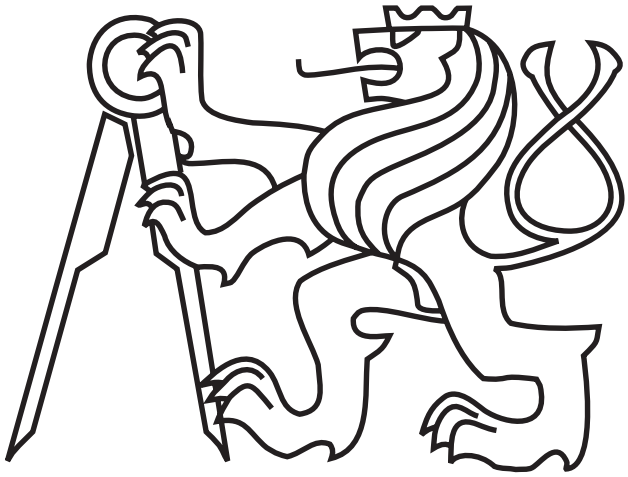
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

BAKALÁŘSKÁ PRÁCE



Tomáš Pytlíček

Navigace mobilního robotu v úlohách hledání utajené cesty

Katedra kybernetiky
Vedoucí bakalářské práce: **Ing. Jan Faigl**

Praha, 2009

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, SW, projekty atd.) uvedené v přiloženém seznamu.

V Praze dne 10. 7. 2009 .....

.....
podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Tomáš P y t l í č e k

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Navigace mobilního robotu v úlohách hledání utajené cesty

Pokyny pro vypracování:

1. Seznamte se s třídou úloh ,,covert robotic navigation", s příbuznými problémy „pursuit-evasion", „hide-and-seek" a přístupy řešení.
2. Vypracujte přehled úloh a přehled přístupů jejich řešení.
3. Seznamte se s robotickým prostředím Player/Stage.
4. Seznamte se s robotickou platformou MORBOT a robotickou platformou projektu SyRoTek.
5. Implementujte základní přístup plánování cesty mobilního robotu založeného na viditelnostním přístupu, tzv. dark path algoritmus.
6. Algoritmus experimentálně ověřte v prostředí simulátoru Stage a na reálném robotu.
7. Algoritmus porovnejte z hlediska různých přístupů hodnocení kvality nalezené cesty.

Seznam odborné literatury:

[1] Marzouqi, M. S. and Jarvis, R.A.: New Visibility-based Path-planning Approach for Covert Robotic Navigation. Robotica, 24(6):759-773, 2006.
[2] http://playerstage.sf.net.
[3] Grimmer, V.: Bakalářská práce - Platforma pro výuku mobilní robotiky. ČVUT v Praze, Fakulta elektrotechnická, katedra kybernetiky, 2008.
[4] Szücsová, H.: Bakalářská práce - Podpůrný software pro výuku mobilní robotiky. ČVUT v Praze, Fakulta elektrotechnická, katedra kybernetiky, 2008.

Vedoucí bakalářské práce: Ing. Jan Faigl

Platnost zadání: do konce zimního semestru 2009/2010

L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 23. 2. 2009

Abstrakt

Bakalářská práce se zabývá úlohami třídy pursuit–evasion a hledáním utajené cesty, které do této třídy patří. Přehled úloh a základních přístupů řešení této třídy úloh je uveden v úvodu práce. Jsou popsány některé problémy, které tato třída zahrnuje a vybrané metody jejich řešení. Některé z uvedených metod jsou použity při řešení hledání utajené cesty. Algoritmus řešení je experimentálně ověřen v různých prostředích a konfiguracích strážců. Použitý algoritmus je komplexní a poskytuje široké možnosti řešení různých variací problému hledání utajené cesty. Všechny možnosti jenž algoritmus umožňuje jsou v práci podrobně popsány. Součástí práce je také ověření navigace robotu podél nalezené utajené cesty a to jak v prostředí simulátoru tak na reálném robotu.

Abstract

The thesis deals with problems from the pursuit–evasion class, especially with covert path searching which belongs to this class. The class is described in the introduction of the thesis. Essential problems of this class and selected solving techniques are presented. Several presented approaches are than used in the covert path searching which is also called Covert Robotics Navigation. The algorithm is experimentally verified in various environments and guarding scenarios. The algorithm is complex and offers wide possibilities to solve variation of the covert navigation problems. All possibilities which the algorithm is able to solved are presented. The experimental verification of the navigation along the found covert path is presented for the simulation and the real robot experiment.

Velmi bych chtěl poděkovat především vedoucímu práce Ing. Janu Faiglovi za trpělivost, cenné rady a množství času, které mi věnoval. Také chci poděkovat celé skupině mobilní robotiky. Nakonec bych rád poděkoval své rodině za poskytovanou podporu.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Mobilní robotika za zabývá problémy řízení pohybu mobilních robotů, zpracováním senzorických dat robotů a plánováním. Mobilní robot je stroj schopný pohybu v prostoru, a to buď dvourozměrném nebo třírozměrném. V případě dvourozměrného prostoru jsou roboty kolové, pásové nebo kráčející. Ve třírozměrném to jsou například bezpilotní letadla - UAV. Své okolí vnímá robot senzory. Senzorů používaných v mobilní robotice je mnoho, nejčastěji se jedná o kamery, lasery, ultrazvukové senzory a odometrické senzory. Mobilní roboty se dají rozlišovat podle míry autonomie. Nejjednodušší robot je teleoperovaný, tj. řídí jej člověk, operátor. Opakem teleoperovaného je robot plně autonomní. Takový robot je schopen pracovat téměř bez zásahu člověka. Člověk jen zadává cíle a robot sám hledá nejvhodnější způsoby jejich dosažení.

Možných aplikací mobilní robotiky existuje velké množství. V domácnostech se lze například setkat s robotickými vysavači či sekačkami. Dalším příkladem jsou pyrotechnické roboty či dálkově řízená bezpilotní letadla. Velmi známá jsou také robotická vozítka provádějící průzkum Marsu. Mnoho úloh je zaměřených na hledání cest v prostředí, či mapování prostředí, s možným následným plněním dalších úkolů. Soubor všech úkolů, které má robot vykonat lze označit za misi robotu.

Tato práce je zaměřená na úlohy ze třídy označované v literatuře jako *pursuit–evasion*. Jedná se o rozsáhlou skupinu úloh, jejichž hlavním společným znakem je, že proti sobě stojí dva protihráči s opačnými zájmy. Slovo *pursuit* znamená stíhání či pronásledování a *evasion* únik, útěk. Pronásledovatel i uprchlík se pohybují ve společném operačním prostoru a souhrnně se označují jako hráči, jinak také agenti. Při řešení těchto úloh je klíčovým problémem schopnost odhalit či dopadnout protihráče, alternativně schopnost nepozorovaně unikat. Lze se setkat se různými variantami, ve kterých je pronásledovatel policista, strážce nebo lovec či predátor a uprchlík pak lupič nebo kořist apod. V konkrétní variantě se úkoly hráčů mohou mírně odlišovat. V uvažovaných úlohách ze třídy *pursuit–evasion* se v rolích hráčů mohou objevit jak roboti, tak lidé, z tohoto důvodu bude dále v práci užíváno životného tvaru slova robot.

Při řešení různých variant úloh je snahou se zaměřit buď na chování pronásledovatele, např. hlídání nějaké prostředí před jeho narušením, nebo na chování uprchlíka, při kterých je naopak snahou nepozorovaně vniknout do nepřátelského území. Je také možné zaměřit se na chování obou agentů, ovšem to je složité. Oba agenti mohou mít různé schopnosti a množství informací. Především se jedná o rychlost a možnosti pohybu a rozpoznávání, míru znalosti prostředí, znalosti o pozici či strategii protivníka.

Úlohy *pursuit-evasion* mají vysoký potenciál využití především u bezpečnostních složek, například u policie či armády. Jedná se buď o různé špionážní mise nebo střežení významných prostorů. Příkladem špionážních úkolů může být monitorování prostoru obsazeného únosci, kde robot může pomocí kamery poskytovat důležité informace. Příkladem střežení prostor je dobře známý problém hlídání galerií tzv. Art Gallery Problem popsaný v 1.2.1. Použitím robotů se pro člověka zmenšuje riziko plynoucí z nebezpečnosti podobných misí. Dá se předpokládat, že význam těchto úloh a používání robotů při jejich řešení bude dále narůstat.

V následující kapitole je uveden základní přehled úloh, který nejdříve představuje používané reprezentace prostředí a základní problém navigace mobilního robota. Dále je uveden seznam typických úloh ze třídy *pursuit–evasion*, na který navazuje popis základních postupů používaných při řešení těchto typů úloh. V druhé polovině kapitoly je představen problém hledání utajené cesty, tzv. skryté navigace (*Covert Robotic Navigation*), který řešen algoritmem *Dark Path*. Experimentální ověření řešení tohoto problému je uvedeno v kapitole 2.

Kapitola 1

Přehled úloh

1.1 Úvod

Jedním z cílů této práce je vytvoření přehledu základních úloh ze třídy *pursuit–evasion*. V mobilní robotice však existuje velké množství jiných úloh a v téměř každé úloze mobilní robotiky tvoří základní problém navigace robota. V různých úlohách je kladen důraz na různé požadavky a pro řešení jednotlivých úloh se používají různé metody. Mají však společný výsledek, kterým je naplánování pohybů robota. Pro různé výpočetní metody hledání vhodné posloupnosti akcí robota jsou vhodné různé reprezentace prostředí. Navigace a pracovní prostředí (resp. způsob reprezentace) tvoří základní pilíře řešení úlohy mobilního robota, proto jsou popisu těchto pojmů věnovány následující oddíly.

1.1.1 Prostředí

Aby bylo možno úlohy řešit, tj. provádět potřebné výpočty, je nutné zavést modely poskytující matematický popis prostředí, ve kterém se agenti pohybují. Tyto modely dělí území na určité myšlené části a je na ně možno nahlížet jak na globální tak lokální úrovni. Při globálním pohledu se sleduje celé území, při lokálním jen malý prostor v okolí robota. Pro různé typy úloh se hodí různé modely, protože na každém modelu se používají odlišné metody řešení, které jsou také různě složité. Každý model i metoda řešení mají své výhody i nevýhody. Modely prostředí lze dělit do následujících tří skupin [10, 13].

Topologická mapa

Topologická mapa je graf, u níž jsou vrcholy grafu určité významné prvky v prostředí. Sousední vrcholy grafů jsou spojeny hranami, které představují cestu z jednoho vrcholu do druhého. Tato cesta je bezkolizní, může však nastat situace, že některé dva vrcholy jsou sousední avšak neexistuje mezi nimi cesta, kterou by robot byl schopen projet. Jednou z výhod tohoto přístupu je, že na tvorbu mapy nemá vliv neurčitost polohy robota. Problémem je spíše určení významných oblastí. Těmi mohou být různé barevné plochy či útvary. Další možností je rozdělení prostředí na množství menších ploch, např. triangulací.

Vrcholy grafu se pak umísťují do středu těchto ploch. Grafová reprezentace má nízké nároky na kapacity záznamu. V paměti jsou uloženy jen významné body a pro každý z nich jeho sousedi, kterých je např. v případě použití triangulace 1.3.3 nejvýše 3. Dále jsou dobře známy algoritmy na prohledávání grafů. Graf se na rozdíl od mřížkové reprezentace skládá z mnohem menšího počtu bodů, a tudíž jsou v určitých případech tyto algoritmy rychlejší.

Mřížková mapa obsazenosti

Pracovní prostor je rozdělen mřížkou na čtverce či šestiúhelníky, které se nazývají buňky. Ve své podstatě se jedná také o graf, kde každou buňku lze považovat za vrchol a se sousedními buňkami ji spojit hranami. Každá buňka má přiřazenu hodnotu, zda je volná, či obsazená tj. je-li na ní překážka. Je možné použít dva základní typy mřížek, čtvercové a šestiúhelníkové. Pro čtvercovou mřížku má každý nehraniční bod 8 sousedů, jejichž vzdálenost je 1 k ortogonálním sousedům nebo $\sqrt{2}$ k diagonálním sousedům. Je možno pracovat se všemi osmi sousedy, tzv. „osmi-okolí“, v některých případech je však vhodné pracovat jen se 4 ortogonálními sousedy, tzv. „čtyř-okolí“. U hexagonálních mřížek má každý vrchol 6 sousedů, jejichž vzdálenost je vždy 1, což má své výhody. Hexagonální mřížky mají však i nevýhody. Především je jejich vytvoření složitější. Existuje také jen 6 směrů, kterými je možné se při plánování cesty vydat. U čtvercové mřížky jich může být až 8.

Jedním z klíčových parametrů mřížkové reprezentace prostředí je rozměr buněk. Ten bývá odvozen od velikosti prostředí. Pokud budou buňky příliš malé lze dosáhnout velmi přesné reprezentace prostředí. Buněk však bude mnoho a výpočet bude taktéž trvat velmi dlouho. Oproti grafu zde také hraje roli přesné určení polohy robota. Aby bylo možné využít potenciál přesné mapy, je nutné také přesně navigovat robota, což může být složité. Při použití příliš velkých buněk se zmenšuje přesnost, ale snižuje výpočetní náročnost. Další nevýhodou mřížky oproti grafu je, že neposkytuje systém na reprezentaci symbolických entit, jako jsou dveře a další objekty.

Přístup řešení úlohy skryté navigace řešený v této práci vyžaduje reprezentaci prostředí jako mřížku. Proto jsou dále uvažovány především čtvercové mřížky s osmi-okolím.

Hybridní přístup

Jedná se o kombinaci předchozích modelů. Mapa se stejně jako v případě grafu skládá z vrcholů a hran. Hrany i uzly lze však popsat i mřížkovou mapou obsazenosti. Hybridní přístup se snaží spojit výhody obou předcházejících přístupů, tj. odolnost při nepřesnostech lokalizace z topologického modelu a přesnost z mřížkové mapy.

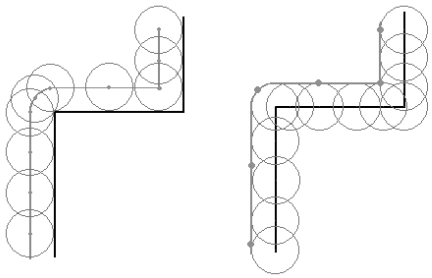
1.1.2 Navigace

Navigace tvoří elementární dovednost mobilního robota a rozumí se jí jednoduché plánování pohybu ze místa A do místa B. Robot se pohybuje v pracovním prostředí, ve kterém

se vyskytují různé překážky. Prostředí může být dvourozměrné nebo třírozměrné. Při řešení úlohy je nutné naplánovat cestu, tj. posloupnost bodů, mezi nimiž je robot schopen se navigovat. Cesta je plánována s ohledem na některé požadavky. Především je požadována cesta bezkolizní. Bezkolizní cesta je taková, která se vyhýbá překážkám. Další požadavky bývají kladeny například na délku cesty nebo bezpečnost cesty.

Při navigaci robota se setkáváme s několika problémy. Jedním z nich je problém lokalizace, jejímž cílem je určit přesnou polohu robota. Polohou robotu ve dvourozměrné prostoru se rozumí souřadnice x, y a natočení φ, v případě třírozměrného prostoru je parametrů šest: x, y, z a tři parametry udávající natočení ve třech osách. Obecné řešení tohoto problému je složité a tvoří základní úlohu mobilní robotiky. Jedním z možných způsobů lokalizace kolového robota je odometrie, která počítá ujetou vzdálenost z odvalování kol robota. Hlavním problémem tohoto přístupu je narůstající chyba, která se nejvíce projevuje při chybě měření natočení robota, například v důsledku proklouznutí koleček. Tato chyba během pohybu robota narůstá a je nezbytné provádět další korekce, například využitím senzorických dat poskytovaných sonary nebo laserovými dálkoměry. Samotná senzorická data jsou však také zatížena zatížená šumem a chybami např. vícenásobné odrazy či pohlcení vyslaného signálu. Pro řešení těchto problémů se používají matematické metody korekce získaných dat, založené především na pravděpodobnostních přístupech [11].

Plánování cesty je jednoduché pro bodového robota. To je robot nulových rozměrů a tudíž není nutné v průběhu plánování cesty řešit některé problémy související například s otázkou, zda-li je robot schopen projet úzkým otvorem v překážce nebo zatočit v místě ohybů. Výsledná cesta však musí být však bezkolizní a musí respektovat skutečné rozměry robota, proto se pro plánování cesty používá takzvaný konfigurační prostor. V tomto prostoru je uvažován bodový robot a však překážky jsou zvětšeny tak, aby při sledování nalezené cesty skutečným robotem nedocházelo ke kolizím s překážkami. V nejjednodušším případě lze získat konfiguračním prostor tzv. „nafouknutím“ překážek. Pokud lze robota považovat za diskového je nafouknutí velmi snadné. V případě použití mřížkového modelu prostředí, spočívá princip této metody v opsání kružnic o poloměru rovném poloměru robota kolem bodů na nichž se nachází překážka. Všechny body ležící uvnitř kružnice se pak považují za překážky, které se tak vlastně zvětší o stejnou velikost, jako se robot zmenšil. Robot si zároveň udržuje určitou vzdálenost od skutečné překážky. Příklad nafouknutí překážek je uveden na obrázku 1.1.



Obrázek 1.1: Nafouknutí překážek.

S problémem navigace se setkáváme v následujících typických úlohách mobilní robotiky:

- explorace - prohledávání a mapování neznámého prostředí,
- inspekce - prozkoumávání či kontrola,
- *pick up and delivery* - vyzvedni a dodej,
- *pursuit–evasion* - stojí proti sobě 2 hráči s odlišnými zájmy.

Posledně jmenovaná třída úloh tvoří hlavní náplň této práce a popis úlohy tohoto typu je věnován následující oddíl.

1.2 Úlohy typu *pursuit–evasion*

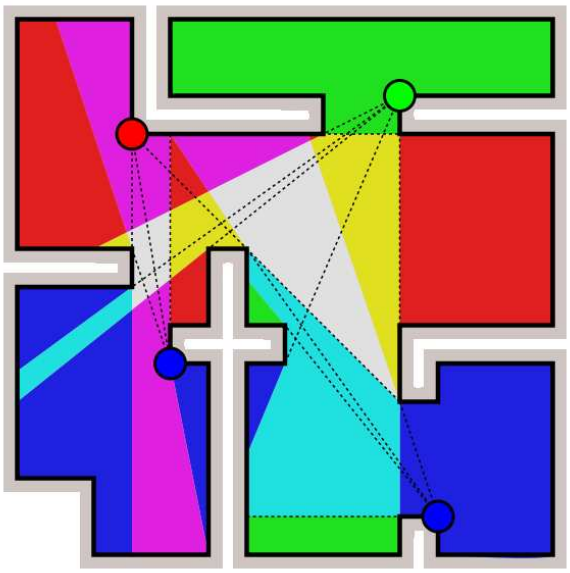
Úlohy ze třídy *pursuit–evasion* si jsou velmi podobné a někdy není možno úlohu přesně klasifikovat, neboť dochází k prolínání různých úhlů pohledu na úlohu. Například úloha střežení galerie je z pohledu stráží tzv. Art Gallery Problem 1.2.1, z pohledu možného vetřelce se jedná o problém skryté navigace 1.2.4. Tento jednoduchý příklad demonstruje, že jednotlivé úlohy mají některé části společné a také jsou ve většině úloh používány stejné přístupy řešení. Některé základní společné přístupy jsou dále uvedeny v oddíle 1.3. V následujících odstavcích jsou představeny nejvýznamnější typy úloh ze třídy *pursuit–evasion*:

- Art Gallery Problem,
- Hide and Seek,
- Capture the Flag,
- Covert Robotics Navigation.

1.2.1 Art Gallery Problem

Jedná se o úlohu, ve které je cílem chránit prostor před mobilními vetřelci rozmístěním statických stráží. Úkolem je najít minimální počet a vhodné rozmístění stráží. Galerie je představována jednoduchým polygonem, tj. takovým, kde se žádné dvě hrany neprotínají [16]. Stráže představují kamery a jsou schopny vidět v rozsahu[1] 360°. Problém zavedl v roce 1973 americký matematik Victor Klee, kdy původem českému matematikovi Vaškovi Chvátalovi položil otázku, kolik stráží je nutno pro střežení galerie. Existuje mnoho variant tohoto problému, kdy jsou např. dány nějaké omezující podmínky, mezi které může patřit nutnost umístění kamer na hrany či vrcholy. Někdy není požadavek střežení celého prostoru, ale jen jeho podmnožiny. Téměř všechny varianty jsou však NP-těžké a lze tak očekávat pouze nalezení přibližného řešení [5]. Příklad řešení Art Gallery je na obrázku 1.2.

[1] Neznamená to, že nutně musí vidět skutečně v celém tomto rozsahu, ale jsou schopny se velmi rychle otáčet a předem nejde nikdy odhadnout, jakým směrem jsou právě natočeny.



Obrázek 1.2: Příklad řešení úlohy hlídání galerie: červené, zelené a modré oblasti vidí jedna stráž, žluté, fialové a světle modré dvě stráže, bílou tři.

1.2.2 Hide and Seek

Český ekvivalent tohoto problému je „hra na schovávanou“. Hry se účastní dvě minimálně jednočlenné skupiny robotů. Úkolem jedné skupiny je vyhledat takové místo, kde by nebyli vidět a druhá skupina robotů se je pak snaží najít. V mnoha případech tvoří druhou ze skupin například lidé. Jednou z aplikací, která se zaměřuje na nalezení chování robotů, kteří se snaží něco najít, je použití robotů při záchranných akcích k vyhledávání lidí v hořích či zřícených budovách. Opačné případy, kdy se robot snaží schovat, jsou různé špionážní mise či monitorování území obsazeného únosci [15].

1.2.3 Capture the Flag (CTF)

Tento typ úloh reprezentuje problém, kdy je úkolem protihráčů nepozorovaně proniknout na území protivníka, z určitých pozic něco získat a vrátit se zpět na své území, přičemž protivník se v tom snaží zabránit. Je nutné zaměřit se jak na ochranu vlastního území, tak nepozorovaně pronikat na území nepřátele [17].

1.2.4 Covert Robotics Navigation

Problém zavedl Ray Jarvis a jeho spolupracovník Mohamed Marzouqi [7] a dá se přeložit jako problém skryté navigace, při které je cílem najít utajenou cestu, tj. takovou cestou, na které nebude robot spatřen. Problém se zabývá navigací mobilního robota či týmu robotů v určitém prostředí za účelem splnění předem daných úkolů, přičemž je důležité soustředit

se na následující optimalizační kritéria: minimalizace času, po který je robot strážemi viděn, minimalizace počtu stráží, jimiž je robot viděn a maximalizace vzdálenosti, ze které je robot viděn. Strážemi mohou být jiní roboti, lidé nebo kamery a podle toho mohou, či nemohou být schopny pohybu. Tento problém je podrobně vysvětlen v oddíle 1.4, ve které je představeno řešení založena na takzvaném *Dark Path* algoritmu, který představuje rozšíření algoritmu *Distance Transform* pro hledání cesty na mřížce v úloha *pursuit–evasion.*

1.3 Základní postupy používané při řešení úloh *pursuit–evasion*

Tak jako u mnoha dalších úloh mobilní robotiky, se při řešení úloh třídy *pursuit–evasion* používají elementární postupy jako je nafukování překážek (popsané v oddíle 1.1.2). Dále se používají specifické metody pro řešená společných základních problémů typických pro třídu úloh *pursuit–evasion.* Jsou to například výpočet viditelných a neviditelných částí pracovního prostorou z jiných částí, hledání vhodných úkrytů apod. Základními pojmy jsou viditelnost a viditelnostní ohodnocení. Samozřejmě je také nutné plánovat cestu bezkolizní, tj. vyhnout se překážkám. Dále jsou popsány některé postupy používané při řešení úloh. Výpočet viditelnostního ohodnocení a algoritmus *Distance Transform* jsou využity při řešení úlohy *Covert Robotics Navigation.*

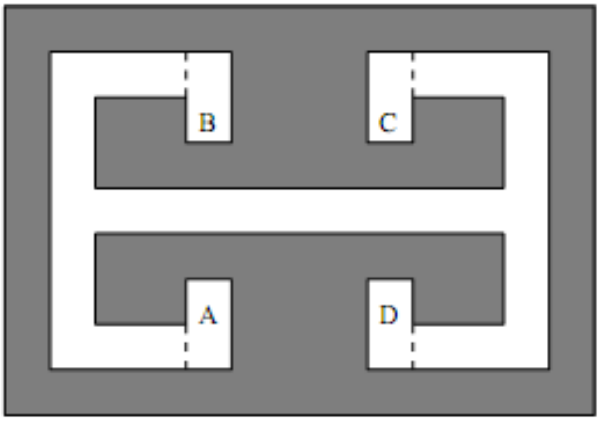
1.3.1 Viditelnost

Jedním z nejdůležitějších pojmů v úlohách *pursuit–evasion* je viditelnost. Pomocí viditelnosti se zjišťuje, které části pracovního prostoru jsou či nejsou z určitých pozic vidět. Viditelnost se většinou užívá v případě reprezentace prostředí grafem nebo geometrické reprezentace např. polygonem. V případě reprezentace mřížkou lze také jednoduše označit, které buňky jsou vidět z jedné konkrétní. Místo pojmu viditelnost je vhodnější zavést pojem *viditelnostní ohodnocení.* Použitím viditelnostního ohodnocení se buňkám přiřazuje určitá hodnota. Ta určuje nejen zda je buňka vidět, ale „jak dobře“ je vidět. Hodnota záleží především na vzdálenosti, z níž je daná buňka pozorována. Vzdálenější buňky jsou „méně“ viditelné než ty bližší, což lépe vystihuje reálné případy. Viditelnostní ohodnocení však může být ovlivněno i dalšími parametry.

1.3.2 Náhodná strategie

Jedním z přístupů řešení úloh střežení či pronásledování je použití náhodné strategie. Jedná se o jednoduchou možnost jak zajistit, aby vetřelec neodhalil strategii strážců, a nemohl tak naplánovat svoji taktiku. Používá se zde pojem viditelnost, kdy se určí části prostoru z nichž je vidět na jiné části prostoru. Operační prostor je rozdělen na různé útvary a model prostředí bývá graf. V tomto případě se předpokládá, že je uprchlík velmi výkonný a zná polohu pronásledovatele kdykoli. Je dokázáno, že v případě jednoduše souvislého prostoru a použití deterministické strategie je zapotřebí Θ $(log\ n)$ [18] pronásledovatelů

k odhalení uprchlíka, kde n je počet vrcholů území. V případě použití náhodné strategie stačí k zachycení jediný pronásledovatel a to i tehdy, kdy uprchlík zná po celou dobu jeho pozici a má neomezenou rychlost. To je dáno tím, že uprchlík sice zná aktuální pozici pronásledovatele, ale nikoli jeho náhodnou strategii, tudíž nezná jeho následnou pozici. Pronásledovatel dokonce nepotřebuje znát strategii uprchlíka, protože je jeho strategie náhodná. Na obrázku 1.3 je uveden jednoduchý příklad prostředí pro demonstraci činnosti náhodné strategie.



Obrázek 1.3: Ukázka použití náhodné strategie - při použití deterministické strategie pronásledovatel nemusí uspět.

Použití náhodné strategie: V příkladě uvedeném na 1.3 by pronásledovatel nikdy nechytil uprchlíka při použití deterministické strategie. Pronásledovatel by mohl například požívat strategii A-B-C-D. Pak by se uprchlík mohl schovat v B, zatímco pronásledovatel by byl v A. Protože je uprchlík rychlý, mohl by se následně přesunout do D, aniž by byl spatřen.

Uprchlík a pronásledovatel se posouvají diskrétně, nejprve uprchlík a poté pronásledovatel. Při použití triangulace se pak dá úloha postupem času zjednodušovat. Pokud pronásledovatel ví, že se uprchlík ukrývá v určité části, soustředí se jen na tuto část grafu (podstrom). Protože graf neobsahuje smyčky, nemůže uprchlík nikudy uniknout. Pokud nezná podstrom, kde se uprchlík nachází, je situaci nutno řešit jiným způsobem.

1.3.3 Triangulace

Některé problémy, jako například hlídání galerie, je možno řešit triangulací. Prostředí se v tomto přístupu rozdělí na trojúhelníky. Platí, že každý jednoduchý polygon může být rozdělen na trojúhelníky a to dokonce v lineárním čase. Je dokázáno, že trojúhelníků je přesně $n-2$, kde n je počet vrcholů. V tomto případě se předpokládá reprezentace prostředí grafem [14]. Každý vrchol má nejvýše tři sousedy. Dále platí, že v jakémkoli konvexním polygonu, tj. takovém, jehož všechny vnitřní úhly jsou menší než 180°, jsou z každého bodu

polygonu vidět všechny ostatní. Pro Art Gallery Problém z toho vyplývá, že takový polygon může být střežen jedinou kamerou. Jelikož je trojúhelník konvexní útvar, lze prostor dělit právě na trojúhelníky. Stejně tak čtverec nebo lichoběžník jsou konvexní tvary. Ovšem polygon nelze vždy dělit na čtverce a lichoběžníky, ale některé trojúhelníky vzniklé při triangulaci lze tak spojit do větších konvexních tvarů. Tím se zmenší množství potřebných stráží. Z výše uvedeného pak vychází, že pro střežení každého polygonu o n vrcholech je zapotřebí maximálně $\frac{n}{3}$ stráží. «««¡ .mine Pro speciální případy, kdy jsou například uvažovány jen ortogonální (pravoúhlé) polygony, je pak zapotřebí jen $\frac{n}{4}$ stráží [5]. ======= Pro speciální případy, kdy jsou například uvažovány jen ortogonální (pravoúhlé) polygony, je pak zapotřebí jen $\frac{n}{4}$ stráží []. »»»»¿ .r69 Ve 3D prostoru je problém složitější. Neplatí totiž, že umístěním stráže do každého vrcholu bude celý prostor monitorován [5]. Důkaz, že pro hlídání galerie postačuje $\frac{n}{3}$ stráží, zjednodušil Steve Fisk pomocí obarvení grafu, vzniklého triangulací polygonu [14].

1.3.4 Algoritmus Distance Transform (DT)

DT algoritmus je založen na *Wavefront* algoritmu, který se považuje za numerickou metodu potenciálových polí [12]. Pro každou volnou buňku, kromě cílové, které je přiřazena hodnota 0, je vygenerována hodnota (potenciál) odpovídající minimálnímu počtu kroků z dané buňky do cíle. Ze startovní buňky pak postupuje vždy k buňce s nejmenší hodnotou až do cíle. Jednou z výhod tohoto algoritmu je, že v případě existence více vhodných cílových pozic je schopen pracovat se všemi. Pak hledá cestu k nejlepšímu z nich, kde nejlepší je ten, k němuž vede nejkratší cesta z dané startovní buňky. Algoritmus začíná generovat hodnoty od cílové buňky jejíž ohodnocení je 0 a v ostatních buňkách je to velmi vysoké číslo (např. počet všech buněk). V dalších krocích přiřazuje hodnoty ostatním buňkám. Pro všech 8 sousedních buněk je zjištěna jejich hodnota a vzdálenost k této buňce. Vzdálenost je eukleidovská (EV), tzn. 1 ke kolmým a $\sqrt{2}$ k úhlopříčným sousedním buňkám. Hodnota přiřazená buňce je pak minimum ohodnocení určité sousední buňky a vzdálenosti k této buňce. Generování je prováděno tak dlouho, dokud se přiřazované hodnoty mění. Hodnota, nebo-li ocenění přiřazená každé buňce, b je následující:

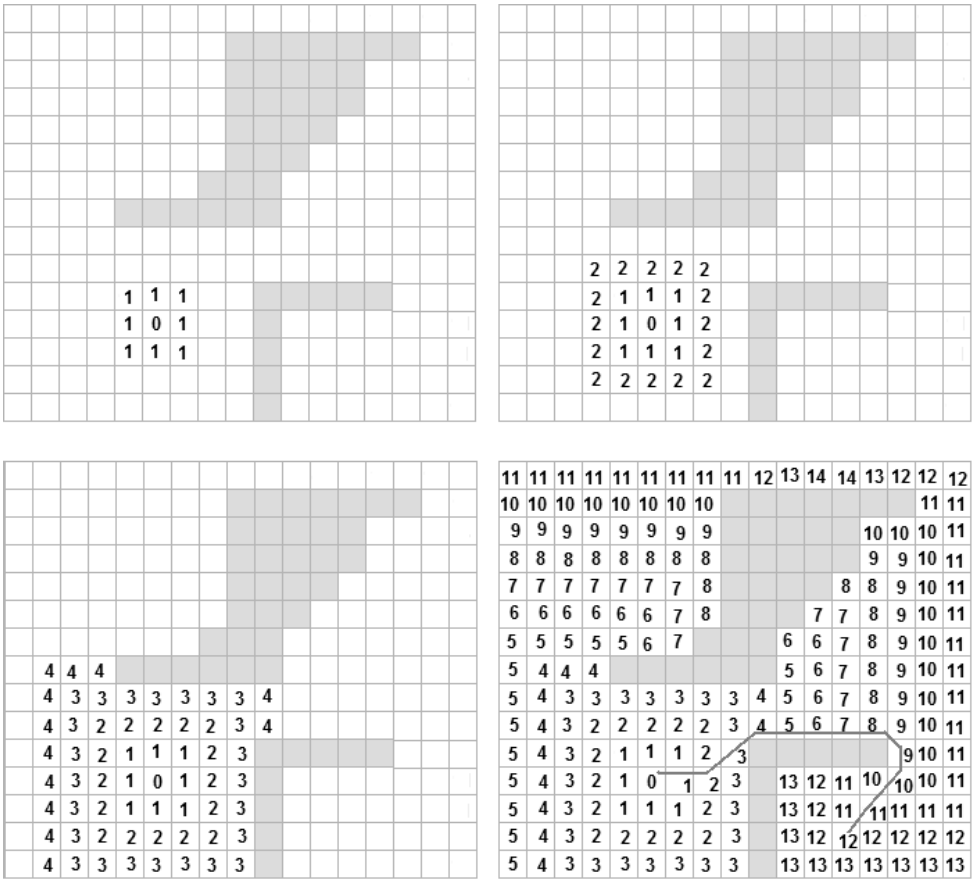
$$cena(b) = \min_{i=1}^{8}[cena(b_i) + EV_{s_i,b}]. \tag{1.1}$$

Výhodou tohoto algoritmu je, že na rozdíl od potenciálových polí nemůže uváznout v lokálním extrému [12]. Další výhodou je, že do ceny oceňující vzdálenost, je možné zanést další faktory. Toho je také využito při řešení problému hledání utajené cesty - *Covert Robotics Navigation*.

Dalších možností využití přidání ocenění je případ robota, který má pomáhat hasičům při vyhledávání a záchraně osob v budovách zasažených požárem. Robot se nesmí dostat do míst zasažených požárem. Zatímco u úlohy skryté navigace jsou pro robota nebezpečná takové pozice, na nichž bude spatřen, u robota je to nebezpečí přímého kontaktu s ohněm. Toto je velmi zajímavá vlastnost DT a lze najít mnoho dalších aplikací, kde je ji možné využít, a které budu přičítat různé faktory [9].

Následující příklad ukazuje možné řešení úlohy chycení uprchlíka algoritmem DT. To znamená docílit toho, aby se uprchlík dostal do pozice, ze které nebude úniku. Uprchlík se snaží dostat do konkrétní bezpečné pozice. Pokud této pozice dosáhne, tak vítězí. Oba znají pozici toho druhého a pohybují se stejnou rychlostí. Pro pronásledovatele i uprchlíka se vytvoří mapa prostředí a vypočítá se DT. Ohodnocení vypočtené algoritmem DT znamená, kolik času hráč potřebuje k dosažení dané pozice ze své současné pozice. Platí, že pozice, kde má pronásledovatel ohodnocení větší než uprchlík, jsou potenciálně bezpečné, ale ne tak, aby zde uprchlík mohl trvale zůstat. Pokud se pronásledovatel pohybuje například dvojnásobnou rychlostí, pak je schopen každé pozice dosáhnout za poloviční čas a ocenění všech jeho pozic se vydělí dvěma [8].

Obrázek 1.4 naznačuje postup výpočtu ohodnocení buněk. Pro jednoduchost znázornění je namísto eukleidovské vzdálenosti brána vždy vzdálenost 1. V obrázku vpravo dole je také zakreslena jedna z možných cest z určité buňky do cíle. Dále je uveden algoritmus na výpočtu DT [6] v pseudokódu 1.



Obrázek 1.4: výpočet DT, šedě překážky, 0 značí cíl. Vpravo dole vyznačena cesta.

Algoritmus 1: Algoritmus Distance Transform.

```
1  for y:=0 to yMax+1 do
2      for x:=0 to xMax+1 do
3          if goal [x,y] then
4              cell [x,y]:=0;
5          else
6              cell [x,y]:=xMax*y Max;
7          end
8      end
9  end
10 repeat
11     for y:=2 to yMax do
12         for x:=2 to xMax do
13             if not blocked [x,y] then
14                 cell [x,y]:= min (cell[x-1,y]+1, cell[x-1,y-1]+√2,cell[x,y-1]+1,
                   cell[x+1,y-1]+√2,cell [x,y]);
15             end
16         end
17     end
18     for y:=yMax-1downto 1 do
19         for x:=xMax-1 downto 1 do
20             if not blocked [x,y] then
21                 cell[x,y]:=min(cell[x+1,y]+1,cell[x+1,y+1]+√2,cell[x,y+1]+1,cell[x-
                   1,y+1]+√2,cell[x,y]);
22             end
23         end
24     end
25 until no change ;
```

1.4 Covert Robotics Navigation

Úloha skryté navigace (*Covert Robotics Navigation*) se zabývá problémem, při němž je třeba robota navigovat tak, aby nebyl spatřen strážemi protihráče, jak bylo zmíněno v oddíle 1.2.4. Možné řešení tohoto problému navrhli Mohamed S. Marzouqi a Ray A. Jarvis z Monash University v Austrálii [7]. Řešení, které navrhli, pracuje s mapou prostředí jako čtvercovou mřížkou s osmiokolím. Přístup řešení byl nazván *Dark Path* algoritmus (DP). Jedná se o algoritmus vycházející z algoritmu DT. Algoritmus DP využívá vlastnosti DT, který do ohodnocení buněk umožňuje kromě vzdálenosti započítat i další faktory. V případě algoritmu DP se jedná o viditelnost, resp. viditelnostní ohodnocení. Viditelnostní ohodnocení je v tomto přístupu důležitý pojem a přesnost jeho výpočtu velmi ovlivňuje výpočet výsledné cesty.

DP algoritmus má několik výhodných vlastností, především to, že dokáže pracovat s různou měrou počátečních znalostí. Znalostmi se rozumí především mapa prostředí a poloha stráží. Všechny znalosti mohou být úplné, částečné nebo žádné. Mapa prostředí je důležitá, aby robot mohl plánovat své pohyby. Pokud mapa není známa, je nutné provést mapování prostředí. To se provádí současně s hledáním skryté cesty, takže množství znalostí o mapě narůstá. Dále je možno znát polohu stráží. Ta je v případě statických stráží neměnná a tudíž se označuje jen jako poloha stráží. V případě pohyblivých stráží je nutno rozlišovat počáteční polohu a aktuální polohu v daném čase. Je možno znát jen počáteční polohu, ale také pohybovou strategii a z ní pak určit polohu v určitém čase. Je uvažován robot vybavený všesměrovým viděním, které slouží k detekci předem neznámých stráží. Schopnosti stráží jsou neznámé a předpokládá se, že jsou podobné schopnostem robota.

Další důležité předpoklady jsou, že prostředí je statické, a překážky jsou neprůhledné a vyšší než jakýkoli agent, tak aby se robot mohl skrýt. Různé možné modifikace jsou popsány v oddíle 1.4.2 na závěr této kapitoly.

Viditelnostní ohodnocení určuje, na které buňky je z jedné konkrétní, na niž je umístěna stráž, vidět přičemž záleží z jaké vzdálenosti. Pro potřeby výpočtu viditelnostního ohodnocení se zavadí viditelnostní mapa. Ta je reprezentována mřížkou se stejnými rozměry jako mapa prostředí. Každé volné buňce je přiřazena její viditelnostní hodnota, kterou je nutné určit co nejpřesněji. Jedním z problémů je jak ji určit, neboť například vzdálenost je měřitelná a tudíž je to ohodnocení velmi přesné, ale viditelnostní ohodnocení závisí na mnoha dalších parametrech, například vlastnostech pozorovatele a ty nemusí být určeny přesně. Podle závislosti viditelnostního ohodnocení na různých parametrech je možné zavést různé modely viditelnosti. Viditelnostní ohodnocení také do značné míry závisí na aktuálních znalostech robota o prostředí. Po výpočtu viditelnostní mapy je použit vzorec (1.2), určující ohodnocení (cenu) každé buňky b.

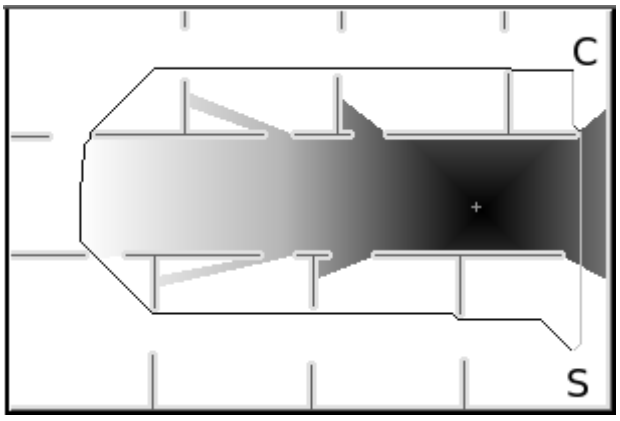
$$cena(b) = \min_{i=1}^{8}[cena(b_i) + EV_{s_i,b}] + \alpha V(b). \tag{1.2}$$

Jednotlivé proměnné znamenají viditelnostní ohodnocení $V(b)$ a míru krytí α, které je dáno reálným nezáporným číslem, $\alpha \geq 0$. Je zřejmé, že pokud je $\alpha = 0$ viditelnost se neuvažuje a výsledná nalezená cesta je nejkratší. Viditelnostní ohodnocení buňky b lze

spočítat dle vzorce (1.3), kde je uvažován jednoduchý model viditelnosti, který uvažuje lineární závislost viditelnostního ohodnocení na vzdálenosti, které s rostoucí vzdáleností klesá.

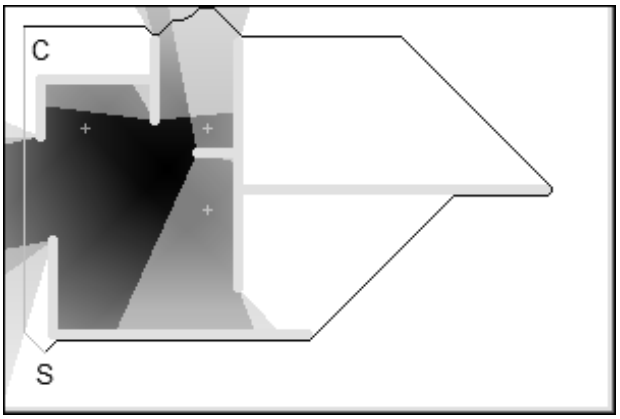
Parametrem α lze vhodně zkombinovat vzdálenost k cíli a míru možnosti být spatřen a nalézt tak nejvhodnější cestu. Výsledná cesta je pak taková, kde suma ujetých vzdáleností mezi jednotlivými buňkami vážená mírou viditelnosti na dané buňce je minimální. To platí i pro případ více cílů, kde je pak jako cíl zvolen ten, do něhož vede cesta s minimálním ohodnocením. Pro různé případy jsou potřeba cesty s různou úrovní krytí. V mnoha případech platí, že pokud je robot spatřen, je mise neúspěšná. Ne vždy je sice možno dosáhnout úplného krytí, ale přesto tento algoritmus dokáže najít takovou cestu, kde bude možnost spatření minimální. Cenou za to, že robot není viděn bývá většinou delší doba trvání mise. Pokud je zapotřebí rychlejší splnění úkolu, je zřejmé, že robot může být spatřen častěji. Je tedy nutné správně zvážit rizika a čas trvání mise, který se předpokládá přímo úměrný ujeté vzdálenosti.

Nastavení koeficientu α je však velmi složité, protože různé faktory ovlivňují, zda je nutné dosáhnout cíle rychle či skrytě. Nelze tedy stanovit jednoznačný postup jeho určení. Následující obrázky ukazují, jak by v určitých případech mohlo vypadat řešení. Světlá křivka je značí nejkratší cestu, tmavá cesta skrytou. Pozice stráží jsou značeny křížky. Na obrázku 1.5 je vidět případ, kdy ve velké vzdálenosti od strážce je robot hůře vidět. Na obrázku 1.6 je případ, kdy je možno volit ze dvou cest, krátké, kde však robot bude spatřen třemi strážemi, či delší, kde bude spatřen jediným strážcem. Více experimentálních výsledků je uvedeno v oddíle 2.



Obrázek 1.5: Příklad utajené cesty - je zvolena delší cesta (tmavá křivka), k zahlédnutí robotu dojde ve velké vzdálenosti, na kterou je robot strážcem hůře vidět.

Po nalezení ohodnocení se robot pohybuje ze startu vždy na sousední buňku s nejmenší hodnotou až do cíle. Základní postup hledání řešení je stejný pro různé úrovně počátečních znalostí (pozice stráže, mapa).



Obrázek 1.6: Příklad utajené cesty - je zvolena delší cesta, na které však dojde k zahlédnutí pouze jediným strážcem.

1.4.1 Výpočet viditelnostního ohodnocení

Výpočet viditelnostního ohodnocení může být náročný. V případě použití jednoduchého modelu viditelnosti nejvíce záleží na úrovni počátečních znalostí. Postupy výpočtu viditelnostního ohodnocení pro různé úrovně znalostí jsou uvedeny v následujících oddílech.

Známá poloha stráží a mapa prostředí

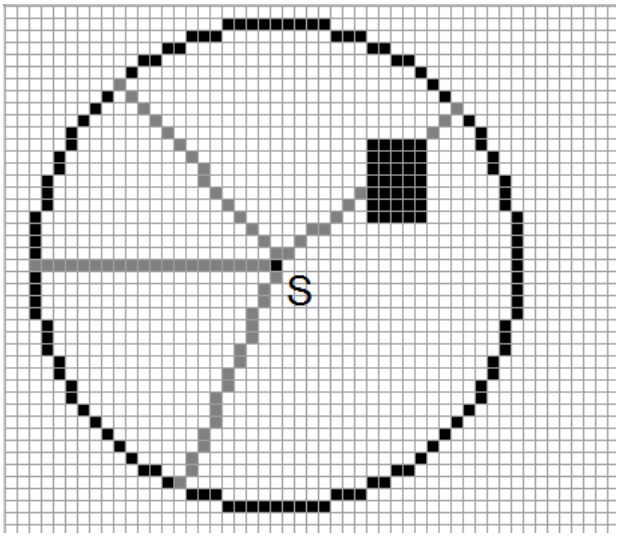
V tomto případě jsou znalosti o prostředí kompletní a výpočet nejjednodušší. Je tedy možno okamžitě ohodnotit všechny buňky a nalézt tak výslednou cestu. Pokud jsou uvažovány dynamické stráže, je nutno znát jejich polohu po celou dobu. Hlavními faktory, které ovlivňují hodnotu viditelnosti dané buňky jsou počet stráží a jejich vzdálenost od buňky. Ve většině případů jsou tyto dva faktory dostačující. Vzdálenost je důležitá jednak proto, že více vzdálený robot je hůře rozpoznatelný a také proto, že pokud už je rozpoznán má více času uniknout. Pokud se stráže pohybují, ale vždy známe jejich pozici, lze po každém kroku viditelnost přepočítat a upravit tak cestu. Podobně lze postupovat i v případě, že se stráž nevyskytuje na předem předpokládané pozici, nebo je naopak odhalena nová stráž.

Výpočet viditelnostního ohodnocení pro každou buňku se provede podle rovnice (1.3), kde $V(b)$ je viditelnostní ohodnocení, d_i je vzdálenost k buňce od stráže i, d_{max} je maximální vzdálenost na jakou stráž vidí. Tato vzdálenost je předpokládána stejná pro všechny stráže. Pokud je tato vzdálenost větší než diagonální délka prostředí l, která je také maximální možnou vzdáleností jakýchkoli dvou agentů, je d_{max} nastavena na $l+1$. S je celkový počet stráží.

$$V(b) = \sum_{i=1}^{S} 1 - \frac{d_i}{d_{max}}, \quad d_i \leq d_{max} \tag{1.3}$$

Pro $d_i > d_{max} : V(b) = 0$. Je patrné, že pro případ jediné stráže je výsledná hodnota každé buňky mezi 0 a 1, kde všechny buňky mající hodnotu 0 jsou pro stráž neviditelné. V algoritmu probíhá výpočet viditelnosti následovně. Nejprve se každé buňce přiřadí po-

čáteční hodnota 0. Pro každou stráž se pak určí buňky, jež vidí. Předpokládá se, že stráž vidí do všech směrů stejně. Za použití Bresenhamova algoritmu, který slouží pro rastrování úseček [1], vede každým směrem myšlená úsečka s počátečním bodem na pozici stráže. Pro každý bod ležící na úsečce se pak spočítá hodnota viditelnosti a přičte se k už vypočtené hodnotě pro danou buňku. Vzdálenost je jen hodnota čítače, jež je zvětšována pro daný směr s každou novou buňkou. Daným směrem se pokračuje, dokud není dosaženo překážky či hranice území. Další body za překážkou jsou pro stráž neviditelné. K zaručení, že se projdou všechny směry lze použít Bresenhamova algoritmu pro kružnici, který na mřížce dokáže vyrastrovat kružnici o daném poloměru a středu [2]. Poloměr kružnice je maximální vzdálenost, na niž stráž vidí a středem je pozice stráže. Po vyrastrování kružnice se jako směry považují jednotlivé body kružnice. Protože při spojování středu kružnice s jejími krajními body je možné, že některé body leží na více úsečkách, je nutné zajistit, že mu bude hodnota přiřazena jen jednou. Postup výpočtu je znázorněn na obrázku 1.7.



Obrázek 1.7: Příklad výpočtu viditelnostního ohodnocení. Stráž je značena *S*. Postupně je ke každému bodu kružnice vedena úsečka a bodům tvořícím úsečku přiřazena hodnota. Body za překážkou jsou pro stráž neviditelné.

Neznámé stráže, známé prostředí

Předpokládá se, že pravděpodobnost spatření stráží na určité buňce je přímo úměrná počtu buněk, z nichž je na buňku vidět. V tom případě buňky, na které je vidět z menšího počtu ostatních buněk, jsou bezpečnější než ty, na které je vidět z více buněk. Je zřejmé, že buňka, na níž je vidět ze všech ostatních, je zajisté viditelná i pro nějakou stráž. Pro tento účel je vytvořena speciální viditelnostní mapa, kde se pro každou buňku určuje tzv. risk faktor. To je číslo od 0 do počtu všech volných buněk. Jednou z metod výpočtu hodnoty je metoda „ray shooting method“. Touto metodou se každé volné buňce postupně v každém

směru zjišťují volné buňky, které jsou vidět. Součet všech zjištěných volných buněk je ohodnocení.

Tento postup je však velmi časově náročný, jeho složitost je $O(n^2)$, kde n je počet volných buněk. Rychlejší metoda využívá toho, že pokud je z buňky A vidět buňka B, je také z B vidět A [12]. Výpočet probíhá tak, že jeden bod, například levý horní roh, je označen jako počáteční. Z tohoto bodu se v určitém směru sčítají viditelné buňky, ovšem nejen pro výchozí buňku, ale pro každou další zjištěnou viditelnou buňku. Tím se postupně zmenšuje počet směrů, ve kterých je viditelnost zjišťována, protože pro další buňky již není nutno počítat viditelné buňky pro směry, ze kterých byla viděna. Na rozdíl od metody, kdy se každá buňka prohledává každým směrem, zde je nutno prohledat každý směr jen pro první buňku. Další možností řešení při neznámých pozicích stráže je předpokládat umístění stráží pomocí pravděpodobnosti. V určitých situacích je pravděpodobnější výskyt stráží v konkrétních místnostech. Je možné použít Gaussovy distribuční funkce v případě, že je známa jen přibližná pozice [12].

Známé stráže, neznámé prostředí

Podobně jako v předchozím případě jsou i zde používány mapa prostředí a mapa viditelnosti. Hodnoty viditelnosti jsou však jen přibližné. Obě mapy jsou pak průběžně s nově získanými informacemi o prostředí aktualizovány. Plánování cesty je v tomto případě rozděleno na:

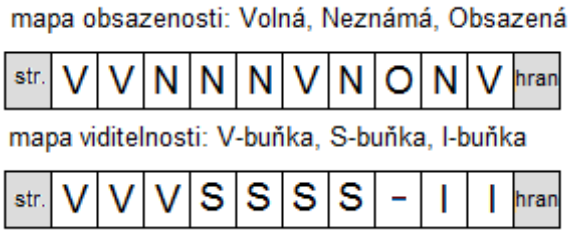
- aktualizaci mapy prostředí, která se děje po určitém počtu kroků,
- aktualizaci viditelnostní mapy, ta se děje současně s aktualizací mapy prostředí,
- přepočet skryté cesty a
- přesun robota o n kroků.

Tyto kroky se opakují dokud robot nedosáhne cílové pozice.

Podobně jako v předchozích případech i zde se buňky mapy prostředí označují jako volné a obsazené, avšak existuje i další možný stav a to buňka „neznámá“. Na počátku jsou všechny buňky mapy prostředí o nichž nejsou žádné informace označeny jako „neznámé“. Později, když je zjištěn skutečný stav buňky, je označena buď jako „volná“ nebo „překážka“.

Dále je pro každou stráž vytvořena také zvláštní soukromá, též privátní, viditelnostní mapa. V každé takové mapě jsou pro buňku tři možné stavy. K určení stavu je nutné zavést pojem viditelnostní linie. Jedná se o množinu bodů ležících na úsečce spojujících danou buňku s určitou stráží. V případě, že viditelnostní linie obsahuje buňku s překážkou, daná stráž na danou buňku nevidí. Taková buňka je pak pro příslušnou stráž označována jako „I-buňka“ (Invisible), neviditelná buňka. Opakem je viditelná buňka „V-buňka“ (Visible), jejíž viditelnostní linie neobsahuje žádnou buňku s překážkou. Posledním typem je „S-buňka“ (unSure), nejistá, jejíž viditelnostní linie neobsahuje žádnou známou obsazenou buňku a nejméně jednu buňku, u níž není znám její stav obsazenosti. Tyto mapy jsou aktualizovány v případě, že je potřeba aktualizovat hlavní viditelnostní mapu. Konečná

hodnota viditelnosti je pak pro každou buňku součet ohodnocení odpovídajících buněk privátních map každé stráže. Určení stavů buněk je pak prováděno pro každý směr od stráže směrem k hranici prostředí.



Obrázek 1.8: Ukázka viditelnostní linie s označením typů buněk.

Viditelnostní ohodnocení je určeno následovně. I-buňky mají hodnotu 0. Hodnota V-buněk je určena rovnicí (1.3). Hodnota S-buněk rovnicí (1.4). Rozdíl je v násobení pravděpodobností, že buňka je V-buňka, která je nepřímo úměrná počtu neznámých buněk pro stráž. Pravděpodobnost je určena jako q^m, kde q je pravděpodobnost, že neznámá buňka je volná a m je počet neznámých buněk na dané viditelnostní čáře. Hodnota q je závislá na prostředí, dá se předpokládat, že je $0,5$.

$$V_u(b) = 1 - \frac{d_i}{d_max} \times q^m \tag{1.4}$$

Po vytvoření hlavní viditelnostní mapy je hledána skrytá cesta použitím rovnice (1.2). Výpočet je tedy shodný jako v případě kompletních znalostí. Robot se pak po cestě pohybuje až do další aktualizace. Jelikož jsou všechny výpočty časově náročné, není vhodné aktualizovat po každém kroku.

Pro zlepšení časové náročnosti existuje několik postupů. Jednak aktualizace privátních map není nutná po každém kroku. Je nutná jen tehdy, pokud je poslední neznámá buňka (S-buňka nebo V-buňka tj. její ohodnocení bylo menší než 0) zjištěna buď jako překážka, nebo volná. V tomto případě je možné přepočítat hodnotu jen dotčených buněk. Pokud se jedná o překážku, pak všechny buňky za ní jsou označeny jako I-buňky. Jedná-li se o volný prostor, jsou všechny následující buňky aktualizovány použitím rovnice (1.4), dokud není dosaženo překážky či hranice. V případě, že je zjištěna více než jedna neznámá buňka (S-buňka nebo V-buňka) jako volný prostor, pak buňky blíže stráži jsou aktualizovány nejdříve. Přitom je nutné každou buňku aktualizovat pouze jednou.

Další snížení výpočetní náročnosti je možné založit na faktu, že viditelnostní mapa nepotřebuje být aktualizována pokud předešlá generovaná cesta obsahuje jen I-buňky. Výjimku tvoří případ, při kterém je na skryté cestě detekována překážka. Pokud není aktualizována mapa, není nutné aktualizovat ani cestu. Další možností je, že robot bude pokračovat v cestě, pokud viditelnostní hodnota nepřekročí určitou přednastavenou hodnotu.

Neznámá pozice stráží, neznámé prostředí

Jedná se o nejsložitější a zároveň nejkomplexnější případ. Řešení je podobné jako v případě neznámých sráží a známého prostředí. Stejně jako v tomto případě platí, že z čím menšího počtu buněk je na konkrétní buňku vidět, tím je menší pravděpodobnost odhalení na této buňce. Hlavním cílem je tedy minimalizace vystavení se volnému prostoru. Hlavním rozdíl je, že neznámé buňky jsou považovány za volný prostor, proto je předpokládán nejhorší možný případ. Na začátku je každé buňce přiřazena maximální hodnota, např. výška × šířka mapy. Tyto hodnoty se postupně upravují, podle toho, jak je odhalována mapa. Bohužel v případě neznámých stráží nelze pro aktualizaci mapy viditelnosti použít postup popsaný pro neznámé prostředí a známé stráže, neboť by výpočet byl velmi časově náročný. Existují jiné metody, u nichž je náročnost menší. Jedna z nich je založena na triangulaci a hledá přibližné ocenění každé buňky [12].

1.4.2 Možnosti rozšíření

Úlohu je teoreticky možno řešit i pro dynamické prostředí. Dynamické překážky jsou například dveře do dalších prostor, které je možno otevírat či zavírat. Pokud ovšem dojde ke změně, nelze jen jednoduše z mapy překážku odstranit či ji tam přidat, je nutné všechny výpočty provést znovu. Jedná se vlastně o nové prostředí. To úlohu výrazně komplikuje, především z pohledu výpočetní náročnosti. Další problém je ve způsobu jak robot zjistí, že ke změně došlo, pokud ke změně dojde v jiné části prostoru, než ve které se robot nachází.

Dalším rozšířením je uvažování průhledných překážky, při kterém je nutné do výpočtu viditelnostního ohodnocení zahrnout i průhlednost překážek. Dále je možné uvažovat, že některé překážky jsou neprůhledné, avšak překonatelné. Jelikož algoritmus používá zvláštní mapu pro prostředí a zvláštní mapu pro výpočet viditelnostního ohodnocení, není složité zanést do každé mapy jiné překážky.

Uvedený algoritmus uvažuje lineární závislost viditelnostního ohodnocení na vzdálenosti, což nemusí odpovídat realitě. Výpočet viditelnostního ohodnocení lze velmi snadno upravit pro jiný model závislosti hodnocení.

Existují i jiné možné způsoby řešení úlohy skryté navigace. Jednou z možností je robot vybavený světlocitlivými senzory. Ten se snaží najít místa s nejmenším osvětlení, neboť je přijat předpoklad, že na těchto místech bude robot nejméně vidět. Tento přístup však do značné míry záleží na světelných podmínkách okolního prostředí. Další používaný postup je minimalizace rozměrů robotů. Je zřejmé, že malý robot, někdy označován jako *Insect* robot, se dokáže lépe skrýt a je hůře rozpoznatelný. I tento přístup má však nevýhody. Především robot se zmenšující velikostí ztrácí i určité schopnosti [12].

Kapitola 2

Experimenty

Cílem bylo zjistit a porovnat výsledky algoritmu hledání utajené cesty v různých situacích, experimenty byly prováděny pro různá prostředí a různá nastavení parametrů algoritmu. Jedním z cílů práce bylo implementovat základní algoritmus pro hledání utajené cesty. Vytvořený program pro nalezení utajené cesty byl následně použit pro nalezení cesty a navigaci robotu podél této cesty. Experimentální ověření řízení robotu bylo provedeno v simulátoru Stage systému Player [4] a na reálném robotu G^2Bot.

2.1 Implementace

Program pro výpočet skryté cesty je implementován v jazyku C++, pro navigaci mobilního robotu je využito rozhraní systému Player/Stage. Algoritmus pro nalezení utajené cesty byl implementován pro případ kompletních znalostí, tj. známa mapa a pozice všech stráží. Stráže jsou uvažovány statické, neboť to j výpočetně méně náročné a není třeba se řešit model pohybů stráží. Mapa prostředí je reprezentována bitmapovým obrázkem, který je použit pro vytvoření mřížkové reprezentace prostředí.

Vstupními parametry programu jsou:

- obrázek s mapou prostředí,
- startovní a cílová pozice, pozice stráží,
- maximální vzdálenost na jakou stráž vidí,
- parametr α,
- rozměr robotu a
- rozměr buňky.

Poslední dva parametry slouží k nafouknutí překážek. Po načtení je vytvořen objekt reprezentující prostředí a každé buňce je přiřazeno, zda je volná či obsazená.

Dále dochází k nafouknutí překážek. Tato metoda byla popsána v oddíle 1.1.2. Velikost nafouknutí překážek je určena z velikostí buňky a velikosti robotu. Pro nafouknutí překážek je použito Bresenhamova algoritmu pro kružnici [2], který dokáže vyrastrovat na mřížce kružnici. Bresenhamův algoritmus pro kružnici se používá i pro výpočet viditelnosti.

Po výpočtu viditelnostního ohodnocení lze provést výpočet pomocí algoritmu DT, do něhož je zahrnuto viditelnostní ohodnocení. Výpočet je proveden s použitím vzorce (1.2). Pro porovnání je také počítána nejkratší cesta, tj. výpočet pro $\alpha = 0$.

Po ohodnocení všech buněk je cesta nalezena gradientním algoritmem a to tak, že počínaje startovní buňkou je do cesty zařazena vždy ta sousední buňka poslední přidané, která má nejmenší hodnotu. Postup končí přidáním cílové buňky. Takto vytvořená cesta obsahuje velké množství bodů (buněk), proto je cesta dále zjednodušena, aby bylo možné při řízení robota zadávat řídicí jednotce pouze ty body na nichž dochází ke změně směru.

Po výpočtu je nalezená cesta použita při navigaci robotu. Program se skládá z následujících kroků (částí):

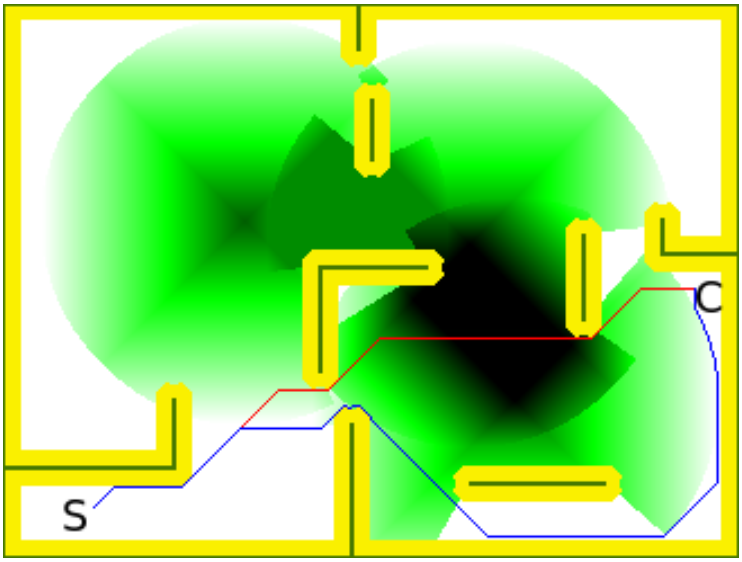
1. načtení mapy,
2. nafouknutí překážek,
3. výpočet viditelnostního ohodnocení,
4. výpočet DP,
5. nalezení cesty,
6. navigace robotu podél nalezené cesty.

Průběh řešení je vizualizován vykreslením dílčích výsledků. Pro vykreslování je použita knihovna SDL [3]. Jsou vykreslovány následující mřížky: mapa prostředí s ohodnocením vzdáleností, tj. pro $\alpha = 0$, mapa s viditelnostním ohodnocením a mapa celkového ohodnocení pro zadané α. Dále je také vykreslena skrytá cesta a jsou rozlišeny překážky nafouklé od skutečných.

V následujících oddílech jsou uvedeny příklady nalezených utajených cest pro různá prostředí. Kromě porovnání délek nalezených cest pro jednotlivé konfigurace je analyzován průběh míry spatření. V závěru jsou uvedeny příklady realizace nalezeného plánu v simulačním prostředí Stage a na reálném robotu.

2.2 Případ 1

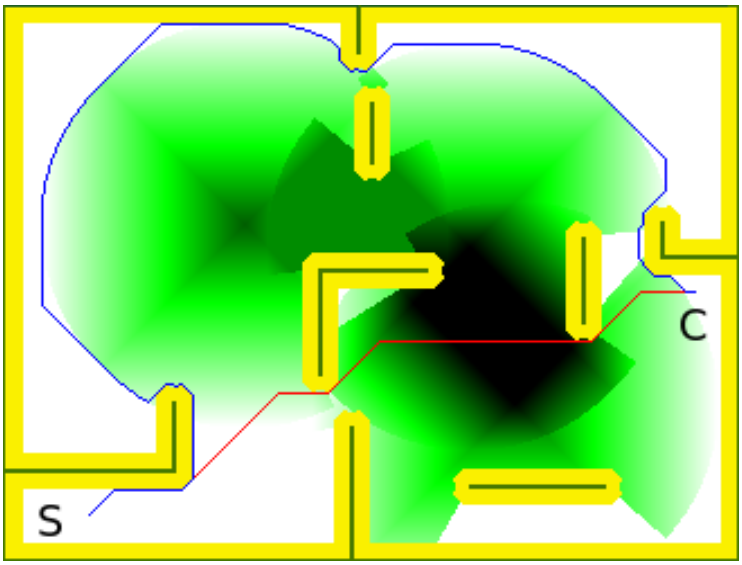
Tento případ se zaměřuje na vliv parametrů α na délky nalezené utajené cesty. Prostředí a umístění strážců je zvoleno tak, aby bylo možné najít krátkou cestu s nízkou mírou krytí tak cestu, která zaručuje vysokou míru krytí a robot není na ni strážemi spatřen. Ze zvolené startovní pozice S do zvolené cílové pozice C tak existuje v prostředí několik cest. Na obrázcích 2.1 a 2.2 jsou zobrazeny dvě různé cesty. V případě na obrázku 2.1



Obrázek 2.1: Nalezená utajená cesta, mapa 1, $\alpha = 30$.

je hodnota parametru α nastavena na 30, v případě na obrázku 2.2 na hodnotu 35. Je-li hodnota $\alpha = 0$, tedy hledání nejkratší cesty, je nalezena cesta v těsné blízkosti dvou stráží, na obrázcích je zakreslena červeně. Se vzrůstající hodnotou α se zvolená cesta od stráží vzdaluje, avšak až do hodnoty $\alpha \approx 4$ vede cesta kolem dvou stráží. Od hodnoty $\alpha \approx 4$ do hodnoty $\alpha \approx 30$ je volena delší cesta, kde dojde k zahlédnutí pouze jednou stráží. Při hodnotě $\alpha \approx 35$ je už volena nejdelší cesta, kde dochází k zahlédnutí z velké dálky. Na obou obrázcích je znázorněno viditelnostní ohodnocení, čím tmavší barva tím je tato hodnota vyšší.

Obrázek 2.3 zobrazuje celkové ohodnocení, tj. jak vliv vzdálenosti, tak vliv viditelnostního ohodnocení. Celkové ohodnocení je použito pro nalezení utajené cesty a je zde zobrazeno pouze pro demonstraci jakých hodnot jednotlivé buňky nabývají. V ostatních případech je použito viditelnostního ohodnocení, ze kterého jsou lépe patrné střežené prostory a způsob jakým se jim utajená cesta vyhýbá. Na obrázku 2.4 je zobrazena míra krytí podél jednotlivých cest. Z grafu je názorně vidět, že v případě nejkratší cesty zobrazené modře je viditelnostní hodnota maximální a míra spatření je maximální po velmi dlouhou dobu. Pro případ $\alpha = 30$, zobrazena červeně je v určitém okamžiku viditelnostní hodnota rovna téměř polovině maximální možné. V případě zelené křivky pro $\alpha = 35$ jsou patrné zákmity, které jsou způsobeny naplánováním cesty přesně podél kružnice na níž stráž vidí.



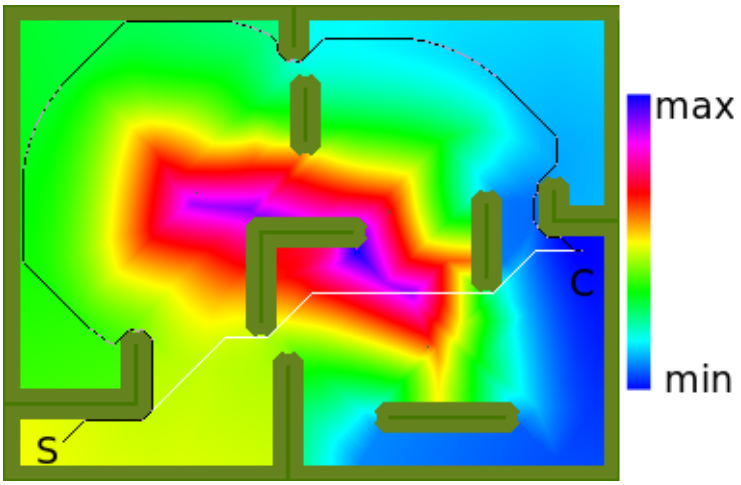
Obrázek 2.2: Nalezená utajená cesta, mapa 1, $\alpha = 35$.

Délky jednotlivých cest jsou shrnuty v tabulce 2.1.

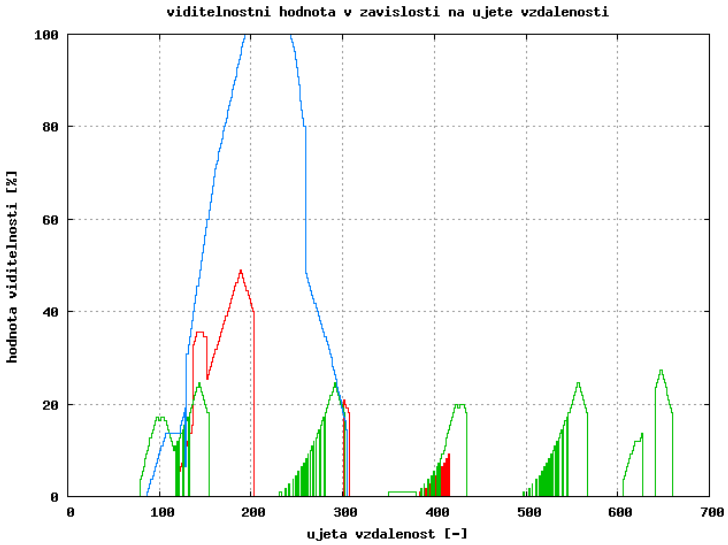
α	**Délka cesty**
0	310,007
30	423,990
35	670,926

Tabulka 2.1: Délka utajené cesty na mapě 1 pro různé hodnoty parametru α.

Z provedených experimentů vyplývá, že hodnota α značným způsobem ovlivňuje výslednou podobu utajené cesty. V závislosti na požadované míře krytí může délka cesta výrazným způsobem delší, přičemž nelze říci, že prodloužení cesty je přímo úměrné požadovanému krytí, neboť krytí je závislé na viditelnostním ohodnocení. Ohodnocení buněk je závislé na charakteru prostředí a vlastnostech stráže, proto může být cesta s vyšším krytí mnohem delší, neboť vyžaduje vyhnutí se strážcům.



Obrázek 2.3: Celkové ohodnocení pro mapu 1, zahrnující jak vzdálenosti tak viditelnostní ohodnocení pro $\alpha = 35$.



Obrázek 2.4: Míra spatření na utajených cestách na mapě 1 pro různé hodnoty parametru α, 100 % je nejvyšší viditelnostní hodnota.

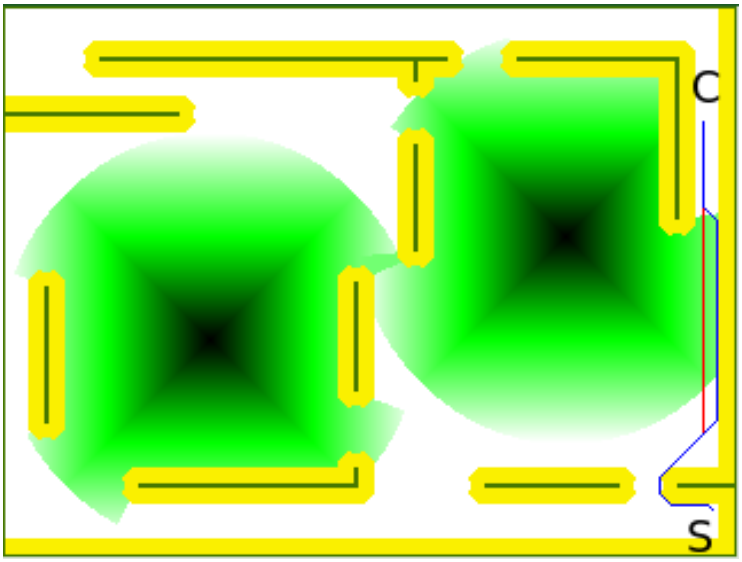
2.3 Případ 2

Tento případ je zaměřen na vliv parametru α na výběr jedné ze pěti možných cest a zjistit při jakém nastavení bude algoritmus volit určitou cestu. Ve všech případech jsou startovní (S), cílová (C) pozice i pozice obou stráží shodné. Výsledná utajená cesta je značena modře, pro srovnání je červeně zakreslena i cesta nejkratší. Jednotlivé délky cest jsou uvedeny v tabulce 2.2.

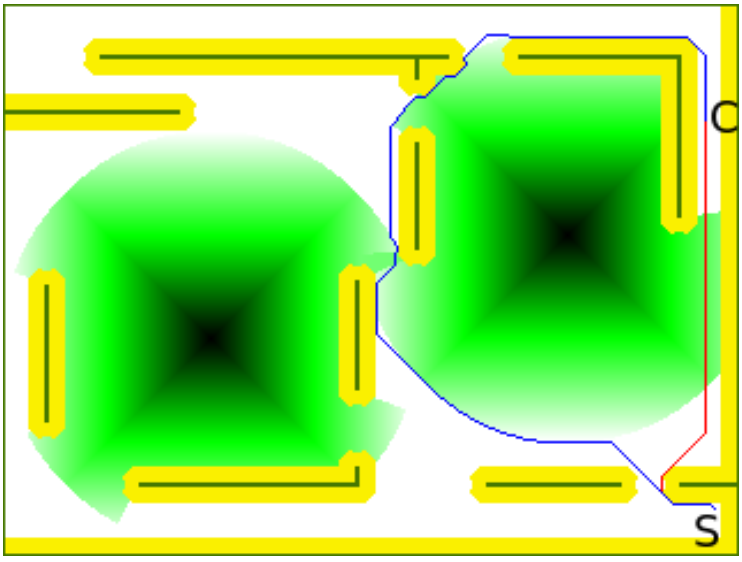
α	**Délka cesty**
0	195,77
20	200,74
30	463,09
40	705,07
50	883,09

Tabulka 2.2: Délka utajené cesty na mapě 2 pro různé hodnoty parametru α.

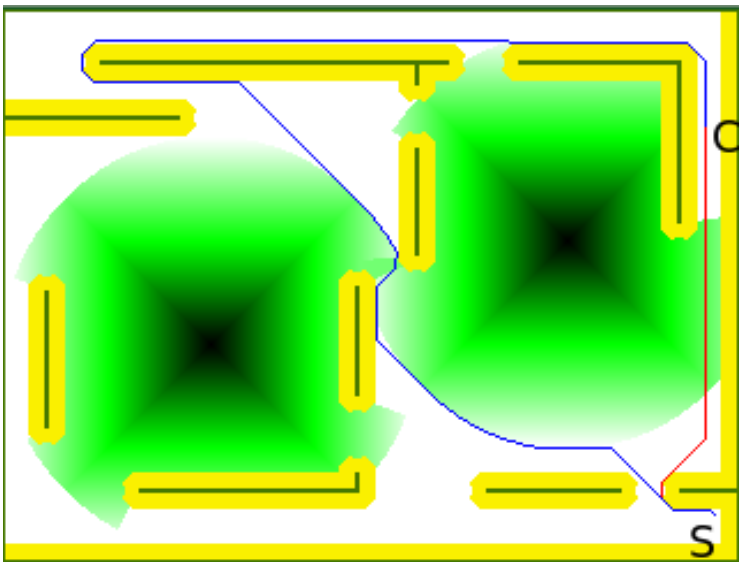
Experimentálně bylo zjištěno, že pro $\alpha \approx 0$ až 0,3 volí algoritmus cestu nejkratší. Pro $\alpha \approx 0{,}4$ až $\alpha \approx 30$ volí cestu zobrazenou na obrázku 2.5. Cestu na obrázku 2.6 volí pro $\alpha \approx$ (30 - 40). Další cesta je pro $\alpha \approx$ (40 - 50). Pro $\alpha > 50$ je vybrána cesta nejdelší, ale také nejvíce krytá. Cesta je zobrazena na obrázku 2.8.



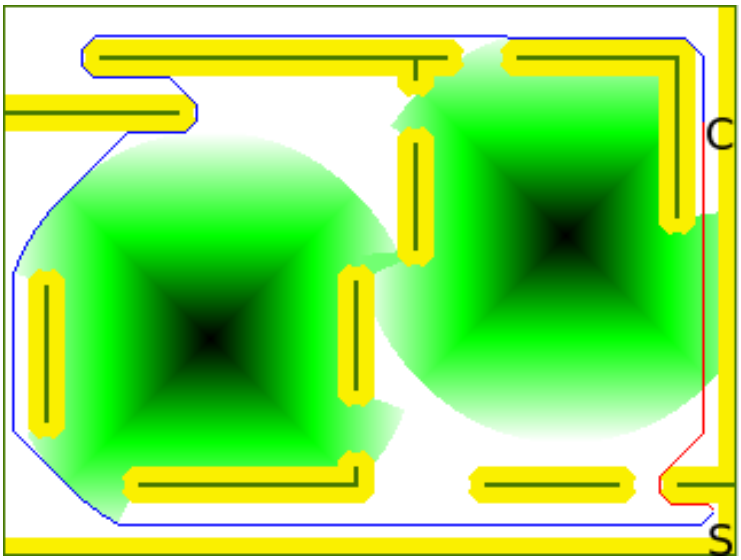
Obrázek 2.5: Nalezená utajená cesta, mapa 2, $\alpha = 20$.



Obrázek 2.6: Nalezená utajená cesta, mapa 2, $\alpha = 30$.



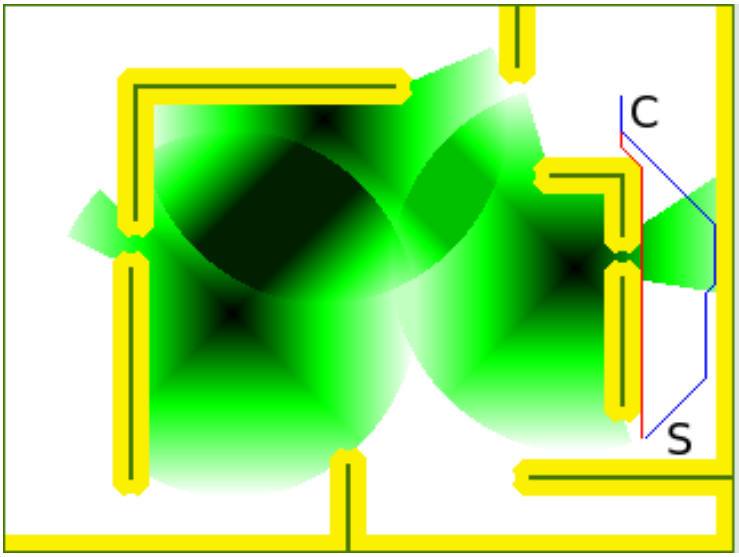
Obrázek 2.7: Nalezená utajená cesta, mapa 2, $\alpha = 40$.



Obrázek 2.8: Nalezená utajená cesta, mapa 2, $\alpha = 50$.

2.4 Případ 3

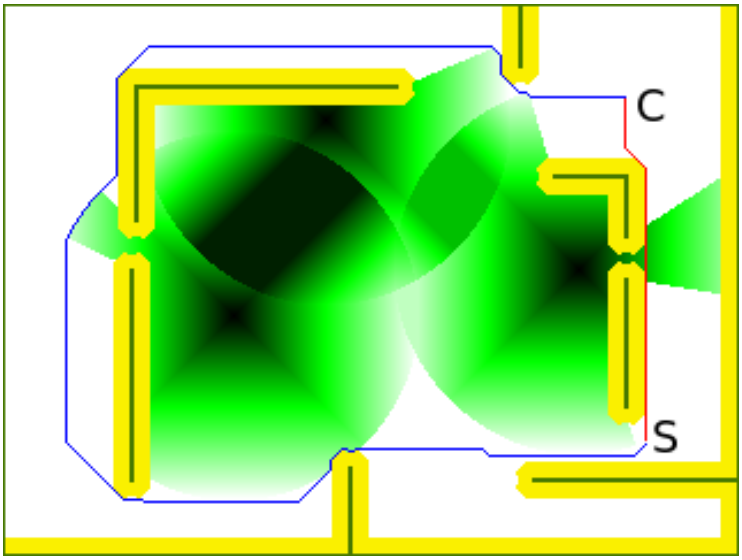
V posledním testovaném případě existují dvě základní cesty. Start a cíl jsou umístěny velmi blízko sebe. První cesta je téměř přímá a krátká, ovšem dojde k zahlédnutí z velmi malé vzdálenosti, avšak po krátkou dobu. Druhá cesta je podstatně delší, a dochází k zahlédnutí z větší vzdálenosti po delší dobu. V případě první cesty se robot s rostoucím α snaží od stráže co nejvíce vzdálit, avšak je omezen hranicí prostoru, modrá cesta na obrázku 2.9. Experimentálně bylo zjištěno, že do hodnoty $\alpha \approx 80$ volí algoritmus první cestu. Druhý případ pro hodnotu $\alpha = 85$ je znázorněn na obrázku 2.10.



Obrázek 2.9: Nalezená utajená cesta, mapa 3, $\alpha = 80$.

α	**Délka cesty**
0	152,728
80	178,409
85	686,492

Tabulka 2.3: Délka utajené cesty na mapě 3 pro různé hodnoty parametru α.



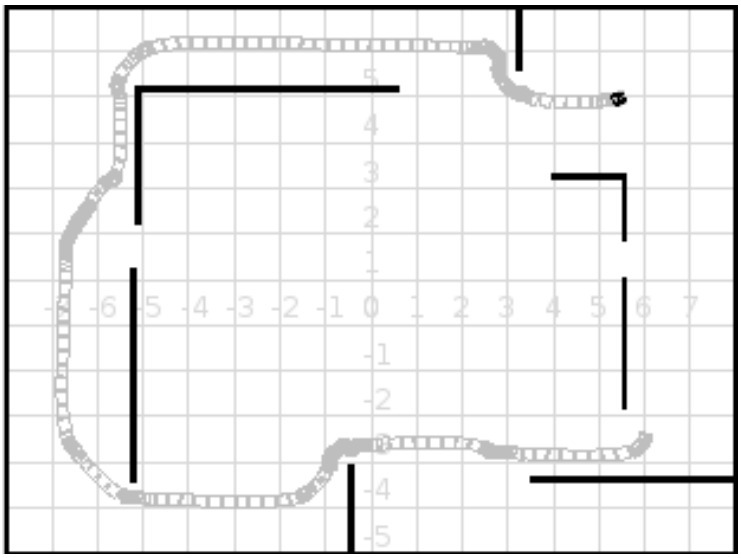
Obrázek 2.10: Nalezená utajená cesta, mapa 3, $\alpha = 85$.

2.5 Navigace podél utajené cesty

Pro vybraná prostředí a nastavení byly provedeny experimenty s robotem. Cílem provedených experimentů bylo ověřit, zda je robot schopen naplánovanou cestu projet. Příklady navigace v simulátoru a v reálném prostředí jsou uvedeny v následujících odstavcích.

2.5.1 Simulace

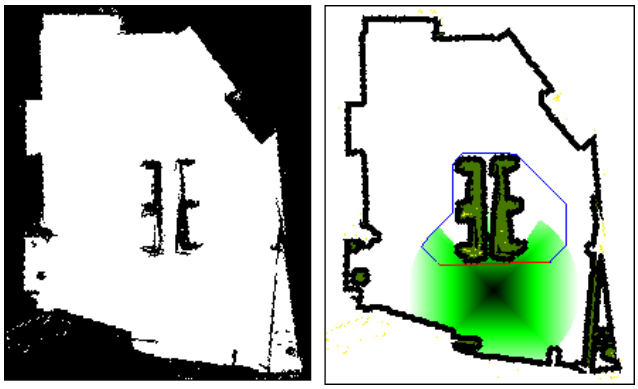
Ověření navigace podél nalezené utajené cesty bylo nejdříve provedeno v prostředí simulátoru Stage. Obrázek 2.11 obsahuje průběh trajektorie robotu navigovaného podél utajené cesty v prostředí případu 3 a parametr $\alpha = 85$. Z obrázku je patrné, že je robot navigován správně. Při navigaci je využíváno řízení robotu do bodu prostřednictvím rozhraní `position2d` systému Player. V případě simulace je odometrie téměř bezchybná (až na umělé definované chyby). Překážky jsou dostatečně nafouklé, proto se jim robot dokáže vyhnout i bez použití dalších senzorů. Menší problém nastává při objíždění stráže, kdy se robot pohybuje po kružnici a je mu zadáváno velké množství bodů blízko sebe. S ohledem na použití způsob řízení se robot pohybuje velmi pomalu. Tento problém by odstranilo kvalitnější nastavení řízení robotu.



Obrázek 2.11: Průběh navigace robotu podél utajené cesty v simulátoru Stage.

2.5.2 Reálný robot

Experiment byl proveden s mobilní robotickou platformou G^2Bot, která je vybavena dostatečně přesnou orometrií a není proto nutné explicitně řešit úlohu lokalizace. Experimentálně bylo prokázáno, že stejným program pro řízení robota v simulátoru, lze řídit i reálného robota. Mapa prostředí zobrazená na obrázku 2.12(a), byla získána mapováním prostředí robotem před zahájením hledání utajené cesty. Velikost buňky byla zvolena 10 cm, velikost prostředí je přibližně 9 m $\times$ 13 m. Rozmístění překážek, určení startovní a cílové pozice, i pozice stráže byly zvoleny tak, aby existovaly 2 cesty. Krátká cesta v těsné blízkosti stráže a dlouhá cesta, na níž je robot zahlédnut na opačné straně prostředí. Naplánovaná trajektorie je zobrazena modře na obrázku 2.12(b). Červeně je zobrazena nejkratší cesta. Na fotografii reálného prostředí 2.13 je zachycen robot jedoucí směrem od strážce, který je reprezentován oranžovou ploškou.



(a) (b)

Obrázek 2.12: Mapa a naplánovaná cesta při reálném experimentu.

Obrázek 2.13: Robot při navigaci podél utajené cesty v reálném prostředí.

Závěr

V práci byla představena třída úloh *pursuit–evasion* a některé možné metody jejich řešení. Dále byla řešena úloha hledání utajené cesty, nebo-li úloha skryté navigace *Covert Robotics Navigation*, jejíž řešení je založeno na rozšíření algoritmu *Distance Transform* o uvažování viditelnostního ohodnocení, tento algoritmus označován jak takzvaný *Dark Path* algoritmus.

Experimentálně bylo ověřeno chování algoritmu v různých prostředí a vliv parametrů α. Pokud je požadována cesta s nejvyšším krytím bez ohledu na ujetou vzdálenost je možné hodnotu α jednoduše nastavit na velmi vysoké číslo. V případě, že záleží na délce, a tedy i čase projetí, utajené cesty, není nastavení vhodné hodnoty parametru α přímočaré a je nutné provést několika výpočtů pro konkrétní prostředí pro různé hodnoty. Poté je možné odhadnout jak parametr přibližně nastavit. Pro různá prostředí a různá nastavení startovní a cílové pozice se hodnota α velmi liší. Zatímco v prvním testovaném prostředí je nejskrytější cesta zvolena už při hodnotě $\alpha = 35$, ve třetím případě je to až pro $\alpha = 85$.

Simulátoru i jízdou reálného robotu bylo ověřeno, že je nalezenou cestu je robot schopen projet. V simulátoru bylo provedeno několik experimentů a u žádného z nich se nevyskytly problémy. Na reálném robotu byl proveden jediný experiment. Chování reálného robotu se od simulátoru lišilo, protože zřejmě došlo ke špatnému nastavení odometrie a počáteční pozice robotu.

Algoritmus DT se v úlohách skryté navigace jeví jako velmi dobrý prostředek hledání cesty s potřebným krytím. Z mnoha provedených experimentů bylo zjištěno, že v případě existence skryté cesty ji dokáže najít. V případě neexistence cesty poskytující úplné krytí dokáže najít takovou, která má krytí nejvyšší možné.

Algoritmus předpokládá velmi jednoduchou závislost viditelnostního ohodnocení na vzdálenosti. Je možné předpokládat, že tato závislost je složitější, a tudíž by bylo možné výpočet viditelnostního ohodnocení pozměnit tak, aby model viditelnosti lépe odpovídal reálným situacím. Zajímavé by bylo i rozšíření algoritmu na dynamické prostředí. Dalším možným rozšířením je úloha skryté navigace týmu spolupracujících robotů. Za úvahu stojí také zda je mřížková reprezentace nejvhodnější, a nebylo by lepší používat např. hybridní modely. Úloha skryté navigace má velký význam a lze očekávat že se bude bude do budoucna ještě zvyšovat. Existuje zde také pohled z druhé strany, tj. ochrany prostorů před vetřelci. I v těchto úlohách se dá předpokládat vývoj a tudíž bude nutné hledat neustále nové metody řešení obou problémů.

Literatura

[1] Bresenham's line algorithm.
URL http://en.wikipedia.org/wiki/Bresenham's_line_algorithm

[2] Midpoint circle algorithm.
URL http://en.wikipedia.org/wiki/Midpoint_circle_algorithm

[3] Simple DirectMedia Layer.
URL http://www.libsdl.org

[4] The Player Project [online]. 2008.
URL http://playerstage.sourceforge.net/

[5] Dagnino, T.: The Art gallery Problem. 1997.
URL http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/97/Thierry/thierry507webprj/artgallery.html

[6] Jarvis, R. A.: Distance Transform Based Visibility Measures for Covert Path Planning in Known but Dynamic Environments. 2004.

[7] Marzouqi, M. S.; Jarvis, R. A.: MECSE-32-2004: "Covert Robotics, Developing Robots for Covert Missions". Technická zpráva, Dept of ECSE, Monash University, Australia, 2004.

[8] Mohamed S. Marzouqi, R. A. J.: Pursuit Games in Obstacle Strewn Fields Using Distance Transforms.

[9] Mohamed S. Marzouqi, R. A. J.: Robot Path Planning in High Risk Fire Front Environments.

[10] Murár, R.; Jurišica, L.: Reprezentácia prostredia mobilného robotického systému. *AT&P journal*, ročník 8, 2005.
URL http://www.atpjournal.sk/casopisy/atp_05/pdf/atp-2005-08-60.pdf

[11] Petr Stěpán, L. P., Miroslav Kulich: Přednášky z předmětu Mobilní robotika. 2009.

[12] Ray A. Jarvis, M. S. M.: New visibility-based path-planning approach for covert robotic navigation. *Robotica*, ročník 24, 2006.

[13] Roman Murár, L. J.: Reprezentácia prostredia mobilného robotického systému. *AT&P journal*, ročník 9, 2005.
URL http://www.atpjournal.sk/casopisy/atp_05/pdf/atp-2005-09-65.pdf

[14] Suri, S.: Polygon Triangulation.
URL www.cs.ucsb.edu/~suri/cs235/Triangulation.pdf

[15] Trafton, J. G.; Schultz, A. C.; Perznowski, D.; aj.: Children and robots learning to play hide and seek.
URL http://www.nrl.navy.mil/aic/iss/pubs/trafton.hideseek.hri.pdf

[16] Vandas, J.: Optimalní umístění kamer do půdorysu, Bakalářská práce. 2006.

[17] Vigna, G.: Capture the flag.
URL http://ictf.cs.ucsb.edu

[18] Volkan Isler, S. K., Sampath Kannan: Randomized Pursuit-Evasion in a Polygonal Environment. 2004.

Obsah CD

Přiložené CD obsahuje zdrojové kódy programu, text bakalářské práce ve formátu PDF a zdrojové kódy celého textu pro systém LaTeX. V následující tabulce je popsána struktura CD.

Adresář	Popis
`src`	zdrojové kódy knihovny
`doc`	zdrojové kódy textu bakalářské práce
`thesis.pdf`	text bakalářské práce

Tabulka 4: Adresářová struktura na CD.