VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

OPTIMÁLNÍ PLÁNOVÁNÍ POHYBU PRO ROBOTA SE VŠESMĚROVÝM PODVOZKEM

OPTIMAL PATH PLANNING FOR OMNIDIRECTIONAL ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ NĚMEC**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2017

Abstrakt

Práce je věnována plánování cesty pro robota, jenž je vybaven všesměrovými koly typu Mecanum a čtvercovým povozkem. Teoretická část je věnována všesměrovým kolům a různým druhům plánovačů se zaměřením na informované plánovače, lokalizace polohy a optimálnosti pohybu. Praktická část je věnována implementaci vybraných plánovačů a porovnání jejich výsledků z pohledu optimálnosti pohybu pro všesměrového robota, kontrola optimálnosti se skládá z porovnání několika vlastností navržené cesty.

Abstract

The work is devoted to planning trips for robot, which is equipped with Mecanum omnidirectional wheels and a square carriage. The theoretical part is devoted omnidirectional wheels and various types of planners With a focus on informed planners, locating the position and movement of optimality. The practical part is devoted to the implementation of selected planners and comparing their results in terms of optimality of movement for omnidirectional robot, optimality check consists of comparing several properties of the proposed path.

Klíčová slova

Robot, plánování cesty, všesměrový, Mecanum kola

Keywords

Robot, path plannig, omnidirectional, Mecanum wheels

Citace

NĚMEC, Lukáš. *OPTIMÁLNÍ PLÁNOVÁNÍ POHYBU PRO ROBOTA SE VŠESMĚROVÝM PODVOZKEM*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

OPTIMÁLNÍ PLÁNOVÁNÍ POHYBU PRO ROBOTA SE VŠESMĚROVÝM PODVOZKEM

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Další informace mi poskytli Ing. Zdeněk Materna a kolektiv, jenž připravil ROS Workshop 2016. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....

Lukáš Němec

14. května 2017

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Jaroslavu Rozmanovi, Ph.D. za jeho pomoc a vedení mé bakalářské práce. Dále děkuji kolektivu, jenž připravil ROS Workshop za uvedení do problematiky daného systému. V neposlední řadě děkuji své rodině a přátelům za podporu a pomoc při studiu.

Obsah

Kapitola 1

Úvod

Ve své práci se zabývám problematikou plánovaní cesty pro všesměrového robota. Oblast plánování cesty robota je důležitá pro rozvíjení autonomního chování robotů v prostoru. Bylo vytvořeno velké množství plánovacích algoritmů s různou kvalitou výsledku. Vývoj těchto technologii je důležitý pro další rozvoj a možnosti nasazení robotů na speciální úkoly. Existuje řada úkolů pro člověka nebezpečných nebo časově velice náročných, kde je může nahradit robot se schopností dosáhnout podobných výsledků. Jedním z těchto úkolů může být výzkum planet v naší sluneční soustavě.

V rámci umělé inteligence se pojem plánování využívá pro popis algoritmu vyhledání sekvence logických operací akcí (transformací), které transformují počáteční stav v prostředí do cílového stavu. V oblasti robotiky se plánováním myslí dosažení cíle bez srážek s překážkami. Plánování se využívá v různých oblastech, jako navigace (automobilů, robotu atd.) v daném prostředí, dále v oblasti počítačové grafiky či průzkum prostoru. Existuje velké množství algoritmů pro plánování, používají se různé technologie (prohledávání grafů, heuristické funkce, atd.). Pro svou práci jsem se rozhodl implementovat jednu z heuristických funkcí, která využívá mřížkové metody dělení prostoru a cenění přestupů mezi body v mřížce. Jedná se o algoritmus, který se vyhýbá lokálním minimům, ale musíme znát předem oblast, ve které se bude robot pohybovat. Cílem mé bakalářské práce je nastudovat různé plánovací algoritmy a s jejich pomocí implementovat navigační systém pro robota se všesměrovým podvozkem.

Kapitola 2

Všesměrová mobilita

Termín všesměrová mobilita je odvozen od schopností robota (zařízení) pohybovat se libovolným směrem bez ohledu na jeho aktuální směrovou orientaci. Roboti jsou často vytvoření pro pohyb v rovinných prostorách, např. robot na čistění podlahy. Pohybuje se tedy ve dvourozměrném prostoru. Normální robot (nevybaven všesměrovými kolečky) se v tomto prostoru dokáže pohybovat třemi směry - vpřed, vzad a okolo svého těžiště. Takovíto roboti nedokáží využít všechny směry pohybu nezávisle, což znamená, že robot si musí nejdříve nastavit směr pohybu úpravou pozice čela (přední části) robota a poté se tímto směrem pohybovat. Robot vybaven klasickými kolečky se nedokáže pohybovat paralelně ke svým osám. Tito roboti jsou označování za tzv. nonholonomní.

Pokud se robot dokáže pohybovat jakýmkoliv směrem ve dvourozměrném prostoru, tak má vlastnost všesměrové mobility, je tedy tzv. holonomní. Všesměroví roboti (vozidla) mají proti klasickým koncepcím, např. auto, výhodu v odlišném řídicím systému. Dokáži se dostat do malých prostor, minimální velikost prostoru je dána jejich šířkou, bez jakýchkoliv obtíží zatímco klasické koncepce musejí konat mnohé a často se opakující operace, např. parkování do řady nebo mezi dvě překážky. Pro všesměrový podvozek je to stejná operace jako jet v před, zatímco klasický podvozek musí využít i operace zatočení a velikost, do které se může vlézt, je též limitovaná jeho maximálním točícím poloměrem. Všesměrový podvozek se tímto stává nejefektivnější technologií pro průchod mezi překážkami. Pratické využití e nasnadě, např. sklady, všeobecný pohyb v objektech a malých prostorách. Všesměrová kolečka tímto dokazují svou efektivitu v manévrování. Všesměrová kola lze rozdělit do dvou kategorií, konvenční a speciální kola.

2.1 Konvenční kola

Především se jedná o otočná kola. Tato kola jsou vybavena otočnými klouby při připojení k podvozku, tím pádem poskytují možnost okamžitého posunu v jakémkoliv směru. Na rozdíl od speciálních kol mají lepší schopnost pohybovat se na nerovném povrchu. Problém těchto kol tkví v tom, že nejsou úplně všesměrové, pokud se chceme pohybovat jiným směrem, než jsou kolečka nastavena, musíme nejdříve změnit jejich nastavení u připojení k podvozku. Po nastavení se lze začít pohybovat. Tato kolečka můžeme vidět např. u nákupních košíků, skladovacích vozíků nebo jiných vozidel, kde přesouváme větší náklad. U všech uvedených příkladů je hlavní hnací silou člověk. Při použití v robotice by nastaly další výdaje s motory. Kolečko by muselo mít vlastní hnací motor a motor pro nastavení otočného kloubu. Tím by se zbytečně navýšila cena a složitost softwaru.

2.2 Speciální kola

Návrh speciálních kol je založen na konceptu aktivního posunu v jednom směru a přitom mít stále možnost pasivního pohybu v jiném směru. Tento koncept zvyšuje flexibilitu pohybu v prostorách s mnoha překážkami. Do tohoto konceptu se zahrnují i Mecanum (švédská) kola nebo speciální kuličková kola. Speciální kola při točení využívají nepřímého pohybu, některé kolečka mohou stát. Mechanismus je tvořen malými otočnými válečky, které jsou umístěny po celém obvodu kola. Kola umožňují normální pohyb vpřed, ale mají možnost kdykoliv změnit směr pohybu i paralelně k osám kol. Kola jsou schopna těchto pohybů, protože jsou válečky umístěné kolmo k osám kol nebo alespoň v jistém úhlu. Pokud jsou na platformě umístěny dvě nebo více speciálních kol, je díky vlastnostem kol platformě umožněno pohybovat se všesměrově.

Mechanismus kuličkového kola využívá aktivní kruh poháněný motorem a převodník pro přenos energie přes válce na kuličku pomocí tření. Tímto způsobem je platforma schopna okamžitého pohybu ve všech směrech.

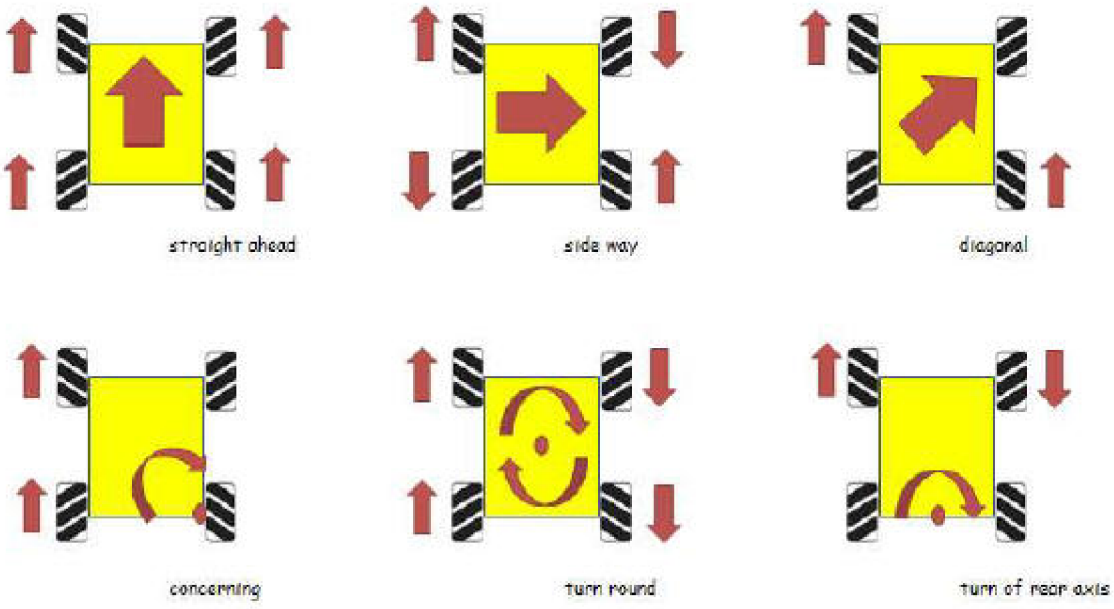
Dalšími speciálními koly jsou Mecanum (švédské) kola využívající právě technologii otočných válečků připojených ke kolu v úhlu.

Obrázek 2.1: Všesměrové kola

2.3 Mechanismus Mecanum kol

Jedno z častěji používaných všesměrových kol je Mecanum kolo. Mecanum nebo také Švédské kolo bylo vynalezeno v roce 1997 inženýrem Bengtem Ilonem, který pracoval pro firmu Mecanum AB. Od této firmy je odvozen název typu kola. Někdy je používán též název podle svého vynálezce Ilonovo kolo.

Kolo je tvořeno jedním kulatým nástavcem, který se může otáčet, a na jeho obvodu jsou otočné válce. Úhel mezi osou válců a centrální osou kola může mít libovolnou hodnotu, ale v případě konvenčního švédského kola je úhel 45°. Umístění v úhlu přenáší část energie pohybu v před, do pohybu kolmo ke směru kol. Naše platforma je tvořena čtyřmi koly umístěny podobně jako na automobilu. Síly, které pohybují koly, se dají sečíst do pohybových vektorů, jejichž výsledkem je konečny směr pohybu platformy. Díky tomu lze správnými směry točení kol dosáhnout jakéhokoliv směru pohybu platformy.



Obrázek 2.2: Možnosti pohybu podvozku s Mecanum koly

Válce jsou tvarovány tak, aby siuleta kola byla kruhová. Může získat tvar válce tak, že vezmeme válec a provedeme řez válcem v námi zvoleném úhlu, po rozříznutí lze vidět tvar otočného válce pro naše kolo. V našem případě je úhel 45°. Tvar válce by měl odpovídat rovnici ze zdroje [3]:

$$\frac{1}{2}x^2 + y^2 - R^2 = 0 \tag{2.1}$$

kde R je vnější obvod kola. Když válec o velikosti, L_r, je menší než vnější odvod kola, R, pak tvar válce může být podobný oblouku o poloměru 2R.

K tomu abychom dosáhli kruhové podoby kola, musíme vypočítat minimální počet válců. Musíme znát délku válce, L_r, a požitím rovnice $n = 2\pi/\phi$ zjistím počet válců n.

Výpočet ϕ je dán rovnicí ze zdroje [3]:

$$\phi = 2 \arcsin \frac{L_r}{2R \sin \gamma} \tag{2.2}$$

Pokud naopak známe počet válců n, můžeme vypočítat jejich délku [3]:

$$L_r = 2R \frac{\sin \phi/2}{\sin \gamma} = 2R \frac{\sin \pi/n}{\sin \gamma} \tag{2.3}$$

Šířku kola lze vypočítat podle rovnice [3]:

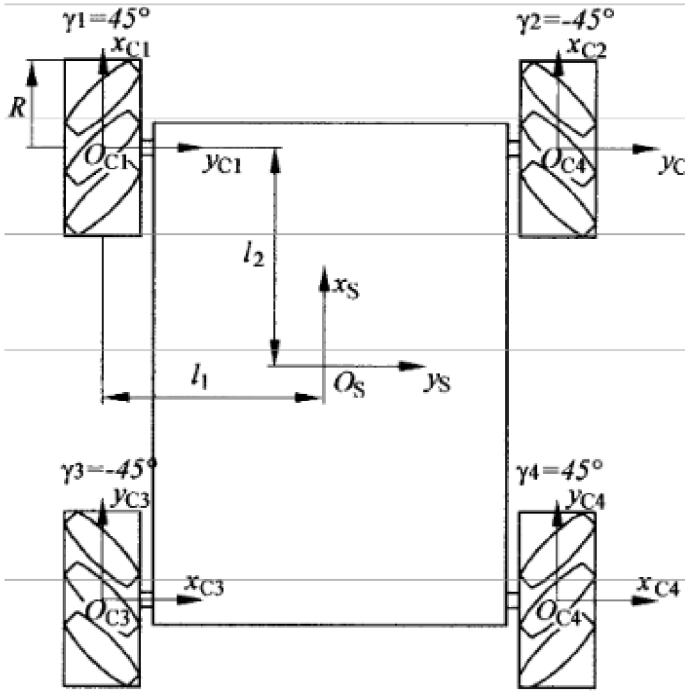
$$L_w = L_r \cos \gamma = 2R \frac{((\sin \pi)/n)}{\tan \gamma} \tag{2.4}$$

Válce nesmí být ani vertikální, ani úplně horizontální. Hlavním cílem je dosáhnout toho, že jediný pohon bude otáčet celým kolem okolo své osy a za pomocí malých změn rychlosti otáčení bude platforma měnit svůj směr. Mecanum všesměrová kola mají tří stupně volnosti složené z otáčení kola, otáčení válců a otáčení skluzu kolem svislé osy procházejícím bodem styku s povrchem. Všesměrová kola se dají dělit podle prvků, na prvky v aktivním směru a pasivním směru. Prvky v aktivním směru se točí spolu s osou válce, který je právě v dotyku se zemí, zatímco pasivní válce jsou kolmé k ose aktivního válce.

Obrázek 2.3: Mecanum kolečko

2.4 Kinematika čtyř kolového podvozku s Mecanum koly

Pokud jsou Mecanum kola připojeny k podvozku způsobem zobrazeném na obrázku 2.2, tak je část rotační síly vložena do směru válečků a část síly do celého kola. V závislosti na rotaci každého kola se výsledně robot pohybuje takovým směrem, jenž vznikl složením vektorů síly pohybu každého jednotlivého kola. Za využití tohoto způsobu není nutné vůbec měnit natočení kol za pomocí přídavných mechanismů (nápravy).



Obrázek 2.4: Rozložení kol u 4 klového mecanum podvozku

Pokud budeme považovat $x_s\ O_s\ y_s$ rámec za vztahující se k podvozku (obr. 2.4), můžeme zapsat rovnici pro pohyb celého robota na základě zdroje [3] jako:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = R/4 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1/(l_1+l_2) & 1/(l_1+l_2) & -1/(l_1+l_2) & 1/(l_1+l_2) \end{bmatrix} . \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{2.5}$$

kde R je poloměr kola, ω_i je úhlová rychlost (rad/s) kola (i = 1, 2, 3, 4), l_1 a l_2 jsou vzdálenosti mezi osami kol a středem podvozku.

Pokud má být rychlost robota skládána, musíme vypočítat úhlovou rychlost pro každé kolo (inversní řešení rychlosti) opět rovnice ze zdroje [3]:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & 1 & -(l_1+l_2) \\ 1 & -1 & (l_1+l_2) \\ 1 & -1 & -(l_1+l_2) \\ 1 & 1 & (l_1+l_2) \end{bmatrix} . \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{2.6}$$

Tato matice se dá zapsat jako čtyři rovnice (zdroj [8]):

$$\omega_1 = 1/R(v_x + v_y - (l_1+l_2)\omega_z) \tag{2.7}$$

$$\omega_2 = 1/R(v_x - v_y + (l_1+l_2)\omega_z) \tag{2.8}$$

$$\omega_1 = 1/R(v_x - v_y - (l_1+l_2)\omega_z) \tag{2.9}$$

$$\omega_1 = 1/R(v_x + v_y + (l_1+l_2)\omega_z) \tag{2.10}$$

2.5 Použití Mecanum kol

Mecanum kola nebo jakákoliv jiná všesměrová kola se využívají pro zařízení, která se musí pohybovat v malých prostorech. Díky jejich možnosti se otáčet namístě a pohybovat se jakýkoliv směrem bez nutnosti měnit natavení kol pro změnu směru pohybu. Tyto vlastnosti kol se dají efektivně použít u zařízení, která se využívají ve skladech, pro přístup do vysokých polic nebo při instalaci zařízení ve výškách[1], dále se ve skladech dají využít tyto kola u přemísťování materiálu nebo výrobků[2].

Dalšími možnostmi těchto kol je jejich využití u transportu osob se sníženou pohyblivostí[3]. Toto využití je velice praktické, dává možnost pohybu ve stísněných koridorech nebo výtazích, kde je nyní možnosti pro tyto přístroje využívat i klasické výtahy s dostatečnými rozměry, kde se vozík pouze otočí o 180° a může vyjet přední částí ven.

Využití kol je možné v mnoha dalších oberech např. vojenství, výzkum cizích planet atd., jsou popsány ve zdroji [1].

[1] https://www.youtube.com/watch?v=IlmKcohyXG0
[2] https://www.youtube.com/watch?v=ZW3OBb_b89k
[3] http://car.pege.org/2006-ever-monaco/wheel-chair.htm

Kapitola 3

Lokalizace robota

Lokalizace je jeden ze základních problémů mobility v robotice. Pro využívání autonomních robotických systémů je daný problém zásadní, protože bez lokalizační metody nevíme, kde se právě robot nachází a není možné ani navigovat v prostoru.

3.1 Mobilní robotika bez lokalizace

Existují alternativní přístupy řízení mobilních robotů. Tyto algoritmy nepotřebují lokalizaci. Jedná se o reaktivní řízení nebo evoluční algoritmy. Tyto formy řízení robota mají své uplatnění, ale výhodnějšími jsou stále formy s lokalizací. Reaktivní řízení reaguje na aktuální stav, neplánuje svou cestu do kroků. Tento způsob ovládání je porovnatelný s algoritmy neinformovaných plánovačů (viz. kapitola 4.1.2). Evoluční (respektive genetické) algoritmy zase neumožňují nahlédnout do nitra řídicího systému, což nemusí být při plnění zodpovědnějších úkolů žádoucí.

3.2 Relativní lokalizace

Za pomoci relativní lokalizace lze určit relativní změnu polohy robota, jeho posun a rotaci, která se změnila posunem z počáteční polohy. Celá změna polohy se rozděluje na dílčí pohyby, které po sloučení určí pozici robota. Na rozdíl od absolutní lokalizace (lokalizace za pomocí GPS a podobných technologii) neumožňují prostředky relativní lokalizace určit přesnou polohu robata, může však zjistit jeho polohu vůči počáteční pozici, pokud je známa, a tím se přiblížit absolutní lokalizaci. Snaha o co nepřesnější lokalizaci je ovšem koplikována akumulací jednotlivých chyb, čímž se celková chyba zvětšuje. Relativní lokalizace patří k nejpoužívanějším prostředkům lokalizace mobilních robotů. Kombinují se s globálními technikami pro odhadování okamžité pozice mezi měřeními a odhady pozice absolutní. Tyto techniky se používají zpravidla v delších intervalech mezi periodicky prováděnými měřeními. V některých případech, kdy relativní lokalizace je přesná, lze zvětšit interval mezi prováděním absolutní lokalizace, protože je náročná na výpočet a zdroje. Spolupráce technik lokálních a globálních (absolutních) technik, které odhadují pozici robota ve světě, může být prohloubena za použití senzorů, které vyžadují kalibraci. Porovnáním výsledků relativní a absolutní lokalizace je možné použít k počáteční kalibraci senzorů.

Všechny relativní lokalizační metody používají pro odhad polohy skládání dílčích pohybů. Způsob jakým jednotlivé metody pracují, se liší.

Jednotlivé dílčí změny pozice:

- přímo naměřené (např. při odometrii)
- vypočteny integrací okamžité rychlosti podle času
- Okamžitou rychlost je možné získat měřením (navigace výpočtem, využití Dopplerova jevu).
- Okamžitou rychlost je možné vypočítat integrací okamžitého zrychlení podle času (inerciální navigace). Přitom zrychlení už lze měřit přímo pomocí akcelerometrů.

Integrace je důvodem malých odchylek u dílčích výpočtů, které se poté skládají do velké chyby.

Seznámíme se s navigací výpočtem jako historicky první metodou tohoto druhu, s odometrií využívající rotační enkodéry jako metodou v mobilní robotice zdaleka nejpoužívanější a dalšími alternativními metodami používajícími jiné senzory pro měření ujeté vzdálenosti nebo rychlosti.

3.2.1 Navigace výpočtem

Nejzákladnější metoda pro odhadování pozice za pomocí výpočtu rychlosti a směru pohybu a času uplynulého od poslední známé pozice. Tato metoda je aplikována již od dob středověku. Byla používána v mořeplavbě, později byla využívána v letectví. Pro její zlepšení se používaly další metody jako hvězdy a orientační body, avšak i zde docházelo k odchylkám (mořské proudy atd.). Relativní lokalizace akumuluje chyby měření, a proto musí být doplněna dalšími metodami pro zlepšení výsledku, přesto se dále musí stále počítat s chybou (kvůli bezpečnosti).

Přes svou jednoduchost a nepřesnost je metoda stále využívána v případech, že ostatní techniky selžou nebo nemohou být z jiných důvodů použity. Navigace výpočtem je základem dalších technik relativní navigace, např. inerciální navigace. Někdy se výraz navigace používá jako společný název pro všechny relativní lokalizační metody.

3.2.2 Odometrie

Relativní lokalizace za pomocí metody odometrie je založena na odhadu změny pozice a orientace kolového robota za pomocí údajů o otáčení jeho hnacích kol nebo běžných kol pomocí rotačních enkodérů. Rotační enkodér je senzor, který předává informaci o rotačním pohybu na elektronický signál. Enkodéry může rozdělit na několik typů:

- Optické, děli se na:
  - transmisivní, využívající přerušování světelného paprsku rotujícím děrovaným diskem (optická závora)
  - reflexivní, využívající plný disk s reflexními a matnými ploškami
- kartáčové, pracující na principu komutátoru
- odporové na principu otočných potenciometrů
- magnetické, měřící změny magnetického pole
- induktivní, využívající principu elektromagnetické indukce

Pro svou spolehlivost, vysoké rozlišení a rozumnou cenu jsou v mobilní robotice nejčastěji používané enkodéry optické, transmisivní a reflexivní.

Enkodéry lze dále dělit podle konstrukce a funkcionality:

Jednokanálové enkodéry

Mají pouze jednobitový výstup (0 nebo 1), a proto neposkytují žádnou další informaci. Navíc jsou chybové, hlavně při nízkých otáčkách u nich hrozí výskyt šumu. Další chyby jsou při zastavení, vyskytne se tam náhodná hodnota, ta se může v čase měnit. Tyto enkodéry se v odometrii nepoužívají.

Inkrementální enkodéry

Též zvané kvadraturní enkodéry (phase quadrature incremental envoders), pracují již se dvěma kanály (dva bity), kanály jsou vzájemně fázově posunuté. Dokáží na rozdíl od jednokanálových enkodéru zjistit směr otáčení a při použití stejného disku mají dvojnásobné rozlišení a netrpí šumem při nízkých otáčkách. Zastaví-li se inkrementální enkodér na jakékoliv hraně, projeví se případný šum přinejhorším oscilací polohy o ± 1. Dále enkodér vyžaduje dostatečnou vzorkovací frekvenci stejně jako jednokanálový. Při překroční maximálních otáček se na rozdíl od jednokanálových enkodérů projeví typicky chyba, k indikaci slouží současná změna hodnot obou kanálů.

Enkodér se dá rozšířit od další funkci přidáním koncového spínače (optické závory). Po přidání doplňku lze enkodér použít i pro měření absolutní úhlové pozice. Po restartu je třeba enkodér vždy inicializovat z koncové polohy. Někdy jsou enkodéry už vybaveny třetím kanálem (index output), který nabývá hodnoty high jednou za 360°.

Absolutní enkodéry

Jsou enkodéry, které mají v kterýkoliv okamžik na svém paralelním výstupu absolutní úhel natočení enkodéru. Jejich rozlišení je dvojnásobný počet kanálů. Musíme počítat s tím, že čím větší rozlišení, tím větší cena. Tento enkodér se používá hlavně v mobilní robotice jako senzor absolutní úhlové pozice. Využití je například u natočení řídícího kola u robotů s pojezdem typu tříkolka a v dalších podobných případech.

Nejčastější formou výstupu enkodéru je Grayův kód, jehož délka kódu je stejná jako délka binárního kódu. Rozdílem je, že při každém přechodu dochází k překlopení právě jednoho bitu v kódu. Zajišťuje to eliminaci problému, který vzniká při přechodovém jevu za použití binárního kódu. Ideální synchronní změna stavu několika kanálu ve stejný okamžik je technicky náročná. Při použití binárního kódu se hodnoty postupně překlápějí, takže na výstup dojde několik (min. jedna) informací, které neodpovídají aktuální pozici absolutního enkodéru s binárním kódem. Při použití Grayova kódu je problém eliminován.

Kapitola 4

Plánování cesty

Kapitola Plánování cesty se zabývá algoritmy, které se používají pro hledání cest v různých prostředích tak, aby se z počáteční pozice dostal robot do cílové pozice, nebo vyhodnotil prostředí za neprůchozí k cíli. Plánovače jsou základně děleny na informované a neinformované. Informované mají k dispozici mapu nebo popis prostoru, kde se robot nachází. Tento prostor je stabilní. Neinformované plánovače využívají pro pohyb v prostoru senzory, čidla a další metody pro určení polohy tak, aby dosáhl cíle. Pokud se robot pohybuje v částečně známém prostoru (dynamickém), kde se překážky mohou pohybovat nebo objevovat nové, tak se využívá kombinace informovaných a neinformovaných metod. Tato kombinace začíná informovanou metodou, která navrhne globální cestu (globální plánovač) a při pohybu se využívají neinformované metody, které se aplikují, pokud se objeví neočekávaná překážka, která není uvedena v mapě (lokální plánovač).

4.1 Plánování cesty pro všesměrového robota

Pro plánování cesty všesměrového robota je mnoho možností. Autonomní metody se většinou skládají ze dvou částí. První část je zmapování prostoru a druhou částí je samotné vyhodnocování nejlepší cesty k cíli. Při plánování cesty se používají stejné algoritmy jako pro klasické roboty. Všesměrový robot má později výhodou, že nemusí vyhledávat cestu korespondující s omezeními jeho pohybu. Všesměrový robot potřebuje jen dostatečnou šířku prostoru na svou nejdelší uhlopříčku a je schopen se dostat kamkoliv bez výpočtů dalšího prostoru na otáčení nebo srovnávaní podvozku.

Při plánování cesty v rámci autonomního ovládání je běžné na počátku určit počáteční pozici robota a jeho cíl. Dále se na začátku vyhodnocuje prostředí a samotné vlastnosti robota, zda je možné se v rámci prostředí pohybovat. Existují dvě hlavní skupiny algoritmů pro plánování pohybu na základě znalostí okolí robota, a to informované metody a neinformované metody. Dále existuje možnost přímého vládání uživatele.

Mnoho z popsaných algoritmů a technologií je posáno ve zdroji [2].

4.1.1 Informované plánování

Informované plánování cesty je založeno na kompletní znalosti prostředí, ve kterém se bude robot pohybovat. Jeho počáteční a cílová pozice a všechny překážky. Robot za pomocí poskytnuté mapy vyhodnotí cestu k cílí a začne se pohybovat podle vyhodnocené trasy. Další možností je postupné vyhodnocování trasy, kdy robot vyhodnotí jeden úsek, posune

se na vyhodnocenou pozici a vyhodnotí další krok. Při tomto přístupu dorazí robot do cíle po několika krocích.

Mřižková dekompoziční metoda

Plánování pomocí mřížky je metoda, které rozdělí celý prostor na buňky. Přesnost a výpočetní složitost závisí na velikosti buňky pro daný model (viz. obrázek 4.3). Mřížkové metody vytvářejí přijatelné cesty. Metody, které využívají pohyb ve všech úhlech, získávají lepší cesty, protože nejsou omezené mřížkou. Je mnoho algoritmů využívající mřížku.

Průchod grafem

Tento algoritmus rozděluje buňky do tří kategorií.

1. Nenavštívené – Ještě nemají přidělenou vzdálenost.
2. Živé – Navštívená buňka s nenavštívenými sousedními buňkami.
3. Mrtvé – Navštívená buňka, která sousedí jen s navštívenými buňkami.

Algoritmus je postaven na procházení front s nenavštívenými buňkami, které se nazývají open (otevřené), od tohoto názvu je odvozen název fronty open queue. Druhá fronta, kde se ukládají navštívené buňky, se nazývá closed queue (Nemusí se jednat o fronty, může se též využít zásobník, záleží na implementaci). Algoritmus vybere jednu buňku se všemi jejími sousedy (nenavštívenými) a zpracovává postupně frontu open, dokud nenalezne cíl. Na tomto typu algoritmu má několik implementací.

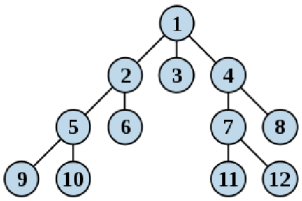
Breadth-First Search (BFS)

Algoritmus prohledávání do šířky (BFS) má jednoduchý princip fungování založený na frontách (FIFO). Algoritmus prochází stromovou strukturu nebo graf. Funkce je následující:

1. Vybereme kořen stromu nebo libovolný uzel grafu a zpracujeme ho.
2. Nové potomky (jen ty co již nejsou ve frontě) uložíme do fronty.
3. Vybereme první uzel z fronty a zpracujeme ho.
4. Pokud je fronta prázdná konec, jinak pokračuj bodem 2.

Využívání fronty má důsledek v tom, že je algoritmus vlnový, to znamená, že jsou uzly zpracovány v pořadí daném jejich vzdálenosti od počátku (kořene). Složitost algoritmu je O(|U|+|H|), kde U je uzel a H hrana. Nevýhodou algoritmu je procházení celého grafu, výhodou je nalezení nejkratší cesty

Použití algoritmu BFS je možné vidět ve zdroji [7].



Obrázek 4.1: Postup algoritmu BFS

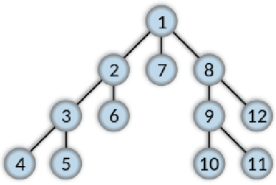
Depth-First Search (DFS)

Algoritmus expanduje buňku s největší hloubkou. Tato metoda zabírá méně paměti nežli BFS (v paměti jsou pouze buňky z počáteční fáze trasy a ty expandované). Metoda je často omezená maximální hloubkou hledání, po dosažení maximální hloubky se volá návratový mechanismus.

Algoritmus:

1. Vyber kořenový uzel nebo libovolný uzel z grafu a zpracuj ho.
2. Ulož nalezené potomky do fronty, pokud tam již nejsou.
3. Vyber první uzel z fronty a expanduj ho.
4. Pokud si nalezl řešení konec (nemusí se ukončovat, projde se pak celý graf a můžeme nález lepší řešení), jinak pokračuj bodem 3.

Algoritmus využívá opět dvou front open a close. Ukládání do fronty open je takové, že každý nově expandovaný uzel své potomky ukládá na začátek je to proto fronta typu zásobníků (FILO). Výhodou oproti BFS jsou menší nároky na paměť. Složitost algoritmu je stejná jako u BFS O(|U|+|H|).

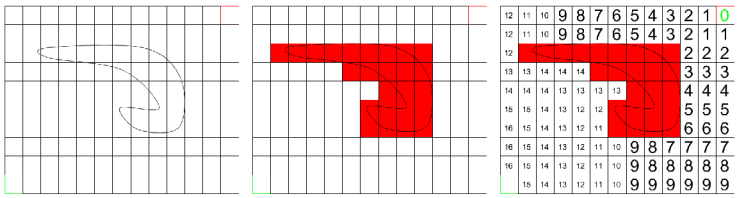


Obrázek 4.2: Postup algoritmu DFS

Poteciální pole

Hodnota je přidělená každé buňce na mapě. Hodnoty se využívají pro plánování trasy. Nejnižší hodnota (0) je přidělena cíli cesty a počáteční pozice je hodnocena jako nejvyšší. Hodnoty se přidělují pomocí algoritmů např. flood fill (viz obr. 2.3 a 2.4 a 4.1). Přechod se vždy plánuje na sousední pozici s nejmenším hodnocením (nejmenší cenou). Hodnota pozice okolo překážek má vyšší hodnotu, tyto hodnoty jsou pro robota nevýhodné, a tak se překážkám vyhýbá. Tento algoritmus má výhodu rychlosti, ale existuje možnost zaseknutí se v lokálním minimu.

Využití této metody můžeme ve zdroji [9] nebo [4].



Obrázek 4.3: a)Mřížková metoda počáteční stav b) řešení překážky (objektu) c) výsledek metody flood fill

Dijkstrův algoritmus

Používají se jeho modifikace A* a B*. Často se využívá k nalezení nejkratší cesty ve mřížce. Na rozdíl od jiných řešení nepoužívá normální frontu, ale prioritní frontu, která se prochází na základě vzdáleností (hodnotou) buněk od cíle. Algoritmy A* a B* mění přístup k procházení fronty nezpracovaných buněk. V závislosti na vzdálenosti od cíle berou v úvahu součet vzdálenosti s ohledem na vzdálenost od počáteční pozice, nebo odhadu odstupu od počáteční pozice. Algoritmy jsou proto více rychlejší v nalezení cesty, ale výsledná cesta nemusí být ta nejkratší.

Dijkstrův algoritmus má složitost O(E Log V), kde V je počet uzlů a E je počet hran toto složit má v případě implementace prioritní fronty jako binární haldy, pokud je procházení fronty sekvenční tak je složitost O(V^2).

Dijkstrův algoritmus je jistým způsobem zobecněný algoritmus vyhledávání do šířky (BFS). V algoritmu je rozdíl šíření rozvoje hran, kde se nevyužívá počet hran od zdroje, ale jejich vzdálenost od zdroje. Díky této funkci se zpracovávají jen hrany s nejkratší cestou. Dijkstrův algoritmus si ponechává všechny hrany v prioritní frontě řazené dle vzdálenosti od zdroje. Na počátku algoritmu má jen zdroj hodnotu 0 (náš cíl nebo zdroj záleží na implementaci), ostatní uzly nemají hodnotu. Při každé iteraci se vybere uzel s nejvyšší prioritou (nejnižší hodnota od zdroje) a tento uzel zpracuje. Přidá se do fronty zpracovaných uzlů a jeho potomci, kteří nejsou ještě ve frontě a ověří, jestli jsou blíže zdroji než byli před zařazením zpracovaného uzlu do fronty zpracovaných. Proto vždy ověřuje [vzdálenos_zpracovaný]+[hodnoty_hrany_(zpracovyný-potomek)]<vzdálenost_pozařzení. Pokud podmínka platí, pak se nastaví potomkovi nová vzdálenost a po průchodu všemi potomky podmínkou se opět vybere uzel s největší prioritou a následuje další iterace. Algoritmus je ukončen, až když je prázdná prioritní fronta. Podmínkou použití Dijkstrův algoritmu je absence záporných hran.

A*

Oproti Dijkstrovu algoritmu používá heuristickou funkci. Heuristická funkce využívá znalost minulých uzlů a zbývající cesty do cíle. Heuristická funkce ovlivňuje řazení uzlů v prioritní frontě. Čím je hodnota heuristické funkce nižší, tím vyšší má prioritu ve frontě. Čerpáme z předpokladů, že uzly ve frontě netvoří kružnici a pro každý cílový uzel se v ní nachází nanejvýš jedna cesta a to doposud nejkratší nalezená. Určení priority ve frontě je závislé na funkci f(x)=h(x)+g(x), kde f(x) je přepokládaná délka cesty, h(x) heuristická funkce pro koncový uzel x a g(x) délka cesty k x.

Heuristická funkce musí splňovat několik podmínek, musí být větší než 0. To znamená, že její hodnota musí být nižší nebo rovna skutečné vzdálenosti z daného uzlu do cíle. Nikdy nemůže být větší nežli reálná vzdálenost z uzlu do cíle h(x)≤h(y)+cost(x,y). Heuristická funkce vzniká na základní znalosti prostoru.

Časová složitost je O(E+V), kde V je počet uzlů a E je počet hran. Reálné zpracování algoritmu A* (Dijkstova algortimu) můžeme vidět ve zdroji [6].

Pseudo-Voronoi Diagramy

Nejkratší cesty nejsou vždy nutné v praxi. Někdy je nutné se např. držet ve větší vzdálenosti od překážek. V kombinaci se širokým vyhledáváním (breadth search) od každé překážky je možné definovat Voronoinovu hranu (Voronoi edge) na místech, kde se překrývají vzdále-

nosti od překážek. Více typickým zpracování Voronoinova algoritmu je, když se mřížkový prostor rozdělí do polygonů, vice informací v kapitole Metody cestovních map.

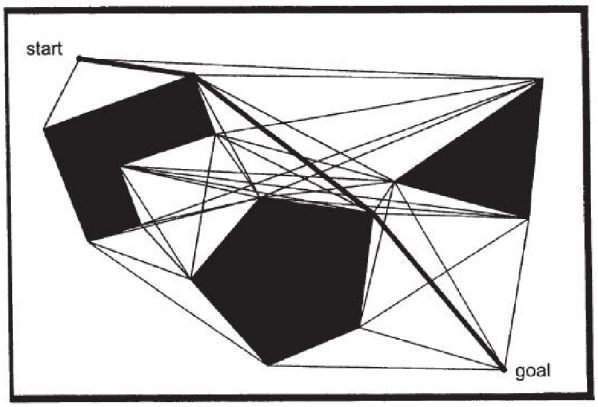
Metody cestovních map

Tyto metody vytvářejí mapy cest z kontinuálního prostoru (Graf zobrazující volný prostor). Hrany grafu jsou proveditelné cesty, po kterých se může robot pohybovat bez omezení. Pro nalezení nejlepší cesty se používají algoritmy pro průchod grafem.

Graf viditelnosti

Jednoduchá struktura, která umožňuje najít nejkratší cestu mezi dvěma body. Hrana překážky je definována jako buňka grafu, stejně jako počáteční a konečný bod. Vždy se spojují jen dvě buňky, spoje nejdou přes překážky. Rychlost algoritmu záleží na počtu hran (překážek), protože se zjišťuje viditelnost z každé buňky a cíle a startu. Pro získání cesty z vytvořeného grafu se používají vyhledávací algoritmy jako Dijkstrův nebo A*.

Graf má několik nevýhod: vede cesty blízko překážek, musí se aproximovat překážky (polygony) a řešení se dotýká překážek. Výhodou je nalezení cesty, pokud nějaká existuje, a je to nejkratší možná cesta.



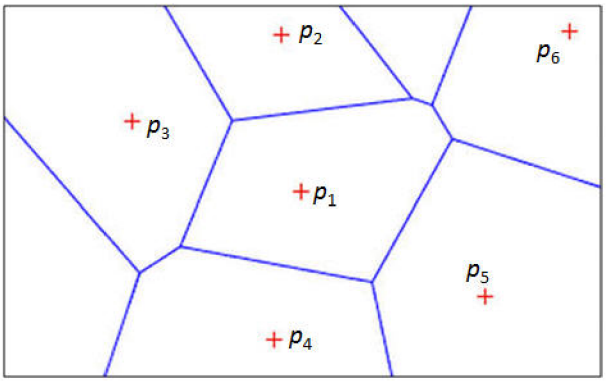
Obrázek 4.4: Ukázka grafu viditelnosti

Voroného diagram

Algoritmus je založen na rovinném grafu, geometrická struktura tvořena body se stejnou vzdálenosti od překážek. Tyto body definují zavřené a otevřené oblasti zvané Voroného buňky (viz obr. 4.4). Robot se může pohybovat po hranách buněk s jistotou, že udržuje maximální vzdálenost od překážky. Vyvaření těchto buněk se nazývá Voroného dekompozice.

Voroného diagram se dá vytvořit za pomocí několika metod. První z metod je naivní algoritmus, dále inkrementální algoritmus, algoritmus rozděl a panuj, zametací „Fortuného“ algoritmus a metoda zdvihu.

Nevýhody algoritmu jsou složitost konstrukce přímo se odvíjející od typu překážek, řešení nejsou minimální. Výhodou je pohyb robota v bezpečné vzdálenosti a nalezení cesty pokud nějaká existuje.



Obrázek 4.5: Voroného diagram

Pravděpodobnostní vzorkovací metoda

Konfigurace popisuje pozici robota. Prostor konfigurace (Configutration space) C_{space} je kolekce všech možných konfigurací. Kolekce konfigurací, které se vyhýbá kolizním s překážkami, se nazývají volný prostor (free space) C_{free}. Volný prostor (C_{free}) je doplňkem oblasti překážek C (obstacle region).

- Pokud robot zastupuje jeden bod ve 2D zobrazení, muže být konfigurace zobrazena za pomocí dvou parametrů (x, y).
- Pokud je robot platforma s n klouby, tak C je n-dimenzionální.

Pro správné plánování je nutné, aby počáteční a cílová konfigurace q_{init}, q_{goal}, musí být součástí C_{free}. Obtíž nalezení kvalitní cesty je ve vyhledání zřetězené křivky v prostoru C_{free}, které umožní zobrazení sekvence povolených pohybů mezi začátkem a cílem.

V nejelementárnější formě funguje pravděpodobnostní plánování ve dvou fázích. První fáze generuje graf cest (road map) a pak vyhledává cestu v grafu. Graf cest je pravděpodobnostní, protože je složen z náhodných konfigurací, které jsou testovány, zda jsou v C_{free} a jestli mohou být připojeny k jiným konfiguracím v C_{free}. Tyto přidané hrany z vhodné náhodné konfigurace jsou opakovány plánovačem, dokud nenalezne přímé spojení do cíle. Ke zrychlení konvergence k cíli je vhodné až po určitém počtu iterací, které se snaží spojit cíl a start přímo.

Největší výhoda v metodě náhodných vzorků je, že může být zobrazena na multidimenzionální konfiguraci prostoru. Pro příklad, pokud je potřeba vytvořit plán pro mechanické rameno s šestistupňovou volností s krokování pohybu po jednom stupni. Pokud je rozsah pohybu kloubu 120°, bude potřeba paměť pro uložení všech vzorků $120^6 = 355.957$ GB. Ukládání pouze náhodných vzorků, které nejsou kolizní s překážkami, není pro paměť taková zátěž.

Hlavním problémem pravděpodobnostních plánovacích metod je velké množství hran. Proto výsledná cesta ze startu do cíle obvykle je křivolaká a obsahuje velké množství zbytečných pohybů. Proto se musí po dokončení plánovaní požít algoritmus pro větší zjemnění cesty (vylepšit ji).

Popis C_{free} jako jeden strom má ale ten následek, že relativně blízké pozice se mohou zdát, že leží daleko od sebe, pokud použijeme pravděpodobnostní směrovače map. Tento problém řeší algoritmus RRT, který vždy vytváří nový strom ke stratu a cíli. Je popsán ve zdroji [3].

4.2 Optimální pohyb robota

Pro co největší efektivitu robota se musí počítat také s jeho energetickými zdroji. Většina robotů je napájena ze svého vlastního energetického zdroje, jako je baterie. Pohyb robota a délka jeho operační doby je závislá na energetickém zdroji. Největší ztrátu energie způsobují senzory a pohyb robota. Omezení funkce senzorů není nejlepší cestou zachování energie, protože tím ztrácíme zpětnou vazbu od robota. Pokud nemůžeme už snižovat energetickou spotřebu součástek robota, tak stále můžeme zachovat energii robota za pomocí efektivního plánování pohybu.

Pro čtyřkolového robota je pohyb tvořen za pomocí skládání vektorů pohybu jednotlivých kol. Již to, že každé kolo má i vedlejší vektor pohybu, zvyšuje ztrátu energie. Možnost robota pohybovat se v úhlech 0° - 360° má za výsledek velkou manévrovatelnost, ale ne každý pohyb je energeticky efektivní. Ve zdroji [10] jsou uvedeny experimenty (obr. 4.10), které zobrazují energetickou ztrátu při pohybu vertikálním, horizontálním a otáčením na místě. Z těchto experimentů lze vyvodit, že pohyb při použití všech kol je energeticky efektivní, ale na obr. 4.10 b) ze zdroje [10] je možné vidět, že pohyb vpravo a vlevo je méně efektivní než pohyb vpřed a vzad. Proto v rámci plánování by měl robot co nejvíce využívat pohyb vpřed a vzad při využití otočení na místě, pokud je možné se takto pohybovat. Dále je zde možnost pohybovat se i v různých jiných úlech (kromě 0°, 90°, 180°, 360°). Při pohybu v těchto úhlech se nevyužívají všechna kola (viz obrázek 2.2). Tento pohyb je tedy ještě více neefektivní, protože dvě aktivní kola musejí, zastat práci dvou neaktivních kol tak, že zdvojnásobí svoji rychlost, aby se robot pohyboval stejnou rychlostí jako vpřed nebo vzad. Neaktivní kola dále zvětšují tření vůči podložce, čímž zvětšují pravděpodobnost změny trasy robota. Se zvyšující rychlostí se odchylka ještě více zvětšuje. Výsledky pokusů diagonálního pohybu jsou znázorněny formou grafů ve zdroji [5] (viz obrázek 4.8) Zdroj [8] obsahuje experiment pohybu, z jehož výsledků lze vidět zdvojnásobení rychlosti dvou aktivních kol (viz obr. 4.9).

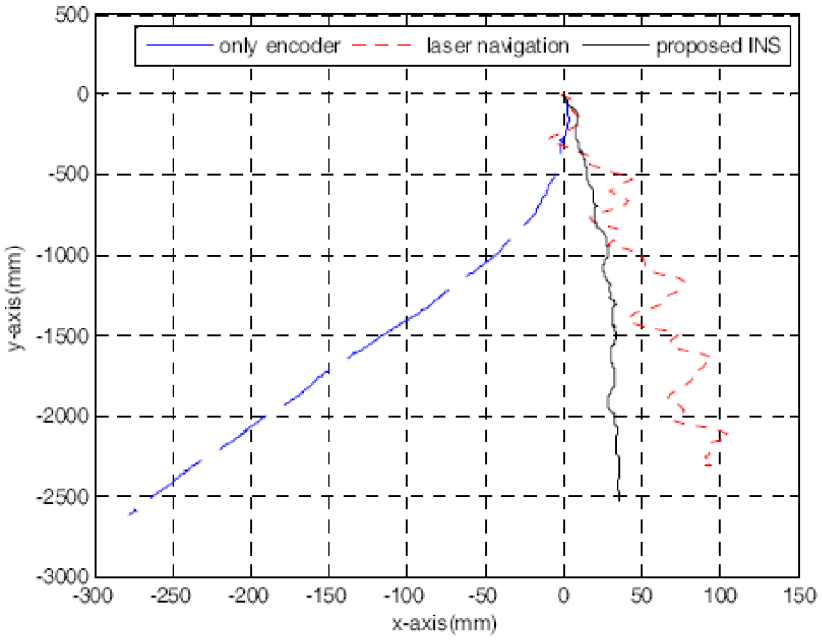
Direction	v_x	v_y	ω_z	**Wheel1**	**Wheel2**	**Wheel3**	**Wheel4**
Forward	5	0	0	4.87	4.87	4.87	4.87
Backward	-5	0	0	-4.87	- 4.87	- 4.87	- 4.87
Left	0	5	0	-4.87	4.87	4.87	-4.87
Right	0	-5	0	4.87	-4.87	-4.87	4.87
Left diagonal forward	5	5	0	0	9.74	9.74	0
Left diagonal	-5	5	0	-9.74	0	0	-9.74

Obrázek 4.6: Tabulka nastavení enkoderu pro směr jízdy

	forward slow	forward fast	backward slow	backward fast	right slow	right fast	left slow	left fast	right rotate slow	right rotate fast	left rotate slow	left rotate fast
experiment one energy cost (J)	323.3	338.5	320.5	341.2	344.7	385	346.7	376.7	323.9	337.6	320.5	337
experiment two energy cost (J)	323.8	339.5	320.6	341.3	346.3	384.2	348.3	377	324.1	337	321.7	336.9
experiment three energy cost (J)	324.9	342.7	321.1	341.2	347.2	384.8	349.3	377	323.2	337.3	322.1	337.6
model simulation energy cost (J)	320.98	340.85	320.98	340.85	346.72	385.84	346.72	385.84	323.25	335.47	323.25	335.47

	forward slow	forward fast	backward slow	backward fast	right slow	right fast	left slow	left fast	right rotate slow	right rotate fast	left rotate slow	left rotate fast
experiment one filtered (J)	18.02	33.22	15.22	35.92	39.42	79.72	41.42	71.42	18.62	32.32	15.22	31.72
experiment two filtered (J)	18.52	34.22	15.32	36.02	41.02	78.92	43.02	71.72	18.82	31.72	16.42	31.62
experiment three filtered (J)	19.62	37.42	15.82	35.92	41.92	79.52	44.02	71.72	17.92	32.02	16.82	32.32
model simulation filtered (J)	15.7	35.54	15.7	35.54	41.44	80.56	41.44	80.56	17.97	30.19	17.97	30.19

Obrázek 4.7: a)Ztráty energie při pohybu b) Ztráty energie při pohybu filtrované výsledky pokus



Obrázek 4.8: Graf diagonálního pohybu robota při rychlosti 200 mm/s

Na problematiku hledání optimální cesty lze nahlížet dále z pohledu prostoru. Při pohybu robot může narazit na rychlejší cestu ve formě přímého koridoru, nebo daný koridor obejít a získat výhodu ve formě větších manévrovací možností. Tato forma přístupu k optimálnosti cesty záleží na přesném určení a definování si cíle robota. Pokud robot funguje na bázi stabilního cíle, tak je rychlejší cesta lepší, ale pokud se cíle robota mohou v průběhu cestování měnit, tak je lepší, když robot využívá cesty s manévrovacími možnostmi.

Dalším důležitým faktorem optimální cesty je rychlost dosažení cíle. Rychlost cesty záleží na délce nalezené cesty. Konvenční modely robota musí řešit při plánování problémy limitů zatáčení a s tímto ohledem se plány cesty s nejkratší cestou nemusí považovat za nerychlejší kvůli situacím v prudkých zatáčkách a orientaci robota na cílové pozici (např. parkování do řady, mnoho malých posunů). Tato omezení se ovšem všesměrových robotů netýkají a plány cest s nejkratší vzdáleností k cíli se mohou označit jako nejrychlejší.

Kapitola 5

Použité technologie

V této kapitole budeme rozebírat technologie a algoritmy, které byly využity při tvorbě plánovače optimální cesty. Pro realizaci plánovače optimálních cest pro všesměrového robota jsme využili výše uvedených algoritmů a technologií. Vytvořený plánovač je vytvořen jako přídavný balíček pro operační sytém robotů (viz. kapitola 5.1). Pro testování optimálnosti cesty je využito několik algoritmů, které jsou popsány níže. Nad výsledky všech plánovacích algoritmů je použit další algoritmus, který určuje nejoptimálnější výsledek. Každý plánovací algoritmus vyhledává nejrychlejší cestu k dosažení cíle.

5.1 Robot Operating System

Robot Operating System (dále jen ROS) je flexibilní framework pro tvorbu softwaru pro roboty. Jedná se o kolekci knihoven, nástrojů a konvencí, které mají zjednodušit návrh a tvorbu funkcí chování pro široké množství robotů. ROS je bezplatný program pro volné využití, kde může každý uživatel využívat balíčky (knihovny) poskytnuté jinými uživateli.

Pro naše testování jsme využili ROS Kinetic, který je neaktuálněší. V rámci ROS využíváme systém pro správu balíčků *catkin*, který zjednodušuje překlad kódu. Dále v rámci ROS využíváme již vytvořené balíčky pro práci s mapami *map_server*, které při vstupu obrázku za pomocí hlavičkového souboru ve formátu *yaml* předají velikost mapy a vytvoří z ní bitmapu, která je nám dána k dispozici. Tento vstup poté pomocí zpráv převezme náš plánovač.

5.2 Použité algoritmy

Pro vytvoření plánovače optimálních cest jsme vytvořili několik algoritmů za pomocí výše uvedených technologii. Algoritmy potřebují několik vstupních parametrů a spuštěný *map_server* s mapu, ve které se pohybujeme. Vstupní parametry jsou dále již jen počáteční a koncový bod, zbylé parametry si program sám dohledá z hlavičkového souboru mapy. Počáteční a koncový bod se získá z grafického programu Rviz, jenž je součástí ROS.

5.3 Plánovač

Hlavní program zastřešuje získání mapy z *map_serveru* a spouštění jednotlivých algoritmů, které jsou níže popsané. Dále zprostředkovává předání startovní a cílové pozice z programu Rviz. Při spuštění programu se čeká na zaslání dat z programu Rviz. Jako první zpráva se

očekává *2D pose estimate* získána z topiku */initialpose*. Druhá zpráva musí být *navigation goal*. Tato zpráva z topiku */move_simple_base/goal* obsahuje cíl cesty robota. Po zadání těchto dvou argumentů již program vyvolá níže uvedené algoritmy a jejich výsledky porovná. Výsledná cesta je ta, která splnila více kritérií a je považována za optimální. Poté program čeká na další cíl, jeho aktuální pozice (cílová pozice) se pokládá za startovní. Program se ukončuje příkazem *CTRL-C*.

5.3.1 Algoritmus Potencionální pole

Algoritmus využívá technologii flood fill vysvětlenou výše. Poté funguje na principu výběru nejlepší hodnoty, kde využívá ještě heuristické funkce *distence*, která vybírá bod blíže k cíli v případě stejných hodnot, z nichž se vybírá další posun. Před samotným algoritmem proběhne flood fill spuštěné mapy z *map_serveru*. Dále proběhne kontrola, zda je v bodech okolo startovního bodu distribuovaná hodnota flood fill. Pokud žádný bod nenese hodnotu, program se ukončí s chybou, že cesta do cíle neexistuje.

Algoritmus postupuje těmito kroky:

1. Kontrola startovní a cílové pozice (test jestli se robot do daného prostoru vejde).
2. Ze startovní pozice vyber body, na které se lze posunout a ulož je do fronty open a startovní bod ulož to fronty close.
3. Vyber nejlepší bod z fronty open za pomocí hodnot polí a funkce *distance*.
4. Najdi z vybraného bodu všechny možné body, kam se lze posunout a které nejsou ve frontě close, ulož vyhovující body do fronty open a vybraný bod do fronty close a smaž jej z fonty open.
5. Pokud je fronta open prázdná tak se program ukončí, protože neexistuje cesta do cíle.
6. Pokud je další posun možný do cíle, ulož cílový bod do fronty close a ukonči vyhledávání a vrať frontu close jako výsledek hledání, pokud ne vrať se do bodu 3.

Algoritmus má dobré výsledky, nachází nejrychlejší spojení ze startu do cíle díky jednoduchému porovnávání hodnot, tzv. chamtivému přístupu (greedy). Jeho hlavní nevýhodou je nebezpečí uvíznutí v lokálním minimu.

5.3.2 Algoritmus A*

Algoritmus A* také využívá technologii flood fill, ale hodnota pole je zde vnímána jako hodnota přechodu, nebo hrany. Výpočet hodnotící funkce f(x)=h(x)+g(x) je pro pohyb horizontální a vertikální upraven na f(x)=g(x), kde g(x) je hodnota získána z flood fill nadané souřadnici, pro diagonální pohyb je hodnota h(x)=2. Algoritmus provede stejnou kontrolu startovního bodu jako u předchozího algoritmu. Dále algoritmus funguje podle výše uvedeného technologie v kapitole 4.1.1.1.

Algoritmus postupuje těmito kroky:

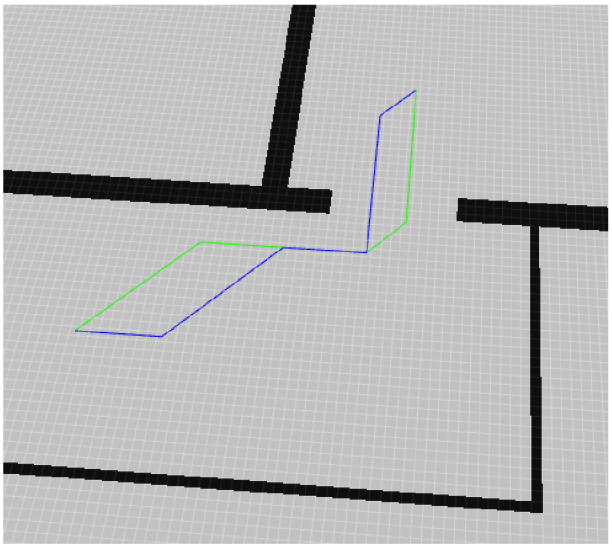
1. Kontrola startovní a cílové pozice (test jestli se robot do daného prostoru vejde).
2. Ze startovní pozice vygeneruj všechny možné posuny do front open.
3. Pokud je fronta open prázdná, ukonči vyhledávání, cesta neexistuje.

4. Vyber nejlepší bod ve frontě open a přesuň ho do fronty close.
5. Zkontroluj zda není poslední bod ve frontě close cíl, pokud je vrať výslednou cestu.
6. Vyber poslední bod v close a vygeneru všechny možné body posunu do fronty open, pokud již nesou ve froně open, nebo close.
7. Pokud se nenalezne žádný nový pohyb, smaž poslední bod z close a pokračuj bodem 4.
8. Pokud jsou nové body, pokračuj bodem 4.

Algoritmus má podobné výsledky jako předchozí algoritmus. A* má výhodu toho, že se nezasekne v lokálním minimu.

5.3.3 Porovnávací algoritmus

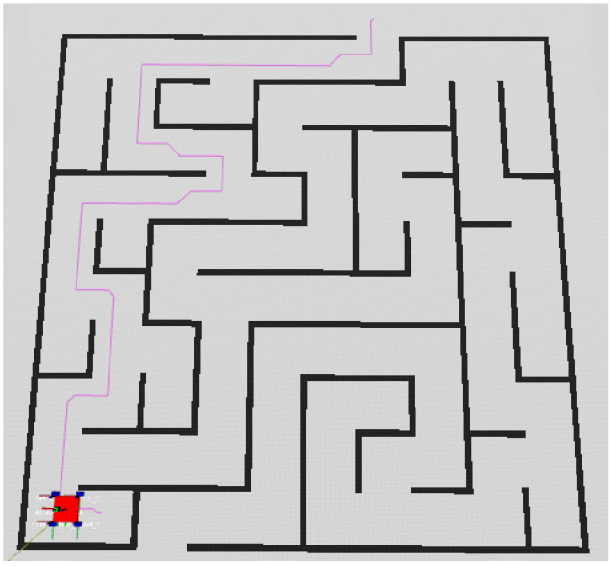
Tento algoritmus porovnává dvě výsledné cesty od výše uvedených vyhledávacích algoritmů. Tyto cesty se porovnávají podle tří parametrů. Prvním je počet posunů (počet polí od startu do cíle), dále se počítá v rámci efektivnosti počet otočení, protože efektivnost je větší při posunech vpřed, a posledním parametrem je počet možných posunů v každé pozici po cestě, čím více má plán cesty těchto možností, tím je větší variabilita. Cesta, které splňuje více parametrů, je považována za optimální.



Obrázek 5.1: Zobrazení výsledku algoritmu, modrá A* zelená Potencionální pole

5.3.4 Zobrazení cesty v Rviz

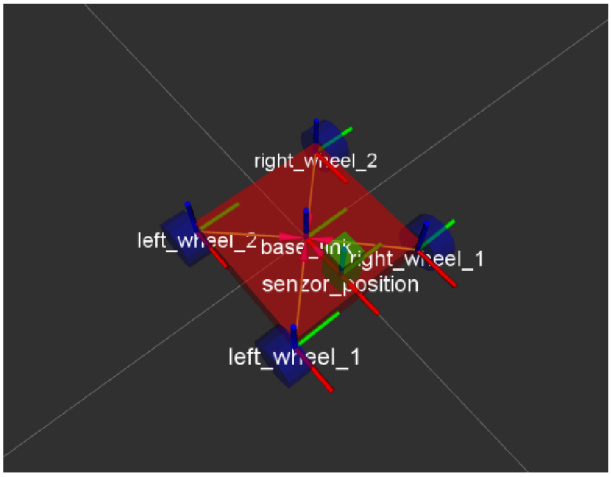
Optimální cesta je zobrazena za pomocí programu Rviz, který je součástí ROS. Cesta je poslána ve zprávě *nav_msgs/Path*. Program Rviz musí být nastaven na zobrazování cesty (topik *Path*) přidáním poslouchání zpráv z daného topiku */plan_cesty*. Po zobrazení je možno zadat znovu cílový bod a po dokončení algoritmů a zvolení optimální cesty se zobrazená cesta o novou část rozšíří.



Obrázek 5.2: Zobrazení cesty a robota v Rviz

5.3.5 Zobrazeni pohybu robota

Pro zobrazení robota je nutný model, tento model je vytvořen pomocí souboru URDF. URDF je založen na značkovacím jazyce XML. URDF je specifikován za pomocí stejnojmenného balíčku v rámci ROS.

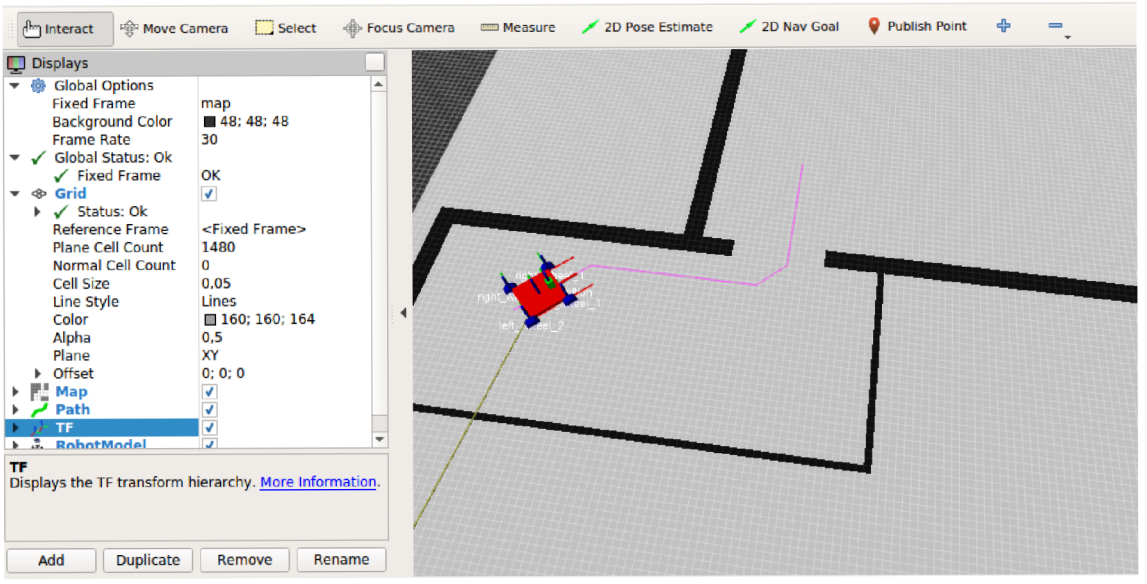


Obrázek 5.3: Model robota

Zobrazení robota v programu Rviz lze provést sledování dvou topiků *TF* a *Robot model*. Pouhým sledováním těchto topiků nezobrazíme robota, ale je nutno vytvořit transformace mezi zobrazenou mapou a robotem. Tyto transformace zajišťuje transformační server, jehož zdrojový kód je v souboru *robot_move_server.cpp*. Hlavní program posílá serveru zprávy a čeká na jeho odezvu o splnění požadavků. Zpráva pro server obsahuje, zda-li se jedná pouze o zobrazení robota na startovní pozici, rezoluci mapy (meter/pixel) a plán cesty, nebo bod

v případě startu. Poté server broadcastově vysílá jednotlivé transformace a tím se robot v programu Rviz začne pohybovat po zobrazené cestě.

Zobrazovací server také využívá informace získané z hodnocení optimálnosti, jedná se o informace možností otočení na místě. Pomocí těchto informací pak v rámci zobrazení pohybu se robot, pokud to lze, pohybuje čelem ke směru pohybu.



Obrázek 5.4: Simulace v programu Rviz

Kapitola 6

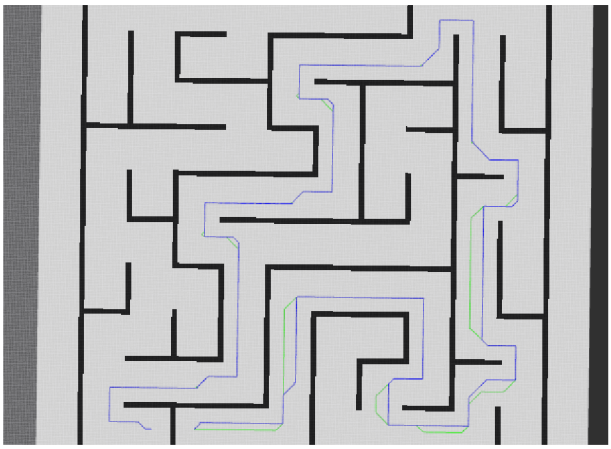
Testování

Testování bylo prováděno v rámci simulací v Rviz s použitím výše uvedených kódů. Byly použity dvě mapy, které jsou součástí *catkin* balíčku *bakalarska_prace* ve složce *map*. Jedná se o mapu *bludiste.png* a mapu *robot-test-map.jpg*. Mapa bludiště byla použita pro simulaci pohybu robota v úzkých prostorech a druhá mapa byla použita pro simulace pohybu robota v otevřených prostorech a místnostech.

6.1 Bludiště

Testovací mapa bludiště byla použitá pro otestování efektivnosti a rozdílnosti plánovacích algoritmů v úzkých prostorech.

První testy algoritmů, jež nebyly obohaceny o heuristické funkce, ukázaly, že algoritmus Potencionálních polí měl výsledky dobré, ale algoritmus A* se projevil jako nedostatečný v ohledu zbytečných pohybů. Robot se při A* pohyboval od strany ke straně, nevyužíval moc přímých pohybů, ale využíval převážně úhlopříčné pohyby. Po přidání heuristické funkce pro úhlopříčný pohyb se výsledky A* začaly podobat algoritmů Potencionálních polí. Následné testy ukázaly, že algoritmy jsou na stejné úrovni, ale jako optimální je častěji vyhodnocen algoritmus A*. Při pohledu na obrázek 6.1 je patrné, že algoritmy většinou využívají stejnou trasu, ale algoritmus Potencionálních polí (zelená čára) občas zbytečně prodlouží svou trasu.

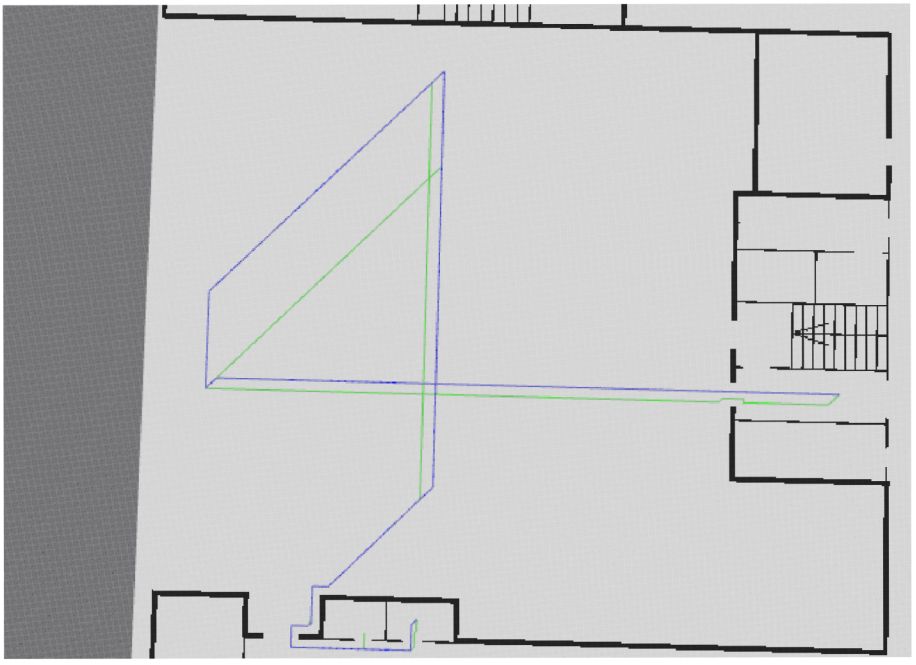


Obrázek 6.1: Test bludiště, modrá A*, zelená Potencionální pole

6.2 Otevřený prostor

Očekávanými výsledky testování v otevřeném prostoru byly hodnoty, které měly upřednostňovat algoritmus Potencionální pole kvůli jeho zpracování jako chamtivého (greedy) algoritmu. Testy probíhaly na mapě *robot-test-map.jpg*, jež obsahuje jak otevřený prostor, tak prostor místností.

Testování prokázalo účinnost heuristické funkce pro algoritmus A*. Díky této funkci se naše očekávání nevyplnila a výsledky testů prokázaly, že algoritmus A* i Potencionální pole mají velice podobnou efektivitu, ale algoritmus A* se vyvaroval zbytečným posunům a opět byl vyhodnocen častěji jako optimální. Obrázek 6.2 zobrazuje test pohybu z budovy přes prostranství do jiné části budovy. Při testu byly použity mezikroky (postupné určování cílů). Výsledky ukazují, že algoritmus Potencionálních polí se snaží dostat co nejdříve na stejnou hladinu v jedné ose jako cíl a to způsobuje jeho případné prodloužení trasy.

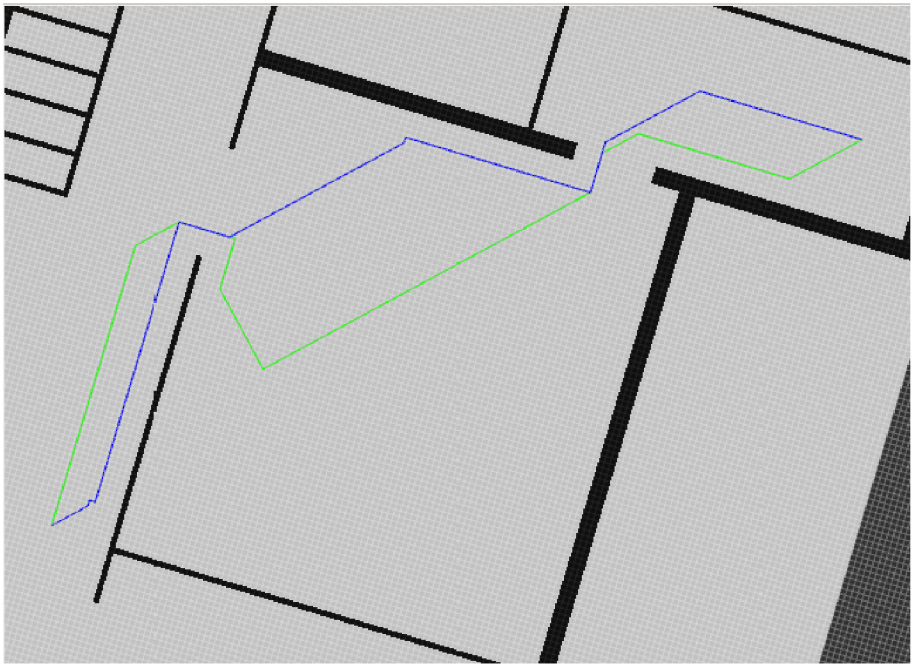


Obrázek 6.2: Test volné prostranství, modrá A*, zelená Potencionální pole

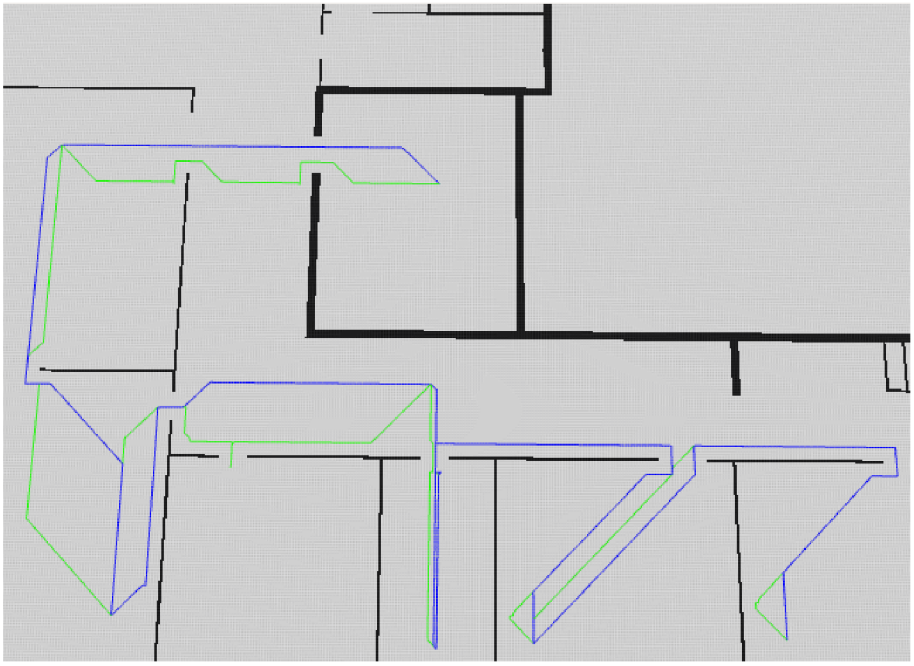
6.3 Místnosti

Testování v místnostech probíhalo na stejné mapě jako testování ve volném prostoru a to na mapě *robot-test-map.jpg*.

Přechozí testy prokázaly kvality algoritmus A* a testy v místnostech je dále potvrzují. A* je opět častěji vybrán jako optimální. Změna nastala v kvalitách algoritmu Potencionálních polí. Doposud se výsledky algoritmů lišily v malé míře, ale pokusy v místnostech jednoznačně ukázaly, že algoritmus Potencionálních polí nemá jen nebezpečí lokálních minim, ale také zbytečně prodlužuje svou trasu. Příkladem může být obrázek 6.3, a v menší míře i obrázek 6.4, který zobrazuje delší cestu s mezikroky do různých místností.



Obrázek 6.3: Test průchod místnostmi, modrá A*, zelená Potencionální pole



Obrázek 6.4: Test průchod místnostmi s mezi korky, modrá A*, zelená Potencionální pole

Kapitola 7

Závěr

Práce je zaměřená na všesměrovou mobilitu robotů s koly typu Mecanum a vytváření optimální cesty k dosažení cíle. Kola typu Mecanum dávají robotům výhody v rámci mobility a pohyblivosti v malých prostorech, ale při aplikaci kol tohoto typu na nerovném povrchu vznikají problémy prokluzování kol, problémy při zdolávaní kopců nebo hrbolů. Robot nemá možnost brzdit. Všechna tato negativa u kol typu Mecanum lze vyřešit za pomocí mechaniky (přidání odpružení a brzd na kolo a nutně i na všechny válce kola). V případě vyřešení těchto problémů by kola typu Mecanum mohla nahradit konvenční kola. Tato práce využívá vlastností kol Mecanum bez těchto vylepšení a na základě této situace je plánovač cest založen na informovaných algoritmech (máme mapu), protože se robot může efektivně pohybovat pouze po rovném povrchu a kola jsou využívána ve skladových prostorech (viz kapitola 2.5) a tyto prostory mají stavební a konstrukční plány, jichž lze využít jako zdroj informací pro plánovač cest. Plánovač cest je vytvořen v Robot Opreating Systém (ROS). ROS je obsahuje velké množství funkcí a podpůrných prostředků pro vytváření programů pro roboty. Náš program plánovače cest využívá tyto podpůrné prvky pro zobrazení výsledků a simulaci pohybu robota. Testování probíhalo v ROS a za pomocí simulací pohybu (viz. kapitola 6). Plánovač má implementované dva informované algoritmy, A* a Potenciální pole (viz. kapitola 4.1.1). Algoritmus Potencionální pole byl vylepšen o funkci rozhodování mezi stejnými hodnotami. Při testech mezi těmito algoritmy bylo dosaženo několika výsledků. Algoritmus A* a algoritmus Potencionálních polí mají podobné výsledky, ale A* se nezasekává v lokálních minimech a zbytečně nenatahuje trasy. V průběhu testování byl použit ještě algoritmus prohledávání do šířky (BFS), ale byl velice pomalý a náročný na paměť, v otevřených prostorech byl naprosto nevyhovující. Testy pro optimální cestu byly vytvořeny podle kapitoly 4.2. Mezi vytvořenými algoritmy se porovnávaly dílčí části a to rychlost, energetická efektivita a mobilnost. Dílčím částem byla dána stejná váha a za optimální cestu jsme považovali cestu s více dílčími částmi (minimálně dva ze tří). Testy prokázaly, že algoritmus A* je vítězem v porovnání s ostatními použitými algoritmy při testování optimálnosti.

Tato práce se dá dále rozvíjet v rámci lokalizace. Aktuálně použitá lokalizace je odometrie (viz. kapitola 3.2.2). Pokud by se odometrie vylepšila o použití i neinformovaných algoritmů (viz. kapitola 4.1.2) pro zlepšení určení pozice za použití senzorů na robotovi, dosáhlo by se přesnějšího dosažení bodů cesty optimální cesty. Přesnost kol Mecanum při použití odometrie je menší s ujetou vzdáleností a rychlostí. Dále by bylo vhodné aplikovat plánovač na robota mimo simulaci a provést testování v reálném prostoru pro odhalení dalších problémů, omezení a jejich následnému vyřešení.

Tato práce odhalila několik omezení kol typu Mecanum. Dále poukázala na omezení při využití odometrie a pomocí simulace na vhodnost algoritmu A* pro použití plánování cesty. Práce rozvinula problematiku optimální cesty, námi rozdělenou na dílčí části rychlosti, energie a mobility. Dali jsme další podměty pro rozvoj této práce a všesměrové robotiky, aby dosáhla lepších výsledků.

Literatura

[1] Adăscăliţei, F.; Doroftei, I.: *Practical Applications for Mobile Robots based on Mecanum Wheels - a Systematic Survey.* [Online; navštíveno 19.12.2016].
URL http://www.incdmtm.ro/mecahitech2011/articole/Pp112-123.pdf

[2] Choset, H.; Lynch, K. M.; Hutchinson, S.; aj.: *Principles of Robot Motion: Theory, Algorithms, and Implementations.* A Bradford Book, 2005, ISBN 978-0262033275.

[3] Doroftei, I.; Grosu, V.; Spinu, V.: *Omnidirectional Mobile Robot - Design and Implementation.* Září 2007, [Online; navštíveno 11.12.2016].
URL https://www.intechopen.com/books/bioinspiration_and_robotics_walking_and_climbing_robots/omnidirectional_mobile_robot_-__design_and_implementation

[4] Kasaei, S.; Kasaei, S.; Kasaei, S.; aj.: *Modeling and Implementation of Omni-directional Soccer Robot with Wide Vision Scope Applied in Robocup-MSL* . [Online; navštíveno 19.12.2016].
URL http://dspace.unimap.edu.my/dspace/bitstream/123456789/7308/1/Modeling%20and%20Implementation.pdf

[5] Kim, J.; Woo, S.; Kim, J.; aj.: *Inertial Navigation System for an Automatic Guided Vehicle with Mecanum Wheels* . [Online; navštíveno 24.1.2017].
URL http://link.springer.com/article/10.1007/s12541-012-0048-9

[6] Stentz, A.: *Optimal and Efficient Path Planning for Partially-Known Environments* . [Online; navštíveno 11.12.2016].
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=351061&tag=1

[7] Sudhagar, K.; Sudhagar, K.; RajaRajeswari, G.: *Intelligent Path Planning Of Mobile Robot Agent By Using Breadth First Search Algorithm* . [Online; navštíveno 11.12.2016].
URL https://www.rroij.com/open-access/intelligent-path-planning-of-mobile-robotagent-by-using-breadth-first-search-algorithm.pdf

[8] Taheri, H.; Qiao, B.; Ghaeminezhad, N.: *Kinematic Model of a Four Mecanum Wheeled Mobile Robot.* [Online; navštíveno 11.12.2016].
URL http://research.ijcaonline.org/volume113/number3/pxc3901586.pdf

[9] Vadakkepat, P.; Tan, K. C.; Ming-Liang, W.: *Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning* . [Online; navštíveno 11.12.2016].
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=870304

[10] Xie, L.; Herberger, W.; Xu, W.: *Experimental validation of energy consumption model for the four-wheeled omnidirectional Mecanum robots for energy-optimal motion control*. [Online; navštíveno 21.12.2016].
URL http://ieeexplore.ieee.org/document/7496410/

Příloha A

Obsah přiloženého paměťového média

- Catkin balíček "bakalarska_prace", jenž obsahuje všechny kódy a součásti pro běh a simulaci bakalářské práce.
- Text této bakalářské práce.
- Manuál pro spuštění a ovládání programu bakalářské práce.