VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Simulace lokalizačního algoritmu nemocničního robotu

Localization Algorithm Simulation for Medical Robot

2018 Adam Vidiševský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Adam Vidiševský**

Studijní program: N2649 Elektrotechnika

Studijní obor: 3901T009 Biomedicínské inženýrství

Téma: Simulace lokalizačního algoritmu nemocničního robotu
Localization Algorithm Simulation for Medical Robot

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Přehled senzorů používaných pro lokalizaci mobilních nemocničních robotů.
2. Přehled existujících lokalizačních metod a algoritmů vhodných pro nemocniční prostředí.
3. Přehled prostředků pro tvorbu simulací (C, C++, C#, Matlab).
4. Návrh lokalizačního algoritmu založeného na evolučním algoritmu a vzájemné korelaci.
5. Implementace lokalizačního algoritmu a jeho optimalizace.
6. Testování navrženého řešení z hlediska výpočetní náročnosti a úspěšnosti zarovnání.
7. Zhodnocení dosažených výsledků závěrečné práce.

Seznam doporučené odborné literatury:

[1] KONEČNÝ, Jaromír. *Principy řízení mobilních servisních robotů.* Ostrava, 2014. Disertační práce. VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra kybernetiky a biomedicínského inženýrství.
[2] GE, Shuzhi S. and Frank L. LEWIS. *Autonomous mobile robots.* Boca Raton: CRC Press, 2006. 736p. ISBN 978-0849337482.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jaromír Konečný, Ph.D.**

Konzultant diplomové práce: doc. Ing. Marek Penhaker, Ph.D.

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2018

.....

Bc. Adam Vidiševský

Rád bych na tomto místě poděkoval svému vedoucímu Ing. Jaromíru Konečnému, Ph.D. za jeho vřelé rady, připomínky a ochotu pomoci, a taktéž doc. Ing. Pavlu Krömerovi, Ph.D. za jeho trpělivost pro mé necitlivé zásahy do jeho kódu, bez nich by tato práce nikdy nevznikla.

Abstrakt

Tato diplomová práce se věnuje metodám lokalizace mobilních nemocničních robotů pomocí zarovnání laserových snímků, simultánní lokalizaci a mapování (SLAM). V úvodu práce jsou představeny a rozděleny již existující metody pro zarovnání snímků. Cílem této diplomové práce je návrh nové metody pro lokalizaci nemocničního robota, za použití evolučního algoritmu, ve spojení se vzájemnou korelací. Jako evoluční algoritmus byla zvolena diferenciální evoluce. Základem metody je využití vytváření nových populací a vyvíjení nových generací diferenciální evoluce pro zarovnání snímků. Vhodnost mutačního kandidáta z dané populace je posouzená pomocí vzájemné korelace, určením korelačního koeficientu pro třídimenzionální korelaci. V další části práce jsou provedena experimentální měření a porovnání s existujícím algoritmem. Experimenty mají za úkol ukázat robustnost metody a výpočetní náročnost.

Klíčová slova: evoluční algoritmus, diferenciání evoluce, vzájemná korelace, zarovnání snímku, nemocniční robot

Abstract

This diploma thesis concerns itself with localization methods of mobile medical robots using laser snapshot, simultaneous localization and mapping (SLAM). At the beginning of this paper various existing methods for snapshot alignment are introduced and categorized. The goal of this diploma thesis is to propose a new method for medical robot localization using evolution algorithm in connection with cross-correlation. Differential evolution was selected as an evolution algorithm. This method is based on creation of new populations and evolution of new generations of differential evolution for the purpose of snapshot alignment. Plausibility of the mutation candidate from a given population is evaluated using cross-corelation, correlation coefficient for three-dimensional correlation. Subsequently, a number of experimental measurement is conducted to show method robustness and computational complexity compared to an existing (benchmark) algorithm.

Key Words: evolution algorithm, differential evolution, correlation, snapshot alignment,mobile medical robot

Obsah

Seznam použitých zkratek a symbolů

LIDAR	–	Light Detection And Ranging
ICP	–	Iterative Closest Point
NDT	–	Normal Distributions Transform
APR	–	Anchor Point Extraction
PSM	–	Polar Scan Matching
CLS	–	Complete Line Segment
HSM	–	Hough Scan Matching
DHS	–	Discrete Hough Spectrum
DHT	–	Discrete Hough Transform
DE	–	Differential Evolution
CR	–	Crossover Rate

Seznam obrázků

Seznam tabulek

1 Úvod

Zvyšující pokrok techniky v oblasti robotiky má za následek i větší poptávku v různých odvětvích. Již od šedesátých let se mobilní roboti začali využívat pro vojenské účely. A i z těchto důvodů bylo potřebné určit lokalizaci robota. Bylo jen otázkou času, kdy zvyšující se výkon senzorů lokalizačních algoritmů umožní použití mobilních robotů i v nemocnicích.

Nemocnice Košice-Šaca a.s. ve spolupráci s Technickou univerzitou v Košicích, Katedrou eletrotechniky a mechatroniky, uvedly do provozu mobilního robota nové generace, který přebere část práce zdravotnického personálu. Pathfinder je unikátní autonomní logistický robot nové generace určený pro transport materiálu a léků z interní lékárny na jednotlivá oddělení v medicínském prostředí. Kromě léků a materiálu dokáže rozvážet i stravu. Převézt může najednou až 60 kilogramů materiálu [28]. Tvůrcem robota je Ing. Ján Bačík, PhD. z Technické univerzity. Jedinečnost systému spočívá v jeho unikátním navigačním systému, který se podobá navigačním systémům v autonomních vozidlech. Pro svou navigaci využívá data z více senzorů, které jsou umístěny na jeho palubě. Tato data následně robot zpracuje a vyhodnotí svou pozici v rámci budovy ve virtuální mapě. Největší výhodou tohoto systému je fakt, že se při implementaci robota a jeho používání vše děje virtuálně a nejsou potřebné žádné zásahy do infrastruktury nemocnice, jakými jsou např. umístění indukčních vedení v podlaze, či lepení různých typů vizuálních navigačních čar. Pokud chce obsluha změnit trasu robota, jednoduše si ji překreslí v mapovém editoru [28].

Pathfinder se umí pohybovat sám. Nejprve si vytvoří mapu prostředí, kterou si načte, a vypočítá si trasu, po které má jít. Dokonce pokud je před ním překážka, sám se zastaví a počká pár sekund a až potom pokračuje v cestě.

Pathfinder přispěje k optimalizaci interních procesů prostřednictvím automatizace nemocniční logistiky. Tento unikátní transportní systém bude pomáhat zdravotnickému personálu při převozu materiálu mezi jednotlivými pracovišti nemocnice. Zdravotníci tak získají více prostoru pro ty, kteří jejich péči nutně potřebují, pro pacienty nemocnice [29].

Pathfinder je autonomní naváděné vozidlo, pohybuje se samo. Na podběhovém podvozku má platformu, tedy prostor pro transportovaný materiál. Robot měří cca 130 cm a váží okolo 85 kg. Pathfinder je bílé barvy, aby v nemocnici nepůsobil rušivě, a během jízdy hraje příjemnou akustickou hudbu (takovýmto způsobem ho lidé na chodbách nepřehlédnou). Ovládá se pomocí aplikace, kterou mají zaměstnanci nainstalovanou v tabletu či mobilním telefonu, nebo přímo na displeji robota. Personál v lékárně otevře zamčené dvířka, naloží požadované léky a zadá, kam má robot jet. Když se robot dostane na zadané oddělení, pošle zprávu personálu, že je přede dveřmi. Zaměstnanec může udělat dvě věci - otevře dvířka a vybere materiál nebo Pathfindera pomocí tabletu či mobilu navede přímo na oddělení. S robotem komunikuje výhradně personál z důvodu zabránění záměny či zneužití přístupu k léčivům. Robot nechodí přímo k pacientům.

Na Slovensku je Pathfinder jediným exemplářem. Jeho inovativní stránkou je, že pro svoji činnost nevyžaduje žádný zásah do infrastruktury nemocnice. Podobný robot se nachází v praž-

ské Fakultní nemocnici v Motole, ale ten funguje na principu magnetických pásek nalepených na zemi. Pathfinder nic takového nepotřebuje - pomocí laserů si udělá mapy, po kterých se bude pohybovat [29].

Dalším využitím nemocničního mobilního robota je v Protonovém centru v Praze. Zde lůžka pacientů sledují pomocí infračervené kamery naváděcí čáry, a pomocí nich jsou pacienti vozeni na ozařovací procedury. Prozatím jsou lůžka provázena nemocničním personálem, nicméně zvyšující se pokrok umožní dopravení lůžka i bez potřebného personálu.

2 Přehled senzorů používaných pro lokalizaci mobilních nemocničních robotů

Mobilní nemocniční roboti pro kontakt s okolním světem využívají senzory, které dodávají robotu data, které pak následně zpracovávají. Senzory lze rozdělit podle vztahu k robotu na interní, měřící parametry robotu a externí, měřící parametry okolí robotu. Interní senzory poskytují robotu informace o jeho subsystémech [14]. Pro účely navigace jsou to informace o akčním subsystému, což jsou obvykle poloha a rychlost jednotlivých pohonů. Externí senzory se dělí na aktivní a pasivní. Pasivní snímače využívají pouze okolní záření, zatímco aktivní mají své vlastní zdroje světla [13].

2.1 Interní senzory

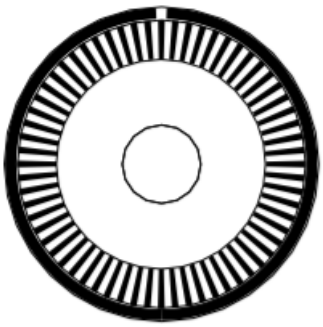
Interní senzory pro určení polohy nebo rychlosti řízení se připojují na hřídel motoru. Dle metody měření se interní senzory dělí na inkrementální a absolutní. Inkrementální senzory jsou založené na principu otáčivého mezikruží s pravidelně se střídajícími průhlednými a neprůhlednými ryskami, které při otáčení přerušují emitované světlo LED diody umístěné na jedné straně mezikruží. Metoda, která určuje polohu a pozici robota z otáček kol a motoru se nazývá odometrie. Používaným zařízením pro tento účel jsou enkodéry, které jsou připojeny ke kotvě motoru, nebo k nápravě kol. Aktuální umístění robota se spočítá z předchozích pozic a průběžné rychlosti [13]. Jako enkodéry se používají tyto senzory:

1. Optické enkodéry
2. Potenciometry
3. Magnetické enkodéry
4. Indukční enkodéry
5. Kapacitní enkodéry

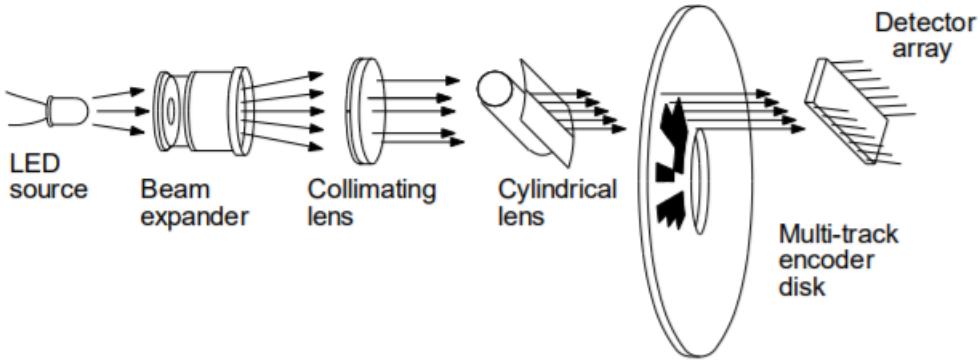
Základním principem enkodéru je měření přítomnosti či nepřítomnosti nějakého dobře detekovatelného materiálu na otáčejícím se kolečku. Nejčastěji se pro odometrii používají optické enkodéry. Optické enkodéry jsou inkrementální a absolutní.

2.1.1 Optické inkrementální enkodery

Nejčastější konstrukční řešení optických inkrementálních enkoderů je použití kolečka, které má na svém povrchu rovnoměrně vyvrtané otvory. Příklad takového kolečka je na obrázku 1. Kolečko se otáčí mezi zdrojem světla a přijímačem. V případě jednokanálového enkodéru je jeden světelný paprsek fokusován na přijímač – fotodetektor. Při otáčení kolečka dochází k přerušování světelného signálu [30]. Dalším možným konstrukčním řešením je umístění zdroje světla



Obrázek 1: Disk používaný pro optický inkrementální senzor



Obrázek 2: Provedení absolutního optického enkodéru

i přijímače na jedné straně kolečka, na které jsou reflexní matné plošky. Jejich střídání, před přijímačem při otáčení kolečka, na něm způsobuje změny napětí v důsledku změny množství detekovaného světla. Na přijímači je naměřen obdélníkový signál, kde každý obdélník odpovídá jednomu otočení o shodný počet stupňů [30]. Pro určení směru se používá enkodér dvoukanálový. Na kolečku je umístěn ještě jeden senzor, který je namontován tak, aby jeho signál byl s původním fázově posunut o 90°.

2.1.2 Absolutní optické enkodéry

Inkrementální enkodéry detekují změnu polohy o pevně daný krok. Pro zjištění přesné polohy v rámci 360° se používají optické enkodéry absolutní. Používají se pouze pro pomalejší otáčky koleček, aby docházelo ke ztrátám co nejméně. Pořadí jednotlivých částí detektoru je na obrázku 2. Pro každý bit je jinak velký otvor v kolečku, což má negativní vliv na výslednou velikost disku. Informace je zakódovaná a je potřebný počet snímačů dle požadovaného rozlišení. Na rozdíl od inkrementálního sériového bitového výstupu, je zde výstup paralelní, což umožňuje zakódovat celé slovo [13].

2.2 Externí senzory

Externí senzory slouží k získání informace o okolí robotu. Externí senzory lze dělit na pasivní, které vyhodnocují pouze přijaté záření z okolí, a aktivní, které vyhodnocují vlastní odražené záření.

2.2.1 Pasivní snímače

Nejvyužívanější pasivní snímače jsou CCD, CMOS a infračervené kamery. Snímací čipy CCD (Charged Coupled Device) se skládají z mnoha světlocitlivých buněk, které při reakci na světlo produkují elektrický signál. Čím více světla dopadne, tím větší náboj vznikne. Data jsou čtena po řádcích. Mimo samotné světlocitlivé buňky čipu je posuvný registr, kam se nejprve přesune náboj z prvního řádku. Ten projde přes zesilovač do A/D převodníku, ze kterého vycházejí digitální data. Poté se všechny řádky přesunou, a do posuvného registru se načte další řádek. Tímto principem jsou pak následně načteny všechny řádky. CMOS čipy jsou tvořeny maticí fotodiod, kde každá buňka má svůj zesilovač a obvod odstraňující šum. Výhodou tohoto přístupu je okamžitý přístup k jednotlivým blokům snímače, takže lze zobrazit pouze oblast zájmu.

Infračervené kamery pracují na odlišném principu než CCD a CMOS snímače. Infračervená energie vyzařovaná tělesy zahřívá prvky v kameře. Tyto prvky pak převádějí teplo v elektrický signál. Infračervené kamery mají oproti normálním kamerám horší rozlišení a pomalejší odezvu [1].

2.2.2 Aktivní snímače - LIDAR

Nejrozšířenějším senzorem v mobilní robotice je bezesporu LIDAR, což je aktivní senzor s laserovým skenovacím zařízením. Princip LIDARu spočívá ve vysílání laserových paprsků. Vyslané paprsky se šíří prostorem a při nárazu na překážku se odráží a vrací zpět. Vzdálenost překážky je určena z doby letu laserového paprsku (Time Of Flight), což umožňuje pořizovat velmi kvalitní prostorová data během krátké doby měření. Skenery lze rozdělit do dvou kategorií, a to na tzv. 2D a 3D skenery. U 2D skenerů je laserový paprsek vychylován pouze v jedné rovině, obvykle v rozsahu do 180 stupňů. Čas skenu trvá 13 sekund při úhlovém rozlišení 1 stupeň [1]. U 3D skenerů je laserový paprsek vychylován do celého zorného pole skeneru.

2.2.3 Aktivní snímače - Ultrazvukové senzory

Ultrazvukové senzory, popřípadě sonary, využívají k měření vzdálenosti k překážce měření doby mezi vysláním akustického signálu a přijetím odraženého akustického signálu. Generátorem ultrazvuku u senzorů je piezoelektrický měnič, jehož základem je piezoelektrický krystal, který dokáže měnit elektrickou energii na mechanickou a naopak. Ultrazvukové senzory poskytují horší data než laserové senzory. Jejich nevýhodou je totiž vysoké tlumení ultrazvukového signálu, což omezuje praktický dosah jen na několik desítek metrů. Vzhledem k poměrně širokému

rozptylu nelze překážku detekovat zcela přesně [14]. Sonar se na robotu umísťuje v přední části robotu. Orientovány jsou tak, aby měřili prostor v rozsahu 180°. V takovém případě robot získá informaci dříve, než pokud by sonar byl umístěn po obvodu robotu. V článku [2] autoři používají na získaná data rozšířený Kalmanův filtr na určení pozice.

3 Přehled existujících lokalizačních metod a algoritmů vhodných pro nemocniční prostředí

V této kapitole jsou popsány základní lokalizační metody používané v mobilní robotice. Lokalizace je proces zjišťování pozice v prostoru z dostupných senzorických dat. Na lokalizaci robotu lze nahlížet jako na problém transformace souřadnic mezi dvěma souřadnými systémy. Robot během navigace a pohybu v tomto postředí vytváří mapu a lokalizuje se v ní. Princip takovéto lokalizace se označuje jako SLAM (Simultaneous Localization and Mapping).

3.1 Algoritmus ICP

ICP (iterative closest point) je iterativní algoritmus, který hledá nejbližší dvojice bodů ve dvou snímcích. Vstupem pro ICP algoritmus jsou dva snímky, referenční a testovací. Výstupem algoritmu je translace a rotace referenčního snímku vůči testovacímu. Algoritmus vytváří dvojice bodů na snímcích a hledá nejbližší body na základě minimalizace rozdílů sumy čtverců. Průběh algoritmu běží, dokud nejsou splněné určité podmínky. Algoritmus je proto výpočetně a časově náročný [3].

Algoritmus lze shrnout do několika kroků:

1. Předzpracování
2. Přiřazení
3. Odmítnutí
4. Určení hodnoty cílové funkce
5. Minimalizace cílové funkce

3.2 Algoritmus NDT

Algoritmus NDT (Normal Distributions Transform) rozděluje skenovací prostor do mřížek pravidelných intervalů, kde každá mřížka je aproximovaná normálním rozdělením. Obrovskou výhodou algoritmu NDT je, že je rychlejší, nepotřebuje totiž hledat pro aktuální snímek žádné referenční mřížky. Rychlost konvergence závisí na velikosti mřížky [4].

Algoritmus funguje následovně:

1. Skenovací prostor je rozdělen na stejně velké mřížky
2. Všechny body $q_i = 1..N$, jsou umístěny do svých příslušných buněk
3. Vypočtení těžiště bodů

$$c = \frac{1}{N} \sum x_q \ , \tag{1}$$

kde N představuje počet bodů v buňce

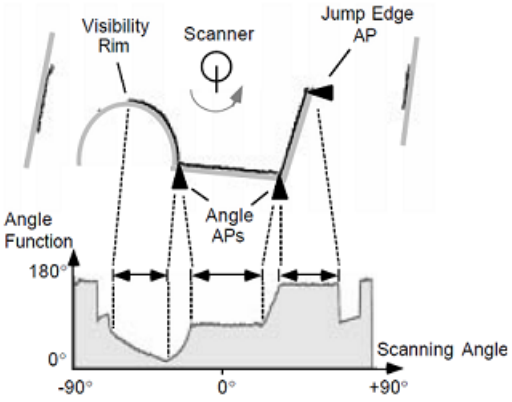
4. Vypočtení kovarianční matice

$$C = \frac{1}{N-1}\sum(q_i - c)(q_i - c)^T \ , \tag{2}$$

5. Určení pravděpodobnosti jedné buňky

$$P = exp\frac{-(p_i - c)^T C^{-1}(p_i - c)}{2} \ , \tag{3}$$

Posledním krokem, po určení normální distribuce všech buněk ve snímku, je zarovnání snímku.



Obrázek 3: Přehled kotevních bodů

3.3 Algoritmus APR

Ve vzniklých snímcích lze pozorovat určité opakující se tzv. kotevní body. Metoda APR je založená na vytvoření sady referenčních snímků a určení kotevních bodů v těchto snímcích. Nový laserový snímek se pak porovnává s touto sadou na základě vybraných kotevních bodů. Na rozdíl od jiných metod, algoritmus APR nevyžaduje informace o aktuální poloze robota [5].

Jako kotevní body se používají:

1. Skokové rohy (Jump Edge Anchors)
2. Úhlové rohy (Angle Edge Anchors)
3. Virtuální rohy (Virtual Edge Anchors)

Skokové rohy se objevují mezi dvěma rovinnými svislými plochami, popřípadě předměty, z nichž jedna rovina je blíže senzoru. Signál senzoru na kraji takovéto plochy „poskočí“ na vzdálenější předmět. Skokové rohy jsou ve snímcích nejsnáze detekovatelné. Jako úhlové rohy se označují průsečnice dvou rovin. Virtuální rohy se určují z histogramů. Nejprve se vytvoří poziční

histogram obou souřadnic, x a y. Poté se v histogramech vyhledají maximální špičky. Jako virtuální roh se nazývá maximum, které se vyskytuje v obou pozičních histogramech na stejném místě. Virtuální kotevní body jsou stabilní a snadno rozpoznatelné v ortogonálních prostředích. Jednotlivé kotevní body jsou znázorněny na obrázku 3.

3.4 Metoda PSM

Metoda PSM (Polar Scan Matching), navržená Albertem Diosim [6] pro srovnání snímků, využívá projekci aktuálního snímku do referenční soustavy souřadnic. PSM je rychlejší než metoda ICP, protože odpadá časově náročné vyhledávání stejných bodů. Metoda srovnává snímky tak, aby reziduální součet čtverců referenčního a aktuálního snímku byl minimální. Metoda sestává z těchto kroků:

1. Předběžné zpracování
2. Projekce snímku
3. Odhad translace
4. Odhad orientace

Při předběžném zpracování jsou odstraňovány hodnoty pohyblivých předmětů a hodnoty mimo rozsah. Jsou odstraněny i hodnoty, které odpovídají nohám židlí a stolů, protože z dlouhodobého hlediska lze očekávat, že nebudou statické. Některé předměty jako třeba sklo, které špatně odráží laserový signál, pak vykazují hodnoty mimo rozsah. Odstraňování hodnot je prováděno mediánovým filtrem. Aplikací mediánovéhu filtru na celý rozsah naměřených hodnot jsou odstraněny i nohy židlí a stolů. Dosahuje se toho nastavením velikosti okna medianového filtru. Při projekci snímku se promítá aktuální snímek do souřadnic referenčního snímku. Cílem je určit, co by naměřil laserový snímač v referenční pozici. Projekce je určena:

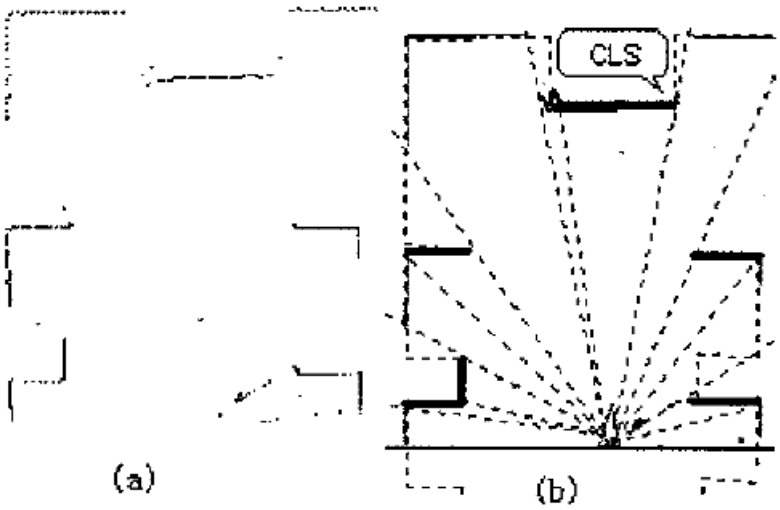
$$r'_{ci} = \sqrt{(r_{ci}\cos(\theta_c + \phi_{ci}) + x_c)^2 + (r_{ci}\sin(\theta_c + \phi_{ci}) + y_c)^2} \ , \tag{4}$$

$$\theta'_{ci} = atan2(r_{ci}\sin(\theta_c + \phi_{ci}) + y_c, r_{ci}\cos(\theta_c + \phi_{ci}) + x_c) \ , \tag{5}$$

kde r'_{ci}, θ'_{ci} jsou přepočítané polarní souřadnice do referenční soustavy, x_c, y_c a θ_c je pozice a orientace v aktuálním snímu, r_{ci}, ϕ_{ci} jsou naměřené hodnoty laserového skeneru v polárních souřadnicích. Přepočítány jsou pouze vzorkovací body, ostatní body r''_{ci} jsou určeny interpolací. Po projekci snímku je známé r''_{ci} a r_{ri}. Cílem je nalézt polohu snímku tak, aby reziduální součet čtverců byl minimální:

$$\sum w_i(r_{ri} - r''_{ci})^2 \ , \tag{6}$$

Změna orientace aktuálního snímku je v polárních souřadnicích znázorněna posunem vlevo nebo vpravo od naměřených hodnot. Potom, co je známá správná lokace a snímky obsahují stejné



Obrázek 4: a) nasnímaný snímek b) vytvoření čárových segmentů ve snímku

statické předměty, je správná orientace nalezena posouváním snímku, dokud nepokryje referenční snímek.

3.5 Metoda CLS

Metoda CLS (Complete Line Segment) patří do skupin metod založených na význačných rysech. Ve snímcích jsou vyhledávány čárové segmenty, podle kterých jsou pak snímky srovnávany. Z aktuálního snímku robota je vytvořena tzv. lokální mapa, která se pak zarovnává do mapy globální. Z relativní polohy těchto dvou map je pak určena pozice robota [7].

V lokální mapě jsou rozlišovány čárové segmenty úplné a neúplné. Jako neúplný čárový segment může být například zeď, před kterou je nějaká překážka. Pouze úplné čárové segmenty jsou použité pro lokalizaci. Čárové segmenty jsou řazeny podle polohy ve snímcích proti směru hodinových ručiček, a jsou uloženy pod sekvenčním číslem. Metoda využívá faktu, že stejný čárový segment má z jakékoliv polohy robota v místnosti pořád stejnou velikost. Po nalezení všech čárových segmentů v aktuálním snímku se vyhledává čárový segment v globální mapě, který má stejnou velikost. Takto se vytvoří několik párů čárových segmentů. Vytvoření čárových segmentů je znázorněno na obrázku 4.

Poté se určuje pravděpodobnost lokalizace dle uložení jiných čárových segmentů. Aby odpadla časově náročná transformace, využívá se k porovnávání relativní umístění, to je dáno středovým bodem, orientací a délkou čárových segmentů. K umístění lokální mapy do globální se pak použije nejpravděpodobnější lokalizace. Mezi dvěma čárovými segmenty a jejich středovými body se v lokální mapě vytvoří tzv. lokalizační vektor. Orientace lokální mapy vůči mapě globální se určuje z vektoru mezi jednotlivými čárovými segmenty. Čárové segmenty z lokální mapy jsou pak zařazeny do mapy globální. Po určení orientace lokální mapy je určena pozice

robota pomocí rovnic:

$$X = x_{\mathrm{G}} - (x_{\mathrm{C}} \cos\theta + y_{\mathrm{C}} \sin\theta) \ , \tag{7}$$

$$Y = y_{\mathrm{G}} - (y_{\mathrm{C}} \cos\theta - x_{\mathrm{C}} \sin\theta) \ , \tag{8}$$

kde x_{C}, y_{C} je pozice středového bodu čárového segmentu v lokální mapě,
x_{G}, y_{G} je pozice středového bodu čárového segmentu v globální mapě a
X, Y je pozice robota.

3.6 Metoda Monte Carlo

Lokalizace Monte Carlo je stochastická metoda navržená v sedmdesátých letech. Tento algoritmus určuje lokalizaci robota pomocí částicového filtrování. Ve vytvořené mapě ze snímku určuje algoritmus pozici a orientaci robota dle jeho pohybů a nasnímaného prostředí. Algoritmus využívá částicový filtr, kde každá částice reprezentuje pravděpodobnost jednotlivých stavů, respektive hypotézu, kde se robot vyskytuje. Na začátku algoritmus začíná s jednotným náhodným rozdělením částic na celý konfigurační prostor, což znamená, že robot zatím nemá žádné informace, a je stejně pravděpodobné, že bude v jakémkoliv bodě prostoru. Jakmile se robot pohne, částice se přesunou, a předpoví nové možné stavy po pohybu. Jakmile snímač robota v nové pozici nasnímá data okolí, částice se na základě rekurzivního Bayesového odhadu přeshlukují. Úkolem metody je docílit, aby předpokládané pozice částic konvergovaly ke skutečným pozicím robota [8].

Nelze přesně předvídat budoucí pozici robota, a proto se generuje spousta náhodných odhadů o příštím umístění robota. Tyto odhady se označují jako částice. Každá částice obsahuje úplný popis možného budoucího stavu. Jakmile robot nasnímá okolí, odstraní se nejméně odpovídající částice a vytvoří se částice nové.

V této metodě se vytváří tzv. domněnka, což je odhad aktuální pozice robota, která se určuje pomocí funkce hustoty pravděpodobnosti, rozložené nad celým stavovým prostorem. Domněnka je reprezentovaná v čase t množinou M částic $X_t = \{x_t^1, x_t^2, ..., x_t^M\}$. Každá částice obsahuje stav, hypotézu robotovi pozice. Opravdovou pozici robota určuje oblast ve stavovém prostoru s částicemi s největší pravděpodobností. Na začátku robot nemá žádné informace, takže částice jsou rovnoměrně rozloženy po celém prostoru. V každém okamžiku algoritmus vezme předešlou domněnku, a dle aktuálního posunu, nových dat obdržených ze senzoru, určí domněnku novou [9].

Při dalším přesunutí robota, algoritmus MCL simuluje pohyb na každou částici v množině. Jakmile snímač robota snímá okolí, zaktualizuje i částice. Pro každou částici spočítá pravděpodobnost, že pokud by byl ve stavu částice, obdržel by stejná data, jako které naměřily senzory. Každé částici pak dle spočítané pravděpodobnosti přiřadí váhu w_t^i. Následně určí nové částice z předešlých domněnek s největší váhou w_t^i. Díky tomu předpokládané pozice částic iterativně konvergují ke skutečným pozicím robota [10].

```
Algoritmus MCL(X_{t-1}, u_t, z_t):
   X'_t = X_t = 0
for m=1 to M:
   x_t^m = motion_update(u_t, x_{t-1}^m)
   w_t^m = sensor_update(z_t, x_t^m)
   X'_t = X'_t + ⟨x_t^m, w_t^m⟩
endfor
for m=1 to M:
   draw x_t^m from X'_t with probability w_t^m
   X_t = X_t + x_t^m
endfor
return X_t
```

3.7 Metoda HSM

Metoda HSM (Hough Scan Matching) využívá pro srovnání snímků Houghovu transformaci. Metoda se pro zpracování obrazu, detekci geometrických čár a křivek v digitálním obraze, využívá už od šedesátých let. HSM je použitelná jak pro lokální tak i globální lokalizaci a je to neiterativní metoda. Pro určení rotace snímků se určuje Houghovo spektrum DHS a pro určení posunu snímků se využívá diskrétní Houghova transformace DHT. Algoritmus funguje následovně:

1. Spočítá se DHT a DHS pro referenční a naměřená data
2. Křížovou korelací se určí hypotézy rotace z lokálních maxim Houghova spektra
3. Pro každou hypotézu se určí posun snímků z DHT

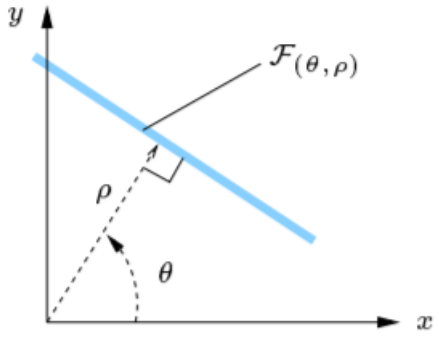
Výhoda Houghovo spektra je, že je invariantní pro translaci vstupních dat a od rotace vstupních dat je posunuté. Křížovou korelací se určí několik možných hypotéz rotace snímků. Houghova transformace zobrazuje $i(s)$ vstupní prostor do parametrického prostoru $HT\{i\}(p)$.

$$HT[F, i](p) = \int_{F_P} i(s)ds \tag{9}$$

Pomocí Houghovy transformace lze vyjádřit i větší hustotu bodů ve snímku jako přímku. Proto je zvolen parametrický prostor. Přímka je vyjádřená pomocí polárních souřadnic na obrázku 5:

$$x\cos\theta + y\sin\theta = \rho\ , \tag{10}$$

kde ρ je vzdálenost přímky od počátku souřadného systému, θ je směr normálového vektoru



Obrázek 5: Vyjádření přímky v polárních souřadnicích

Vnitřní prostor je konečná množina bodů $P = p_j$, což jsou naměřené hodnoty ze senzoru. Pro vstupní prostor platí, pokud $s := (x, y)$ pak $i(s) = \sum \delta(s - p_j)$ kde δ je diracovo delta.

Houghovo spektrum je definováno jako:

$$HS_g[i](\delta) := g[HT[i](\delta, .)] \tag{11}$$

kde g je invariantní funkcionál.

3.8 Korelační metody

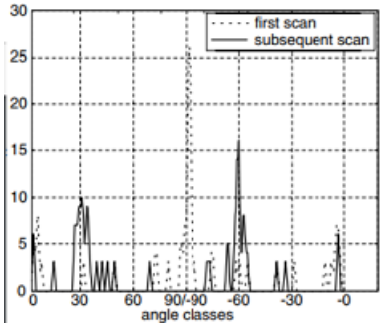
Korelační metody využívají podobnosti v naměřených datech, a pomocí minimálního korelačního koeficientu hledají nejlepší posun snímků, a díky tomu určení pozice robota. V článku [15] je korelace použitá na sadu histogramů. Z naměřeného snímku jsou určeny tři histogramy, histogram pozice X, pozice Y a úhlový histogram. Histogramy ze dvou snímků jsou diskrétní, a proto mohou být porovnány diskrétní korelační funkcí:

$$c(j) = \sum_{i=1}^{n} h_1(i) h_2((i + j) \setminus n) \ , \tag{12}$$

kde c je vzájemná korelace, j je fázový posun, h_1 a h_2 jsou histogramy porovnávaných snímků,\ operátor modulo, n je počet tříd histogramu

Příklad porovnání úhlových histogramů dvou snímků je na obrázku 4. V článku [15] je korelační metoda porovnávána s odometrií s překvapivými výsledky. Zatímco u odometrie s použitím Kalmanova filtru byla chybovost určení pozice robota snížena o 45 %, při použití histogramů a korelace byla menší o 80 %.

V článku [16] Murray, Erwin a Wermter určují pozici robota pomocí korelace zvukových záznamů. Inspirovali se sluchovým systémem savců. Ve svém experimentu na robota umístili dva mikrofony, které jsou od sebe posunuty ve známé vzdálenosti a snímají určitý zvukový zdroj. Autoři článku Murray, Erwin Wermter používají sekundovou nahrávku. Pomocí vzájemné



Obrázek 6: Přehled kotevních bodů

korelace časového zpoždění zvukového záznamu mezi mikrofony určují azimut – úhel, který svírá zdroj s robotem. Po určení azimutu určují i vzdálenost zvukového zdroje pomocí korelace:

$$Corr(g,h)_j(t) = \sum_{k=0}^{N-1} g_{j+k} h_k \ , \tag{13}$$

kde g, h jsou zvukové záznamy a t maximální časová odchylka

Všem bodům v prostoru jsou přiřazeny vektory. Pomocí vzájemné korelace je vyhledávány maximální hodnota vektoru:

$$length(C) = (length(A) - length(B)) - 1 \ , \tag{14}$$

kde A a B jsou vektory, směřující k mikrofonům, C je vektor určují pozici maximální korelace mezi signály g a h s časovým posunem. Metoda je časově náročná. Určení pozice bylo 1-2 sekundy opožděné za reálným posunem robota. Přesnost metody je 97 %.

V disertační práci [17] autor využívá korelaci pro srovnání snímků přímo ze surových naměřených dat. Snahou o snížení časové náročnosti se omezuje pouze na okraje snímků – hranice polygonů ve snímcích. Pro tyto účely vzájemnou konvenční korelaci modifikuje na:

$$(r * s)(p_x, p_y, \omega) = \begin{pmatrix} r(\Phi)\cos(\Phi) \\ r(\Phi)\sin(\Phi) \end{pmatrix} \odot \begin{pmatrix} s(\Phi+\omega)\cos(\Phi+\omega) + p_x \\ s(\Phi+\omega)\sin(\Phi+\omega) + p_y \end{pmatrix} , \tag{15}$$

kde r a s jsou naměřené hodnoty, p_x a p_y hodnoty transformace.

Modifikovaná korelační funkce vyjadřuje míru zarovnání snímků. Výsledkem operace $\odot$ je pak takové číslo, které ukazuje na míru podobnosti dvou laserových snímků. Hledaná transformace je taková, která je maximální. Operace $\odot$ se využívá k určení korelačního koeficientu, pro jehož určení je potřeba trojrozměrná funkce, což zvyšuje časové nároky výpočetního programu. Další možností jak definovat korelační koeficient průniku polygonů korelovaných snímků, protože korelovat celé snímky je časově a výpočetně náročné, je jednodušší snímek nejdříve převést na rastr a korelovat pouze hranici polygonu snímku. Algoritmus korelačního koeficientu funguje, prochází hranici testovaného snímku a porovnává ji s hranicí referenčního snímku. Pokud se na stejné ad-

rese v příslušné buňce vyskytuje stejná hodnota, korelační koeficient se zvyší o jedna. Omezením se pouze na hranice polygonu, lze metodu využít i v reálném čase. Algoritmus je robustní vůči dynamyckým objektům. Autor publikace [17] porovnává algoritmus se stávajícími metodami ICP, CLS.

3.9 Metody využívající diferenciální evoluci pro lokalizaci mobilních robotů

Princip diferenciální evoluce navrhl Storn a Price [11]. Vycházejí z hlavních myšlenek evolučních algoritmů a navrhují jiný přístup. Pro optimalizaci evoluce používají diferenci dvou náhodně vybraných vektorů z populace.

3.9.1 Diferenciální evoluční filtr

Moreno a Garrido na základě myšlenek diferenciální evoluce vytvořili nelineární evoluční lokalizační filtr [18] pro globální lokalizaci robota. Tento přístup je robustní, účinný, pracuje přímo s naměřenými daty a nevyhledává žádné význačné rysy. Cílem této metody je určit pozici robota možnými převáženými stavy. Metoda funguje rekurzivně, podobně jako metoda Monte Carlo.

Algoritmus evolučního filtru sestává z těchto kroků:

1. Inicializace

   Určí se počáteční řešení, vytvoří se hodnota fitness pro každý bod a určí se počáteční pozice.

2. Evoluční hledání

   Z naměřených a počátečních hodnot se určí ztrátová funkce. Vytvoří se nová generace pomocí perturbačního vektoru. Následuje křížení.

3. Aktualizace

   Nejlepší prvek populace nahradí počáteční odhad pozice.

Evoluční filtry pracují s hypotetickými stavy, avšak nevyužívají distribuci pravděpodobnosti. Používají n dimenzionální vektory $x_i^k = (x_{i,1}^k, ..., x_{i,n}^k)^T$, kde i je kandidát řešení a k je iterační krok. Počáteční populace je vybraná náhodně tak, aby pokryla celý prostor. Diferenciální evoluční filtr vytváří nový parametrický vektor rozdílem váhových vektorů, dvou členů populace. Výsledný vektor se porovnává s předem určeným vektorem z populace a pokud je menší, v populaci ho nahradí. Tato základní myšlenka diferenciální evoluce je v diferenciálním evolučním filtru rozšířena o další perturbační vektor, pro který platí:

$$v = x_i^k + L(x_b^k - x_i^k) + F(x_{r_2}^k - x_{r_3}^k) \ , \tag{16}$$

kde x_i^k je parametrický perturbační vektor k-té iterace, x_b^k je nejlepší vektor z populace k-té iterace, $x_{r_2}^k$ a $x_{r_3}^k$ jsou parametrické vektory náhodně vybrané z populace, L a K jsou konstantní váhové faktory, které určují zesílení diferenční variace.

Diference je následovaná křížením:

$$u_{i,j}^k = \begin{cases} v_{i,j}^k, & p_{i,j}^k < \delta \\ x_{i,j}^k \end{cases} , \tag{17}$$

kde $p_{i,j}^k$ je náhodně vybraná hodnota z intervalu [0, 1] pro každý parametr j populace i-tého kroku k a δ je kontrolní hodnota křížení.

Autoři článku [19] diferenciální evoluční filtr rozšířili o další mechanismy. Přidali další prahový odmítavý pás, díky kterému je kandidát v populaci nahrazen, jen pokud je potomek výrazně lepší. Rozšířili algoritmus o vyřazovací operátor, který urychluje konvergenci algoritmu tím, že eliminuje nejhorší kandidáty v populaci v každé iteraci. Pozměnili perturbační zesilovací faktor F tak, aby se přizpůsoboval a zvyšoval, pokud populace nesměřuje do slibných oblastí. S takto upraveným algoritmem provedli dva experimenty. V prvním experimentu lokalizovali pozici robota v místnosti bez pohyblivých předmětů. Ve druhém experimentu přidali pohyblivé předměty a špatně rozpoznatelné překážky. Dle těchto experimentů je algoritmus vysoce robustní a málo náchylný na šum. Přesné výsledky experimentu lze dohledat v [19].

3.9.2 Diferenciální evoluční filtr s Markovovými řetězci

Dobré výsledky má i kombinace diferenciálního evolučního filtru s Markovskými řetězci (KL Based Differential Evolution Markov Chain Global Localization Filter) [20]. Markovské řetězce vyjadřují přechody mezi jednotlivými stavy, které se uskutečňují v libovolně blízkých časových okamžicích. Náhodné proměnné nabývají hodnoty přiřazené určitým stavům [21]. Cílem algoritmu je převést částice z metody diferenciální evoluce na částice Markovských řetězců.

Na začátku se počáteční populace rovnoměrně rozloží na celý prostor. Každý kandidát představuje Markovův řetězec a v každé iteraci jsou vytvářené nové řetězce. Nový potenciální kandidát je generován mutací v DE. V následujícím výběrovém mechanismu se určí přijetí nebo odmítnutí nového kandidáta. Výhodou kombinace DE a MC je využití výhod obou těchto metod. Diferenciální evoluce účinně vyhledává možné stavy v prostoru a Markovovy řetězce účinně prostor vzorkují. Pro konverzi částic je důležité dodržet podmínku, že pokud vzorek x_i^j je z cílové distribuce, i další vzorek x_i^{j+1} musí být ze stejné distribuce.

$$x_{i*}^j = x_i^j + F(x_{r_1}^j - x_{r_2}^j) + e \ , \tag{18}$$

kde e je normální distribuce v d dimenzionálního prostoru a F je mutační konstanta.

V každé iteraci se opakují veškeré operace pro všechny kandidáty a vytvářejí celou populaci pro další iteraci. Na každého člena populace lze nahlížet jako na Markovův řetězec, který se vyvíjí ve stejném místě s nejlepší hodnotou fitness diferenciální evoluce. Cyklus pokračuje, dokud není splněna konvergenční podmínka.

```
Algoritmus DE_MC_GL
   for i = 1 : N_p do                          inicializace N_p Markovovych retezcu
      x_i^0 = uniform(free_map)
   end for
   j = 1
   while (Konvergencni podminka) do
      for i = 1 : N_p do
         x_{i*}^j = x_i^j + F(x_{r_1}^j - x_{r_2}^j) + e                Mutace
         r_log = fitness(x_{i*}^j) - fitness(x_i^j)
         u ~ U_(0,1)
         if r_log < log u then                    selekce, dalsi vzorek retezce
            x_i^{j+1} = x_{i*}^j
         else
            x_i^{j+1} = x_i^j
         end if
      end for
         j ← i + 1
         optimalni lokace = x_i^j : min{fitness(x^j)}
         kontrola konvergencni podminky
   end while
end
```

3.9.3 Hybridní adaptivní diferenciální evoluce pro lokalizaci

Hybridní adaptivní evoluce modifikuje principy diferenciální evoluce. Jedním z nejdůležitějších parametrů v diferenciální evoluci je váhový faktor, taktéž nazývaný mutační konstantou F. Před začátkem optimalizace se tento parametr vynuluje a nastavuje pro každého člena populace zvlášť. Hybridní adaptivní evoluce upravuje mutační konstantu F pro každého člena populace v iteraci, podle blízkosti k optimální oblasti. Autoři článku [22] porovnali hybridní adaptivní evoluci s diferenciální evolucí pro globální lokalizaci. U hybridní evoluce prokázali rychlejší konvergenci a vyšší přesnost lokalizace.

4 Přehled prostředků pro tvorbu simulací (C, C++, C#, Matlab)

V této kapitole jsou uvedeny programovací jazyky používající se na tvorbu simulací. Jsou uvedeny jejich výhody i nevýhody. Nejobecnější rozdělení jazyků je na nízko-úrovňové a vysoko-úrovňové. Nižší programovací jazyky jsou primitivní jazyky, které využívají příkazy procesoru. Výhodou je, že daný program lze co nejlépe optimalizovat. Programátor není ničím omezen a může využít všechny funkčnosti zařízení. Dalším možným dělením je na jazyky interpretované a neinterpretované. Interpretované jazyky potřebují kompilátor, kterým je kód překládaný za běhu programu. Výhodou je, že programátor nemusí deklarovat proměnné ani velikost programu, avšak program je pomalejší ve srovnání s neinterpretovaným jazykem. Kompilované jazyky jsou rychlejší, po zkompilování mohou být spouštěny samostatně.

4.1 Jazyk C

Programovací jazyk C vyvinuli v sedmdesátých letech Ken Thompson a Dennis Ritchie v Bellových laboratořích firmy American Telephone and Telegraph. Jazyk C je nízko-úrovňový jazyk, který byl původně zamýšlen na programování systémů. Program napsaný v jazyku C lze efektivně přeložit do strojového kódu. Jazyk C má několik poměrně jednoduchých pravidel, pomocí kterých je možné vytvářet a skládat jednotlivé úseky programů do větších celků. V roce 1984 vznikla první verze americké národní normy jazyka pod označením ANSI C. Programovací jazyk C podporuje strukturované programování, umožňuje přímý přístup do paměti pomocí ukazatelů a aplikace v něm jsou kompaktní. Hodí se na systémové i aplikační programování [23].

4.2 Jazyk C++

Jazyk vytvořil Bjarne Stroustrup. C++ je programovací jazyk s podobnou syntaxí jako C. Původně byl jazyk C++ vyvíjen pro vyřešení jedné konkrétní simulace s velice specifickými požadavky, která napovídala spíše na využití jiného jazyka než C [24]. Jeho schopnosti sahají od nízko-úrovňového programování až po vysoko-úrovňové. Podporuje jak objektové programování, tak i procedurální programování. Výhodou objektového programování při tvorbě simulací je, že lze využívat již předdefinované objektové třídy. Na simulace lze tedy využívat již předefinované objekty a v rámci dědičnosti je rozšiřovat. Jazyky C/C++ umožňují využít v simulacích generování náhodných čísel pomocí funkcí rand a srand. Náhodná čísla se generují pomocí rekurentně zadaných posloupností, takže se nejedná o čistě náhodný proces [25].

4.3 Jazyk C#

Jazyk C# je vysoko-úrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft s platformou .NET Framework. Nejedná se o jazyk interpretovaný, ale už přímo o spustitelný kód, generovaný automaticky těsně před spuštěním programu. Technologie .NET se skládá z CLR (Common Language Runtime) a z knihoven tříd systému .NET Framework. CLR je společným

běhovým prostředím pro programy psané v různých jazycích. Díky CLR spolu mohou jednotlivé části programu navzájem spolupracovat, přestože byly napsané v jiných programovacích jazycích. Výhodou C sharp je automatický managenet paměti tzv. Garbage collection [27].

4.4 Matlab

Jedná se o interaktivní programové prostředí a skriptovací programovací jazyk. Program je vyvíjen společností MathWorks. Původně byl jazyk určen pro matematické účely, postupem času byl Matlab rozšiřován o další funkce.

Matlab umožňuje provádět simulace v nadstavbovém nástroji Simulink, který je určen k modelování a simulačním výpočtům dynamických systémů, popsaných blokovým schématem sestaveným z prvků vestavěných knihoven, nebo bloků definovaných uživatelem. Nechybí popis často používaných základních prvků knihoven. Příkladem vestavěných funkcí je Simulace Monte Carlo [26].

5 Návrh lokalizačního algoritmu založeného na evolučním algoritmu a vzájemné korelaci

V předchozí části byli popsané různé metody pro zarovnání snímků. Cílem nové metody je nalézt afinní transformaci p, která nejlépe spojí testovací snímek s referenčním snímkem. Množina všech afinních transformací p je tvořena posunem p_x, p_y, a úhlem natočení snímků ω. Navržená metoda lokalizačního algoritmu spojuje dva algoritmy – diferenciální evoluci, která hledá afinní transformaci p, a vzájemnou korelaci, která určí, nakolik je nalezená transformace vhodná.

5.1 Diferenciální evoluce

Diferenciální evoluce (DE) je efektivní a velmi jednoduchý algoritmus, který vychází z algoritmu genetického žíhání. Poprvé byl publikován Kennethem V. Pricem v roce 1994 [11]. Název diferenciální evoluce popisuje princip algoritmu, kdy k vygenerování zkušebního výsledku dochází pomocí diference dvou náhodných jedinců. DE algoritmy mají rychlost konvergence, která je při porovnání s klasickými algoritmy viditelně menší.

Diferenciální evoluce probíhá cyklicky v generacích s cílem nalézt nejvhodnější populaci jedinců. Z prohledávané oblasti S se vybírá uspořádaná $N - tice$ bodů, které se říká populace bodů a bývá označována jako P. Každý z bodů populace P je adeptem na řešení. Populace se za běhu vyvíjí, takže z $g - té$ generace populace P se vyvine $(g + 1)$ generace populace P. Každý prvek populace si hledá neustále lepší umístění v prohledávané oblasti S. Vývoj prvků populace probíhá s využitím evolučních operátorů mutace, křížení a výběru.

Nejdříve se vytvoří počáteční generace P_0 z populace P, která se skládá z N bodů. Každý bod z počáteční generace je vybrán náhodně. Z prvků počáteční generace populace P, se spočítá hodnota účelové funkce. Cyklus se poté opakuje. Z aktuální generace P_g se vytváří nová generace Q_g tak, že ke každému bodu x_i aktuální generace P_g je vytvořen pokusný bod y, který vzniká pomocí operace mutace a křížením náhodných bodů P_g. Mutací vznikne mutant u, křížením pak z mutanta u a původního bodu x_i vznikne pokusný bod y. Do nové generace Q_g populace pak postupuje lepší bod z této dvojice. Jakmile je generace Q_g kompletní, stává se Q_g novou generací P_{g+1} populace. Při splnění ukončovací podmínky se cyklus tvorby nových generací populace P ukončí a výstupem algoritmu jsou nejlepší body poslední vytvořené generace populace P.

Mutací vzniká k bodu x_i populace mutant u. V následujících popsaných mutacích jsou r_1, r_2, r_3, r_4, r_5 navzájem různé a náhodně vybrané body z aktuální generace P_g. F je mutační konstanta, x_{best} je nejlepší bod aktuální generace P_g mutace, což znamená, že pro bod x_{best} je účelová funkce menší než v ostatních bodech.

Typy mutací [11]:

1. rand/1
$$u = r_1 + F(r_2 - r_3) \tag{19}$$

2. rand/2
$$u = r_1 + F(r_2 - r_3) + F(r_4 - r_5) \tag{20}$$

3. best/1
$$u = x_{best} + F(r_1 - r_2) \tag{21}$$

4. best/2
$$u = x_{best} + F(r_1 - r_2) + F(r_3 - r_4) \tag{22}$$

5. current-to-best/1
$$u = x_i + F(x_{best} - x_i) + F(r_1 - r_2) \tag{23}$$

6. current-to-best/2
$$u = x_i + F(x_{best} - x_i) + F(r_1 - r_2) + F(r_3 - r_4) \tag{24}$$

7. rand-to-best/1
$$u = r_1 + F(x_{best} - r_1) + F(r_2 - r_3) \tag{25}$$

8. rand-to-best/2
$$u = r_1 + F(x_{best} - r_1) + F(r_2 - r_3) + F(r_3 - r_4) \tag{26}$$

9. either-or
$$u = \begin{cases} r_1 + F(r_2 - r_3), & rnd < F \\ r_1 + 0.5(F+1)(r_2 + r_3 - 2r_1) \end{cases} \tag{27}$$

Mutace v DE je následována křížením, kdy se vytváří pokusný bod y, potomek mutanta u a bodu populace x_i, a to záměnou některých souřadnic x_i za hodnoty souřadnic mutanta u. V DE se používá křížení binomické a exponenciální.

Diferenciální evoluce je zapsaná v tomto pseudokódu:

```
Algoritmus DE
   vygeneruj pocatecni generaci populace P_0 = (x_1, x_2, ..., x_N) populace P
   vypocitej hodnotu ucelove funkce ve vsech bodech generace P_0
   g = 0
   repeat
      Q_g = P_g
      for i = 1 to N do
         vytvor pokusny bod y k bodu x_i
         if f(y)≤f(x_i) then
            do Q_g vloz na misto bodu x_i bod y
         end if
      end for
   P_{g+1} = Q_g
   g = g + 1
   until ukoncovaci podminka
```

Dle [12] lze DE algoritmus zjednodušeně popsat následovně. V algoritmu je potřeba nejdříve nastavit parametry, které ovlivňují průběh evoluce. Jedná se o mutační konstantu F, práh křížení CR, počet jedinců v populaci NP, počet argumentů účelové funkce D a vzorového jedince, dle kterého se vygeneruje prvotní populace jedinců.

Aktuálně vybraný jedinec se označuje jako aktivní. Náhodně se vyberou tři jedinci z populace. První dva se od sebe odečtou, čímž vznikne diferenční vektor. Diferenční vektor se vynásobí mutační konstantou F, čímž vznikne váhový diferenční vektor, jehož přičtením ke třetímu vzniká šumový vektor. Pro jeden prvek z aktivního jedince a šumového vektoru se vygeneruje náhodné číslo z intervalu 0-1, které se porovnává s prahem křížení CR. Pokud je číslo menší, do nového zkušebního jedince se vloží prvek z šumového vektoru. Pokud je číslo vetší než CR, do zkušebního jedince je umístěn prvek z aktivního jedince. Cyklus evolučního algoritmu běží do doby, než je vytvořen určitý počet generací.

5.2 Vzájemná korelace

V předchozím textu byl popsán algoritmus diferenciální evoluce. Tento algoritmus v průběhu svých iterací vybere tři členy z populace r_1, r_2, r_3, vytvoří z nich mutačního třírozměrného mutačního kandidáta u, který je následně využit pro zarovnání laserových snímků. Dvě jeho souřadnice jsou pro relativní translaci a jedna souřadnice pro rotaci. Pro vznik dalších generací je nezbytné určit vhodnost každého mutačního kandidáta v populaci vůči původnímu kandidátovi pomocí účelové funkce. Z kandidátů, kteří jsou nejvíce vhodní, je vytvořena další generace. V této práci je jako účelová funkce využitá vzájemná korelace.

Robot poskytuje naměřené hodnoty v polárních souřadnicích. Každá hodnota určuje vzdálenost překážky r při kroku laserového snímače 0,25°. Naměřený snímek je tedy množina $A : r(\Phi)$.

Mějme dva snímky, jeden referenční $A_{\mathrm{REF}} : r(\Phi)$ a druhý testovací $B_{\mathrm{TEST}} : r(\Phi)$, lišící se rotací a translací. Pro výpočty je vhodnější snímky převést do souřadnic kartézských. Pro transformaci do kartézských souřadnic platí:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r(\Phi)\cos(\Phi) \\ r(\Phi)\sin(\Phi) \end{pmatrix} \tag{28}$$

Diferenciální evoluce poskytuje možnou transformaci snímků p_x, p_y, ω. Dosazením do modifikované korelace dostaneme:

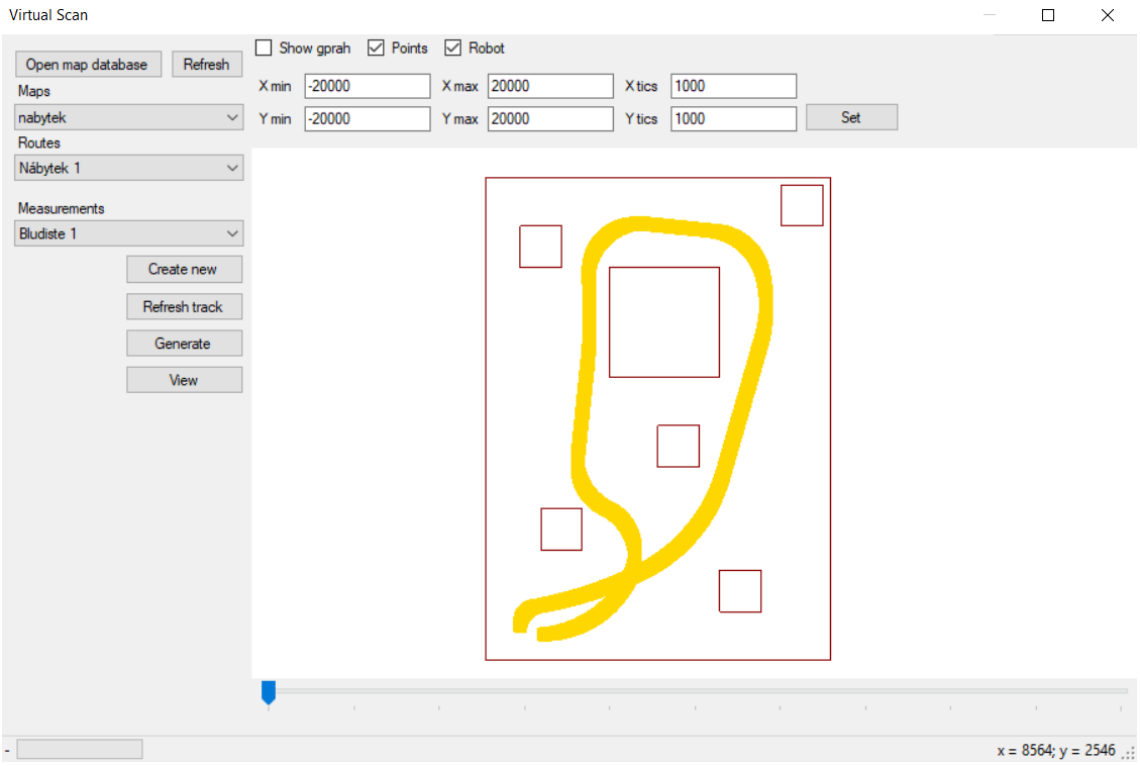
$$(A_{\mathrm{REF}} * B_{\mathrm{TEST}})(p_x, p_y, \omega) = \begin{pmatrix} A_{\mathrm{REF}}(\Phi)\cos(\Phi) \\ A_{\mathrm{REF}}(\Phi)\sin(\Phi) \end{pmatrix} \odot \begin{pmatrix} B_{\mathrm{TEST}}(\Phi+\omega)\cos(\Phi+\omega) + p_x \\ B_{\mathrm{TEST}}(\Phi+\omega)\sin(\Phi+\omega) + p_y \end{pmatrix} \tag{29}$$

Výsledkem operace je číslo z množiny kladných čísel, korelační koeficient, který ukazuje na míru podobnosti dvou laserových snímků. Celý proces funguje následovně. Diferenciální evoluce z jedné populace nabídne pro zarovnání tři hodnoty p_x, p_y, ω. Testovací snímek je o tyto hodnoty posunut a zarovnán. Pomocí vzájemné korelace jsou testovací a referenční snímek porovnány. Kandidáti, kteří mají nejvyšší vzájemnou korelaci, největší fit, jsou využity pro vznik dalších generací. Algoritmus se ukončí, jakmile je splněn určený počet populací a generací.

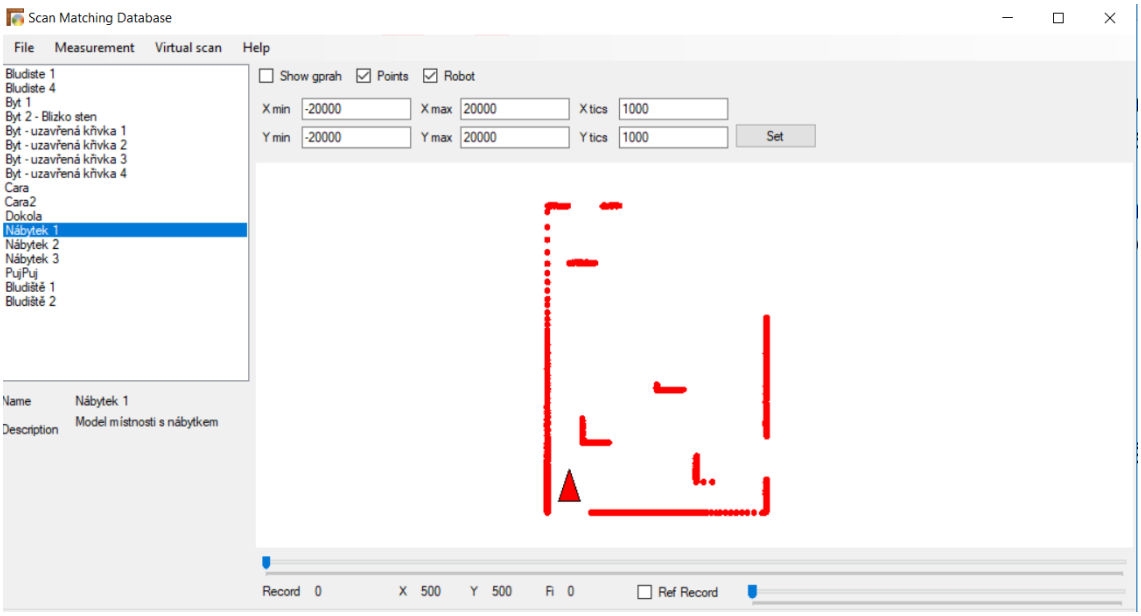
5.3 Program simulující pohyb robota

Pro testování algoritmu je nezbytné vyzkoušet program na naměřených datech. Existují dvě možnosti jak tato data získat. Buď testovat na reálném robotu, který při průchodu místností ji naskenuje laserovým paprskem, a tato data pak poskytne. Nebo mít vhodný simulační program, který tento úkol simuluje. Pro testování navržené metody v této diplomové práci, byla využitá druhá možnost. Jako nejvhodnější program se pro tento úkol jeví simulační program *ScanMatchingDatabase* navržený ing. Jaromírem Konečným.

Tento jednoduchý program byl navržen v programovacím jazyce *C#*. Program nejen, že věrně napodobuje reálné chování robota a věrně skenuje místnost, ale lze i jednoduchými příkazy v *XML* kódu nadefinovat jakoukoliv místnost, a to různých tvarů, dále různé překážky, např. židle, stoly, otevřené dveře atd. A lze i nadefinovat různou cestu robota. Program *ScanMatchingDatabase* následně dle navržené cesty robota místností vygeneruje naměřená data a exportuje do formátu *csv*. Na obrázku 7 je ukázka návrhu cesty robota zadefinovanou místností. Na obrázku 8 je znázorněno, jak vypadají výstupní data ze simulačního programu.



Obrázek 7: Ukázka návrhu cesty robota a místnosti



Obrázek 8: Ukázka výstupních dat robota při průchodu místností

6 Implementace lokalizačního algoritmu a jeho optimalizace

V této kapitole bude popisován postup řešení, vytvořený program a implementace algoritmu v jazyce C++ ve vývojovém prostředí Visual studio 2015. Zkompilovaný program lze spustit přes příkazový řádek. Uživatel zadá data ve formátu tabulkových dat .csv, která se mají zpracovat. Uživatel také může nastavit jednotlivé parametry diferenciální evoluce. Veškerý výčet možností je uveden v tabulce 1.

Pokud uživatel zadá pouze jméno souboru a vybere algoritmus diferenciální evoluce, ostatní parametry se nastaví na přednastavené hodnoty, tj. počet populací 100, počet generací 10000, práh křížení 0.9, a mutační konstanta 0.9.

Po spuštění programu se nastaví hodnoty diferenciální evoluce a načtou se hodnoty ze vstupního souboru. Dle zadaných indexů se v souboru vyberou hodnoty pro referenční a testovací snímek. Pokud uživatel indexy nezadal, data v prvním řádku v souboru se použijí pro referenční snímek a ve druhém řádku pro snímek testovací. Po načtení souboru následují jednotlivé fáze diferenciální evoluce v tomto pořadí:

1. Inicializace prvotní populace
2. Mutace
3. Křížení

Při inicializaci prvotní populace se vytváří zadaný počet vektorů. Každý vektor obsahuje tři složky, které jsou vybírány náhodně z intervalu od nuly do jedné. Po vytvoření populace následuje mutace. Pro každý vektor se postupně v populaci určí mutační kandidát. Pokud uživatel nevybral určitý typ mutace, je přednastaven standardní typ mutace rand/1:

$$u = R_1 + F(R_2 - R_3) \ , \tag{30}$$

kde F je mutační konstanta. Vektory R_1, R_2 a R_3 jsou vybírány z populace náhodně. Ve fázi křížení, dochází k promíchání složek původního vektoru se složkami mutačního kandidáta. Náhodně se určí číslo v intervalu od nuly do jedné. Pokud je toto číslo vyšší nežli práh křížení, složka původního vektoru a mutačního se zamění. Takto vytvořený mutační vektor je poté posouzen účelovou funkcí – vzájemnou korelací snímků.

Pro potřeby korelace a porovnávání snímků jsou vytvořená dvourozměrná pole, tzv. mapy. Robot svým laserovým snímačem naměří vzdálenosti překážek r v místnosti. Laserový senzor se pohybuje s krokem 0.25° a nasnímá tak 1082 hodnot v rozmezí 45 – 225°. Data ze snímku, naměřená robotem, jsou přepočítaná do kartézských souřadnic dle rovnice 28. Kde r je vzdálenost naměřené překážky v místnosti a úhel Φ je úhlový krok pro laserový snímač. Kartézské souřadnice dat jsou poté pro potřeby korelační matice transfomovány pomocí vztahů:

$$X_{\text{upravené}} = \frac{(X - \text{MIN}_{\text{mapy}}) \times \text{Velikost}_{\text{mapy}}}{(\text{MAX}_{\text{mapy}} - \text{MIN}_{\text{mapy}})} \ , \tag{31}$$

Příkaz	**Popis**
-h	zobrazí nápovědu
-file + jméno souboru	zadání vstupní souboru, který se bude zpracovávat
-alg + de nebo ide	výběr algoritmu diferenciální evoluce
-reference + celé číslo	nastaví index pro referenční snímek v souboru
-current+ celé číslo	nastaví index pro testovaný snímek v souboru
-gen+ celé číslo	nastaví počet generací
-pop+ celé číslo	nastaví počet populací
-F + celé číslo	nastaví mutační konstantu
- C + celé číslo	nastaví práh křížení
-time + celé číslo	nastaví čas trvání

Tabulka 1: Nastavitelné parametry navrženého programu

$$Y_{\text{upravené}} = \frac{(Y - \text{MIN}_{\text{mapy}}) \times \text{Velikost}_{\text{mapy}}}{(\text{MAX}_{\text{mapy}} - \text{MIN}_{\text{mapy}})} \ . \tag{32}$$

Tyto transformační vztahy vstupní data přeškálují. Pokud je například nejvzdálenější překážka v naměřených datech:

$$r = 8\,000 \text{ (mm)} \ ,$$

pak je dostatečné nastavit minimum a maximum na:

$$\text{MIN}_{\text{mapy}} = \ -10\ 000$$

$$\text{MAX}_{\text{mapy}} = \ 10\ 000$$

Velikost mapy je zvolena na hodnotu 100 z důvodu výpočetní náročnosti. Čím bude větší velikost mapy, tím stoupá výpočetní náročnost celé operace. Při tomto nastavení má jedna buňka v mapě rozsah 200, což znamená, že do jedné buňky v mapě se vejde 200 hodnot vstupních dat.

Mapa je poté vyplněna následujícím způsobem: buňka v mapě je vyplněná jedničkou, pokud se index mapy shoduje s kartézskými souřadnicemi bodu ve snímku. Na zbylých buňkách v mapě jsou nuly. Mapa je vytvořena jednak pro referenční snímek, jednak pro snímek testovací. Složky vektoru mutačního kandidáta u určí posun testovacího snímku v mapě. První dvě složky u_1, u_2 jsou použité pro posun v souřadnicích x, y, třetí složka u_3 je použitá pro natočení testovacího snímku o úhel θ. Vzájemnou korelací se pak tyto vytvořené mapy porovnají buňku po buňce dle vztahu 29. Výsledkem je celé kladné číslo zvyšující se s lepším zarovnáním testovacího snímku na referenční. Pokud je posun příliš velký, snímky se nezakryjí ani v jednom bodě a výstupem bude nula.

Výsledná hodnota slouží k posouzení vhodnosti mutačního kandidáta u. Pokud mutační vektor u má vyšší výsledek vzájemné korelace, v populaci původní vektor nahradí. Po vyzkou-

šení všech prvků v populaci je vytvořená nová generace z již upravené populace. Algoritmus je ukončen proběhnutím všech zadaných generací.

Na obrázcích 9 a10 je graficky znázorněn průběh navrženého programu. Aplikace byla spuštěna se vstupními parametry:

$$počet\ populací = \ 100$$

$$počet\ populací = \ 10\ 000$$

$$F = \ 0.5$$

$$CR = \ 0.9$$

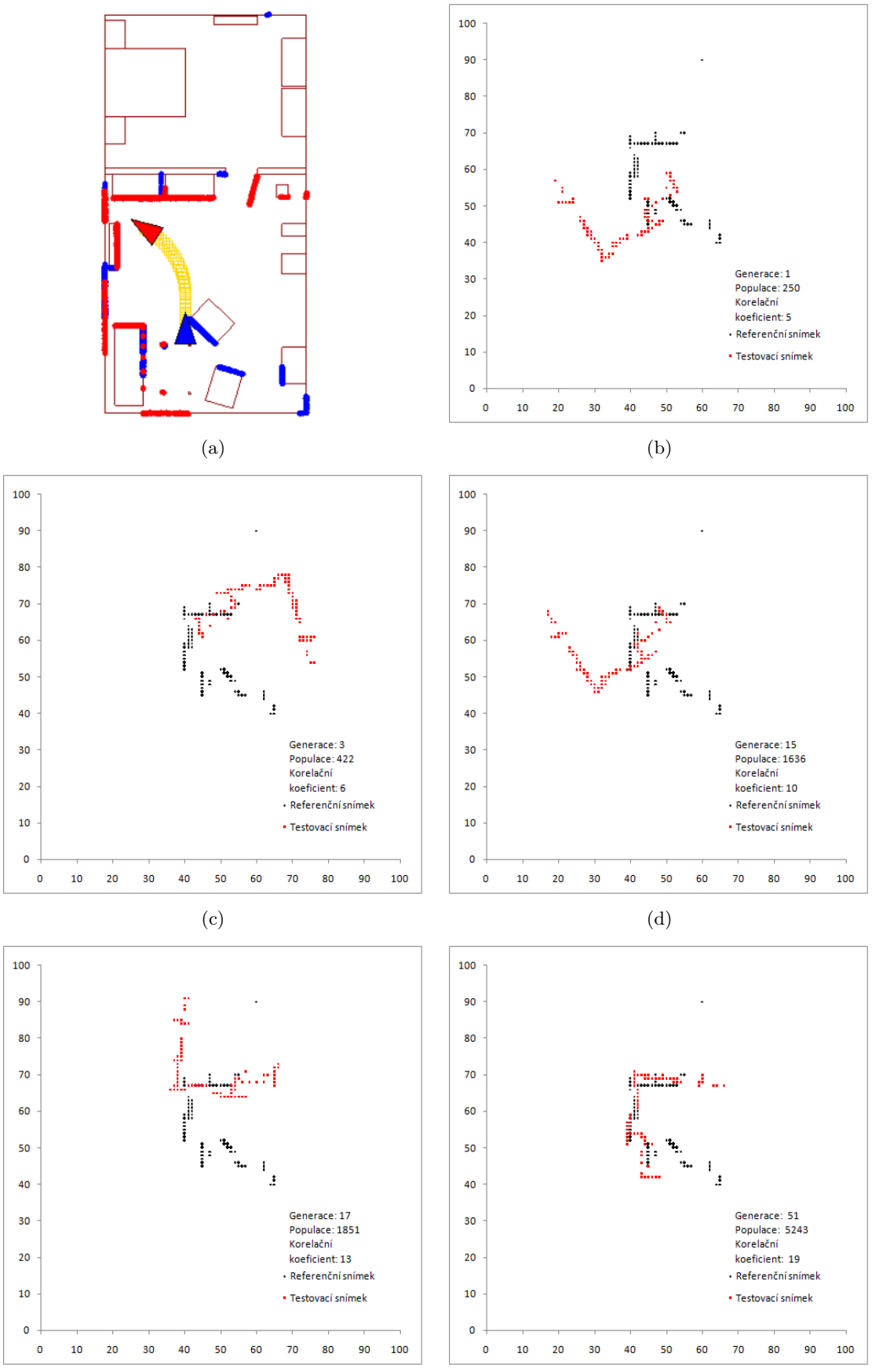
Na prvním obrázku 9(a) je znázorněno reálné posunutí robota o hodnoty $x = -911$ (mm), $y = 2\,579$ (mm) a $\theta = -57$ (°).

Na dalších obrázcích je znázorněn postupný průběh iterace diferenciální evoluce. Přesněji řečeno je znározněn uskutečněný posun pro mutačního kandidáta z populace, který má vyšší korelační koeficient než ten předešlý. V další generaci následně kandidát původního člena nahradí. Na obrázcích je černou barvou znázorněn referenční snímek a červenou barvou snímek testovací s použitým posunem, který nabízí vybraný kandidát. Snímky jsou transformovány tak, jak vstupují do posuzovací mapy. U každého obrázku je vypsáno, o kterou populaci a generaci se jedná, a jaký má kandidát korelační koeficient. Na obrázku 9(b) je prvotní odhadnutí. Snímky se již částečně dotýkají, a proto korelační koeficient začíná narůstat. V průběhu dalších generací se již odhad posunu zpřesňuje až ve výsledný odhad na obrázku 10(l), na kterém jsou snímky dokonale zarovnány. Nalezený posun na posledním obrázku je $x = -913$ (mm), $y = 2\,582$ (mm) a $\theta = -56$ (°).

6.1 Korelační mapa s váženými body

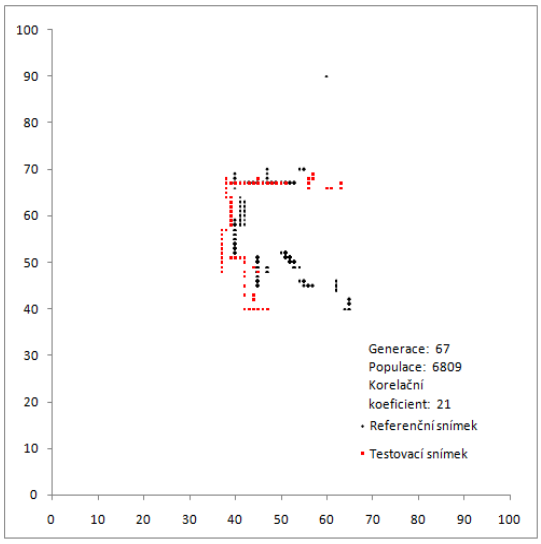
Naměřené hodnoty jsou před naplněním korelační mapy transformovány pomocí vztahu 31. Při naplňování je do buňky v mapě vyplněná jednička, pokud tam naměřená hodnota dle indexů patří. Ostatní naměřené hodnoty, pokud by patřili do stejné buňky, jsou již zahazovány. Laserový senzor pro překážky, které jsou k němu blíže, naměří více hodnot. Proto je nasnadě vytvořit vážené buňky v mapě. V buňce nebude jednička, ale počet bodů, který do rozsahu buňky v mapě z naměřených dat patří. Na obrázcích 11 a 12 jsou použita stejná naměřená data jako v předešlém příkladu. Avšak pro posouzení vhodnosti kandidáta je použita korelační mapa s váženými body. Překážky, které jsou k robotu blíže, mají vyšší číslo v buňkách. Na obrázku 11(a) je opět ukázán reálný posun. Na dalších 11(b) až 12(l) je použit posun kandidáta z dalších generací se zvyšujícím se korelačním koeficientem.

Z obrázků je patrné, že opět dojde k zarovnání snímku. Avšak v tomto případě k chybnému. Vážené buňky v mapě se na sebe zarovnají z důvodu vyššího korelačního koeficientu. V případě větších posunů robota, jako je v tomto pokusu, nastane chybné zarovnání. Mapa s váženými body

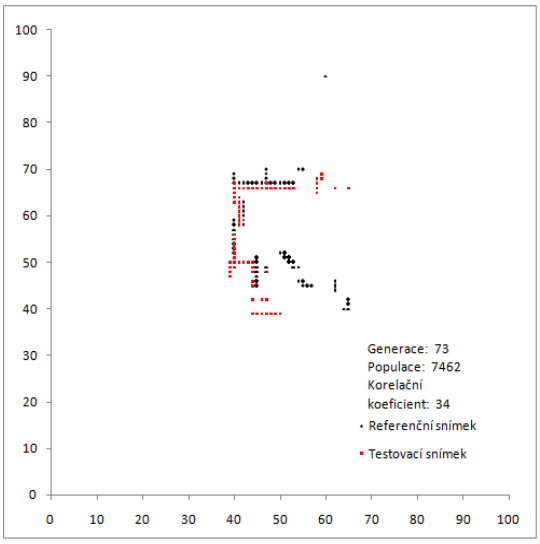


(a) (b) (c) (d) (e) (f)

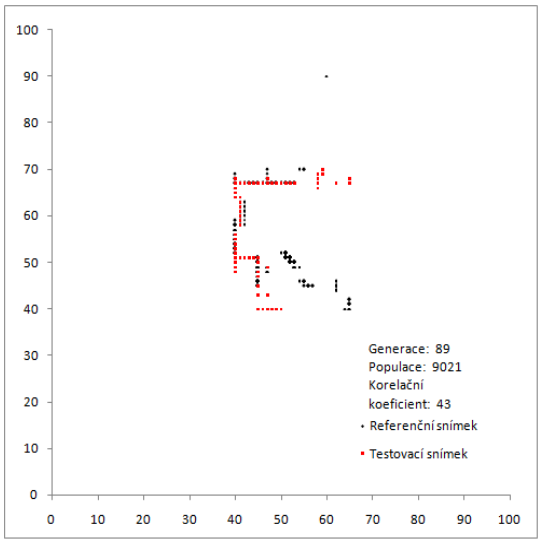
Obrázek 9: Grafické zobrazení průběhu aplikace



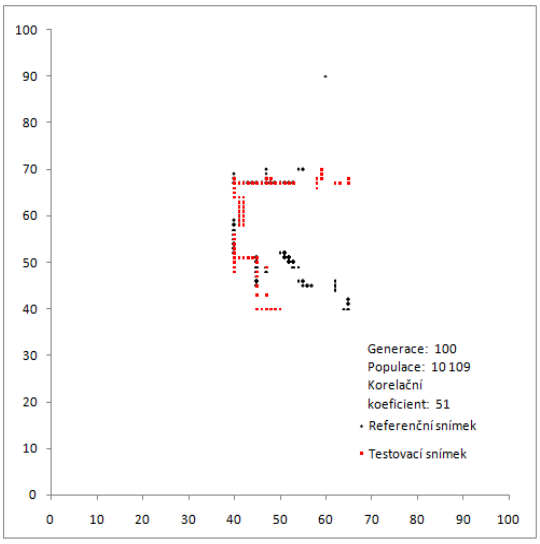
(g)



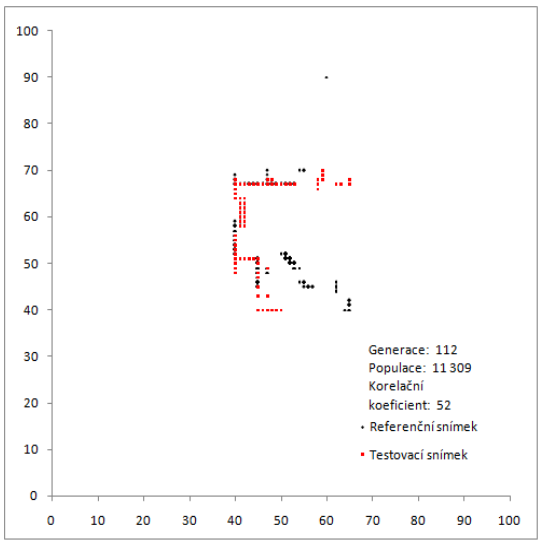
(h)



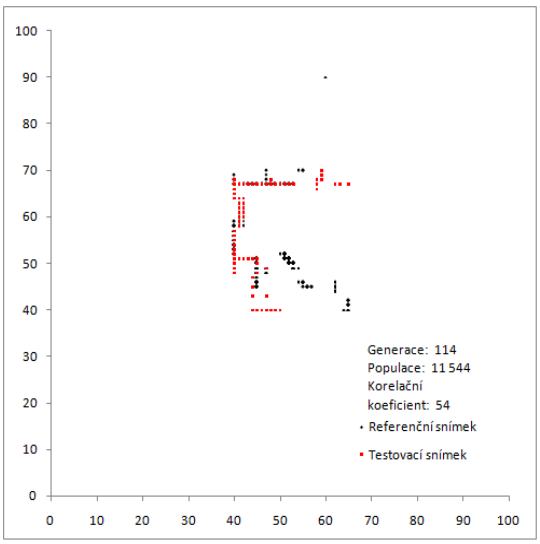
(i)



(j)



(k)



(l)

Obrázek 10: Grafické zobrazení průběhu aplikace

by měla smysl pouze pro menší posuny. V reálné situaci nelze dopředu poznat, jestli posun bude malý či velký. A proto, aby algoritmus byl co nejvíce obecný pro veškeré situace, které mohou nastat, nelze korelační mapu s váženými body dál používat.

6.2 SLAM – Průchod bludištěm

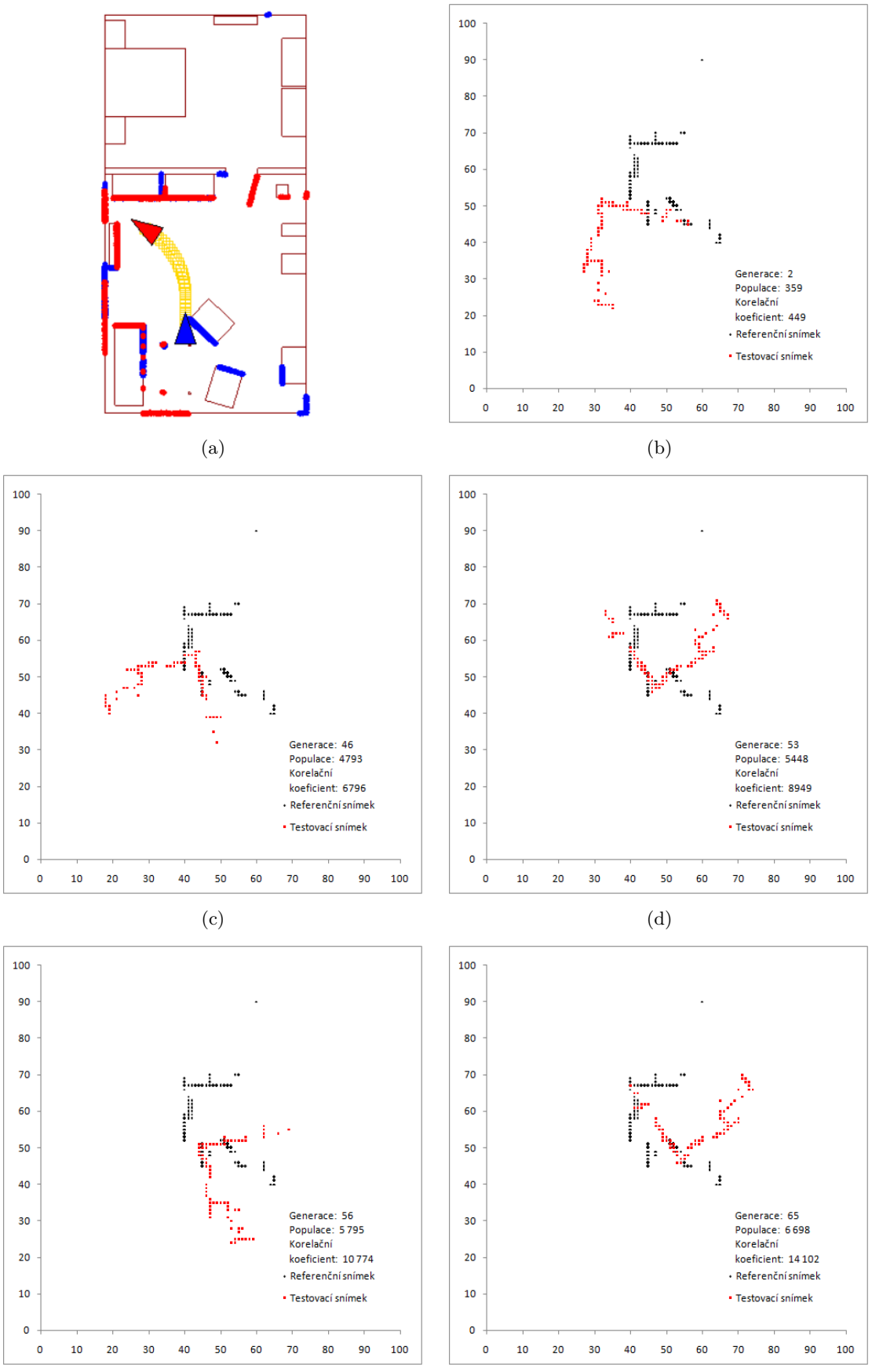
Dalším experimentem pro ověření algoritmu a nalezení případných chyb v aplikaci je pokus nechat robota projít bludištěm. V tomto experimentu robot lokalizuje svou pozici pomocí porovnávání snímku. Naměřená data pro průchod bludištěm byla simulována v programu ScanMatchingDatabase. Výstup programu se uložil do souboru ve formátu .csv, ve kterém je na každém dalším řádku uložen další posun robota v bludišti. Celkem soubor obsahuje 76 pozic. Na každém řádku v souboru jsou simulovány hodnoty laserového senzoru z právě dané pozice v bludišti. V tomto experimentu je pozice robota zjištěna ze srovnání na sebe snímků, po sobě jdoucích v souboru. Aplikace ScanMatchingDatabase vypisuje i reálné posuny, proto je možné skutečné posuny a vypočtené posuny porovnat. Reálné posuny robota jsou v souboru vztažené vůči první pozici robota. Robot se v průběhu cesty bludištěm různě otáčí, proto je nutné pozici robota přepočítat do nové soustavy souřadnic pomocí transformačních vztahů, pro rotaci:

$$x^I = x\cos(\alpha) - y\sin(\alpha)\ , \tag{33}$$

$$y^I = x\sin(\alpha) + y\cos(\alpha)\ , \tag{34}$$

kde x, y jsou původní souřadnice, x^I, y^I jsou transformované souřadnice, α je úhel o který se robot otočí oproti prvnímu snímku v souboru. Může se stát, že jsou dva po sobě jdoucí snímky otočeny oproti první pozici o 90°. Program snímky zarovná sice bez použití transformace reálného posunu vůči prvnímu snímku do nové soustavy, ale eviduje posun v jiné ose.

Výsledný průchod robota bludištěm je na obrázku 13. Černou barvou je v bludišti na obrázku naznačena ideální lokalizace, červenou barvou lokalizace robota spočítaná programem. Na obrázku je patrná odchylka spočítaného průchodu, která průběžně integračně narůstá. Je to z důvodu určování umístění robota, vždy z dvou po sobě jdoucích snímků. Malá chyba postupně narůstá. Největším podílem na tomto růstu chyby je otáčení robota o 90°. Na tomto experimentu se projevila negativní vlastnost korelace, chybné zarovnávání, pokud je ve snímku více rovných stejných ploch. Lze si představit nereálnou idealizovanou situaci, kdy robot prochází nekonečně dlouhou chodbou. V takovém případě nelze ze dvou stejných snímků určit opravdovou novou pozici. Program snímky zarovná chybně, nerozpozná posun, a nerozezná ani rotaci, natočení robota. Pokud by v takovémto nereálném idealizovaném příkladu, byl i dostatek překážek, jako jsou například židle nebo stoly, program by snímky zarovnal již dobře. Stejný problém nastává, i pokud je místnost středově souměrná, popřípadě se v místnosti objevují stejné tvary. V takových případech dochází ke špatnému určení otočení robota. Algoritmus má tendenci snímky

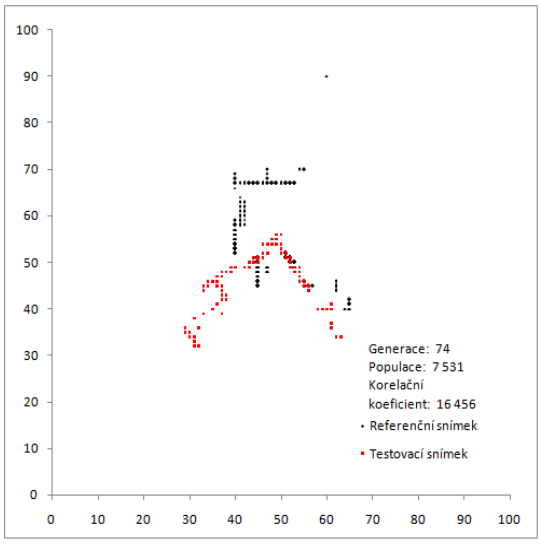


(a) (b)

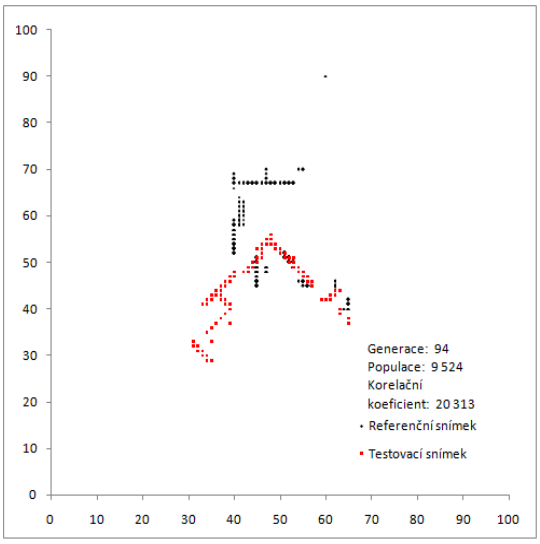
(c) (d)

(e) (f)

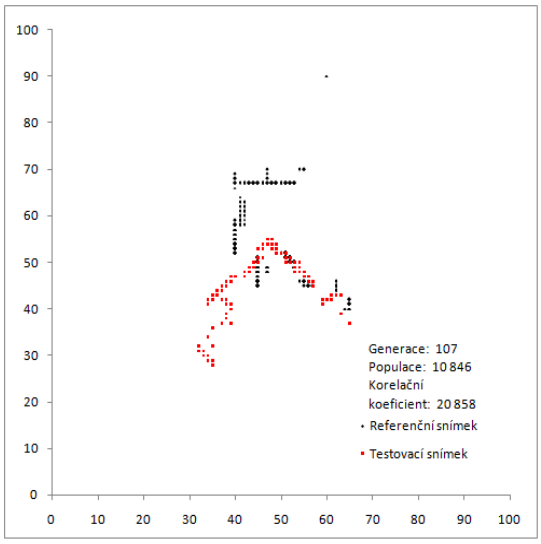
Obrázek 11: Grafické zobrazení průběhu aplikace s mapou s váženými body



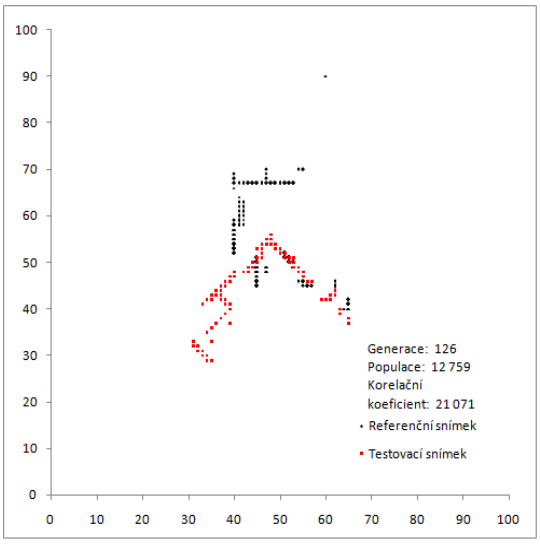
(g)



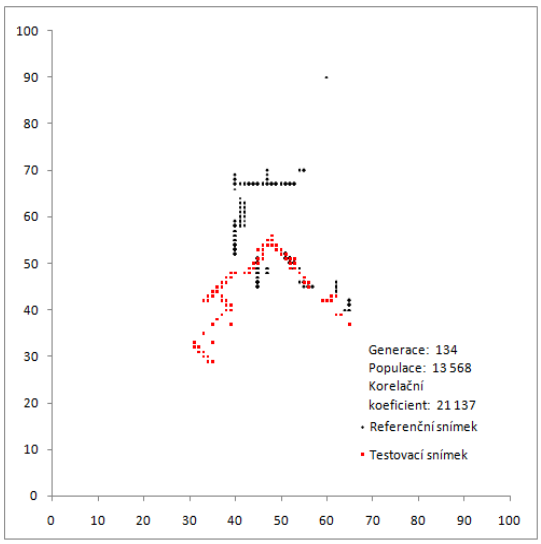
(h)



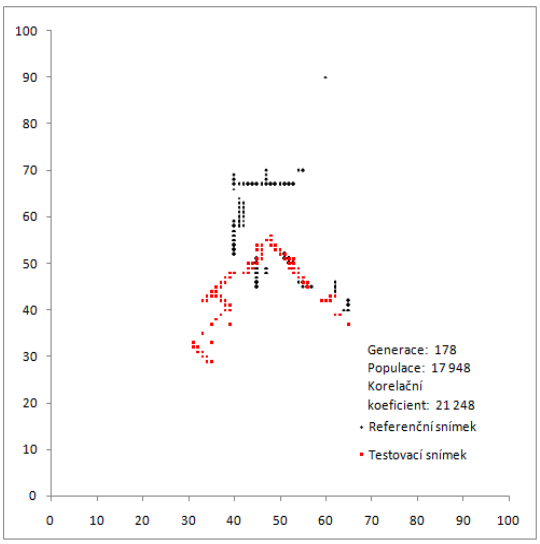
(i)



(j)

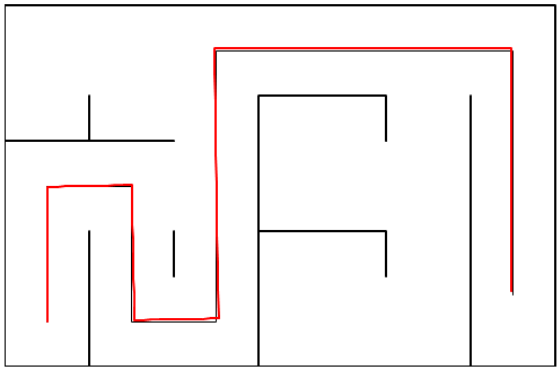


(k)



(l)

Obrázek 12: Grafické zobrazení průběhu aplikace s mapou s váženými body



Obrázek 13: Průchod Bludištěm

otočit o 180°. V takových případech osově symetrických místnostích, je program nutné pustit několikrát a vybrat nejčastější nalezený posun.

6.3 Jiné typy mutace DE

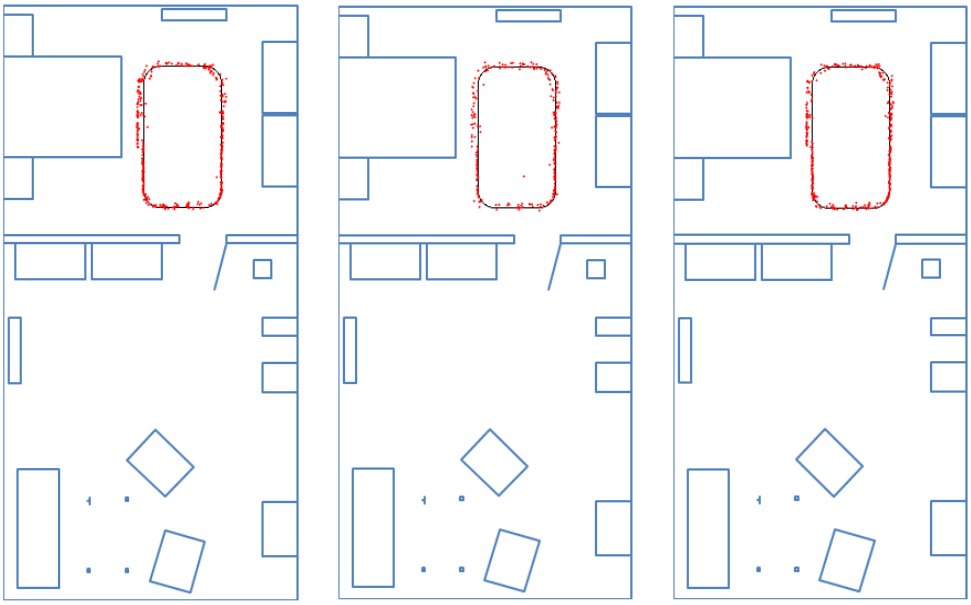
Navržený lokalizační algoritmus umožňuje vybrat i jiné typy mutace diferenciální evoluce. Kromě standardní mutace rand/1 (rovnice 19) lze vybrat mutaci Current-to-best/1 (rovnice 23) a mutaci either-or (rovnice 27). Mutace Current-to-best/1 využívá pro vytvoření nového mutačního člena u nejlepšího kandidáta x_{BEST} z předešlé generace. Při mutaci either-or vystupuje nová proměnná rnd, která je vybraná náhodně z intervalu nula až jedna. Pokud bude proměnná rnd menší, nežli mutační konstanta F, pro vytvoření mutačního kandidáta u se použije klasická mutace rand/1. Pokud bude větší, použije se pro vytvoření mutačního kandidáta u rovnice 27.

Pro vyzkoušení popsaných typů mutací jsem zvolil pokus porovnávání následných posunů robota v místnosti vůči první pozici. Graficky je pokus znázorněn na obrázku 14. Robot v místnosti vykoná znázorněný pohyb, postupně mění veškeré své tři souřadnice a vrátí se do původní pozice. V každé nové pozici vytvoří sken. Na obrázku 14 je černou barvou znázorněno opravdové umístění robota, červenou barvou pozice určená programem. Program porovnává postupně všechny skeny vždy vůči prvnímu, referenčnímu, ten je neměnný. Pro každý porovnaný sken se určí pozice robota v místnosti. Se vzdalováním robota od referenční pozice dochází k postupnému snižování společných bodů ve snímcích.

V tabulce 2 jsou vypsané směrodatné odchylky a úspěšnost pokusu pro různé typy mutací diferenciální evoluce. Směrodatné odchylky jsou určeny pomocí vztahu:

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - x)^2}\ , \tag{35}$$

kde x je reálná hodnota dané souřadnice v místnosti, x_i je spočítaná souřadnice algoritmem, N je počet měření. Směrodatná odchylka je určena ze všech pozic v tomto experimentu, proto je



a - rand/1 b - current-to-best/1 c - either-or

Obrázek 14: Jiné typy mutace DE

$N = 460$. Vlivem negativních vlastností korelace dochází k chybnému zarovnánávání. Z tohoto důvodu je u každého typu mutace stanovena úspěšnost λ správného zarovnání:

$$\lambda = \frac{N_V - N_S}{N_V} * 100 \ , \tag{36}$$

kde N_S je počet špatně zarovnaných snímků, N_V je celkový počet snímků v souboru naměřených dat.

	Rand/1	**Current-to-best**	**Either-or**
σ_x **(mm)**	52	162	46
σ_y **(mm)**	152	124	38
λ (%)	80	54	79

Tabulka 2: Výsledky testu typu mutací

Z obrázku 14 a z tabulky 2 je patrné, že nejvyšší úspěšnost má standardní mutace rand/1. Mutace current-to-best1, tedy tvoření nové populace pomocí nejlepšího kandidáta v populaci, má úspěšnost nejnižší. Směrodatné odchylky jsou zhruba u všech mutací řádově stejné.

7 Testování navrženého řešení z hlediska výpočetní náročnosti a úspěšnosti zarovnání

V této kapitole budou ukázány výsledky experimentálního ověření navrhované metody. Navrhovaný program byl testován z hlediska výpočetní náročnosti a úspěšnosti zarovnání. Navrhovaná metoda, zarovnávání pomocí diferenciální evoluce ve spojení se vzájemnou korelací, je porovnávána s metodou, která taktéž používá diferenciální evoluci, ale jako účelová funkce se používá počítání euklidovské vzdálenosti.

7.1 Popis porovnávacího programu

Referenční algoritmus, který poslouží pro porovnání s navrženou metodou, používá taktéž evoluční algoritmus, diferenciální evoluci. Avšak jako účelovou funkci pro posouzení vhodnosti kandidáta z populace, používá výpočet euklidovské vzdáleností bodů. Přesněji, sumu minimálních vzdáleností bodů v referenčním a posuzovaném snímku. Účelová funkce pracuje následovně. Snímek se posune. Vybere se první bod z referenčního snímku. Najde se bod z porovnávacího snímku, který má nejmenší vzdálenost. Vybere se další bod z referenčního snímku, nalezne se opět bod, který má minimální vzdálenost a přičte se k předešlé určené vzdálenosti předešlého bodu v referenčím snímku. Výsledkem je suma nejmenších vzdáleností bodů ve snímcích.

Sumační minimální euklidovská vzdálenost v pseudokódě:

Posuzovany snimek B se posune
for $i = 0$ az do poctu bodu v referencnim snimku $A(i)$
    vyber bod z referenciho snimku $A(i)$
        for $j = 0$ az do poctu bodu ve snimku $B(j)$
            najdi minimalni euklidovskou vzdalenost
            $\sqrt{(x_{Ai} - x_{Bj})^2 + (y_{Ai} - y_{Bj})^2}$
        dokud nenajdes minimalni
        end for
 pricti k predesle a vezmi dalsi bod z referencho snimku A
 end for

Takto určená suma posuzuje, nakolik je mutační kandidát vhodný k nahrazení původního člena. Tvorba populací a generací v diferenciální evoluci je totožná jako v navrhované metodě.

7.2 Popis experimentu

Data pro posouzení obou algoritmů byla vytvořena v programu ScanMatchingDatabase, který byl popsán v předešlé kapitole této diplomové práce. Program generuje realistická data, která se ukládají v souborech typu .csv, simuluje funkce laseru SICK LMS 100, který emituje laserový paprsek

v rozmezí úhlu 45° až 225°, a zjišťuje tak vzdálenost k nejbližší překážce. Úhlové rozlišení emulovaného snímače je 0,25°. Minimální měřitelná vzdálenost je 500 mm, maximální měřitelná vzdálenost 20 000 mm. Simulační program umožňuje snadné vytváření uživatelsky definovaných schémat prostředí, vektorových map. A taktéž umožňuje definovat různé trajektorie pohybů robota v definovaném prostředí. Vektorové mapy jsou uloženy v souborech XML a simulační program je používá ke generování odpovídajících skenů LiDARu.

Hlavní výhodou simulačního přístupu je jednoduchost v generování dat a jejich zápisu ve srovnání s reálnými laserovými senzory. Nemluvě o ceně reálných laserových senzorů. Simulační data obsahují polohu a orientaci robota na mapě a data získaná simulačním LiDAREM. Polohu a orientaci lze použít pro srovnání vypočítané pozice. Pro potřeby srovnávacího experimentu byla v programu ScanMatchingDatabase vytvořena vektorová mapa a nasimulován pohyb robota. V každé pozici robota byla vygenerována odpovídající data LiDARu.

Aby se u obou algoritmů posoudili schopnosti určit pozici robota za různých podmínek, vybírá se, pro potřeby experimentu, ze simulačních dat náhodně 20 snímků, které slouží jako referenční. Porovnávací snímky se vybírají ze souboru dat ve vzdálenosti časového kroku 5, 10 a 20 od referenčního snímku. Jinými slovy, náhodně se vybírá referenční snímek ze souboru naměřených hodnot. Posuzovaný snímek se vybírá ve vzdálenosti daného časového kroku od referenčního snímku. Jako časový krok je myšlen posun robota od referenční pozice. V souboru naměřených hodnot je každá nová pozice na novém řádku. Lze na to pohlížet i jako na posun v tomto souboru o daný krok. Parametry diferenciální evoluce obou algoritmů byly nastaveny stejně:

$$počet\ populací = 20$$

$$početgenerací = 5\ 000$$

$$F = 50$$

$$CR = 0.9$$

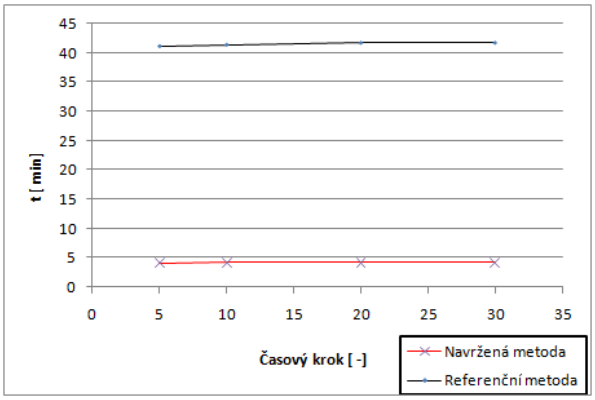
Všechny experimenty, tedy kombinace referenčního a posuzovaného snímku, proběhly 50 krát. Oba algoritmy byly spuštěny na stejném počítači, aby se zajistilo zpracování na stejném hardwarovém procesoru, kvůli posouzení časové náročnosti algoritmu.

7.3 Popis vyhodnocení výsledků

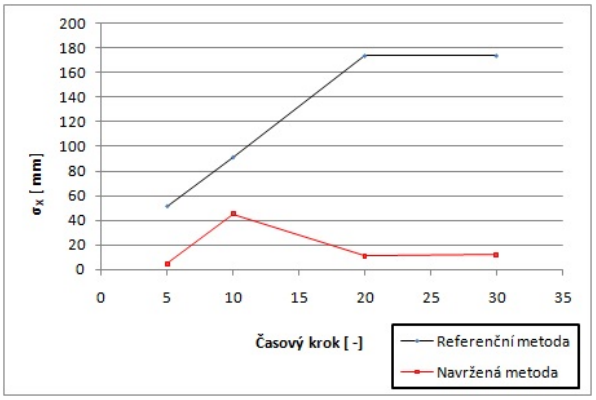
V tomto odstavci bude popsán výsledný naměřený výstup obou metod a vyhodnocování výsledků. Výstupem obou algoritmů jsou textové soubory, které obsahují vypočítané hodnoty. V souboru je vypsaný vždy kandidát z populace z dané generace, který splňuje kritéria, tj. jeho hodnoty pro posun v souřadnicích x, y, a úhel otočení θ. Soubor dále obsahuje čas v milisekundách, za který došlo k nalezení kandidáta, a jeho korelační koeficient. Každý textový soubor obsahuje jinou kombinaci referenčního a porovnávaného snímku. Náhodně bylo vybráno 80 refe-

renčních snímků. Ke každému referenčnímu snímku byl vybrán porovnávaný snímek s časovým krokem 5, 10 a 20, a každá kombinace proběhla 50 krát. Počet výsledných textových souborů pro jednu metodu je 12 000.

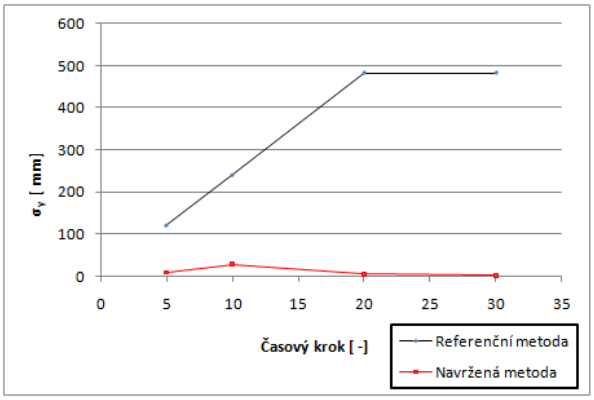
Pro zpracování naměřených výsledků byl vytvořen skript, který seřadí 50 opakování pro danou kombinaci do jednoho textového souboru. Z těchto 50-ti měření byla určena směrodatná odchylka dle vztahu 35 a úspěšnost výsledků dle rovnice 36.



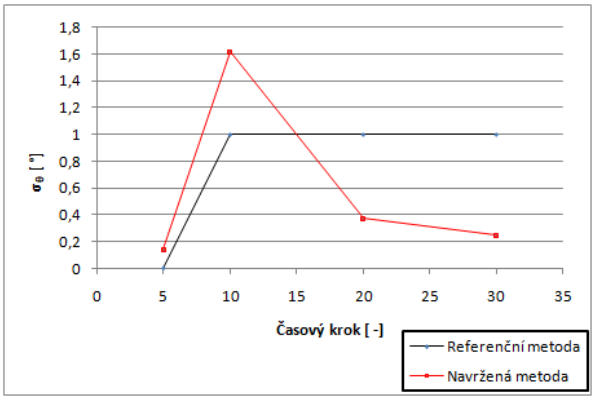
(a)



(b)



(d)



(e)

Obrázek 15: Grafické zobrazení směrodatných odchylek a časové závislosti pro referenční snímek 70

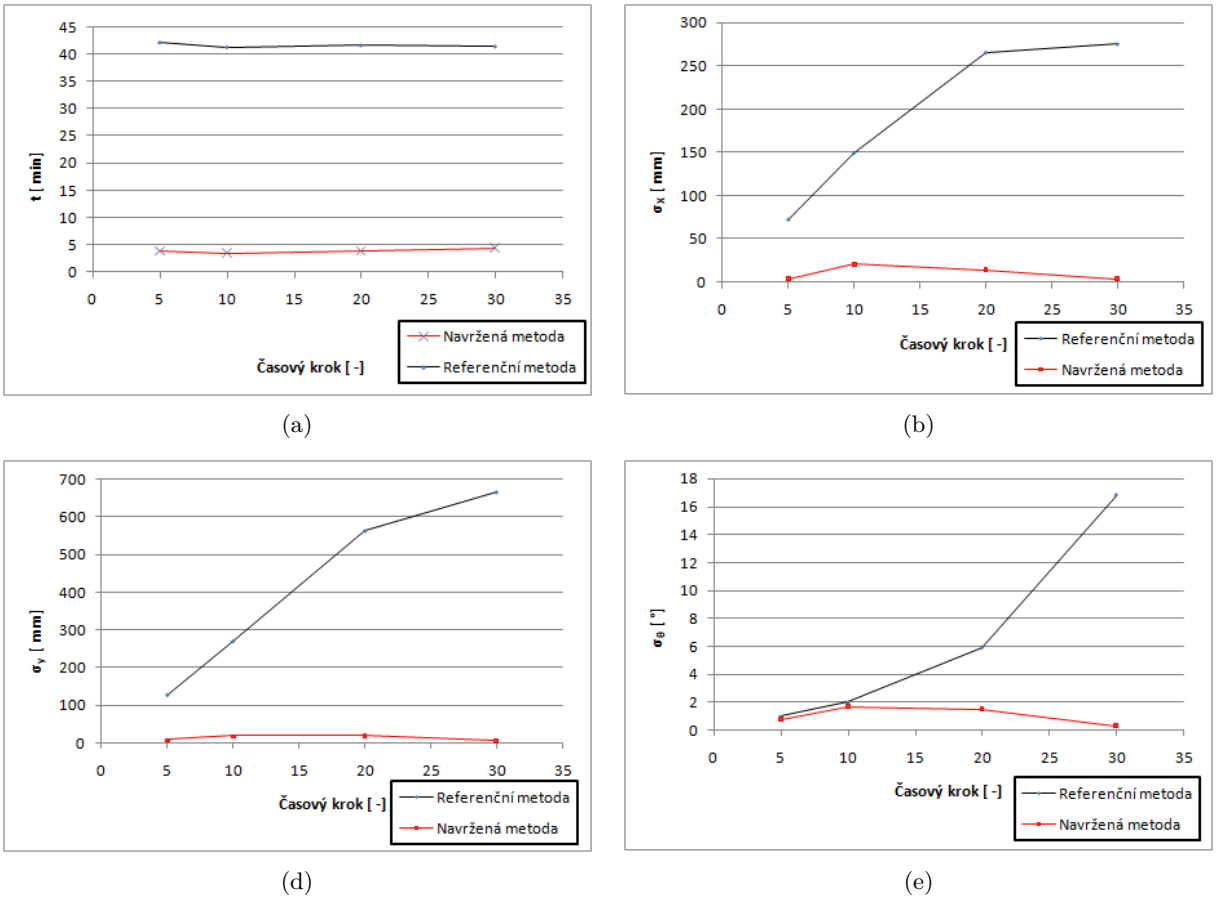
Časový krok	**5**	**10**	**20**	**30**
λ (%)	88	88	84	88

Tabulka 3: Úspěšnost zarovnání navržené metody pro referenční snímek 70

Vlivem negativních vlastností vzájemné korelace dochází k chybnému natočení snímků. Korelační koeficient pro dva snímky se pohybuje v rozmezí 100 až 150. Je to dáno počtem bodů v korelační matici. A snižuje se společným počtem bodů. Při chybném natočení začnou další změny mutačních kandidátů stagnovat pouze pro malé změny. Což má za následek, že tvorba

dalších generací již nevede ke správnému natočení. Korelační koeficient v takovém případě nebývá vyšší než 30. Proto lze tyto výsledky odfiltrovat. Veškeré navrhované posuny s korelačním koeficientem nižším než 30 jsou brány jako chybné. Nejsou brány v úvahu při počítání směrodatných odchylek a jsou započítány do neúspěšných výsledků.

Na obrázku 15 jsou graficky znázorněny výsledky. Z náhodně vybraných referenčních snímků je zde pro ukázku vybraný sedmdesátý snímek ze souboru naměřených dat. Na obrázku 16 je referenční snímek sto čtyřicátý.



Obrázek 16: Grafické zobrazení směrodatných odchylek a časové závislosti pro referenční snímek 140

Časový krok	**5**	**10**	**20**	**30**
λ (%)	88	88	86	76

Tabulka 4: Úspěšnost zarovnání navržené metody pro referenční snímek 140

Na obrázcích jsou vyneseny vypočítané hodnoty v závislosti na časovém kroku, na vzdalování od referenčního snímku. Červená barva znázorňuje navrženou metodu, diferenciální evoluci ve spojení se vzájemnou korelací, a černá barva metodu využívající diferenciální evoluci ve spo-

jení s euklidovskou vzdáleností. Na obrázku 15a je zobrazena časová závislost. Je patrné, že korelační metoda je skoro desetkrát rychlejší. Je to způsobeno korelační mapou, ve které většina buněk obsahuje prázdné pole, tedy nulu. Zatímco při počítání euklidovské vzdálenosti se hledá bod v posuzovaném snímku, který bude ke snímku nejblíže, což je časově náročné. Na obrázku 15$b - d$ jsou zobrazeny spočítané směrodatné odchylky pro nabízený posun v souřadnicích x, y, a úhel θ. I na těchto obrázcích jsou patrné menší směrodatné odchylky navrhované metody, nežli metody referenční, nicméně chyby jsou řádově srovnatelné. Na obrázku 15 je u referenční metody patrné zhoršovaní určených pozic se vzdalováním od referenčního snímku. Avšak při posouzení dalších referenčních snímků to není patrné, například na obrázku 16, nelze proto mluvit o negativní vlastnosti referenční metody.

V tabulce 3 a 4 jsou vypsané úspěšnosti λ zarovnání snímků, vypočítané dle vztahu 36. Úspěšnost zarovnání pro navrhovanou metodu se pohybuje okolo 85 %. Referenční metoda má úspěšnost 100 %.

8 Zhodnocení dosažených výsledků závěrečné práce

V této kapitole bude popsána diskuze nad dosaženými výsledky. Cílem této práce bylo nalézt funkční algoritmus založený na evolučním algoritmu. Pro porovnávání snímků se jevila nejvhodnější diferenciální evoluce, a to kvůli své nízké časové náročnosti, a jednodušší implementaci. Nemocniční mobilní roboti obsahují své vlastní jednodeskové mikropočítače. V dnešní době je na trhu nabízen mikropočítač RASPBERRY Pi, jehož jádro běží na operačním systému LINUX. Z tohoto důvodu byla celá aplikace psaná v programovacím jazyku C++, kvůli snadné budoucí přenositelnosti zkompilované aplikace na mikropočítač. Tento přístup má řadu úskalí, jednak musí programátor sledovat sám vytváření konstruktorů a destruktorů a jednak ovládání programu přes příkazovou řádku. V reálném použití aplikace na robotu lze do budoucna tento koncept vylepšit. Bude zapotřebí vytvořit další aplikaci, která již bude uživatelsky příjemnější, ve vysokoúrovňovém objektově orientovaném programovacím jazyku jako je například C Sharp, popřípadě Android, kvůli jeho rozšířenosti na mobilních telefonech a tabletech. Touto aplikací pak robota bude možné ovládat a nastavovat parametry diferenciální evoluce. Největším úskalím diferenciální evoluce je totiž nastavení mutační konstanty. Při vyšší hodnotě dochází k stagnaci v lokálních minimech. Taktéž by se v této aplikaci rovnou mohla vykreslovat mapa a nalezená lokalizace robota.

Při implementaci korelační mapy byla vyzkoušená mapa s váženými buňkami. Myšlenka tohoto nápadu spočívala v tom, že při transformaci skenu do korelační mapy dochází k jisté aproximaci. Nasnímaná data jsou rozložena rovnoměrně v prostoru, což je dané úhlovým rozlišením LIDARu. Avšak překážky, které jsou k LIDARu blíže, jsou i k sobě umístěním blíž. Při transformaci skenu do korelační mapy byl proto vytvořen koncept vytváření vážených buněk, tedy překážky, které jsou k robotu blíže, mají v buňce vyšší hodnotu, dle počtu překážek. Při experimentech se však koncept s váženými buňkami jevil jako chybný. Při velkých posunech robota totiž dochází k chybnému zarovnání snímků. Zarovnají se na sebe buňky s větší váhou. Koncept by byl použitelný pouze na malé posuny. Avšak lokační algoritmus by měl obecně fungovat na veškeré posuny robota.

Dalším vyzkoušeným experimentem byl pokus pro přístup SLAM. Průchod robota bludištěm. Výsledek je sice uspokojivý, ale do budoucna jej lze vylepšit. V tomto experimentu byla lokace robota zjištěna vždy ze dvou po sobě jdoucích snímků, což vede ke vzrůstající postupné chybě. Do budoucna by šel tento přístup vylepšit, neporovnávat pouze dva snímky, ale postupně by se snímek umísťoval do mapy, zda-li se snímek zarovnal s nějakou chybou, a až pak určit umístění robota.

Při finálním experimentu se projevila sice menší časová náročnost navržené metody, ale taky nižší úspěšnost zarovnání. Je to způsobeno spojením negativních vlastností diferenciální evoluce a negativních vlastností vzájemné korelace. Diferenciální evoluce způsobuje stagnaci v lokálních minimech. Vzájemná korelace způsobuje chybné natočení snímků. Spojením těchto negativních vlastností dochází k špatnému zarovnání. Snímek se špatně natočí a další mutační kandidáti

již nenabízejí dostatečný posun, aby se snímek natočil zpátky. Korelační koeficient takových chybných výsledků je mnohem nižší, než při správném natočení. Proto při zpracování výsledků ve finálním experimentu byly veškeré nalezené posuny s nižším korelačním koeficientem odfiltrovány a brány jako chybné. I toto filtrování by šlo do budoucna vylepšit.

Veškeré experimenty v této práci prokázaly vysokou úspěšnost zarovnání a časovou úsporu ve srovnání se stávajícími metodami, proto nic nebrání použití tohoto algoritmu ve skutečném nemocničním prostředí.

Navržený algoritmus lze však použít i v jiných oblastech v nemocnicích. Algoritmus srovnávající snímky lze použít i pro nalezení posunu CT snímků při radioterapii řízené obrazem. Pacient je při takové radioterapii před ozářením srovnáván na snímky vytvořené na simulátoru. Laborant, který přístroj ovládá, tento posun hledá ručně. Zde by se dal do budoucna použít taktéž lokalizační algoritmus navržený v této diplomové práci.

Diferenciální evoluce ve spojení se vzájemnou korelací by se dala využít i při plánování léčby radioterapie. Při tomto plánování radiologický fyzik v plánovacím systému vybírá nastavení urychlovače, kolimátoru a stolu tak, aby v zakresleném objemu byla co největší dávka. Pokud by fyzik zakreslil do CT snímků, kde má být jaká dávka, opět by se zjednodušeně jednalo o zarovnání snímků, a proto by šel využít navržený algoritmus v této diplomové práci. Nicméně je důležité dodat, že při optimalizaci plánování se již nejedná pouze o tři neznámé, jako při hledání lokalizace nemocničního robota. Avšak výhodou diferenciální evoluce je právě možná rozšiřitelnost na větší počet neznámých.

9 Závěr

Tato diplomová práce se zabývá vývojem nové metody zarovnání snímků laserového senzoru nemocničního robota. Navržená metoda je založená na evolučním algoritmu a vzájemné korelaci. Metoda má vysokou účinnost zarovnání. Navrženou metodu lze rovněž použít pro přístup SLAM.

V úvodu jsou zmínění existující nemocniční roboti, kteří používají lokalizační algoritmy. Jedná se především o nemocničního robota Pathfinder využívajícího se pro rozvoz léků v košické nemocnici. Robot byl vytvořen ve spolupráci s Technickou univerzitou v Košicích.

Kapitola 2 se zabývá přehledem senzorů, které se používají pro lokalizaci nemocničních robotů. Jsou zde zmíněny interní senzory, optické inkrementální enkodery, absolutní optické enkodéry a externí senzory. Mezi nejpoužívanější senzory patří aktivní snímače – LIDAR. Pro nemocniční roboty by se daly využít i aktivní snímače – ultrazvukové senzory. Jsou cenově dostupnější, ale za to nejsou tak přesné.

Kapitola 3 se zabývá rešerší existujících metod používaných pro lokalizaci robotu. V rešeršní práci jsou rozebrány algoritmy ICP, NDT, APR, PSM, CLS, Monte Carlo a HSM. Všechny algoritmy jsou matematicky popsány. V této kapitole jsou zmíněny algoritmy využívající korelační metody, algoritmy využívající diferenciální evoluci a algoritmus využívající diferenciální evoluční filtr s Markovými řetězci.

Kapitola 4 se zabývá přehledem prostředků pro tvorbu simulací. Hlavním cílem této kapitoly je nalezení nejvhodnějšího programovacího jazyka pro vytvoření nové metody. Kvůli použitelnosti programu přímo v hardwarovém prostředí robota se jako nejvhodnější jazyk jeví C++. Pro pozdější ovládání robota zvenčí je vhodné napsat dodatečnou aplikaci v jazyku C sharp.

Kapitola 5 navazuje na předchozí rešerši. Je zde navrhnuta nová lokalizační metoda založená na evolučním algoritmu a vzájemné korelaci. Uvádím zde základní princip diferenciální evoluce a detailně popisuji její průběh. Dále je zde popis existujících typů mutací a křížení členů, programu simulujícího pohyb robota a vysvětlení principů vzájemné korelace.

Kapitola 6 se zabývá implementací lokalizačního algoritmu a jeho optimalizací. Je zde popsáno ovládání programu, jeho spuštění a potřebné parametry, které je třeba zadat, aby program fungoval správně, vliv mutační konstany a prahu křížení na výsledné srovnání snímků. Dále je v této kapitole popsán průběh programu, vytváření populací a generací a popis účelové funkce založené na vzájemné korelaci. Vysvětleno je naplňování korelační mapy, výpočet korelačního koeficientu a jeho vliv na vytvoření dalších generací. Je zde uvedena prvotní úvaha vylepšení korelační mapy váženými buňkami. Tato úvaha však byla chybná. Taktéž je v této kapitole popsán prvotní experiment metodou SLAM, průchod bludištěm a integrační růst chyby při metodě SLAM. Experimenty byly prováděny za účelem ověření funkčnosti algoritmu a zjištění jeho vlastností. Na konci kapitoly jsem vyzkoušel různé typy mutací, abych nalezl nejvhodnější typ mutace pro závěrečný srovnávací experiment.

Kapitola 7 se zabývá finálním experimentem, který porovnává navržený algoritmus se stávající metodou. Určuje přesnosti zarovnání a časové náročnosti algoritmů. Je zde popsáno, jak

experiment probíhal a vysvětlena referenční metoda.

V diskuzi této práce jsou popsány dosažené výsledky. Je zde zdůvodněn výběr programovacího jazyka a budoucí možné vylepšení. Jsou zde popsány negativní vlastnosti navržené metody způsobené stagnací diferenciální evoluce a problém korelace, kdy dochází k chybnému zarovnání snímků. Je zde popsáno jednak možné řešení a vylepšení do budoucna, jednak možné využití algoritmu i v jiných nemocničních oblastech.

Literatura

[1] Ge, S., Lewis,F.L.: *Autonomous mobile robots.* CRC/Taylor (2006)

[2] Choi, M., Choi, J., Chung, W.: *Correlation-based scan matching using ultrasonic sensors for ekf localization.* ADVANCED ROBOTICS - UTRECHT (2012)

[3] BESL, P., MCKAY, N. A.: *Method for Registration of 3-D Shapes.* IEEE Trans. on Pattern Analysis and Machine Intelligence (Los Alamitos, CA, USA: IEEE Computer Society), Vol. 17, No.8 (1992)

[4] TAKUBO, T., KAMINADE, T.: *NDT scan matching method for high resolution grid map.* In Proc. of the 2009 IEEE International Conference on Intelligent robots and Systems, st. Louis, USA (2009)

[5] WEBER, J., JORG, K., PUTTKAMER, E.: *APR - Global scan matching using anchor point relationships.*, IOS press, pp. 471-478, (2000)

[6] DIOSI, A., KLEEMAN, L.: *Fast Laser Scan Matching using Polar Coordinates.* Vol. 26, No. 10, October 2007

[7] ZEZHONG, X., JILIN, L., ZHIYU, X.: *Scan matching based on CLS relationship.* Gansha, China, October 2003

[8] FOX, D., WOLFRAM, B., DELLAERT, F., THURN, S.: *Monte Carlo Localization for Mobile Robots.* Computer Science Department, Carnegie Mellon University, Pittsburgh PA. 1999

[9] FOX, D., WOLFRAM, B., DELLAERT, F., THURN, S.: *Robust Monte Carlo Localization for Mobile Robots.* Computer Science Department. University of Freiburg, Germany. 2001

[10] FOX, D., WOLFRAM, B., DELLAERT, F., THURN, S.: *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots.* v Proceedings of the Sixteenth National Conference of Artificial Intelligence. Orlando, USA, 1999

[11] STORN, R., PRICE, V.: *Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces.* J. Global optimization, 1997

[12] KARGER, M.: *Algoritmus diferenciální evoluce s prvky deterministického chaosu (ChaosDE) v prostředí Mathematica.* 2011

[13] BORENSTEIN, J., EVERETT, H.,R., FENG, L.: *"Where am I?"Sensors and Methods for Mobile Robot Positioning*, The University of Michigan, 1996

[14] NOVAK, P.: *MOBILNÍ ROBOTY - Pohony, Senzory řízení*, Vydavetel BEN, 2005

[15] IVANJKO, E., BASIC, D.,B., PETROVIC, I.: *Correlation Based Approach to Mobile Robot Pose Tracking in Unknown Environments*, University of Zagreb, 2007

[16] MURRAY, J., C., ERWIN, H., WERMTER, S.: *Robotic Sound-Source Localization and Tracking Using Interaural Time Difference and Cross Correlation*, University of Sunderland, NeuroBotics, 2004

[17] KONEČNÝ, J.: *Principy řízení mobilních servisních robotů.* Disertační práce, VSB-Technická univerzita, Ostrava 2014

[18] MORENO, L.,E., GARRIDO, S., MUNOZ, L.,M.: *Evolutionary filter for robust mobile robot global localization*, Robotics And Autonomous Systems, 2006

[19] MARTIN,F., MUNOZ, L.,M.,GARRIDO, S., BLANCO, D., MORENO, L.: *L1 - norm global localization based on a Differential Evolution Filter*, Intelligent Signal Processing, WISP, 2009

[20] MARTIN,F., MORENO, L.,GARRIDO, S.,BLANCO, D.: *Kullback-Leibler Divergence-Based Differntial Evolution Markov Chain Filter for Global Localization of Mobile Robots*,sensors ISSN 1424-8220, 2015

[21] HORDĚJČUKOVÁ, K.: *Spojité Markovské řetězce a jejich použití.* Bakalářská práce, Masarykova univerzita - přírodovědcká fakulta, Brno 2015

[22] BASHIRI, M., VATANKHAH, H., GHIDARY, S., S.: *Hybrid adaptive differential evolution for mobile robot localization*, Intelligent Service Robotics, 2012

[23] KERNIGHAN, W., B., RITCHIE, M., D.: *Programovací jazyk C*, Brno, Computer Press, 2013

[24] JOINES, A., J., ROBERTS, D., S.: *An Introduction to object-oriented simulation in C++*, Winter Simulation Conference, 1997

[25] PRECHELT, L.: *An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program*, Technical Report 2000-5, 2000

[26] KREJČÍ, A., REITNGER, J., TIHELKA, D., VAŇEK, J.: *Úvod do programového prostředí MATLAB*, Fakulta aplikovaných věd Západočeské Univerzity v Plzni, 2014

[27] PETZOLD, CH.: *Programování Microsoft Windows v jazyce C#*, SoftPress, 2003

[28] Košice:DNES[online]: *Éra robotov prichádza už aj do nemocničního prostredia.* Veronika Janušková. Poslední změna 1.1.2018 [cit. 1.1.2018]. Dostupné z https://kosicednes.sk/udalosti/era-robotov-prichadza-uz-aj-do-nemocnicneho-prostredia/

[29] Korzár Košice[online]: *V nemocnici v Šaci rozváža lieky robot novej generácie*. Milan Kapusta. Poslední změna 1.1.2018 [cit. 1.1.2018]. Dostupné z https://kosice.korzar.sme.sk/c/20612344/v-nemocnici-v-saci-rozvaza-lieky-robot-novej-generacie.html#ixzz52NWpfd9c

[30] Enkodéry [online]: *Senzory pro detekci pohybu kol*. Martin Dlouhý a Zbyněk Winkler. Poslední změna 30.10.2003 [cit. 1.1.2018]. Dostupné z https://robotika.cz/guide/encoders/cs

A Přílohy

Přiložené CD obsahuje:

- PDF diplomové práce.
- LATEX zdrojové soubory včetně obrázků.
- Vyhodnocené soubory experimentů v aplikaci EXCEL.
- Naměřená data, která byla použita v diplomové práci.
- Zdrojové soubory a projekt aplikace pro zarovnání snímků.
- Zkompilovanou aplikaci pro zarovnání snímků.