TECHNICKÁ ZPRÁVA

Aplikace algirtmů RRT a A* pro plánování trasy kolaborativního robota UR3

Martin Juříček

Faculty of Mechanical Engineering, Brno University of Technology
Institute of Automation and Computer Science
Technicka 2896/2, Brno 616 69, Czech Republic
200543@vutbr.cz

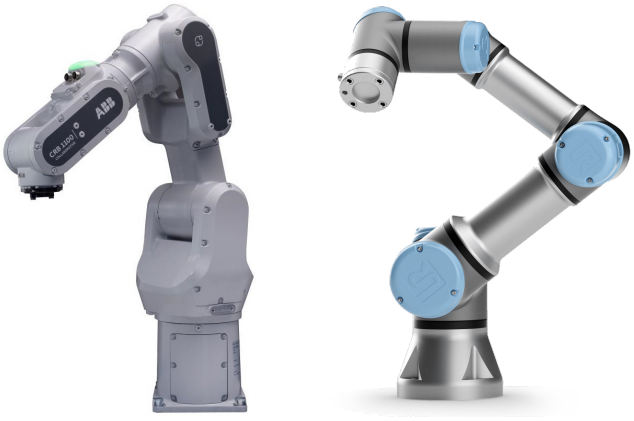
Abstrakt: *Tato technická zpráva pojednává o aplikací algoritmů RRT a A* pro plánování jednoduché trasy kolaborativního robota UR3 s využitím ROS. Také je provedena komparace algoritmů RRT a A*, zakončená diskuzí o možném rozšíření.*

Klíčová slova: *umělá inteligence, RRT, A*, ROS, Universal Robots, UR3*

1 Úvod

Robotizace výrobních stanovišť vzhledem k vypuknutí pandemie onemocnění COVID-19 nabývá stále více na váze, jelikož i za nepřízdnivé pandemické situace mohou stále fungovat klíčová výrobní oddělení. Jednou z možností jak robotizovat je použití konvenčních průmyslových robotických ramen, či případně čím dál tím více oblíbenějších kolobarotivních robotů. Tito kolaborativní roboti jsou primárně navrženi pro spolupráci s člověkem, avšak tato vlastnost je vykoupena za cenu nižší rychlosti, dosahu a nosnosti oproti výhradně průmyslovým robotům. Přesto své popularity nabývájí především díky modularitě, jednoduchého programování a integrace do výrobního prostředí. I přes tento trend firma ABB přišla s novou generací kolaborativních robotů a především kobot SWIFTI svou rychlostí a přesností se začíná blížit parametrům konvenčních robotických ramen, ale uchovává si vlastnosti bezpečného kolaborativního robota. K tomu, aby bylo možné tyto robotické ramena řídit a programovat je do určitých pozic, je možné použít hned několik metod. Jednou z metod je aplikace plánovacích algoritmů, které patří mezi metody umělé inteligence. Spojením plánovacích algoritmů s inverzní kinematikou a dynamykou, při integraci hardwarového kontroléru, lze vytvořit řídící systém, který nabízí v odladěnější verzi takřka všichni výrobci robotů.

Ikonou kolaborativních robotů se stali roboti od firmy Universal Robots. Tato dánská firma se stala jedním z největších globálních distributorů kobotů. Své kolaborativní roboty firma poskytuje do menších, středních i velkých korporátních podniků. Důkazem, že se jedná o ikonu, je dostupnost velkého množství softwarových balíků, které lze integrovat do různých vývojových prostředí, simulačních prostředí, či použití k řešení různých problémů, které lze spatřit v repozitářích na serveru Github.



(a) ABB SWIFTI (b) Universal Robots UR3

Obrázek 1: Kolaborativní roboti

2 Plánovací algoritmy

Své pole působnosti plánovací algoritmy nacházejí nejenom na poli robotiky, ale jsou i například klíčovou součástí při řešení pohybu umělé inteligence v různých počitačových hrách. Je to tedy integrální část, díky které se postavy ovládané počítačem efektivně pohybují směrem k cílům a vyhýbají se překážkám a možným rizikům. Tento princip se dá přenést do robotiky a proto se staly neodmyslitelnou součástí. Existujé celá řada různých algoritmů založených na odlišných principech a přístupech. Mezi jedny z nejznámnějších můžeme řadit i algoritmus A* a RRT.

2.1 A*

První zmínky o algoritmu A* se datují až do roku 1964, kdy Nils Nilsson přišel s heuristickým vylepšením Dijkstrova algoritmu v rámci projektu the Shakey Project. Jednalo se o projekt, kde mobilní robot mohl plánovat své vlastní akce. Následně algoritmus A* se stal jednou z nejoblíbenějších voleb pro hledání cest, jelikož je poměrně flexibilní a lze jej použít v širokém spektru aplikací [1].

A* staví na principech Dijkstrova algoritmu, přičemž s cílem rychlejšího řešení, když čelí problému hledání nejkratší cesty mezi dvěma uzly. Zavedením již zmiňované heuristiky, pomáhá rozhodnout o dalším uzlu, který je třeba vzít v úvahu při jeho pohybu po cestě. Dijkstrův algoritmus nabývá své výhody pokud řešíme nalezení nejkratší cesty mezi počátečním uzlem a všemi ostatními uzly, kdežto algoritmus A* se liší tím, že hledá pouze nejkratší cestu mezi počátečním uzlem a cílovým uzlem [2].

Zjednodušená podstata algoritmu je ta, že v každém kroku se vybere uzel podle hodnoty **f** což je parametr rovný součtu dvou dalších parametrů **g**(náklady na pohyb) a **h**(odhadovaná cena pohybu, často označována jako heuristika). Tedy v každém kroku se vybere uzel s nejnižším **f** a tento uzel se zpracuje, dokud není nalezen koncový bod. Parametr **h**, lze blíže popsat jako odhadované náklady na pohyby, které se mají přesunout z dané polohy do konečného cíle. Parametr **g** je náklad na pohyb pro přesun z počátečního bodu do definované polohy, kdy je sledována cesta generovaná k tomu, aby se tam dostal [3].

2.2 RRT

Algoritmus RRT(Rapidly-exploring Random Tree v českém překladu rychle rostoucí náhodný strom) vyvinul Steven M. LaValle a James Kuffner, za účelem efektivního vyhledávání v nekonvexních vysokodimenzionálních prostorech. Tento algoritmus nabyl velké popularity pro řešení problému plánování pohybu v různých oblastech robotiky. Nedostatky původního algoritmu při řešení odlišných problémů pak odlaďují rozšířené verze algoritmu, či různé modifikace s jinými algoritmy [4].

Ve své nejjednodušší podobě je konstruován postupem, kdy každý náhodný vzorek polohy v prostoru je připojen k nejbližšímu vzorku, který je sou částí stromu. Algoritmus přírustkovým způsobem vytváří stromovou strukturu a velmi rychle prorůstá prostředím. S postupujícími iteracemi se prudce zkracuje očekávaná vzdálenost mezi náhodným vzorkem a nejbližším bodem na stromu [5].

Výhodou algoritmu je jeho možná rychlost a implementace. Ve srovnání s jinými algoritmy plánování cesty je RRT poměrně rychlý. Časově nejnákladnější částí je hledání nejbližšího souseda, čas tohoto procesu roste v závislosti na počtu generovaných vrcholů.

Obecnou nevýhodou algoritmu RRT je, že v reálných aplikacích samotný algoritmus nestačí a je většinou rozšířen například o vyhlazování trajektorie a jiné.

3 Popis programu

Cílem semestrálního projektu bylo naprogramování plánovacích algoritmů A* a RRT pro hledání cesty robotického ramene. Program semestrálního projektu je naprogramován v jazyce python ve verzi 2. Zvolena byla tato verze programovacího jazyka python z důvodu kompatibility s knihovnami z frameworku MoveIT a ROS. Ačkoliv python 2 není nejaktuálnější verzí, je stále používán v ROS distribucích Kinetic Kane a Melodic Morenia, přičemž distribuce Melodic Morenia je stále považována za aktuální a má životnost do roku 2023.

Program byl vyvíjen v hostitelském operačním systému linux Ubuntu 18.04 a byla použita ROS distribuce Melodic Morenia. Tato distribuce je prímarně podporována pro linuxové distribuce Ubuntu 18.04, Debian 9, ale i také pro Windows 10. K naprogramování a testování byly použity knihovny:

- RosPy
- MoveItCommander
- GeometryMsg
- NumPy

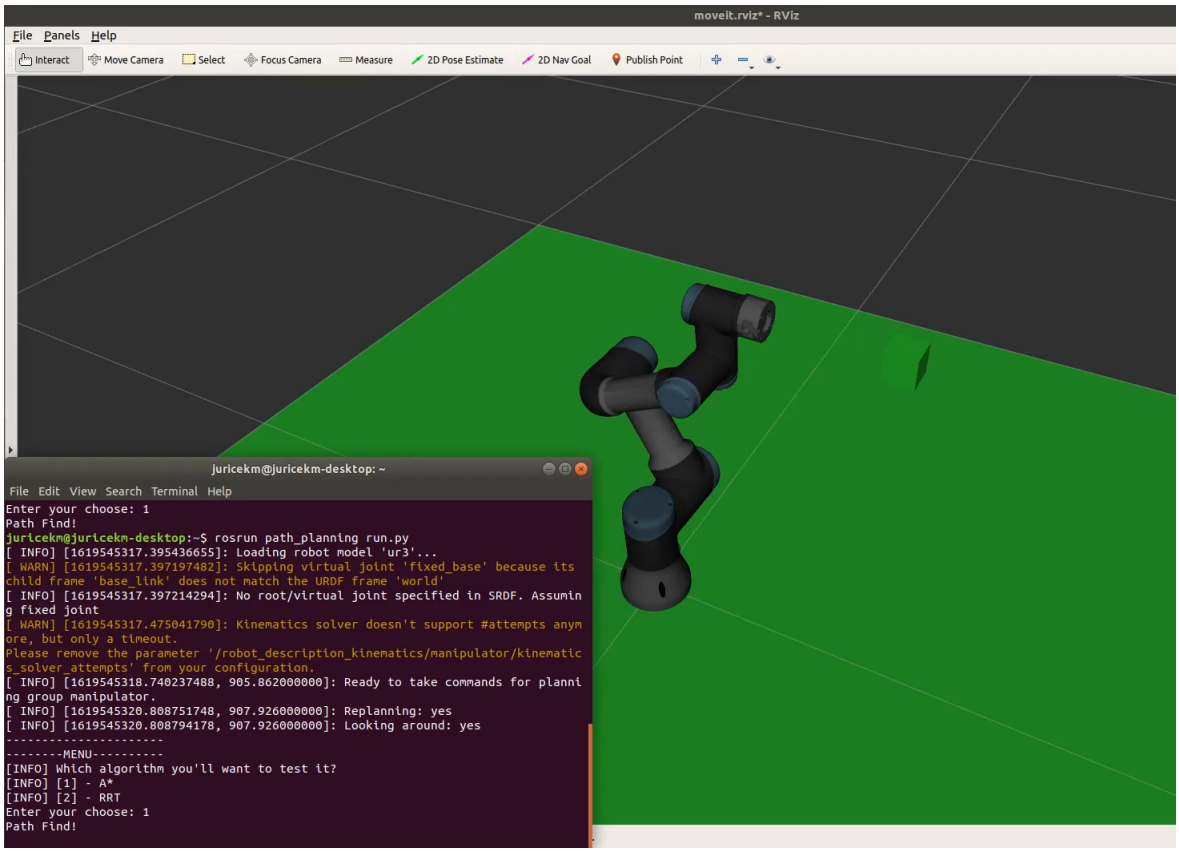
- Plotly
- Standardní knihovny Python 2

K samotnému otestování funkčnosti je použito simulační prostředí Gazebo, framework MoveIT a vizualizační okno ROS Rviz. Před spuštěním hlavního programu, je tedy třeba provést inicializaci a otevření simulačního prostředí Gazebo spolu s modelem kolaborativního robota UR3, spuštění řídícího programu frameworku MoveIT pro kolaborativního robota UR3 od společnosti Universal Robots a poté je tu i možnost spuštění vizualizačního okna ROS Rviz.

Aby bylo možné spustit hlavní program, byl vytvořen ROS balík *path_planning*, který v sobě integruje základní manifest a složku s výslednými python skripty. Hlavní uživatelský program se skladá z jednoduchého konzolového menu, kde při spuštění skriptu *run.py*, se v prvé řadě provede inicializace ROS a robot se přemístí do inicializační polohy, následně může uživatel v jednoduchém switchi spouštět hledání trasy pomocí algoritmů A* nebo RRT. V tomto hlavním uživatelském programu je také definován testovací prostor, počáteční a cílový bod, spolu s možnostmi nastavení samotných algoritmů. Po výpočtu trasy, jsou následně body poslány řídícímu skriptu, kde je proveden výpočet kartézské cesty a samotná simulace provedení pohybu.

Řídící skript *motion_planning.py* se opírá o knihovny frameworku MoveIt a ROS. V tomto skriptu jsou naprogramované funkce jako je inicializace podlahy, pohyb do inicializačního stavu a pohyb dle kartézského výpočtu.

Současně se v ROS balíku nachazejí skripty *rrt.py* a *a_star.py*. Tyto skripty integrují v sobě jádro algoritmů RRT a A*. K výpočetu je použita jako heuristika euklidovská vzdálenost, ale může být použita i například Manhattanská či Minkowskiho vzdálenost. A proto je možné v balíku i spatřit smoatný skript *heuristic.py*. K vizualizaci výsledné trajektorie je volán skript *graph.py*, který následně vykreslí ve webovém prohlížeči nalezené body trajektorie.



Obrázek 2: Zobrazení testovaní aplikace ROS Rviz

4 Výsledky testování

Pro možnosti porovnání a testování algoritmů, byl vytvořen definovaný testovací prostor, který má v geometrickém pojetí tvar kvádru. Tento prostor však může být definován i kupříkladu "koulí", jako plný rozsah pracovního

prosotru kolaborativního robota. Definovaný testovací prostor se nachází v pracovním prostoru kolaborativního robota, přičemž unvitř něho je integrován startovací a koncový testovací bod.

Výsledné téstování je shrnuto v tabulce 1, kde je možné porovnat čas výpočtu obou algoritmů pro jednodoché nalezení cesty bez překážek pro definovaný startovací bod S=[0.13, 0.11, 0.55] a koncový bod K=[0.35, 0.11, 0.55]. Oba algoritmy mají své přednosti a proto byly nastaveny tak, aby bylo možné je co nejtěsněji porovnat. Jelikož algoritmus A* spadá pod kategorii informovaných algoritmů, byl vytvořen prostor, který má parametry testovacího prostoru, přičemž současně byl modifikován rozdělením do 100 buňek v každém směru. U algoritmu RRT nebyly potřeba tyto úpravy a body jsou generovány v pracovním rozsahu, avšak bylo nastaveno 10 možných pokusů nalezení cesty s 500 iteracemi.

Tabulka 1: Výpočetní rychlosti nalezení trasy v definovaném testovacím prostředí

měření	A*	RRT
č. 1	0.14 s	0.59 s
č. 2	0.59 s	0.01 s
č. 3	0.11 s	0.13 s
č. 4	0.10 s	0.92 s
č. 5	0.08 s	0.06 s

Při porovnáním s integrovaným a odladěným algoritmem RRT z frameworku MoveIT, byl naprogramovaný algoritmus RRT řádově 100x pomalejší.

Ačkoliv algoritmus A* pro definovaný problém je obecně brán jako nevhodný, při správné modifikaci a definování problému, může naskýtat i jistý smysl použití. Své výhody nachází například v tom faktu, pakliže existuje alespoň jedna cesta, tento algoritmus ji najde, současně tento algoritmus poskytuje jednoznačné optimální řešení, což může být využito při repetitivních pohybech. Kdežto algoritmus RRT je obecně definován jako vhodný a je jeden z nejznámějších pro tuto aplikaci. Tento algoritmus může cestu najít i mnohem rychleji než algoritmus A*, ale obvykleji mu hledání trvalo delší dobu, své výhody naskýtá v poměrně jednoduchém přístupu kuproti algoritmu A*, avšak to je vykoupenu za cenu možnosti i nenalezeni cesty v definovaném nastavení.

5 Závěr

Výsledkem práce je naprogramování funkčních algoritmů RRT a A*. Tudíž je možné tuto aplikaci testovat i přímo na reálném robotovi, přičemž je jen nutné dostahovat balík pro ROS *Universal Robots ROS Driver*. Algoritmy byly naprogramovány s jistými niancemi ku možnému elegantnějšímu řešení, avšak tyto úpravy byly provedeny především z důvodu pocohpení samotných algoritmů a také neplagiátorského přístupu.

Práce integruje v sobě základní verzi algoritmu RRT a A*, ale existují i rozšířené varianty jako například Bi-RRT, či RRT-Smart nebo RRT-GPU. Verze RRT-GPU se jeví jako nejschůdnější varianta pro definovaný problém, jelikož je určena pro trojrozměrnou implementaci, kdy využívá hardwarovou akceleraci grafické karty. Algoritmy RRT obecně v dané problematice vykazují velkou rychlost, avšak ne vždy naleznou optimalní řešení, proto jako konkurenta můžeme považovat i například algoritmus PRMK a jeho rozšířené varianty. Algoritmus PRMK sice vykazuje známky delšího výpočtu, avšak cesty jsou mnohem optimálnější a možné i je použít pro replánování.

Jelikož algoritmus RRT nevykazoval očekávanou rychlost nalezení cesty, bylo zkoušeno použití i jiných alternativ jako byla například knihovna CuPy. CuPy je kompatibilní knihovna ke knihovně NumPy, která využivá k výpočtu grafické procesory. Po otestování algoritmu RRT při využití knihovny CuPy, bylo vypozorováno, že rychlost je nižší cirka o 0,001 s jak u knihovny NumPy. Zpomalení docházelo při výpočtu heuristiky, přičemž byl potvrzen fakt, že tato knihovna nachází větší potenciál při zpracování mnohonásobně většího počtu dat, než bylo použito při řešení algoritmu RRT.

Práce má potenciál na rozšíření o další algoritmy či adaptace pro rozšířené vaianty samotných algoritmů. Další část rozšíření práce se může také zabývát vyhlazením plánované trajektorie. K tomuto problému jsou například využívané Beziérové křivky nebo B-spline a jiné.

Jedním z hlavním nedostatků této aplikace je výpočetní čas algoritmů, využití programovacího jazyka python je příjemné pro vývojaře, avšak pro reálné aplikace je vhodnější použít programovací jazyk C++, který je mnohonásobně rychlejší. Jedním ze zajimavých konkoruntů co se na poli výpočetní rychlosti týče ku C++ by byl programovací jazyk Julia. Tento programovací jazyk integruje také balíky pro ROS, pod názvem RobotOS.jl a současně poskytuje vývojaří poměrně příjemnou syntaxi a téměř srovnatelnou výpočetní rychlost jako C++, díky jádru, které je napsáno v jazyce C. Jednou z dalších alternativ pro zrychlení výpočtu je použití grafické karty on společnosti Nvidia při použíí technologie CUDA, která umožnujě psát programy v C/C++ pro GPU.

Odkazy

[1] NILSSON, Nils J. *The Quest for Artificial Intelligence* [online]. Cambridge: Cambridge University Press., 2009 [cit. 2021-04-28]. Dostupné z: https://ai.stanford.edu/~nilsson/QAI/qai.pdf.

[2] *A* search algorithm* [online]. isaac computer science, 2020 [cit. 2021-04-28]. Dostupné z: https://isaaccs.org/c/dsa_search_a_star.

[3] *Introduction to A** [online]. Red Blob Games, 2020 [cit. 2021-04-28]. Dostupné z: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[4] VONASEK, Vojtech; FAIGL, Jan; KRAJNÍK, Tomáš; PŘEUČIL, Libor. RRT-path – A Guided Rapidly Exploring Random Tree. In: [online]. 2009, sv. 396, s. 307–316 [cit. 2021-04-28]. ISBN 978-1-84882-984-8. Dostupné z DOI: 10.1007/978-1-84882-985-5_28.

[5] KNISPEL, L. *Pokročilé plánování cesty robotu (RRT)* [online]. Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012 [cit. 2021-04-28]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=57322.