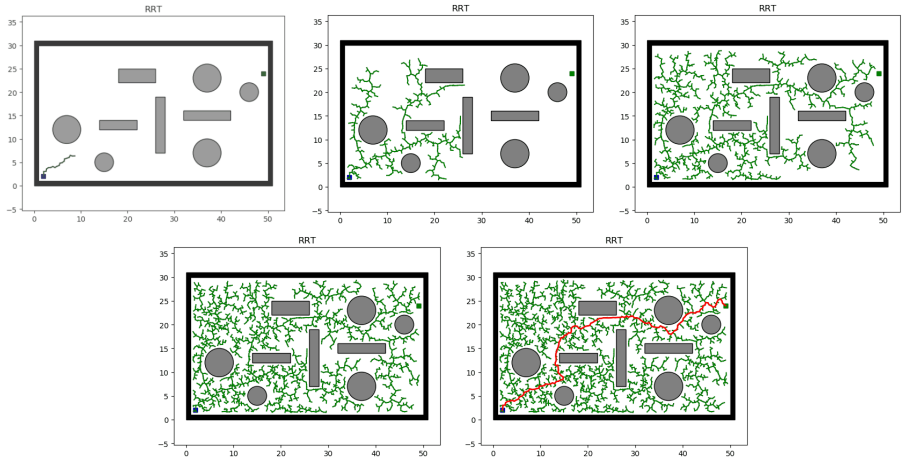
Problém hľadania cesty (path planning)

- Vstupom je počiatočná konfigurácia robota a poloha cieľa
- Výstupom je postupnosť krokov tvoriacich cestu
- Klasické algoritmy sú založené na generovaní náhodných grafov a pri veľkom počte stupňov voľnosti sa stávajú neefektívnymi
- Požiadavky na nájdenú cestu sú bezpečnosť a efektívnosť

Rapid-exploring random trees (RRT)

- Algoritmus[LaValle, 1998] na hľadanie cesty, založený na generovaní stromovej štruktúry, ktorej vrcholy sú náhodné vzorky z priestoru konfigurácií
- Na začiatku sa v algoritme nachádza iba počiatočná konfigurácia
- Algoritmus v každom kroku pomocou heuristiky vygeneruje novú konfiguráciu, a spojí ju s najbližšou konfiguráciou do ktorej je možné sa dostať bez kolízie
- Tento krok sa opakuje až kým nie je dosiahnutý cieľ, alebo nie je prekročený maximálny počet krokov

Rapid-exploring random trees (RRT)



Obr.: Vizualizácia 2D verzie RRT algoritmu. [Garg, 2023]

Učenie posilňovaním

- Vstupom učiaceho algoritmu je množina stavov a množina akcií agenta
- Výstupom je naučená stratégia reprezentovaná funkciou, ktorá vyjadruje optimálnu akciu pre konkrétny stav
- Učenie prebieha v krokoch, pričom v kroku j sa agent nachádza v stave s, následne vykoná akciu a, a obdrží nový stav s' a príslušnú odmenu $R(s, a, s')$, kde R je funkcia odmeny

Učenie posilňovaním

- Cieľom je maximalizovať súčet odmien v dlhodobom horizonte

$$\sum_{i=0}^{T} \gamma^i R_i, \gamma \in (0, 1)$$

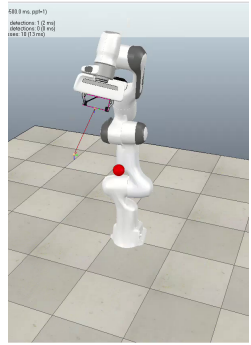
Implementácia

- Robotický simulátor CoppeliaSim[Rohmer et al., 2013]
- Robot Franka Emika Panda[Gaz et al., 2019]

Obr.: Panda

Implementácia - situácia bez prekážok

- Stav je reprezentovaný polohou cieľa a polohami jednotlivých kĺbov
- Akcia je reprezentovaná vektorom uhlových rýchlostí, ktoré budú nastavené jednotlivým kĺbom počas najbližšieho kroku
- Cieľ - naučiť agenta nájsť čo najefektívnejšiu cestu zo vstupnej konfigurácie až ku cieľu



Obr.: Panda

Implementácia - situácia bez prekážok

- Implementovali sme niekoľko funkcií odmeny, založených na myšlienke "riedkej odmeny".
- Agenta sme trénovali pomocou 4 algoritmov - DDPG, TD3, PPO, SAC
- Použili sme implementáciu trénovacích algoritmov z knižnice `stable-baselines3`[Raffin et al., 2021]
- Knižnica pre neurónové siete - PyTorch[Paszke et al., 2019]
- Na vyhodnocovanie sme použili 500 ciest ktoré sme našli pomocou vzorkovacích algoritmov RRT a PRM

Výsledky - situácia bez prekážok

- Na vyhodnocovanie efektívnosti májdenej cesty sme definovali tieto metriky:
  ρ_{joint} predstavuje priemerný pomer dĺžok ciest v kĺbovom priestore nájdených vzorkovacími algoritmami a naším modelom
  $\rho_{cartesian}$ predstavuje priemerný pomer dĺžok ciest chápadla v 3D priestore nájdených vzorkovacími algoritmami a naším modelom
  ρ_{find} predstavuje percento prípadov keď sa modelu podarilo nájsť cestu
- Z experimentov vyplýva, že pri maximálnej uhlovej rýchlosti 0.2 sa podarilo úspešne natrénovať modely pomocou algoritmov DDPG, TD3 a SAC
- Trénovanie pomocou PPO algoritmu nebolo úspešné

Výsledky - situácia bez prekážok

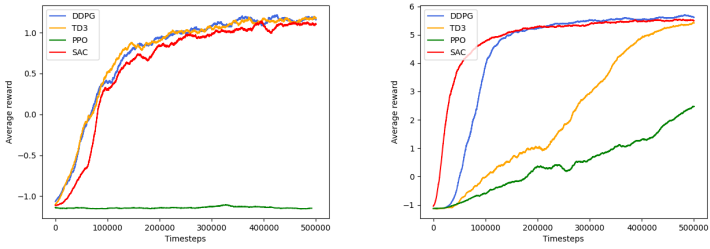
Algorithm	ρ_{joint}	$\rho_{cartesian}$	ρ_{find}
DDPG	0.325	0.742	0.995
TD3	0.366	0.744	1.0
SAC	0.449	0.787	1.0
PPO	0.212	0.659	0.513

Tabuľka: Výsledky trénovania s `max_speed=0.2`

Algorithm	ρ_{joint}	$\rho_{cartesian}$	ρ_{find}
DDPG	0.321	0.491	0.988
TD3	0.216	0.617	1.0
SAC	0.281	0.562	0.994
PPO	0.143	0.462	0.997

Tabuľka: Výsledky trénovania s `max_speed=1.0`

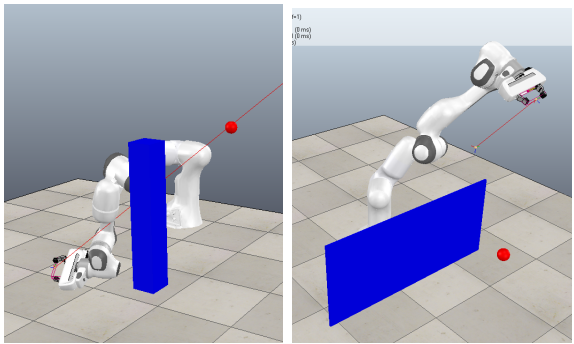
Výsledky - situácia bez prekážok



Obr.: Averaged reward from the training with setting `max_speed=0.2` and `max_speed=1.0`

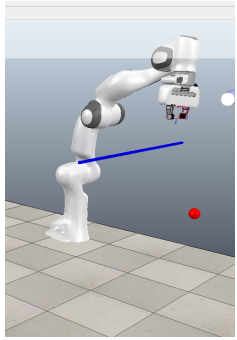
Implementácia - situácia s prekážkami

- Implementovali sme možnosť pridať do simulácie prekážky
- V našich experimentoch sme narábali len s prekážkami tvaru kvádra
- Každá prekážka je reprezentovaná polohou svojho ťažiska a dĺžkami hrán



Obr.: Ukážky statických prekážok

Implementácia - situácia s prekážkami



Obr.: Ukážka premenlivej prekážky

Implementácia - situácia s prekážkami

- Implementovali sme dva typy prekážok
- Statické prekážky - počas celého procesu učenia nemenili (ich reprezentácia nie je zahrnutá v stave agenta)
- Premenlivé prekážky - ktorých poloha a veľkosť sa menia v zadanom rozsahu vždy po skončení epizódy (ich reprezentácia je zahrnutá v stave agenta)

Výsledky - situácia s prekážkami

- Úspešnosť natrénovaného modelu silno závisí od rôznych hyperparametrov, ako je veľkosť, vzialenosť od robota a podobne
- Je ťažké nájsť oblasť do ktorej umiestniť prekážky, tak aby úloha bola stále zaujímavá a zároveň agent dokázal získať dostatok pozitívnej skúsenosti
- Definovali sme metriku na vyhodnocovanie bezpečnosti natrénovaného modelu $\rho_{collide}$ ako priemerný počet kolízií za jednu epizódu

Výsledky - situácia s prekážkami

Algoritmus	experiment	ρ_{find}	$\rho_{collide}$
DDPG	stĺp	0.291	5.837
TD3	stĺp	0.279	5.578
SAC	stĺp	0.016	6.902
DDPG	stena	0.251	14.654
TD3	stena	0.035	14.592
SAC	stena	0.287	12.473

Tabuľka: výsledky experimentov so statickými prekážkami

Algoritmus	ρ_{find}	$\rho_{collide}$
DDPG	0.073	7.795
TD3	0.077	6.435
SAC	0.213	11.192

Tabuľka: výsledky experimentu s premenlivou prekážkou

Výsledky - situácia s prekážkami

- Nepodarilo sa natrénovať spoľahlivý model, ktorý by nachádzal bezpečné cesty

Pripomienky oponenta

- Vysvetlite význam konštanty „c“ označovanej ako hyperparameter v rovnici 4.1 Spomínaná rovnica je $f_{const}(p) = c$. V tejto rovnici, c predstavuje kladnú konštantu ktorá má motivovať agenta nachádzať cieľ
- Vysvetlite ako je vyjadrený vektor „v“ reprezentujúcich stav premenlivých prekážok na neoznačenej rovnici na strane 21 v kapitole 4.1.
  Príslušná rovnica je $\boldsymbol{s}_j = (\boldsymbol{t}, \boldsymbol{\theta}_j, \boldsymbol{v})$, pričom $\boldsymbol{s}_j$ predstavuje stav agenta v kroku j, a $\boldsymbol{t}, \boldsymbol{\theta}_j, \boldsymbol{v}$ sú postupne poloha cieľa, vekto polôh kĺbov a vektor reprezentujúci stav premenlivých prekážok. Stav každej prekážky je reprezentovaný polohou ťažiska a dĺžkami hrán.

Pripomienky oponenta

- Vysvetlite význam rovníc 4.5 a 4.6. Aký vplyv má dráha nájdená algoritmami PRM a RRT ?
  Rovnice 4.5 a 4.6 obsahujú definície metrík $\rho_{joint}, \rho_{cartesian}$. Dráha nájdená algoritmami PRM a RRT slúži ako baseline voči ktorej je náš model porovnávaný.
- Uveďte výpočtový čas RL algoritmov hľadania dráhy a porovnajte voči PRM a RRT. Aký vplyv má zložitosť prostredia ?
  Výpočtový čas týchto dvoch metód nie je možné porovnávať, pretože riešia iný problém. Implementácia PRM a RRT použitá na evaluáciu natrénovaných modelov má ešte jeden parameter, finálnu orientáciu chápadla. Zložitosť prostredia výrazne limitovala schopnosť modelu získať skúsenosť potrebnú na splnenie úlohy.

Pripomienky oponenta

- Aký vplyv má maximálna rýchlosť `max_speed` na algoritmus? Má vplyv aj počas využívania natrénovaného modelu alebo iba počas trénovania ?
  Menšie hodnoty parametra `max_speed` môžu pomôcť algoritmu získať lepšiu skúsenosť a v konečnom dôsledku hľadať efektívnejšie cesty. Zároveň ale príliš malé hodnoty môžu zabrániť získaniu pozitívnej skúsenosti, obzvlášť pri niektorých algoritmoch (PPO).
  Pri používaní natrénovaného modelu bude `max_speed` predstavovať maximálnu absolútnu hodnotu zložiek vektora akcíi generovaného modelom.

Ďakujem za pozornosť

Citácie I

Garg, S. (2023).
Motion planning algorithms for robots.

Gaz, C., Cognetti, M., Oliva, A., Robuffo Giordano, P., and De Luca, A. (2019).
Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization.
IEEE Robotics and Automation Letters, 4(4):4147–4154.

LaValle, S. M. (1998).
Rapidly-exploring random trees : a new tool for path planning.
The annual research report.

Citácie II


Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019).
Pytorch: An imperative style, high-performance deep learning library.


Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021).
Stable-baselines3: Reliable reinforcement learning implementations.
Journal of Machine Learning Research, 22(268):1–8.


Rohmer, E., Singh, S. P. N., and Freese, M. (2013).
Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework.
In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*.