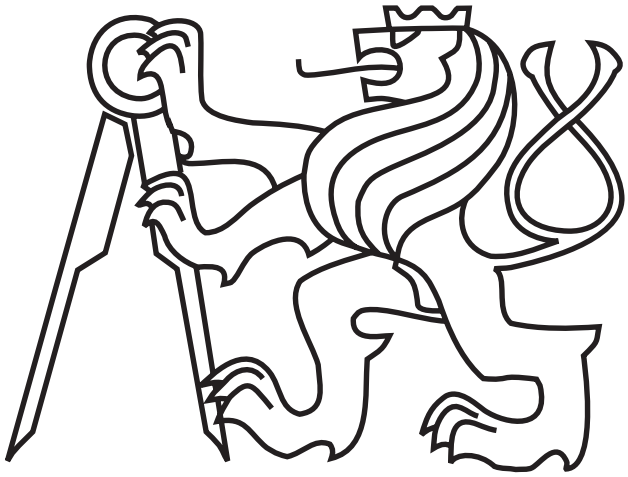
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

DIPLOMOVÁ PRÁCE



Jan Mačák

Multirobotická kooperativní inspekce prostředí

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

Praha, 2009

Prohlášení

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

V Praze dne 21.5.2003 .....

.....
podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Jan M a č á k

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný

Obor: Kybernetika a měření, blok KM2 – Umělá inteligence

Název tématu: Multirobotická kooperativní inspekce prostředí

Pokyny pro vypracování:

1. Seznamte se s úlohou multi-robotické inspekce prostředí.
2. Implementujte algoritmus pro řešení multirobotické inspekce prostředí.
3. Algoritmus ověřte na experimentálních robotických platformách Gerstnerovy laboratoře.
4. Proveďte porovnání s jinými přístupy.

Seznam odborné literatury:

[1] Choset, H.; Lynch, K.M.; Hutchinson, S.; Burgard, W.; Kavraki, L.E.; Thrun, S.: Principles of Robot Motion. The MIT Press, 2005.
[2] Steven, M.; LaValle, S.M.: Planning Algorithms. Cambridge University Press, 2006.

Vedoucí diplomové práce: Ing. Jan Faigl

Platnost zadání: do konce zimního semestru 2009/2010

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Vladimír Mařík, CSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 3. 9. 2008

Abstrakt

Diplomová práce se zabývá řešením multirobotické kooperativní inspekční úlohy. Úloha je řešena dekompozičním přístupem jako problém hlídání galerie a problém více obchodních cestujících. Cesty mezi jednotlivými měřicími místy jsou plánovány využitím harmonických funkcí, v práci je popsáno několik způsobů plánování cesty. Je popsána významná výhoda plánování využitím harmonických funkcí, a to možnost zobecnění cíle z bodového na plošný s libovolným polygonálním tvarem hranice a snadnost plánování v prostředí s více cíli. Navržené postupy jsou ověřeny řadou experimentů i na reálné robotické platformě. Součet délek cest řešení inspekční úlohy s cestami plánovanými v harmonickém potenciálu je o několik jednotek procent větší než u stejného řešení s cestami plánovanými grafem viditelnosti. Analýza rychlostního profilu ukazuje, že řešení s cestami dle harmonického potenciálu lze realizovat a dokončit v kratším čase. V práci jsou uvedeny výsledky experimentu hledání řešení inspekční úlohy využitím neuronové sítě a zobecněných míst.

Abstract

The thesis deals with multirobotic cooperative inspection task. The solution of the inspection task is obtained via decomposition into independent subproblems – the Art Gallery Problem and the multiple Traveling Salesmen Problem. The path between each pair of measurement places is planned by a harmonic potential field. The important advantage of planning using harmonic potential beyond obstacle avoidance is described. This advantage is a possibility to generalise goal from single point to an arbitrary polygon and ease of planning in multiple goal environment. Described methods are verified in a set of experiments. The sum of route lengths of the solution of inspection task planned by the harmonic potential field is slightly greater than in the identic solution planned by the visibility graph while realisation time of solution is by units of percent lower in case of planning in the harmonic potential field. Generalised goals are used in experiment on finding the inspection task solution based on artifitial neural network.

Děkuji Ing. Janu Faiglovi za obětavé vedení a řadu cenných námětů a připomínek. Děkuji Ing. Vojtěchu Vonáskovi za pomoc při provádění reálného experimentu.

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Mobilní robotika

Význam Čapkova slova robot již není v dnešní době třeba vysvětlovat, stejně tak si lze snadno odvodit, co v daném kontextu znamená přívlastek *mobilní*, a snad ještě jednodušší je představit si některé konkrétní exempláře mobilních robotů, např. vozítko NASA pro průzkum Marsu při misi Pathfinder - Sojourner [5] nebo robotického psa Aibo. Mobilní robot je stroj, který je schopen se pohybovat v určitém prostředí a vykazuje jistou, alespoň minimální, míru autonomity.

Mobilní robotika obohacuje klasickou robotiku o další možnosti aplikací a je ideálním řešením v případech, kdy je pro splnění úkolu nasazení lidí buď nemožné, např. průzkum vzdálených planet, nebo příliš nebezpečné, např. zneškodňování nálože, nebo jen neekonomické. Mobilní robotika také přináší spousty zajímavých úloh a problémů, například jak řídit a přesně určovat polohu robotu v prostředí, jak vnímat okolní prostředí a stavět mapy, jak v těchto mapách plánovat a řídit pohyb. Jednou z úloh, které mobilní robotika řeší, je i úloha inspekce známého prostředí.

Úloha inspekce

Úloha inspekce prostředí mobilním robotem je definována mapou a vlastnostmi prostředí a vlastnostmi mobilního robotu, zejména pozorovacími schopnostmi. Cílem je projet zadanou mapu tak, aby každý bod mapy byl během cesty alespoň jednou naměřen/pozorován a přitom byla cesta optimální podle zadaného kritéria – nejkratší, nejrychlejší, nejméně energeticky náročná apod. Nejvýznamnějším faktorem, který má vliv na výslednou naplánovanou trasu, je dosah měřicího senzoru. V následujícím textu bude užíván termín *viditelnost*. Ta je určena jak technickými parametry robotu – dosahem a pozorovacím úhlem senzorů, nejčastěji kamer, tak fyzikálními vlastnostmi prostředí, ve kterém se robot pohybuje – např. osvětlení, kouř, různé stíny.

Úloha má velké množství praktických aplikací, namátkou lze zmínit hlídání letišť, nemocnic, muzeí, kancelářských budov a ostatních soukromých či veřejných prostor, systematické prohledávání hořících budov, vyhledávání nastražených bomb či jiných nebezpečných předmětů. Méně tradičními aplikacemi na hranici definice úlohy jsou automatické vysávání nebo údržba trávníků – tyto aplikace lze formulovat lépe jako problém *pokrytí*. V těchto případech je většinou snadné získat vhodnou mapu prostředí reprezentující překážky, respektive volný prostor, ve kterém se má robot pohybovat. Dobře k tomu lze využít

architektonických plánů budov.

Jedním ze současných výzkumných trendů je studium možností nasazení celého společenství spolupracujících robotů za účelem rychlejšího dokončení úlohy, případně minimalizace nebo maximalizace jiného kritéria. Úlohou multirobotické inspekce je tedy rozdělení úkolů pro jednotlivé spolupracující roboty tak, aby byly co nejlépe splněny zadané požadavky. Pro nasazení v praxi je dále ještě třeba vyřešit množství dalších problémů, především způsob komunikace. K tomu lze využít např. výsledků studia multiagentních systému a tato práce se tomuto problému nevěnuje.

Jelikož je úloha inspekce velmi komplexní problém, je vhodné ji řešit dekompozicí na víceméně nezávislé podproblémy, ideálně takové, u kterých je známé řešení. Úlohu robotické inspekce lze rozložit na problém nalezení minimální množiny měřicích míst, pro kterou je zajištěno, že každý bod prostoru je naměřitelný alespoň z jednoho místa takové množiny. Tento typ úloh lze v literatuře najít jako tzv. *problém hlídání galerie* (*Art Gallery Problem*). Dalším podproblémem je nalezení nejvhodnějšího pořadí projetí jednotlivých měřicích míst, které zajistí nejkratší možnou dobu provedení inspekce nebo minimální energetické nároky. Tuto úlohu lze formulovat jako problém obchodního cestujícího. Nakonec na nejnižší úrovni plánování je třeba nalézt optimální cesty a trajektorie mezi jednotlivými měřicími místy.

Nalézt optimální trajektorii mezi dvěma body přímo je velmi těžké, neboť to vede na řešení variační úlohy nad množinou trajektorií, kterou je navíc těžké správně zadefinovat. Proto se volí rozdělený postup, kdy se nejprve najde cesta jako posloupnost bodů, kterými musí robot projet, aby dosáhl daného cíle, a poté se nalezená cesta převede na trajektorii v závislosti na použitém modelu robotu. Přestože i tento přístup má svá úskalí, je snáze řešitelný.

Možností jak naplánovat cestu mezi dvěma body na známé mapě je celá řada. Oblíbenou metodou je například prohledávání konfiguračního prostoru [23] diskretizovaného na volné a obsazené buňky, hledání cesty na grafu viditelnosti daného konfiguračního prostoru nebo využití potenciálového pole jako heuristiky pro pohyb k cíli v konfiguračním prostoru [29], případně randomizovaný přístup v podobě hledání cesty metodami PRM(*Probabilistic Road Map*) [19] nebo RRT (*Rapid-exploring Random Tree*) [24].

Tato práce se zabývá především hledáním cest využitím harmonického potenciálu, neboť lze předpokládat, že tato metoda bude vytvářet hladké cesty, které jsou jednodušeji převeditelné na trajektorie a lépe realizovatelné velmi rychlými nebo těžkými roboty, a bude dobře respektovat jednotlivé překážky ve volném prostoru.

Další kapitoly jsou koncipovány následovně: kapitola 1 je věnována popisu dekompozice problému a řešení jednotlivých podproblémů, v kapitole 2 jsou rozebrány a popsány vlastnosti a možnosti využití harmonických funkcí pro plánování cesty volným prostorem. Kapitola 3 se zabývá praktickým ověřením a srovnáním postupů navržených v kapitole 2.

Kapitola 1

Definice úlohy

Úlohu multirobotické inspekce lze formulovat jako *mWRP* (*multiple Watchmen Routes Problem*) [15], což znamená pro danou mapu prostředí P nalézt m optimálních tras takových, aby bylo možné každý bod $p \in P$ alespoň z jedné trasy naměřit. Běžně používanými kritérii optimality těchto tras jsou kritéria *MinMax* minimalizující délku nejdelší cesty, nebo *MinSum* minimalizující součet délek všech tras. Pro obě úlohy je dokázáno, že jsou NP-těžké [27].

mWRP lze řešit metodou rozkladu na víceméně nezávislé podproblémy:

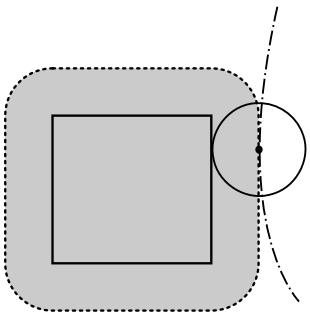
1. nalezení minimální množiny míst, jejichž navštívení garantuje splnění úlohy inspekce,
2. nalezení tras robotů, to jest optimálního pořadí a rozdělení návštívení míst pro jednotlivé roboty,
3. nalezení optimálních cest pro pohyb mezi jednotlivými místy a
4. realizace inspekce, to jest projetí nalezených cest.

Mapou prostředí P je v tomto případě myšlen popis volného prostoru, ve kterém se mají roboty pohybovat.

Na tomto místě je vhodné přesně vymezit pojmy trasa, cesta a trajektorie.

- Trasa – pořadí míst, která daný robot postupně při inspekční úloze navštíví.
- Cesta – posloupnost bodů ve volném prostoru, jimiž robot musí projet, aby se dostal z místa A do místa B.
- Trajektorie – cesta včetně času a řídicích veličin.

V této práci bude uvažována reprezentace volného prostoru polygonem s děrami. Tento volný prostor bude statický, tzn. překážky se nebudou v čase přemisťovat. Robot bude uvažován diferenciální s diskovým půdorysem. Tím lze dosáhnout po úpravě mapy abstrakce bodového robotu, jak je naznačeno na obrázku 1.1. Roboty budou vybaveny identickými všesměrovými měřicími senzory s definovaným dosahem.



Obrázek 1.1: Úprava mapy pro robot s diskovým půdorysem. Po „nafouknutí“ mapy Minkovského sumou lze robot považovat za bodový. Původní stav je vykreslen plnou čarou, stav po úpravě je naznačen přerušovanou čarou.

1.1 Nalezení množiny měřicích míst

Tuto podúlohu lze řešit jako *AGP* (*Art Gallery Problem*). Cílem *AGP* je na dané mapě P nalézt minimální množinu $G \subset P$ míst (strážců), ze kterých je viditelný kterýkoliv bod $p \in P$. Pojem *viditelnost* v tomto případě znamená, že existuje spojnice bodů $g \in G$ a $p \in P$ ležící celá ve volném prostoru [15]. Pracuje-li se s omezeným dosahem (omezenou viditelností) robotů – budiž označeno d_{max} – , pak lze definovat termín *d-viditelnost*, kde d je délka nejkratší ze spojnic g, p ležících ve volném prostoru. Logicky je pak bod p z g viditelný, pokud je $d \leq d_{max}$. Pro zjednodušení úlohy lze předpokládat, že všechny roboty mají stejnou viditelnost.

Je dokázáno, že k pokrytí jednoho libovolného polygonu o n vrcholech je zapotřebí nejvýše $\frac{n}{3}$ bodových strážců [13] a obecněji pro polygon o n vrcholech a h děrách maximálně $\frac{n+h}{3}$ strážců [8]. Pro praktické problémy jsou tyto horní odhady příliš vysoké a je snaha počet měřicích míst minimalizovat, aby se následně prováděné fáze hledání řešení co nejvíce zjednodušily. Nalezení takové množiny je opět NP-těžká úloha. Existuje mnoho přístupů, které hledají aproximaci řešení v polynomiálním čase, jedním z nich je [14].

V případě omezené viditelnosti, která lépe modeluje možnosti současných senzorů a může zohledňovat i fyzikální podmínky prostředí, je k pokrytí prostoru třeba více stážců než v případě neomezené viditelnosti. Jedním z algoritmů pro řešení úlohy s omezenou viditelností je [18]. Tento algoritmus vyžívá randomizovaného dvojnásobného vzorkování prostoru. Algoritmus pracuje ve třech krocích

1. náhodný výběr bodu p a výpočet oblasti $\mathcal{O}(p)$, která je z tohoto bodu viditelná,
2. náhodné navzorkování oblasti $\mathcal{O}(p)$ a výběr bodu p^* pokrývajícího největší oblast a
3. přidání bodu p^* do množiny strážců a odečet pokryté oblasti od dosud nepokrytého prostoru,

které jsou opakovány, dokud není celý volný prostor kompletně pokryt.

Postup navržený v [20] je na rozdíl od předchozího postupu deterministický a množinu strážců hledá postupným dělením prostoru na konvexní polygony, které lze pokrýt jedním strážcem. Výstupem tohoto přístupu mohou být krom bodových strážců i přímkové segmenty, na které lze strážce rozmístit libovolně, aniž by to narušilo splnění úlohy.

Při rozmisťování strážců se nesmí zapomínat ani na jejich fyzické rozměry - aby se strážce na dané místo vůbec dostal. Navíc je vhodné kvůli minimalizaci délky trasy inspekce strážce umisťovat do co nejmenší a nejkompaktnější oblasti. Za těchto okolností totiž lze předpokládat dokončení inspekce v co nejkratším čase a s co nejmenšími energetickými nároky. Tyto požadavky zohledňuje algoritmus *Boundary Placement* navržený v [15].

1.2 Nalezení pořadí navštívení míst

Jakmile je známa množina strážců G, je třeba určit, který z robotů a v jakém pořadí jednotlivá místa projede. Tato formulace vede na řešení *mTSP* – problému více obchodních cestujích (*multiple Traveling Salesmen Problem*). Jedná se opět o NP-těžkou úlohu. Ta je zobecněním úlohy jediného obchodního cestujícího.

1.2.1 Problém jediného obchodního cestujícího

V klasické formulaci diskrétní matematiky se jedná o hledání nejlevnějšího Hamiltonovského cyklu v úplném ohodnoceném neorientovaném grafu. Tato formulace je tzv. symetrický *TSP*, neboť $d(A, B) = d(B, A)$, kde $d(A, B)$ označuje ohodnocení hrany (A, B). Je-li splněna trojúhelníková nerovnost, jedná se o metrický symetrický *TSP* – může se použít Manhantannská, Eukleidovská nebo jakákoliv jiná metrika – např. i čas nutný k přechodu z bodu A do bodu B. V této práci je jako metriky použito délky cesty mezi jednotlivými body. [1]

Přístupů k získání řešení problému obchodního cestujícího je více, nejčastější dva jsou:

- Hledání přesného řešení, které je však v rozumném čase nalezitelné pouze pro jednodušší problémy. U nejprimitivnějšího přístupu (hrubou silou) je třeba vygenerovat a otestovat všechny kombinace hran tvořící Hamiltonovský cyklus a vybrat z nich tu nejlevnější. Takový postup má složitost $O(n!)$ a je tím pádem použitelný jen pro velmi malé počty míst. Jednou z možností urychlení je efektivnější prohledávání použitím algoritmu větví a mezí, nebo využití lineárního či dynamického programování, případně metody sečných rovin – těmito přístupy byly v poslední době získány zajímavé výsledky [3].

- Druhou možností je heuristický nebo aproximační přístup, který ovšem vede na suboptimální řešení. Otázkou je, o kolik je takové řešení horší než přesné a zda tento nedostatek vyváží menší výpočetní složitost. Typickým postupem je vygenerování

[1] Existuje mnoho dalších modifikací, nejznámější je asymetrický *TSP*, který lze modelovat na orientovaném úplném grafu, a u kterého neplatí $d(A, B) = d(B, A)$.

základního řešení hledáním nejbližšího nenavštíveného souseda. Takové řešení není více než $1,25\times$ horší, než přesné řešení [7]. Dále se dá řešení iterativně vylepšovat například heuristikou *k-opt* [7], nebo jinou randomizovanou metodou. Nejjednodušší variantou *k-opt* heuristiky je heuristika 2-*opt* zlepšující řešení prohazováním dvou hran a vybíráním lepší varianty. Pokud již nelze řešení záměnou žádných dvou hran zlepšit, je takové řešení 2-optimální. Aproximované 2-optimální řešení *TSP* produkuje o cca 9 % horší řešení než optimum [7]. Jiným přístupem je heuristika *GENIUS*, která využívá speciální procedury pro sestavení Hamiltonovského cyklu *GENI – Generalized Insertion* a postoptimalizační procedury *US – String-Unstring* [17]. V této práci je použito heuristiky *GENIUS* s *MinMax* kritériem minimalizujícím délku nejdelší cesty.

1.2.2 Problém více obchodních cestujících

Jak již bylo uvedeno dříve, jedná se o zobecnění úlohy jednoho obchodního cestujícího. V tomto případě je cílem pro k cestujících nalézt k uzavřených cest optimálních dle zvoleného kritéria.

Při hledání optimálního řešení *mTSP* vůči kritériu *MinSum* lze úlohu převést na řešení *TSP*, v případě kritéria *MinMax* již situace tak snadná není [6].

Hledání přesného řešení lze formulovat jako úlohu celočíselného programování [6], to znamená minimalizaci funkce

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij},$$

kde c_{ij} je cena hrany z uzlu i do uzlu j a $x_{ij} \in \{0,1\}$ podle toho, zda daná hrana v cestě použita je nebo není, za omezení

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 2,..,n,$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 2,..,n,$$

což znamená, že každé místo krom výchozího je navštíveno právě jednou, a omezení, že výchozí místo je navštíveno právě k krát

$$\sum_{i=2}^{n} x_{i1} = k,$$

$$\sum_{j=2}^{n} x_{1j} = k,$$

kde k je počet robotů, a dalších omezení eliminujících subcykly, což jsou uzavřené cesty, které neobsahují výchozí uzel (*depot*). Další možností je použití algoritmu *Branch and Bound* [1].

Heuristické přístupy pro nalezení suboptimálního řešení *mTSP* využívají heuristik navržených pro *TSP* – Lin Kernigan, k-opt –, neuronových sítí [22] nebo genetických a evolučních algoritmů [22].

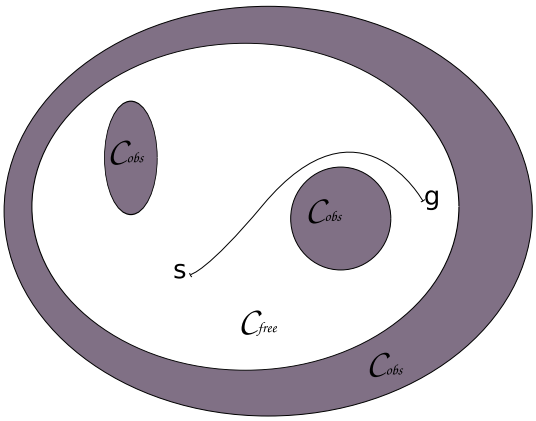
Další možností, jak nalézt pořadí projetí jednotlivých míst, je použít shlukové analýzy a nad každým shlukem poté nalézt řešení pro jednoho obchodního cestujícího. Takové řešení má v inspekční úloze také své opodstatnění, neboť jeho důsledkem jsou disjunktní operační prostory jednotlivých robotů a je tak již ve fázi plánování vyloučena kolize dvou robotů. K rozdělení měřicích míst na k shluků, kde k je počet robotů lze použít algoritmus *K-means* [25], jehož konvergence je zaručena a principálně spočívá ve dvou krocích:

1. přiřazení každého bodu k nejbližšímu těžišti shluku,
2. aktualizace poloh těžišť jednotlivých shluků,

prováděných opakovaně, dokud se příslušnost bodů ke shluku neustálí. Následně lze ke každému shluku přidat jeden společný bod, ze kterého budou roboty vyjíždět (*depot*).

1.3 Nalezení cesty mezi jednotlivými místy

Nejzákladnějším požadavkem při generování cesty mezi dvěma body je vyhnutí se všem překážkám. Jsou-li překážky v čase neměnné, jde o plánování ve stacionárním prostředí, v opačném případě jde o prostředí dynamické.



Obrázek 1.2: Znázornění konfiguračního prostoru, $\mathcal{C}_{free}$ je volný podprostor, $\mathcal{C}_{obs}$ je podprostor překážek, $\mathcal{C}_{free} \cup \mathcal{C}_{obs} = \mathcal{C}$, dvojice (s, g) znázorňuje cestu volným prostorem.

Plánování cest pro mobilní roboty probíhá prohledáváním *konfiguračního prostoru*. To je obecně n-dimenzionální prostor obsahující všechny možné konfigurace (stavy) robotu. Dimenze je nejčastěji dána počtem stupňů volnosti robotu. Takový prostor lze rozdělit na dva typy disjunktních podprostorů. Volné podprostory – to jsou podprostory konfigurací, kterých může robot nabývat – a podprostory překážek, které jsou tvořeny nepřístupnými

konfiguracemi, jak je znázorněno na obrázku 1.2. Úkolem plánovacího algoritmu je provést prohledání konfiguračního prostoru a odpovědět na otázku, zda existuje bezkolizní cesta z bodu A do bodu B a pokud existuje, tak ji najít, nejčastěji jako posloupnost souřadnic. Důležitou vlastností plánovacího algoritmu je jeho *úplnost*. Algoritmus je *úplný*, pokud v konečném čase správně odpoví, zda cesta existuje a v konečném čase pak cestu najde. Konfigurační prostor vzhledem k jeho mohutnosti nelze celý prohledat, a proto je nutné buď prostor při prohledávání vzorkovat, pak jde o plánování vzorkováním, nebo jeho hranici exaktně popsat množinou geometrických primitiv, pak se jedná o kombinatorické plánování [24]. Obě metody mají své výhody a nevýhody.

Plánování vzorkováním

Jak již bylo uvedeno, jednou možností je plánování vzorkováním (*Sampling Based*) a detekcí kolize. Nevýhodou tohoto přístupu je jeho *neúplnost* ve smyslu prohledání celého konfiguračního prostoru. Pokud algoritmus vzorkuje konfigurační prostor hustě, pak lze říct, že cestu najde v konečném čase, pokud existuje. Pokud cesta neexistuje, algoritmus nikdy neskončí. Typickým zástupcem tohoto typu jsou RDT (*Rapidly exploring Dense Tree*) algoritmy a RRT (*Rapidly exploring Random Tree*) jako jeho nejznámější varianta. Výhodou těchto algoritmů je jejich efektivnost a jednoduchost.

Kombinatorické plánování

Druhou možností je kombinatorické plánování, kdy je spojitý problém převeden na diskrétní přesně – například na graf polygonů –, a proto je možno otázku neexistence cesty zodpovědět v konečném čase. Jeho nevýhodou je větší složitost a těžší nasazení pro řešení praktických problémů. Důležitým aspektem je volba reprezentace překážek, vhodná pro plánování je například polygonální mapa.

Použitelné heuristiky

V případě vzorkovaného prohledávání lze docílit zvýšení rychlosti nalezení řešení použitím dodatečné heuristiky, která vybírá nejvhodější směr růstu k cíli a může zohledňovat vzdálenosti od překážek. Jako vhodné heuristiky lze použít potenciálového pole, které každému bodu konfiguračního prostoru přiřadí hodnotu potenciálu tak, že blízkost k překážce bude hodnotu potenciálu zvyšovat a blízkost k cíli bude hodnotu snižovat. Potenciál lze definovat dle [12] jako

$$U(q) = U_{goal}(q) + U_{obstacle}(q),$$

kde

$$U_{goal}(q) = \begin{cases} \frac{1}{2}\xi d^2(q, q_{goal}), & \text{pokud } d(q, q_{goal}) \leq d^*_{goal} \\ d^*_{goal}\xi d(q, q_{goal}) - \frac{1}{2}\xi (d^*_{goal})^2, & \text{jinak} \end{cases}$$

a

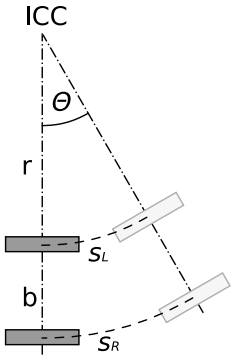
$$U_{obstacle}(q) = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & \text{pokud } D(q) \leq Q^* \\ 0, & \text{jinak,} \end{cases}$$

kde d a D jsou vzdálenosti od cíle, resp. nejbližší překážky, a ξ a η jsou parametry ovlivňující tvar potenciálu. Prohledávací algoritmus pak postupuje směrem největšího klesání. Taková heuristika funguje výborně, pokud mají všechny překážky konvexní tvar, nebo pokud cíl není zastíněn dvěma podobnými překážkami vedle sebe. V opačném případě mohou vzniknout v potenciálovém poli lokální minima, ve kterých algoritmus může uváznout. Z takového uváznutí se lze dostat například použitím simulovaného žíhání nebo lze zvolit potenciálové pole, které lokální extrémy nemá. Takové vlastnosti má *harmonická funkce*.

Hodnotu harmonické funkce nelze pro libovolný bod samostatně určit jako vektorový součet sil od jednotlivých překážek a cíle, neboť hodnota harmonického potenciálu v každém bodě závisí na okolních hodnotách. Proto není harmonická funkce příliš vhodná k přímému použití jako heuristiky pro směr vzorkování, protože vyžaduje výrazně složitější výpočet. Harmonická funkce je řešením diferenciální rovnice nad konfiguračním prostorem a její průběh určuje cestu do cíle ze všech bodů prostoru. Jedná se tedy o metodu *úplnou*, která generuje hladké cesty. Nevýhodou tohoto přístupu je větší časová složitost výpočtu, výhodou může být znalost cest ze všech bodů do jednoho cíle. To umožňuje opakované využití jednou vypočítaného potenciálu pro jiné výchozí polohy a stejný cíl a tím redukuje celkový čas potřebný pro naplánování jedné cesty.

1.4 Jízda po nalezené cestě

Aby bylo možné nalezenou cestu projet, je třeba vygenerovat průběhy řídicích veličin. Ty jsou závislé na konkrétním modelu robotu, respektive na modelu jeho pohybu. Jedním z nejjednodušších modelů je model diferenciální. Robot tohoto typu má dvě nezávisle hnaná kola na společné ose, jak je naznačeno na obrázku 1.3.



Obrázek 1.3: Diferenciální model pohybu.

V tomto případě se zatáčení realizuje rozdílnými rychlostmi levého a pravého kola. Pro natočení o úhel Θ je třeba, aby byl rozdíl ujetých drah

$$\Delta s = s_L - s_R = \Theta r - \Theta(r + b) = \Theta b,$$

kde r je vzdálenost bližšího kola od bodu otáčení ICC (*Instantaneous Center of Curvature*) a b je vzdálenost mezi koly.

Řídicí veličiny pro tento model pohybu lze získat poměrně jednoduše, pro cesty s málo změnami směru a dlouhými rovnými úseky je vhodné použít řízení *Turn–Move*. To znamená rovné úseky projet maximální rychlostí a v bodech změny směru zastavit a provést otočení na místě. Takové řízení zajišťuje poměrně přesné projetí zadané cesty, ale z energetického hlediska kvůli častému rozjíždění a zastavování není optimální.

Pokud je cesta hustě navzorkovaná a téměř hladká – s malými změnami úhlů mezi jednotlivými segmenty cesty, je lepší použít řízení bez zastavování. To znamená snažit se celou cestu projet maximální rychlostí bez zastavení, jen v bodech, kde se významněji mění směr, zpomalit tak, aby byla splněna dynamická, respektive kinodynamická omezení. Dá se předpokládat, že takové řízení má menší energetické nároky, neboť práce nutná k přesunu robotu z místa A do místa B po dráze s vychází ze vztahu

$$W = \int_A^B m\mathbf{a}(s)d\mathbf{s},$$

kde m je hmotnost robotu a $\mathbf{a}(s)$ jeho zrychlení v závislosti na poloze. To znamená, že čím jsou menší změny rychlosti během jízdy, tím méně spotřebuje robot energie.

Kapitola 2

Harmonický potenciál

2.1 Teoretický rozbor

Jak již bylo naznačeno v kapitole 1.3, harmonických funkcí lze užít k plánování cest pro inspekční úlohu. Oproti klasickému potenciálovému plánování nemohou uvnitř definičního oboru mít lokální extrémy, a proto netrpí možným uváznutím. Naproti tomu je jejich výpočet výrazně složitější.

2.1.1 Harmonické funkce

Jedním z axiomů potenciálového pole (harmonické funkce) f je, že splňuje na celém definičním oboru Laplaceovu rovnici

$$\Delta f = 0, \tag{2.1}$$

kde Δ je Laplaceův operátor

$$\Delta = \sum_{k=1}^{n} \frac{\partial^2}{\partial x_k^2}.$$

Důležitým důsledkem splnění Laplaceovy rovnice je nulovost integrálu

$$\int_{\partial\Omega} \frac{\partial u}{\partial \mathbf{n}} dS = 0 \tag{2.2}$$

po hranici definičního oboru. Důkaz vychází z Greenovy identity

$$\int_{\Omega} (u\Delta v - v\Delta u) dV = \int_{\partial\Omega} (u\frac{\partial v}{\partial \mathbf{n}} - v\frac{\partial u}{\partial \mathbf{n}}) dS,$$

bude-li $v(x_1, ..x_n) = 1$, což je jistě harmonická funkce, a u harmonická, pak integrál na levé straně bude 0 a na pravé straně vznikne integrál (2.2).

Využitím vlastnosti střední hodnoty [4] se dá dokázat neexistence extrémů uvnitř definičního oboru, tzn. případné globální extrémy leží pouze na hranici definičního oboru takové funkce.

Z tohoto důvodu (absence lokálních extrémů) lze daný typ funkcí použít k plánování (hledání) cesty mobilního robotu a to poměrně přímočaře – geometrická mapa prostředí se převede na množinu okrajových podmínek, k těmto podmínkám se přidá okrajová podmínka určující pozici cíle a vyřeší se parciální diferenciální rovnice (2.1). Aplikací operátoru gradient

$$\nabla = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_n} \right)$$

vznikne z funkce f vektorové pole, které každému bodu z definičního oboru funkce f přiřazuje směrový vektor $d\mathbf{s}$. Cesta vzniklá složením těchto vektorů povede do cíle a bude vždy spojitá a hladká[1]. Vzhledem k tomu, že při řešení inspekční úlohy volný prostor už zohledňuje rozměry robotu, konfiguračním prostorem je upravená dvourozměrná mapa a Laplaceovu rovnici stačí řešit v R^2.

V praxi je však velmi těžké hledat analytické řešení Laplaceovy rovnice, a proto je třeba využít některé z numerických metod. Ty však jsou řešením pouze přibližným a navíc s nespojitými derivacemi, což podstatně snižuje kvalitu výsledné cesty. Nejběžnějšími numerickými metodami pro aproximaci diferenciálních rovnic jsou metoda konečných diferencí a metoda konečných prvků. Obě zmíněné metody v zásadě pracují na podobném principu a to takovém, že diferenciální rovnici nad danou oblastí – definičním oborem – nahradí soustavou lineárních rovnic pro vybrané body oblasti. Počet a rozmístění těchto bodů potom určuje přesnost aproximace. Metody se od sebe liší způsobem volby bodů oblasti a způsobem sestavení lineárních rovnic.

2.1.2 Metoda konečných diferencí

Jak již název napovídá, princip této metody tkví v aproximaci derivací v parciální diferenciální rovnici (dále *PDE*) diferencemi mezi sousedními body oblasti.

Postup hledání řešení lze dekomponovat do několika kroků:

1. na oblasti se zadefinuje ortonormální síť bodů (vrcholů), pro které se budou sestavovat tzv. *síťové rovnice*. Tyto rovnice svazují hodnoty sousedních uzlů sítě. Každému uzlu sítě náleží právě jedna rovnice, výsledná soustava tedy bude mít hodnost n, kde n je počet uzlů.

2. Diferenciální rovnice se aproximuje diferenční rovnicí. V tomto případě

   $$\frac{\partial f}{\partial x} \approx \frac{1}{h} \left(f(x+h, y) - f(x, y) \right),$$

   respektive

   $$\frac{\partial f}{\partial x} \approx \frac{1}{h} \left(f(x, y) - f(x-h, y) \right),$$

   kde h je vzdálenost sousedních uzlů mřížky, dále

   $$\frac{\partial^2 f}{\partial x^2} \approx \frac{1}{h^2} \left(f(x+h, y) - 2f(x, y) + f(x-h, y) \right);$$

[1] Ve smyslu plynulé změny směru.

pro směr y obdobně a výsledná aproximace rovnice tedy bude

$$\frac{1}{h^2}\left(f(x+h,y)+f(x,y+h)-4f(x,y)+f(x-h,y)+f(x,y-h)\right)\approx 0. \qquad (2.3)$$

Z této diferenční rovnice se sestaví soustava síťových rovnic.

3. Dalším krokem je zavedení okrajových podmínek.

   - Pokud je v daném bodu sítě definována Dirichletova podmínka

     $$f(x,y)=g(x,y),$$

     stačí příslušnou síťovou rovnici (2.3) nahradit touto podmínkou.
   - Pokud je v daném bodě definována Neumannova podmínka

     $$\frac{\partial f}{\partial \mathbf{n}}=g$$

     a normálový vektor je ve směru x, pak je třeba příslušnou rovnici modifikovat na

     $$\frac{1}{h}\left(f(x+h,y)-f(x,y)\right)=g,$$

     podobně pro směr y, obdobný vztah platí i pro vektory v opačném směru x, resp. y.
   - Pokud body sítě nejsou zároveň body zadané okrajové podmínky, nebo normálový vektor $\mathbf{n}$ není ve směru některého z bázových vektorů prostoru, je třeba okrajové podmínky aproximovat.

Přepíší-li se výsledné rovnice do maticového tvaru

$$\mathbf{A}_h\mathbf{u}_h=\mathbf{b}_h,$$

lze soustavu snadno vyřešit některou ze standardních metod řešení soustav – např. gaussovou eliminační metodou, nebo metodou konjugovaných gradientů. Dle [32] je matice $\mathbf{A}_h$ vždy symetrická a velmi řídká, neboť v každém řádku má maximálně pět nenulových hodnot. Vhodným očíslováním uzlů lze navíc dosáhnout toho, že bude pásová – tzn. všechny nenulové prvky budou ležet na diagonálách. V důsledku těchto vlastností lze i rozsáhlé soustavy řešit na dnešní běžně dostupné výpočetní technice řádově v jednotkách sekund.

2.1.3 Metoda konečných prvků

Teoreticky jde o modifikaci Galerkinovy diskretizační metody, která hledá přibližné řešení *PDE* jako řešení variační úlohy nad prostorem (Sobolevovým[2]) přípustných funkcí.

[2]Sobolevův prostor $\mathcal{H}^k(\Omega)$ je prostor všech funkcí $f\in\mathcal{L}_2(\Omega)$, jejichž všechny zobecněné derivace až do řádu k včetně jsou taktéž z $\mathcal{L}_2(\Omega)$, kde $\mathcal{L}_2(\Omega)$ je Lebesgueův prostor funkcí integrovatelných se čtvercem na Ω – nad definičním oborem diferenciální rovnice.

Pro metodu konečných prvků je charakteristické nepravidelné rozmístění bodů, které pokrývají definiční obor dané *PDE*. Tyto body tvoří vrcholy tzv. *konečných prvků*, nejčastěji trojúhelníků nebo čtyřúhelníků. Použije-li se trojúhelníkových prvků, je třeba oblast, na které je *PDE* definována, triangulovat. Ideálně tak, že všechny vzniklé trojuhelníky budou co nejvíce rovnostranné, neboť pak bude mít aproximace řešení nejmenší chybu, protože jako důsledek Věty 4.3 (s.310) [32] pro odhad chyby aproximace na celé oblasti Ω platí nerovnost

$$||f - f_h||_{\mathcal{L}_2(\Omega)} \leq Ch^2|f|_{\mathcal{H}^2(\Omega)},$$

kde $f \in \mathcal{H}^2(\Omega)$ je hledaným řešením (funkcí ze Sobolevova prostoru), f_h její aproximace, $h = \max\{h_s, s = 1, .., S\}$, h_s je délka nejdelší strany s-tého trojúhelníku a C je konstanta. Ke každému z vrcholů – uzlového bodu triangulace –, M_n, $n = 1, 2, ..., N$, přísluší právě jedna bázová funkce v_n, která má následující vlastnosti:

- nad každým trojúhelníkem T_s, kde s je globální index, jehož jedním z vrcholů je M_n, má tvar
$$N^s = a^s + b^s x + c^s y, \tag{2.4}$$
- a splňuje interpolační podmínky
$$v_n(M_n) = 1$$
a
$$v_n(M_m) = 0$$
pro všechna $m \neq n$. To znamená, že je nenulová jen na trojúhelnících přímo sousedících s vrcholem M_n. Množina takto definovaných bázových funkcí je lineárně nezávislá a tvoří prostor V^h dimenze N [26].

Přibližným Galerkinovým řešením Laplaceovy rovnice bude po částech lineární funkce

$$u_h = \sum_{n=1}^{N} U_n v_n,$$

která splňuje rovnost

$$a(u_h, v_h) = F(v_h), \tag{2.5}$$

kde

$$a(u, v) = \iint_{\Omega} \nabla u \cdot \nabla v dx dy + \int_{\partial\Omega} \sigma uv dS \tag{2.6}$$

a

$$F(v) = \int_{\partial\Omega} gv dS,$$

kde σ a g jsou parametry okrajové podmínky

$$\sigma u + \frac{\partial u}{\partial \mathbf{n}} = g,$$

pro všechny funkce z prostoru V^h. Vzhledem k linearitě prostoru stačí vzít všechny bázové funkce prostoru V^h a postupně je dosazovat do rovnice (2.5). Tak vznikne soustava N lineárních rovnic

$$\mathbf{A}_h\mathbf{U}_h = \mathbf{F}_h.$$

Matice $\mathbf{A}_h$ je opět řídká, ale obecně už ne pásová, pásovosti se dá za určitých okolností dosáhnout vhodným očíslováním vrcholů oblasti. Tato soustava je řešitelná klasickými postupy. Jelikož je v této metodě potřeba počítat hodnoty hraničních integrálů, je nutné ještě popsat numerickou metodu k jejich výpočtu.

Gaussovy kvadraturní vzorce

Jedná se o numerický výpočet hodnoty určitého integrálu typu

$$I_w(f) = \int_a^b w(x)f(x)dx$$

jako lineární kombinace funkčních hodnot integrované funkce v několika bodech definičního oboru, tedy

$$I_{n+1}(f) = \sum_{i=0}^{n} H_i f(a_i),$$

kde H_i jsou váhy a a_i body z daného intervalu. Číslo n určuje počet použitých bodů – je jich $n+1$ – a také zásadně ovlivňuje řád kvadraturního vzorce – ten udává maximální stupeň polynomu, který bude vzorec ještě integrovat přesně.

Jelikož je třeba integrovat polynomy na konečných intervalech, lze použít Legendrova-Gaussova vzorce [31], což je kvadraturní vzorec pro interval $\langle -1, 1\rangle$ a $w(x) = 1$ a jehož řád je $2n+1$. Body a_i budou kořeny Legendrova polynomu P_{n+1}, který je definován předpisem

$$P_n(x) = \frac{d^n}{dx^n}\left[\frac{1}{2^n n!}\left(x^2-1\right)^n\right].$$

V tomto případě postačí $n = 1$, neboť se v úloze vyskytují polynomy maximálně druhého stupně. Tedy

$$P_2(x) = \frac{1}{2}\left(3x^2-1\right)$$

a příslušné kořeny budou $a_{1,2} = \pm\sqrt{\frac{1}{3}}$.

Váhy H_i jsou definovány jako

$$H_i = -\frac{2}{(n+2)P_{n+2}(a_i)P'_{n+1}(a_i)}.$$

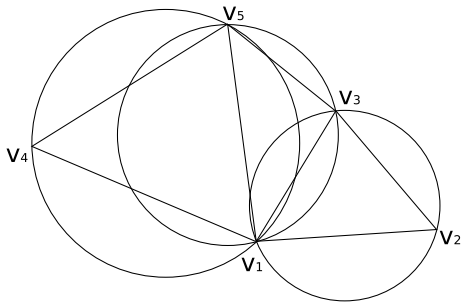
a v tomto případě jsou po dosazení $H_{1,2} = 1$.
Jelikož je třeba počítat integrály v mezích $\langle a, b\rangle$, je nutné tento inteval převést na $\langle -1, 1\rangle$. To lze udělat jednoduchou substitucí

$$y = \frac{2x-a-b}{b-a}.$$

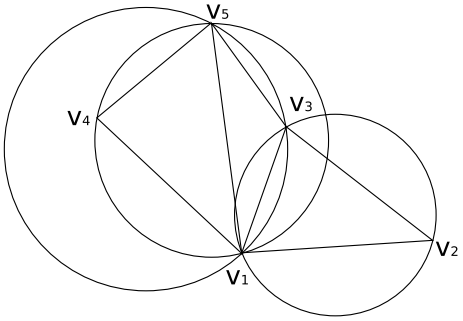
Triangulace oblasti

Ideální triangulace – ta s nejmenší aproximační chybou – je ta s rovnostrannými trojúhelníky. Obecně tuto triangulaci sestavit nelze a je tedy třeba hledat takovou triangulaci, která toto kritérium bude splňovat co nejlépe. Takovou triangulací je *Delaunayova* triangulace, o které je dokázáno, že maximalizuje nejmenší úhel každého trojúhelníku [28].

Delaunayovou triangulací je každá triangulace, pro jejíž všechny trojúhelníky platí, že v kružnici opsané danému trojúhelníku neleží žádný jiný vrchol triangulace, pro lepší názornost je situace ilustrována na obrázku 2.1. K triangulaci lze využít volně dostupné knihovny



Obrázek 2.1: Delaunayova triangulace – v žádné opsané kružnici neleží více než 3 vrcholy.



Obrázek 2.2: Triangulace, která není Delaunayova, v levé a prostřední kružnici leží vždy čtyři body (v_1, v_3, v_4 a v_5).

algoritmů výpočetní geometrie CGAL [11], která vytváření Delaunayovských triangulací podporuje. Podobu výsledné triangulace lze v základu ovlivňovat dvěma parametry – maximální délkou hrany triangulace (strany trojúhelníka) a minimální velikostí vnitřních úhlů, ta se zadává přes parametr B daný předpisem

$$\sin \alpha_{min} = \frac{1}{2B}.$$

Použitý algoritmus zaručeně konverguje pro hodnoty úhlu větší než 20,7 °, tedy $B = 0{,}125$ [9].

Druhou možností, jak určovat vlastnosti triangulace, je definice vlastních podmínek, za jakých bude iterativní algoritmus CGALu považovat aktuální trojúhelník za dostatečně kvalitní. Tak lze dosáhnout výrazného zjemnění triangulace v hraničních oblastech a následného zpřesnění aproximace harmonické funkce nad danou oblastí, aniž by došlo k extrémnímu nárůstu počtu trojúhelníků a samozřejmě i nutného výpočetního času a paměťového prostoru.

2.1.4 Algoritmizace

Zatímco u metody konečných diferencí je popis uvedený v kapitole 2.1.2 v podstatě snadno naprogramovatelným algoritmem – za předpokladu, že se k řešení vzniklé soustavy

lineárních rovnic použije některý z dostupných nástrojů pro výpočet, algoritmizace metody konečných prvků již tak přímočará není. Je třeba počítat koeficienty jednotlivých prvků a určité integrály pro okrajové podmínky a z těchto elementů poté sestavit soustavu lineárních rovnic.

Metoda konečných diferencí

Nejsnadnější cestou k sestavení globální matice $\mathbf{A}_h$ metody konečných diferencí je

1. vytvoření globální matice pro obdélníkovou oblast s rozměry $(\max x - \min x, \max y - \min y)/h$, kde se uzly mřížky očíslují zleva doprava po řádcích, respektive shora dolů po sloupcích. Globální matice pak bude pásová s pěti diagonálami ve tvaru

$$\mathbf{A}_h = \begin{pmatrix} 4 & -1 & 0 & . & . & . & 0 & -1 & 0 & . & 0 \\ -1 & 4 & -1 & 0 & . & . & . & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & . & . & . & 0 & -1 & 0 \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ 0 & . & -1 & 0 & . & . & . & 0 & -1 & 4 & -1 \\ 0 & . & 0 & -1 & 0 & . & . & . & 0 & -1 & 4 \end{pmatrix},$$

   kde mezi první a druhou, respektive čtvrtou a pátou diagonálou bude $(\max x - \min x)/h$ nul,

2. vytvoření vektoru pravé strany, který bude nulový,

3. modifikace řádků a sloupců pro uzly, ve kterých je zadána okrajová podmínka,

4. modifikace příslušných řádků vektoru pravé strany,

5. vyloučení řádků a sloupců z matice a vektoru získaných v prvním kroku pro uzly, které leží mimo definiční oblast rovnice.

Tento postup je však vhodný jen v případě, že uzlů (řádků a sloupců), které je třeba vyloučit, je výrazně menší počet než je celkový počet uzlů, neboť jen v tomto případě nebude rozměr matice $\mathbf{A}_h$ z prvního kroku příliš nadsazen a nedojde tak ke zbytečnému plýtvání pamětí.

Metoda konečných prvků

Praktický postup, jak výpočet metodou konečných prvků naprogramovat je zhruba následující:

1. triangulace oblasti,

2. výpočet parametrů a, b a c ze vztahu (2.4) pro jednotlivé trojúhelníky,

3. sestavení lokální prvkové matice $\mathbf{A}^s$ řádu 3 pro jednotlivé trojúhelníky, koeficienty jsou dány vztahem

$$a_{ij}^s = \iint_{T_s} \left(b_i^s b_j^s + c_i^s c_j^s\right) dxdy,$$

který vznikne dosazením rovnice (2.4) do bilineární formy (2.6).

4. Zahrnutí příspěvků hraničních integrálů do $\mathbf{A}^s$, sestavení vektoru $\mathbf{F}^s$ s příspěvky hraničních integrálů k pravým stranám rovnice (2.5). Velikosti příspěvků pro hraniční stranu M_iM_j jsou dány vztahy

$$a_{ij} = a_{ij} + \int_{M_iM_j} \sigma N_k N_l ds,$$

kde $k = i, j$ a $l = i, j$, a

$$f_i = f_i + \int_{M_iM_j} g N_i ds,$$

$$f_j = f_j + \int_{M_iM_j} g N_j ds.$$

K výpočtu hodnot těchto integrálů lze využít zmíněných Gaussových kvadraturních vzorců s řádem 2, které dají pro tuto třídu integrálů přesné výsledky.

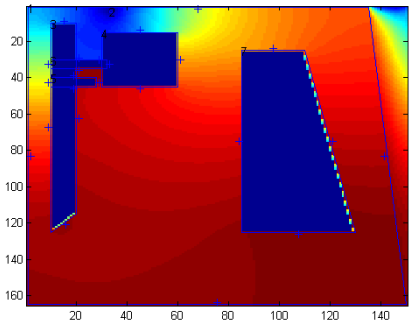
5. Sestavení globální matice $\mathbf{A}_h$ z prvkových matic jednotlivých trojúhelníků.

6. Vyřešení vzniklé soustavy lineárních rovnic.

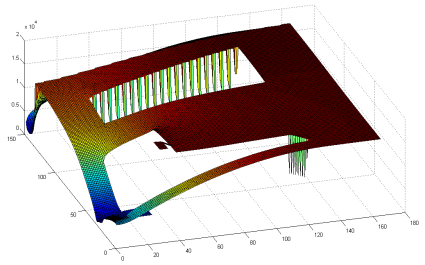
2.2 Srovnání výsledků jednotlivých metod

Obecně velkou výhodou metody konečných diferencí je, že je velmi snadno naprogramovatelná a velmi jednoduše modifikovatelná pro výpočet různých PDE. Další velkou výhodou je jednoduché rozšíření do vyšší dimenze a tedy možnost plánování ve vícedimenzionálním konfiguračním prostoru. Ve vyšších dimenzích však vede na obrovskou, i když velmi řídkou globální matici. Při řešení inspekční úlohy tato výhoda nemá velké využití, neboť se uvažuje plánování ve 2D konfiguračním prostoru – nad polygonální mapou – pro bezrozměrný robot[3], a převažují nevýhody. Velkou nevýhodou je nemožnost modifikace sítě na nepravoúhlé mapy tak, aby nebylo nutné hraniční křivky aproximovat. Mohou tak vzniknout podél hranice oblasti, na kterých je Laplaceova rovnice velmi nepřesně aproximovaná, což by v praxi znamenalo, že pro dané body by nebylo možné najít cestu. Tento problém je možné řešit přidáním uzlů v první řadě – v podstatě se jedná o posunutí hranice oblasti do překážky a zavedení přechodové podmínky – za hranicí oblasti k uzlům sítě, jak je vidět na obrázku 2.3.

[3]Rozměry robotu jsou zohledněny u překážek, které jsou o danou velikost nafouknuty. Toho lze dosáhnout *Minkovského sumou* mapy s kružnicí, která reprezentuje tvar (největší z rozměrů) robotu.



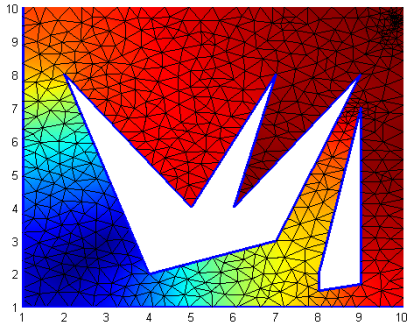
(a)



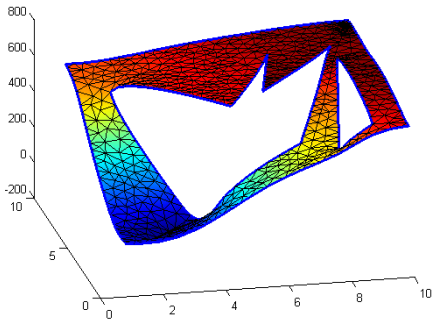
(b)

Obrázek 2.3: Řešení Laplaceovy rovnice metodou konečných diferencí, na (a) modré úsečky znázorňují okrajové podmínky, značky po stranách úseček znázorňují směr vnitřní normály. (b) 3D zobrazení.

Metoda konečných prvků zmíněnou nevýhodu nemá, přesně triangulovat lze téměř jakoukoliv oblast – vyjma nelineárních okrajových podmínek. Vhodnou triangulací lze dosáhnout malého počtu uzlů, což znamená i menšího řádu řešené soustavy a úloha je tak vyřešena rychleji než metodou konečných diferencí a to se srovnatelnou kvalitou. Ilustrace řešení Laplaceovy rovnice metodou konečných prvků je na obrázku 2.4. Metoda konečných prvků se z výše popsaných důvodů – lepší aproximace hranic prostředí, variabilní tvar konečných prvků – jeví pro řešení inspekční úlohy jako lepší a dále stačí pracovat jen s ní, i když navržené postupy hledání výsledné cesty v harmonickém potenciálovém poli se samozřejmě dají aplikovat i na pole vypočtené metodou konečných diferencí.



(a)



(b)

Obrázek 2.4: Řešení Laplaceovy rovnice metodou konečných prvků, na (a) výškový profil, tučné černé úsečky značí Neumannovy okrajové podmínky. (b) 3D zobrazení.

2.3 Vliv okrajových podmínek

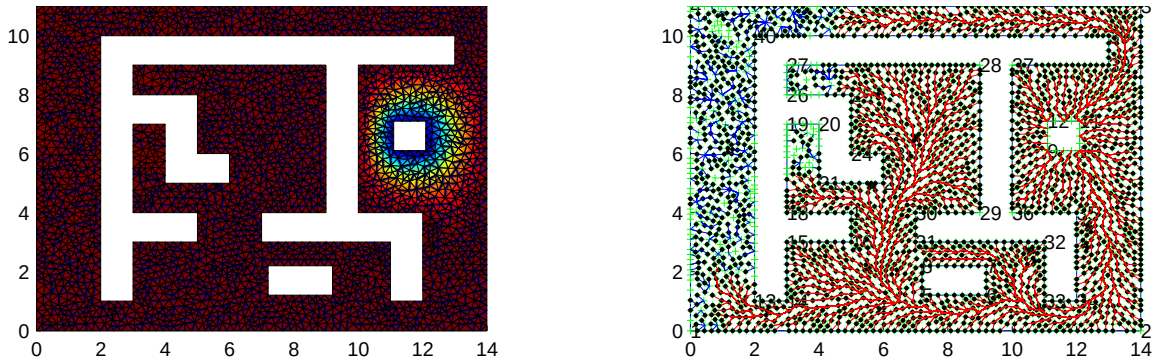
Jelikož je Laplaceova rovnice parciální diferenciální rovnicí druhého řádu, je možné zadavát jako okrajovou podmínku jak hodnotu funkce v daném bodě nebo hranici – již zmíněná Dirichletova podmínka, tak hodnotu derivace ve směru vnější normály hranice – Neumannova podmínka. Proto je možné hledat harmonickou funkci jako řešení čistě Dirichletova problému, kdy jsou všechny okrajové podmínky Dirichletovy, nebo naopak jako čistě Neumannův problém, kdy jsou všechny okrajové podmínky prvního řádu, případně jako řešení smíšeného problému.

2.3.1 Čistě Dirichletův problém

Popis problému pouze Dirichletovými okrajovými podmínkami

$$f(x, y) = g$$

je zřejmě nejpřirozenější, pro lidskou mysl nejsnáze představitelný: potenciál klesá od nejvyšších hodnot – překážky – k nejnižší hodnotě - cíli. Tento přístup ale začne být záhy limitován výpočetní přesností počítače, neboť potenciál od cíle roste velmi strmě, aby u nejbližší překážky nabyl svého maxima. Rozdíly potenciálů vzdálenějších oblastí jsou potom pod rozlišovací schopnost standardně používaných datových typů, jak je ilustrováno na obrázku 2.5(a). Oblastí, na které se dá v tomto případě plánovat, je oblast pokrytá červenými cestami na obrázku 2.5(b). Zvětšení výpočetní přesnosti nemá velký smysl, jen se tím případně rozšíří oblast, na které lze ve výsledku najít cestu, na úkor rychlosti výpočtu řešení.



(a) (b)

Obrázek 2.5: Řešení čistě Dirichletova problému. (a) průběh potenciálu, (b) nalezené cesty k cíli.

2.3.2 Čistě Neumannův problém

V tomto případě se problém popíše výhradně Neumannovými podmínkami, tedy

$$\frac{\partial u}{\partial \mathbf{n}} = g,$$

kde $\mathbf{n}$ značí vnější normálový vektor elementu hranice oblasti (obrázek 2.6). Takto specifikované okrajové podmínky vedou k řešení, které není narozdíl od čistě Dirichletova problému limitováno výpočetní přesností, ale na druhou stranu generují singulární matici $\mathbf{A}$, protože v případě, že není zadaná žádná podmínka na funkční hodnotu, existuje celý prostor hledaných řešení lišících se o konstantu. S tímto problémem se lze samozřejmě vyrovnat několika způsoby [10]. Nejjednodušším způsobem je dodatečně podmínit hodnotu potenciálu v jednom z uzlů, to odpovídá operaci

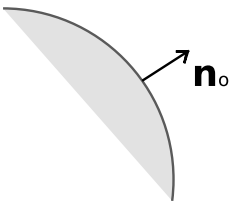
$$\mathbf{A}_m = \mathbf{I}' \cdot \mathbf{A} \cdot \mathbf{I},$$

$$\mathbf{F}_m = \mathbf{I}' \cdot \mathbf{F}$$

a

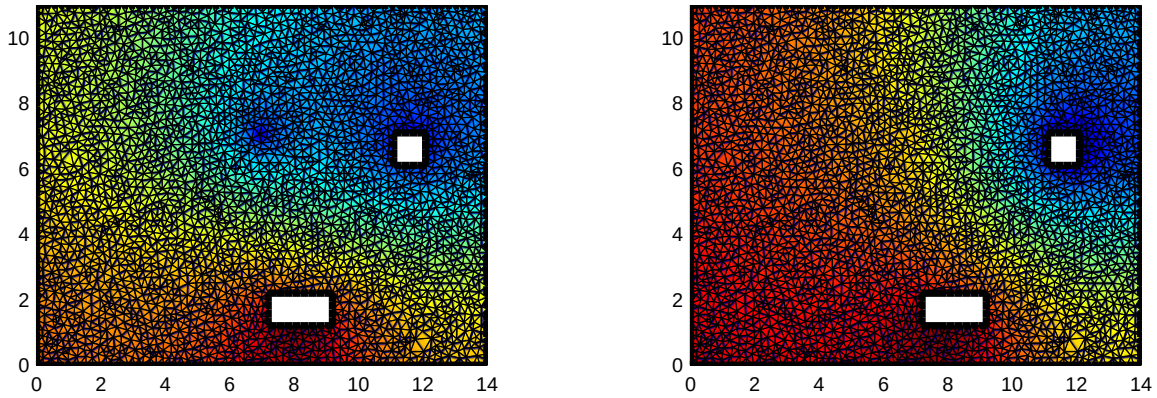
$$\mathbf{u} = \mathbf{I} \cdot \mathbf{u}_m,$$

kde $\mathbf{I}$ je jednotková matice s jednou nulou na hlavní diagonále a $\mathbf{u}_m$ je řešení modifikované soustavy. V podstatě to znamená, že se vypustí rovnice pro jeden uzel a potenciál v daném bodě se nastaví na hodnotu 0. Po výpočtu ostatních hodnot se vektor obsahující řešení doplní o vypuštěnou hodnotu – to odpovídá násobení zleva maticí $\mathbf{I}$. Bohužel uzel, který bude vypuštěn, nelze vybrat úplně náhodně, protože tato volba ovlivňuje podmíněnost dané soustavy a tedy kvalitu výsledného řešení. Nejjednoušší volbou je podmínit některý z uzlů cílové okrajové podmínky – tam je jediné globální minimum a jeho hodnota není až tak důležitá. Ačkoliv podmínka (2.2) musí být splněna vždy, v případě čistě Neumannova



Obrázek 2.6: Vnější normálový vektor.

problému a nastavení podmínek po celé hranici to platí dvojnásob. I při nesplnění této podmínky se sice nějaký výsledek získá, ale nemusí to být harmonická funkce – uvnitř def. oboru se mohou objevit lokální extrémy viz obrázek 2.7(a). Za povšimnutí stojí i lokální maximum na obrázku 2.7(b) v případě, kdy je integrál přes hranici nulový až na výpočetní nepřesnost $(-2,2204 \cdot 10^{16})$. To dobře ilustruje špatnou podmíněnost tohoto problému.



Obrázek 2.7: Řešení čistě Neumannova problému. (b) dodržena podmínka (2.2), (a) nedodržena podmínka (2.2).

2.3.3 Smíšený problém

V praxi se jako nejvýhodnější jeví problém popsat jak Neumannovými, tak Dirichletovými podmínkami, neboť v tomto případě je řešení nejméně citlivé na správně zadané okrajové podmínky a nejsou kladeny požadavky na jednotnost podmínek. Konkrétně lze hranici cíle nastavit funkční hodnotu jako Dirichletovu podmínku a všechny překážky, kterým je třeba se vyhnout, popsat Neumannovými podmínkami s kladnými derivacemi – aby potenciál kolem překážek rostl.

V tomto případě, stejně jako v případě čistě Dirichletova problému je podmínka (2.2) splněna implicitně a není potřeba se o její splnění starat, což v praxi přináši výrazné zjednodušení situace. Další výhodou volby smíšeného popisu je nesingularita globální matice problému a tedy jednodušší výpočet výsledného řešení.

2.4 Generování cesty

Zadefinují-li se hranicím oblasti hodnoty okrajových podmínek a do mapy je umístěna cílová pozice, které je také nastavena okrajová podmínka, vypočtené uzlové hodnoty harmonického potenciálu ohodnotí uzly triangulace, buďto k cíli sestupně, nebo vzestupně. Uzly triangulace s funkční hodnotou potenciálu pak tvoří ohodnocený graf, ve kterém se cesta vyhledává.

V obecném případě je ohodnocení uzlů nemonotónní a k nalezení cesty z vybraného uzlu do cílového uzlu je třeba použít některý z prohledávacích algoritmů – do šířky, do hloubky, apod. K transformaci tohoto obecného ohodnocení na monotónně klesající lze použít Dijkstrův algoritmus pro nalezení nejkratší cesty ze všech uzlů grafu do daného cíle. Pro vyhledání cesty v takto monotónně ohodnoceném grafu pak stačí použít gradientní metody, která v každém kroku bude vybírat nejlevnější variantu.

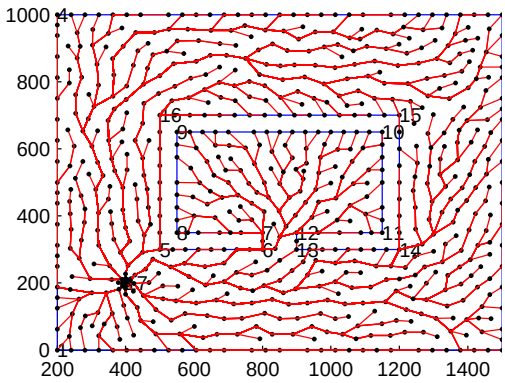
Podobného monotónního ohodnocení lze dosáhnout i použitím harmonického potenciálu, který proti Dijkstrově algoritmu navíc navádí robot od překážky. Další výhodou ohodnocení grafu harmonickým potenciálem je možnost specifikovat obecnější cíl, než je jeden uzel – cílem může být celá oblast s libovolnou hranicí, případně může být oblastí více.

2.4.1 Cesta po vrcholech triangulace

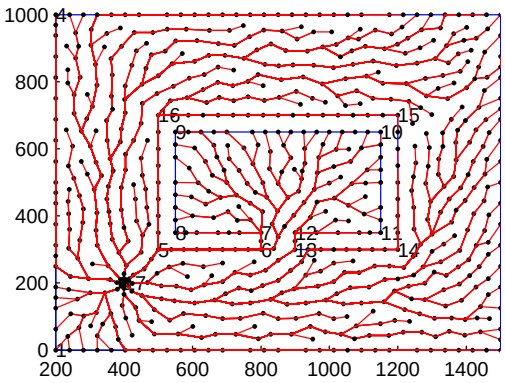
Generovat cestu lze více způsoby. Tím nejjednodušším a nejbezpečnějším způsobem – z hlediska vyhýbání se překážce –, je procházet sousední uzly triangulace a vždy vybírat ten s nejmenší hodnotou

$$u_{i+1} = min\left\{f(u_i) - f(u)|u \in U_s\right\},$$

kde U_s je množina sousedních uzlů. Takový přístup však negeneruje hladké ani euklidovsky krátké cesty, jak lze vidět na obrázku 2.8.



Obrázek 2.8: Cesta z jednotlivých uzlů generované metodou minima.



Obrázek 2.9: Cesta z jednotlivých uzlů generované metodou nejmenší derivace.

Obdobným postupem, který generuje plynulejší cesty, je vybírání uzlu ve směru nejmenší (největší záporné) derivace

$$u_{i+1} = min\left\{\frac{f(u_i) - f(u)}{||u_i - u||}|u \in U_s\right\},$$

kde u_i značí současnou pozici (uzel) a U_s je množina sousedních uzlů. Jde vlastně o zobecnění předchozí metody, neboť v okamžiku, kdy budou trojúhelníky rovnostranné resp. všechny stejné, tak obě metody dají stejný výsledek. Cesty generované touto metodou jsou na obrázku 2.9. Tyto dva postupy mají tu výhodu, že nemohou robot navést do překážky, neboť se pohybují pouze po vrcholech triangulace a oblasti pokryté překážkami triangulovány nejsou.

2.4.2 Cesta po aproximovaném gradientu

Připustí-li se možnost, že cesta může být složena nejen z uzlů triangulace, ale i z bodů ve volném prostoru, je možno na základě ohodnoceného grafu generovat v podstatě hladké cesty (v limitním případě). Princip spočívá v nalezenení tří nejbližších uzlů pro danou pozici – jakoukoliv, i mimo uzel – těmito třemi uzly proložit rovinu a analyticky nalézt směr největšího klesání vůči rovině $z = 0$. V tomto směru pak postoupit o definovaný krok. Délka kroku souvisí s velikostí jednotlivých trojúhelníků a není vhodné ji volit výrazně menší než je průměrná délka hrany triangulace – nemá velký smysl učinit krok tak malý, aby pro nově dosaženou pozici byly nejbližší ty samé tři uzly. S rostoucí délkou kroku klesá výpočetní čas potřebný k vyhledání cesty. Na druhou stranu, čím kratší bude krok, tím hladší bude výsledná cesta.

Tento postup má však jednu zásadní nevýhodu, může robot navést přímo do překážky. To v případě, že se jedná o úzkou překážku a jeden z nejbližších uzlů leží na „druhé straně“ překážky. Druhý případ může nastat, je-li velikost kroku větší než maximální možná volná vzdálenost před překážkou. Zatímco selhání druhého typu lze řešit poměrně efektivně variabilní velikostí kroku a detekcí překročení hranice, selhání prvního typu se ošetřují těžko, ideální je zmenšit stranu triangulace – tím se počet případů výrazně redukuje, ale problém to neřeší.

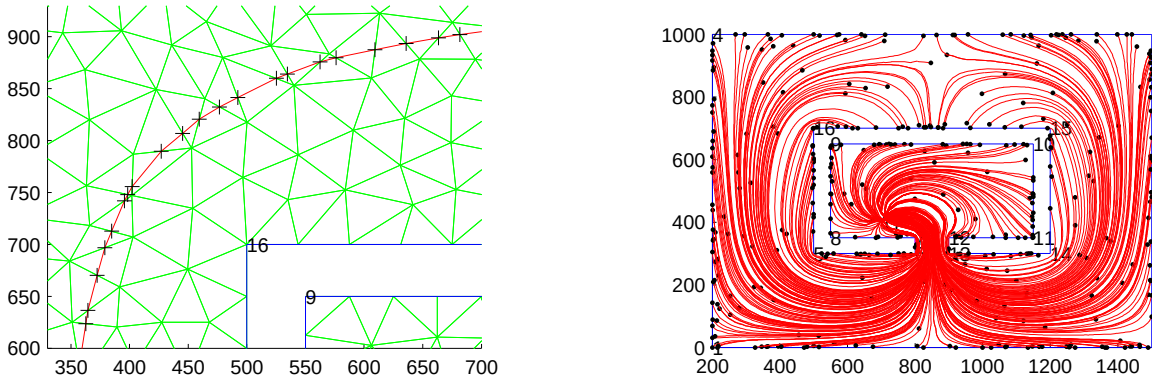
Cesta po aproximovaném gradientu nad konečným prvkem

Vhodnější postup, jak generovat kvalitní hladkou cestu, je postupovat po gradientech jednotlivých prvků triangulace, detail cesty je na obrázku 2.10(a). U tohoto postupu odpadá nebezpečí zásahu překážky, neboť se cesta plánuje jen nad trojúhelníky pokrytou oblastí. Další výhodou je možnost dynamicky určovat délku kroku tak, aby byl každý z trojúhelníků překonán právě jedním krokem. Algoritmus lze rozdělit na dvě fáze, které se opakují, dokud není dosaženo cíle:

1. hledání trojúhelníku, ve kterém se algoritmus právě nachází,
2. výpočet gradientu a posun v jeho směru do dalšího trojúhelníku.

Problém nastává v okamžiku, kdy gradienty sousedících trojúhelníků jdou proti sobě a hledání cesty se zastaví, protože se neví, kterým směrem pokračovat. Z tohoto důvodu je třeba algoritmus rozšířit např. tak, že se v takovém případě provede krok ve směru součtu gradientů, jinou možností je provést krok do společného uzlu s nižší hodnotou. Cesty nalezené touto metodou jsou na obrázku 2.10(b).

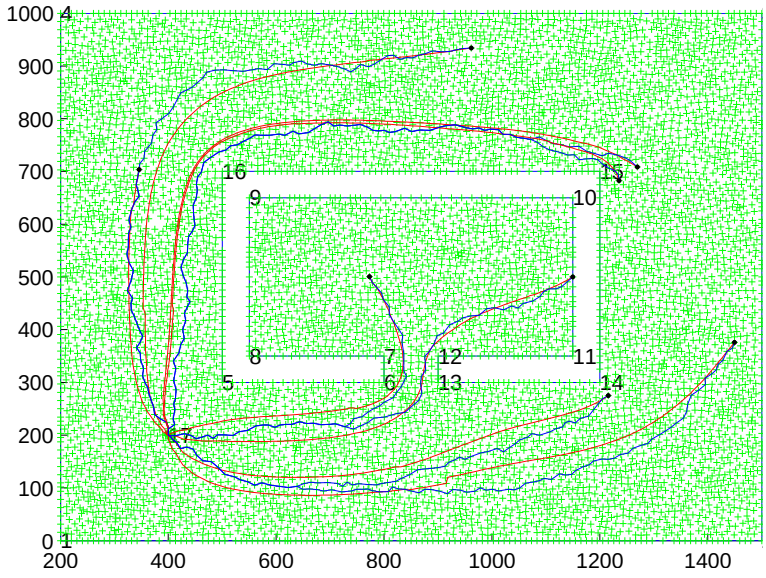
Pro efektivní vyhledávání trojúhelníku vhodného pro naplánování dalšího kroku je vhodné k uložení všech trojúhelníků využít KD strom, který bude vyhledávat nejbližší trojúhelníky na základě vzdálenosti jejich těžišť od aktuální pozice. V této práci je využito implementace LIBKDTREE [21]. Ještě vhodnější variantou, jak efektivně vyhledávat sousední trojúhelníky, je uložení triangulace jako spojového seznamu, kdy bude mít každý prvek triangulace u sebe uloženy odkazy na své sousedy.



(a) (b)

Obrázek 2.10: (a) Detail cesty nalezené jako posloupnost gradientů jednotlivých trojúhelníků, černé křížky označují jednotlivé lineární segmenty cesty. (b) Cesty z jednotlivých uzlů generované metodou prokládání rovinou. Mapa BugTrap.

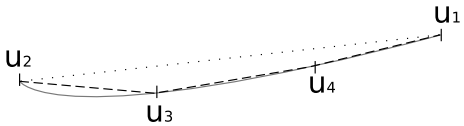
Zajímavé, i když předpokládatelné, je, že se zmenšující se velikostí strany triangulace se cesty vygenerované jednotlivými metodami k sobě přibližují a dá se předpokládat, že v limitním případě, kdy $d \to 0$, všechny tři postupy dají stejnou cestu a to tu, která by vzešla z aplikace gradientu na analytické řešení Laplaceovy rovnice. Postupná konvergence cest je naznačena na obrázku 2.11.



Obrázek 2.11: Přibližování se cest získaných různými způsoby při malé straně triangulace, maximální délka hrany triangulace je v tomto případě $d = 20$.

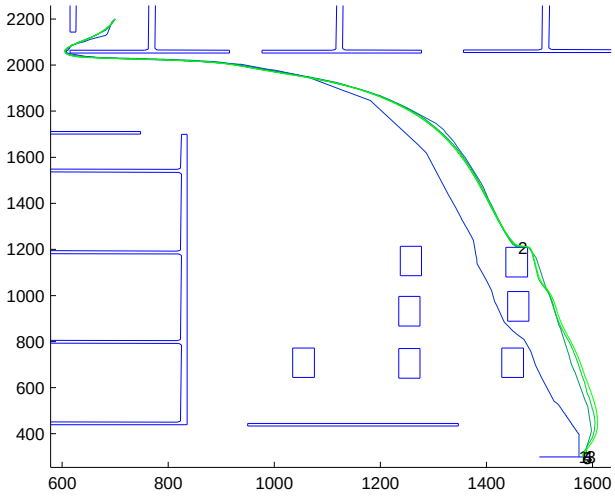
Iterativní cesta po aproximovaném gradientu

Pokud se hledá cesta ve velmi rozsáhlé mapě s málo nebo malými překážkami, pro výše zmíněné postupy generování cesty je třeba jemné triangulace v celé oblasti, neboť obecně neexistuje apriorní informace o tom, kudy přibližně cesta povede a hrubá triangulace vede na velké chyby v aproximaci řešení, což v důsledku zvyšuje riziko zásahu překážky při plánování po aproximovaném gradientu, jak je vidět na obrázku 2.12.



Obrázek 2.12: Důsledek příliš hrubé triangulace. V případě, že u_1 a u_2 tvoří jeden konečný prvek, aproximace řešení lineární funkcí bude natolik nepřesná, že gradient nad tímto prvkem povede přímo do překážky, naproti tomu při rozdělení na uzly u_1, u_2, u_3 a u_4 směřuje aproximovaný gradient prvku u_2, u_3 od překážky.

V tomto případě lze použít iterovaného hledání cesty postupným zjemňováním triangulace. Princip spočívá v přidávání cesty nalezené v předchozí iteraci jako dodatečnou podmínku triangulace, což má za následek jemnější triangulaci, ale pouze nad oblastí, kudy povede výsledná cesta. V každé iteraci tedy algoritmus dává přijatelné řešení, jehož kvalita – hladkost, délka – se zlepšuje. Příklad cesty nalezené iterativní metodou je na obrázku 2.13.

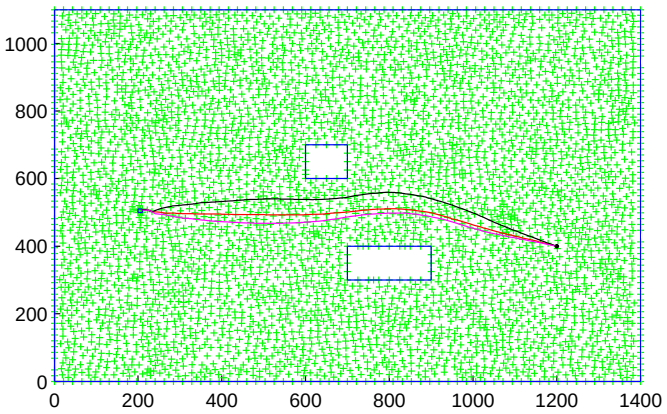


Obrázek 2.13: Průběh vylepšování výsledné cesty iterativním algoritmem. Výsledná cesta získaná v 5. iteraci je obarvena světle zeleně. Dále si lze všimnout, že v druhé iteraci skončilo hledání cesty nárazem do překážky. Přidáním této neúspěšné trajektorie dojde ke zjemnění triangulace před danou překážkou a tím k lepší aproximaci řešení Laplaceovy rovnice a v důsledku bude v dané oblasti převažovat gradient, který bude robot navádět od překážky.

Algoritmus lze ukončit po definovaném počtu iterací nebo poté, co jsou splněna kritéria pro kvalitu řešení. Iterované hledání cesty je vhodné také použít v případě, kdy metoda s aproximovaným gradientem selže a nalezená cesta vede do překážky.

2.4.3 Vliv hodnot okrajových podmínek

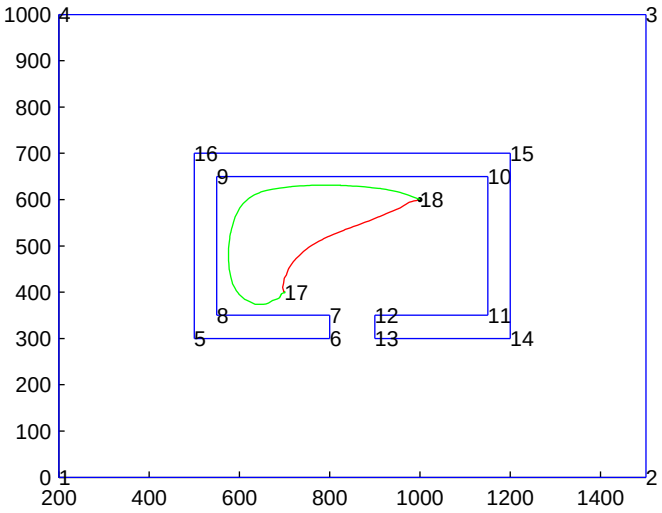
Na výsledný průběh harmonické funkce má vliv samozřejmě nejen volba typu okrajové podmínky, ale i její číselná hodnota, a to zejména u Neumannových podmínek. Zde velikost derivace u jednotlivých překážek určuje preferovanou vzdálenost nalezených cest od dané překážky. Výsledná vzdálenost, ve které bude robot míjet danou překážku, není dána konkrétní hodnotou podmínky, ale poměrem hodnot překážek v nejbližším okolí, jak je vidět na obrázku 2.14.



Obrázek 2.14: Různé varianty nalezené cesty v závislosti na volbě hodnot Neumannových podmínek ohraničujících překážky. V případě červené cesty jsou hodnoty totožné, v případě černé resp. fialové je hodnota vzdálenější překážky dvojnásobná.

V případě, kdy není třeba znát cesty ze všech bodů prostoru, ale pouze cestu z bodu A do bodu B, lze zvolit ještě jiný způsob popisu. Startu (A) i cíli (B) nastavit Dirichletovu podmínku, startu kladnou, cíli zápornou a všem překážkám a hranicím nulovou Neumannovu podmínku. Takové harmonické pole bude sice do cíle navádět pouze z daného bodu, ale pro cesty uvnitř téměř uzavřené oblasti[4] bude dávat mnohem lepší cesty, než při parametrizaci pole pro hledání cest z celé mapy. Příklad takové cesty včetně srovnání je na obrázku 2.15. Je vidět, že takto získaná cesta je výrazně kratší než cesta získaná při plánování pro všechny body mapy. Ovšem výpočet pole pro jednu takovou cestu zabere stejně času, jako výpočet pole s jinou parametrizací pro nalezení všech cest, a proto se pro plánování inspekční úlohy v okamžiku, kdy je třeba spočítat vzdálenosti mezi všemi měřicími místy příliš nehodí.

[4]Typickým příkladem je mapa *BugTrap*.



Obrázek 2.15: Zelená cesta byla získána plánováním pro všechny body mapy, červená plánováním pro dva body mapy.

2.5 Zobecněná místa

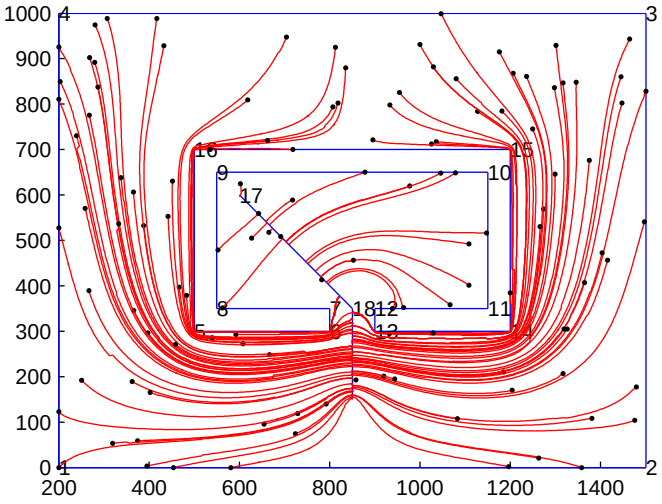
Jednou z unikátních možností při plánování využitím harmonického potenciálu je možnost definovat jako cíl obecnější útvar, než je bezrozměrný bod – např. libovolný, i nekonvexní polygon, případně je možné takových cílů definovat více. V takovém případě se hledá cesta, která vede podle gradientu pole na nejbližší bod z množiny bodů tvořících cíl, jak je naznačeno na obrázku 2.16. Této vlastnosti lze využít v okamžiku, kdy je třeba se po optimální cestě dostat do nějaké oblasti a na konkrétní poloze v rámci oblasti již moc nezáleží. Této vlastnosti lze dobře využít ve spojení s řešením *AGP* pro segmentové strážce [20].

Přibližně podobného výsledku při plánování podle nejkratší cesty grafem lze dosáhnout jedině tak, že se cílová oblast navzorkuje vybraným počtem bodů, a pak se pro každý bod zvlášť najde cesta ze zadaného místa. Ze všech takto nalezených cest se pak vybere ta nejlepší.

Trvá-li nalezení jedné cesty využitím Dijkstrova algoritmu čas n a nalezení cesty harmonickým potenciálem N, pak existuje m takové, že platí

$$m \cdot n \geq N,$$

kde m je počet vzorků cílové oblasti.



Obrázek 2.16: Ilustrace použití zobecněného cíle a následně nalezených cest pro několik náhodně vybraných míst mapy.

2.5.1 Rozklad mapy

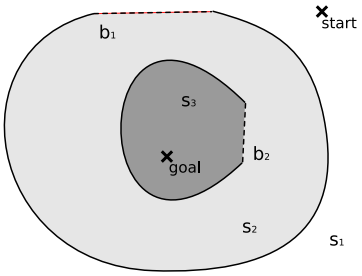
Zobecněných cílů lze využít také k rozkladu velmi rozsáhlé nebo velmi členité mapy na menší části, ve kterých je možno plánovat výrazně rychleji. V takovém případě bude cílový polygon v jednom segmentu hranicí pro druhý segment, jak je naznačeno na obrázku 2.17.

Rozdělené oblasti tvoří uzly grafu, sousedící oblasti jsou v tomto grafu spojeny neohodnocenými neorientovanými hranami. Plánování pak probíhá ve dvou krocích. Nejprve se prohledáním grafu zjistí, kterými oblastmi cesta povede, a v druhém kroku se najdou jednotlivé části cesty nad jednotlivými segmenty. Výsledná cesta se získá zřetězením těchto částí cest. Obecně je třeba spočítat N potenciálů,

$$N = \sum_{i=1}^{n} n_{bi},$$

kde n je počet segmentů a n_{bi} počet podcílů – míst, přes která jsou jednotlivé segmenty propojeny – na daném segmentu.

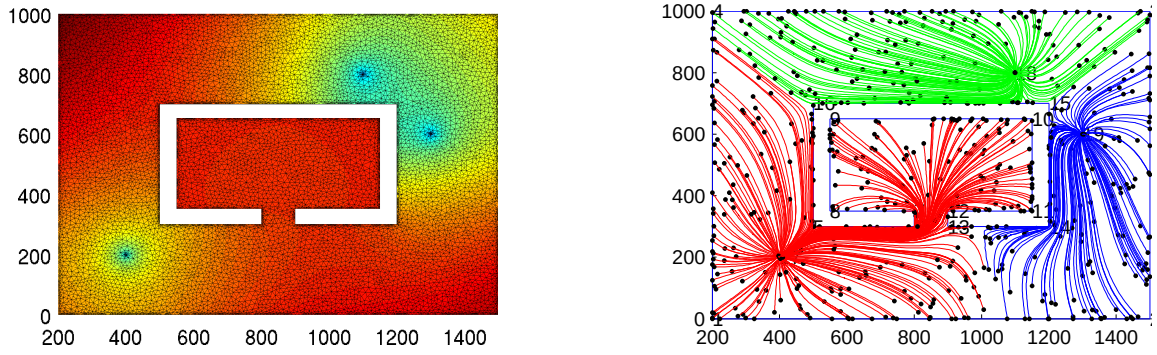
Nevýhodou tohoto přístupu je nutnost zpracování mapy za účelem nalezení těchto podcílů, což není triviální úloha. Jednou z možností je umístit podcíle do nejužších míst mapy, kterými je třeba projíždět. Např. pokud se inspekce provádí v budově s místnostmi, dají se podcíle umístit do všech vchodů místností, pak probíhá plánování nezávisle pro každou místnost. Další možností je použít k rozkladu mapy některé dělící úsečky získané při řešení *AGP* rozdělováním na konvexní polygony.



Obrázek 2.17: Rozdělení mapy na segmenty s_1, s_2 a s_3 se dvěma zobecněnými cíly b_1 a b_2 (označeny přerušovanou čarou), nad každým ze segmentů lze harmonický potenciál počítat nezávisle. V zobrazeném případě povede cesta přes všechny tři segmenty, nejprve se nalezne cesta od startu na podcíl b_1, následně cesta od podcíle b_1 k podcíli b_2 a z podcíle b_2 do cíle (*goal*).

2.5.2 Více cílů

Jinou aplikací je jízda do nejbližšího z více možných cílů, takovým cílem může být v praxi například dobíjecí stanice. Atraktivitu jednotlivých cílu je ještě možné ovlivňovat nastavením okrajové podmínky – předepsanou hodnotou potenciálu v těchto místech. Čím nižší bude hodnota, tím bude cíl přitahovat roboty z většího okolí na úkor ostatních cílů. Vliv hodnot potenciálů v jednotlivých cílech je opět jako v případě podkapitoly 2.4.3 vázán na hodnoty okolních cílů a rozhoduje jejich poměr. Příklad ilustruje obrázek 2.18.



(a) (b)

Obrázek 2.18: (a) Průběh harmonického potenciálu pro více cílů, modrá barva značí minimum – polohy cílů. (b) Rozdělení mapy na oblasti podle působnosti zadaných cílů.

Kapitola 3

Experimenty

V této kapitole jsou experimentálně ověřeny navržené postupy a teoretická tvrzení z předchozí kapitoly. Je zkoumán vliv parametrů triangulace na rychlost výpočtu, cesty získané plánováním v harmonickém potenciálu jsou srovnány s cestami získanými plánováním podle Dijkstrova ohodnocení určujícího nejkratší cestu v grafu, v tomto případě jsou hrany triangulace ohodnoceny Eukleidovskou vzdáleností. Dále je srovnána délka cest jdoucích po uzlech triangulace a postupujících přes jednotlivé konečné prvky v závislosti na parametrech triangulace. Druhá polovina kapiloty je věnována řešení inspekční úlohy. Jsou srovnány navržené přístupy řešení, dále jsou zhodnoceny rozdíly v délce a času inspekce při plánování cest harmonickým potenciálem a grafem viditelnosti. Je popsán experiment využívající k nalezení řešení inspekční úlohy zobecněných míst a neuronové sítě a v závěru kapitoly je proveden experiment s robotem v reálném prostředí.

Experimenty využívají třech různých map, uměle vytvořené *BugTrap* na obrázku 3.7(a) a dvou map vycházejících z reálného prostředí – *TestArea* na obrázku 3.9(b) a *jari-huge* na obrázku 3.1(b).

Experimenty jsou prováděny v MATLABu spuštěném v prostředí linux na počítači s procesorem *Core 2 Duo* 1,8GHz, 2GB RAM. K triangulaci je využito knihovny CGAL, triangulace a výpočet koeficientů jednotlivých konečných prvků je realizován v *C++* jako *mex* funkce. V *C++* je také implementováno hledání cesty v grafu triangulace a taktéž je z MATLABu voláno jako *mex* funkce. *C++* kódy jsou kompilovány kompilátorem *g++* s optimalizací *02*. K řešení úlohy více obchodních cestujících je využito programu *mtsplib* [33] implementujícího heuristiku *GENIUS*, k výpočtu přesného řešení úlohy *TSP* je využito volně dostupné aplikace *Concorde* [2].

3.1 Vliv parametrů triangulace

Jak již bylo naznačeno dříve, konkrétní podoba triangulace má velký vliv na kvalitu aproximace harmonické funkce a tedy i na kvalitu nalezených cest. Nejzákladnějším pozorováním je, že v případě, kdy je triangulace příliš řídká – zejména v blízkém okolí překážek a hranic oblasti –, hrozí nebezpečí střetu robotu s překážkou. Možností jak tento problém

omezit je více a každá z nich má svá pro a proti.

3.1.1 Velmi jemná triangulace

Jednou možností je zvolit malou hodnotu pro maximální délku hrany triangulace. To vede na velmi jemnou triangulaci, která produkuje kvalitní cesty po celé oblasti. Velmi limitujícím faktorem v tomto případě je však velmi velké množství trojúhelníku na rozsáhlých mapách a s tím spojené velké pamětové a časové nároky na výpočet.

V tabulce 3.1 jsou uvedeny průměrné počty vrcholů a trojúhelníků triangulace a průměrné časy potřebné k výpočtu harmonického potenciálu pro mapu *BugTrap*. Je vidět, že pro malé problémy je výpočet dostatečně rychlý, jakmile má triangulace více než cca 10 000 trojúhelníků, výpočet zabere řádově jednotky sekund. V tabulce jsou uvedeny i jednotlivé mezičasy výpočtů a je vidět, že nejvíce času trvá sestavování globální matice v MATLABu.

Z implementačního hlediska je zajímavá otázka, zda a případně o kolik by se výpočet urychlil realizací sestavení globální matice jako *mex* funkce v *C++*.

d_{max}	n_{vertex}	n_{Δ}	t_{triang}	t_{pp}	t_{global}	t_{total}
1 000	179,0	277,0	0,003	0,000	0,057	0,057
500	179,0	277,0	0,003	0,002	0,065	0,067
200	218,4	346,8	0,002	0,001	0,074	0,075
150	288,5	471,9	0,002	0,003	0,094	0,097
100	438,6	747,2	0,004	0,002	0,137	0,139
80	624,3	1 080,8	0,009	0,003	0,201	0,204
60	975,3	1 728,6	0,010	0,011	0,331	0,342
50	1 395,4	2 539,1	0,014	0,021	0,507	0,528
40	2 130,0	3 920,9	0,026	0,045	0,848	0,893
30	3 644,0	6 851,8	0,040	0,134	1,765	1,899
20	7 856,6	15 070,2	0,084	0,586	5,229	5,815
15	13 916,5	26 961,0	0,158	1,884	17,117	19,001
10	30 749,9	60 208,4	0,386	10,082	92,677	102,759

Tabulka 3.1: Počet trojúhelníků a vrcholů v závislosti na maximální přípustné délce hrany triangulace. d_{max} je maximální délka hrany, n_{vertex} je průměrný počet vrcholů, n_{Δ} je průměrný počet trojúhelníků, t_{triang} je čas triangulace, t_{pp} čas výpočtů nad jednotlivými trojúhelníky, t_{global} čas potřebný k sestavení globální matice a vyřešení a t_{total} je celkový čas výpočtu, všechny časy jsou v sekundách.

3.1.2 Po okrajích zjemněná triangulace

Druhou možností je zvolit jemnější triangulaci pouze podél hranic oblasti. Toho lze docílit modifikací nastavených omezení triangulace v knihovně CGAL, konkrétně specifikací maximální délky hrany v případě, že některá z hran daného trojúhelníku leží na hranici

d_{max}	$d_{boundary}$	n_{vertex}	n_{Δ}	t_{triang}	t_{pp}	t_{global}	t_{total}
100	60	3 729,0	6 104,0	0,075	0,130	1,432	1,562
100	50	4 002,0	6 613,0	0,074	0,142	1,524	1,666
100	40	4 311,0	7 171,0	0,078	0,165	1,658	1,823
100	30	4 767,0	7 948,0	0,087	0,202	1,898	2,100
100	20	5 991,0	9 806,0	0,116	0,308	2,457	2,765
100	10	8 388,0	13 974,0	0,151	0,610	4,109	4,719
100	5	11 972,0	19 604,0	0,227	1,212	7,499	8,711

Tabulka 3.2: Počet trojúhelníků a vrcholů v závislosti na maximální přípustné délce hrany triangulace. Význam symbolů je stejný jako v tabulce 3.1, $d_{boundary}$ je maximální přípustná délka hrany triangulace pro trojúhelníky ležící na hranici oblasti. t_{triang} je čas triangulace, t_{pp} čas výpočtů nad jednotlivými trojúhelníky, t_{global} čas potřebný k sestavení globální matice a vyřešení a t_{total} je celkový čas výpočtu, všechny časy jsou v sekundách.

oblasti. Výsledná triangulace pak má na volném prostoru relativně větší trojúhelníky, zatímco podél hranic trojúhelníky menší a výrazně se tak eliminuje riziko navedení robotu do překážky. Nevýhodou tohoto řešení je také nárůst počtu trojúhelníků a problémů s tím spojených, i když ne v takovém rozsahu jako při stejnoměrné triangulaci na celé mapě.

V tabulce 3.2 jsou uvedeny počty vrcholů, trojúhelníků a průměrné časy výpočtu. Je vidět, že při dosažení víceméně stejné přesnosti – stejně husté triangulace – podél hranice, kde je to pro plánování nejdůležitější, jsou časy výpočtu výrazně nižší (4 s versus 19 s), což zejména při plánování pro úlohu inspekce, kdy je třeba napočítat vzdálenosti mezi všemi měřicími místy, přináší výraznou úsporu času.

3.2 Srovnání výsledných cest

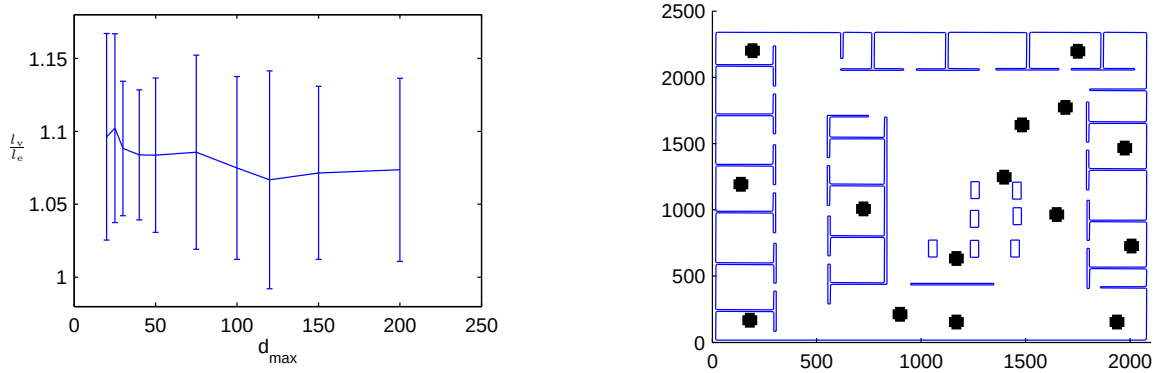
Efektivitu a kvalitu plánování v harmonického potenciálu je vhodné srovnat s jinými přístupy, aby se ukázalo, zda taková metoda má šanci v konkurenci obstát. Jako srovnávací metody poslouží plánování prohledáním grafu ohodnoceným Dijkstrovým algoritmem ohodnocujícím uzly podle jejich Eukleidovské vzdálenosti[1], neboť nabízí stejný výstup, jako plánování využitím harmonického potenciálu – pro jedno zpracování mapy dává cesty ze všech bodů mapy do příslušného cíle.

3.2.1 Srovnání cesty po vrcholech triangulace a cesty po prvcích

Rozdíl mezi cestou po vrcholech triangulace a cestou po prvcích triangulace je naznačen v kapitole 2.4 na obrázcích 2.9 a 2.10(b). Graf na obrázku 3.1(a) znázorňuje poměr délek cest po vrcholech a po prvcích v závislosti na maximální přípustné délce hrany triangulace. K vyhodnocení je použito všech cest mezi 15-ti vybranými místy na mapě *jari-huge*, tzn. 105

[1]Dále označováno pouze jako Dijkstrovo ohodnocení.

cest pro každou úroveň jemnosti triangulace. Z grafu je vidět, že cesty po vrcholech jsou vždy přibližně o 5-10 % delší.



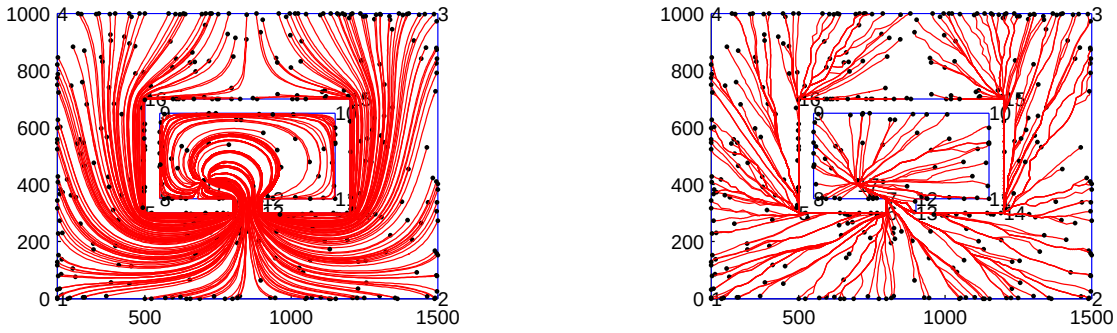
(a) (b)

Obrázek 3.1: (a) Závislost poměru délek cest na maximální délce hrany triangulace, l_v značí délku cesty po vrcholech triangulace, l_e označuje délku cesty po trojúhelnících. (b) Polohy testovacích míst.

3.2.2 Srovnání s ohodnocením Dijkstrovým algoritmem

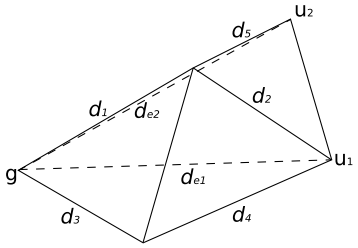
Pro obě metody lze použít stejného algoritmu hledání cesty a stejné triangulace, neboť se od sebe liší pouze ohodnocením uzlů grafu. V případě ohodnocení podle Dijkstrova algoritmu je hodnota v uzlu triangulace dána délkou nejkratší cesty v grafu mezi daným uzlem a cílovou pozicí. Daného ohodnocení lze také využít k aproximaci gradientu, ale výsledná funkce bude mít výrazně jiný průběh a bude upřednostňovat nejkratší cesty. Jak rozdílné jsou cesty v závislosti na typu ohodnocení je vidět na obrázku 3.2(a), respektive na obrázku 3.2(b). Již z obrázků je patrné, že cesty získané harmonickým potenciálem jsou euklidovsky delší, nejvýrazněji se to projevuje na krátkých vzdálenostech, kdy gradient harmonického potenciálu není osově symetrický a cesty jsou zbytečně zakřivené a dlouhé. Je však také vidět, že cesty nalezené po ohodnocení podle Dijkstrova algoritmu sice respektují překážky, ale pohybují se v jejich těsné blízkosti a s prudkou změnou tvaru překážky se prudce – nehladce – mění i směr cesty, což může při řízení robotu dělat potíže.

Směrová nestálost cest (klikatost) volným prostorem pozorovatelná na obrázku 3.2(b) je důsledkem triangulace oblasti a ohodnocování uzlů jen na základě vzniklého grafu, který již neobsahuje informace o vzájemné poloze uzlů v prostoru, a tudíž není možno určit skutečnou nejkratší vzdálenost pro danou dvojici (uzel, cíl), jak je naznačeno na obrázku 3.3. V důsledku toho dva uzly Eukleidovsky přibližně stejně daleko mají podle Dijkstrova algoritmu výrazně jiné ohodnocení, což vede na lokálně nekonzistentní gradient. Ze srovnání vyplývá, že ve volném prostoru, tzn. pokud je zajištěna přímá viditelnost, dává lepší cesty graf ohodnocený Dijkstrovým algoritmem – ve smyslu Eukleidovské délky a profilu



(a) (b)

Obrázek 3.2: (a) Naplánované cesty využitím harmonického potenciálu, polygonální mapa BugTrap, 7910 vrcholů (13841 trojúhelníků). (b) Tatáž úloha jako na obrázku (a), identická triangulace, ohodnocení uzlů na základě Dijkstrova algoritmu, cesty podle aproximovaného gradientu.



Obrázek 3.3: Ohodnocení Dijkstrovým algoritmem pro uzel u_1 vůči cíli g bude buď d_1+d_2, nebo d_3+d_4, ale to je vždy víc, než euklidovská vzdálenost d_{e1}, naproti tomu ohodnocení uzlu u_2 bude d_1+d_5, což je téměř stejné jako euklidovská vzdálenost d_{e2} dvojice (g, u_2).

trajektorie –, neboť se jedná v podstatě o rovné cesty a harmonický potenciál má tendence hledat zakřivenené cesty. Naopak v okamžiku, kdy není cíl přímo dosažitelný, graf ohodnocený Dijkstrovým algoritmem vygeneruje cestu vedoucí po hranici překážky, neboť ta je Eukleidovsky nejkratší, kdežto ohodnocení harmonickým potenciálem povede robot ve větší vzdálenosti od překážky.

mapa	**počet vrcholů**	**počet cest**	$r(l_{harm}, l_{sgp})$
jari-huge	46 616	11 654	1,0801
bugtrap	14 254	14 254	1,2531
testarea	24 546	12 273	1,1122
celkově		38 181	1,1550

Tabulka 3.3: Průměrný poměr délek cest.

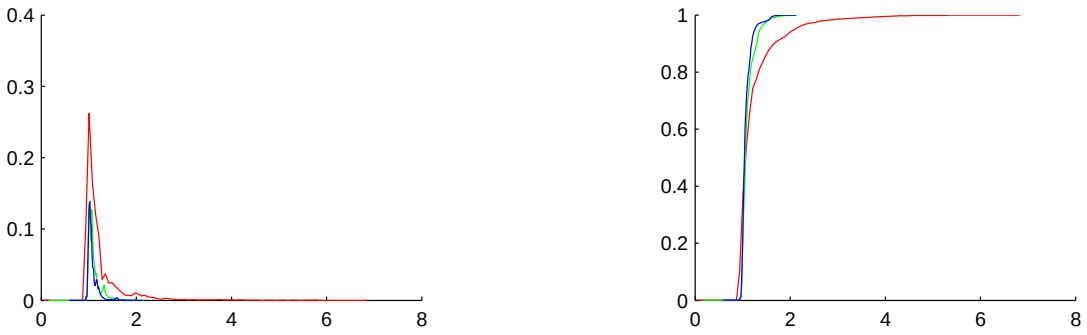
Provede-li se srovnání délek cest a to tak, že pro každou dvojici (start, cíl) se nalezne

cesta oběma metodami a určí se poměr jejich délek

$$r(l_{harm}, l_{sgp}) = \frac{l_{harm}}{l_{sgp}},$$

dá se určit průměrný rozdíl délek. Jak lze vidět z tabulky 3.3, jsou cesty získané plánováním v harmonickém potenciálu průměrně o 15,5 % delší než cesty získané plánováním nad uzly ohodnocenými Dijkstrovým algoritmem. Tyto hodnoty se ještě pro každou mapu mírně liší, na mapě *BugTrap* je rozdíl 25,3 %, na mapě *jari-huge* je rozdíl 8,01 %. Výchylka je částečně způsobená také volbou cíle nad danou mapou, čísla v tabulce 3.3 jsou průměrována pro dvě náhodně vybraná místa v každé mapě a pro náhodně vybrané cesty vedoucí z celé mapy do tohoto místa.

Na obrázku 3.4(a) je histogram, na obrázku 3.4(b) odhad distribuční funkce poměru délek cest pro mapu *TestArea*. Z odhadu distribuční funkce je vidět, že hodnoty poměrů $r(l_{harm}, l_{sgp})$ nemají velký rozptyl, zejména pro mapy s jemnou triangulací.



(a) (b)

Obrázek 3.4: Rozložení poměrů délek cest získaných plánováním v harmonickém potenciálu a v ohodnocení Dijkstrovým algoritmem. Červený průběh je pro mapu *BugTrap*, zelený pro *TestArea* a modrý pro *jari-huge*. (a) histogram, (b) odhad distribuční funkce.

Srovnají-li se časy nutné k průjezdu cesty nalezené ohodnocením harmonickým potenciálem a cesty nalezené Dijkstrovým algoritmem, ukáže se, že se dají projet ve víceméně stejném čase. To je částečně dáno parametry modelu robotu, zejména velkým dopředným zrychlením. Použité parametry modelu odpovídající robotu G^{2}BOT^2 jsou uvedeny v tabulce 3.4.

Z nalezené cesty lze pro daný model robotu vygenerovat trajektorii, tzn. časové průběhy řídicích veličin. Pro cesty získané plánováním v harmonickém potenciálu není vhodné trajektorii generovat přístupem *Turn-Move*, neboť se cesta skládá z velkého množství krátkých lineárních úseku a robot by se neustále rozjížděl a zastavoval. Vzhledem k relativně malým změnám směru mezi po sobě jdoucími segmenty cesty lze předpokládat, že robot

[2]Použit také při reálném experimentu.

parametr	$v_{\max}$	$a_{\max}$	L	r	$\omega_{\max}$	$\epsilon_{\max}$
hodnota	0,466	1,000	0,3775	0,047	$\frac{2}{L}v_{\max}$	$\frac{2}{L}a_{\max}$

Tabulka 3.4: Parametry diferenciálního modelu robotu pro výpočet rychlostního profilu. $v_{\max}$ je nejvyšší dopředná rychlost, $a_{\max}$ dopředné zrychlení, L vzdálenost hnaných koleček, r poloměr kola, $\omega_{\max}$ úhlová rychlost a $\epsilon_{\max}$ úhlové zrychlení.

dokáže cestu projet bez zastavování a dopřednou rychlost snižuje pouze omezení na dopředné zrychlení a úhlovou rychlost. Pro každý bod cesty, kde se mění směr, se určí křivost cesty jako

$$K(s) = \frac{x(s)'y(s)'' - y(s)'x(s)''}{\sqrt[3]{(x(s)'^2 + y(s)'^2)^2}},$$

kde x' a x'', respektive y' a y'' jsou první a druhé derivace křivky dle dané proměnné – v tomto případě jsou nahrazeny prvními a druhými diferencemi. Maximální dopředná rychlost v daném bodě při splnění omezení na maximální úhlovou rychlost se určí podle vztahu

$$v = \frac{\omega_{\max}}{K}.$$

Rychlosti vyšší než maximální dopředná rychlost se zmenší na tuto hodnotu. Výsledná posloupnost rychlostí se ještě upraví, aby mezi jednotlivými rychlostmi bylo splněno omezení na maximální zrychlení. Omezení na úhlové zrychlení je relaxováno.

Určí-li se poměr časů nutných k projetí nalezených cest pro daný model srovnávanými metodami opět jako

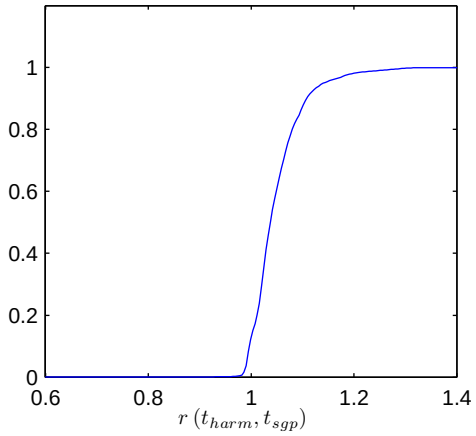
$$r(t_{harm}, t_{sgp}) = \frac{t_{harm}}{t_{sgp}},$$

kde t_{harm} je čas nutný k projetí cesty získané v harmonickém potenciálu a t_{sgp} je čas nutný k projetí cesty získané ohodnocením Dijkstrovým algoritmem, bude z 7800 náhodně vybraných cest do pěti náhodně vybraných cílů na mapě *BugTrap* průměrný poměr 1,052 a standardní odchylka 0,1168. To znamená, že cesty získané využitím potenciálu jsou časově průměrně přibližně o 5 % delší. Na obrázku 3.5 je znázorněna odpovídající distribuční funkce.

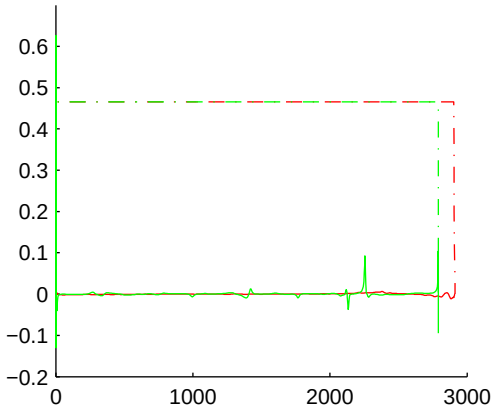
Na obrázku 3.6 je vykresleno srovnání typických rychlostních profilů pro dané dvě ohodnocení. Je vidět, že cesty získané plánováním harmonickým potenciálem mají menší úhlové rychlosti – to znamená, že méně často výrazně mění směr a dá se říci, že pro robot s nezanedbatelnou hmotností budou lépe realizovatelné. S méně častou a více plynulou změnou směru také teoreticky vzniká úspora energie, neboť robot nemusí výrazně zpomalovat, aby projel ostřejší zatáčku.

3.3 Srovnání cesty po segmentované mapě

Na obrázku 3.7(b) je zobrazeno několik cest nalezených přístupem rozdělení mapy na menší části popsaným v podkapitole 2.5.1. Je vidět, že rozčleněním a nezávislým plánová-

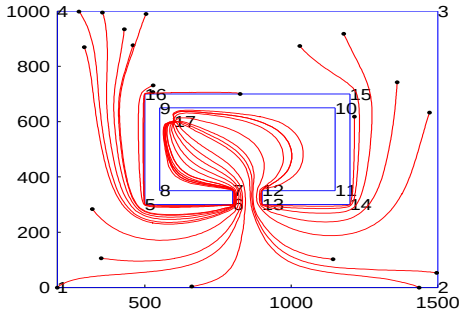


Obrázek 3.5: Distribuční funkce poměrů časů nutných k projetí cest nalezených srovnávanými metodami.

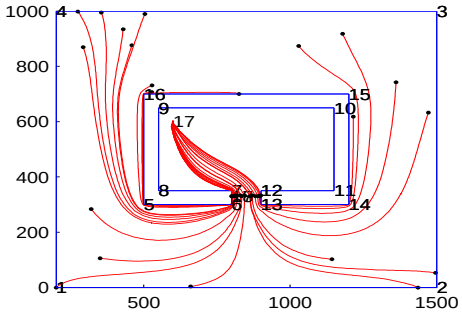


Obrázek 3.6: Srovnání rychlostních profilů. Čerchované průběhy jsou dopředné rychlosti, plné čáry jsou úhlové rychlosti. Červený průběh je pro cestu nalezenou harmonickým potenciálem, zelený průběh je pro cestu podle Dijkstrova ohodnocení.

ním nad těmito částmi lze dosáhnout kratších cest. Další věcí, která stojí za povšimnutí, je relativní hladkost cest v bodech, kde jsou jednotlivé části navázány. Vyjádřeno číselně jsou cesty na obrázku 3.7(a) průměrně o 11 % delší než na obrázku 3.7(b) – vzorek 2009 náhodně vybraných cest do stejného cíle znázorněného na obrázku 3.7(a), respektive 3.7(b), průměrný poměr délek je 1,1101, standardní odchylka 0,1531.



(a)



(b)

Obrázek 3.7: (a) Cesty z několika vybraných uzlů generované nad nerozdělenou mapou *BugTrap*. (b) Cesty ze stejných uzlů jako na obrázku (a), mapa je rozdělena na dvě části – vnitřní a vnější oblast okolo centrální překážky, na obrázku je rozdělení znázorněno černými body tvořícími úsečku v nejužším místě mapy.

3.4 Řešení inspekční úlohy

Využije-li se cest získaných plánováním nad grafem ohodnoceným harmonickým potenciálem k sestavení matice vzdálenosti mezi všemi měřicími místy nalezenými jako řešení *AGP*, lze přikročit k hledání optimálního pořadí navštívení míst. A to jednak jako řešení úlohy *mTSP* heuristickým přístupem *GENIUS*, tak alternativní dvoukrokovou metodou rozdělení na shluky metodou k-means a následného řešení jednoduchého *TSP*.

Dá se předpokládat, že každá metoda povede na jinak strukturované cesty po mapě. Zatímco cesty získané v prvním případě budou přibližně stejně dlouhé a navzájem velmi propletené, v druhém případě se dá předpokládat spíše opak – cesty budou různě dlouhé a jejich větší části budou ležet v oddělených lokalitách. V obou případech je zajímavé sledovat délku nejkratší a nejdelší cesty a celkový součet délek všech cest v závislosti na počtu kooperujících robotů.

3.4.1 Řešení jako úloha *mTSP*

V případě řešení použitím heuristiky *GENIUS* je jako kritérium optimality zvoleno kritérium *MinMax*. Jak je vidět v tabulce 3.5 nebo 3.6 pro mapu *TestArea*, tak v tabulce 3.7 pro mapu *jari-huge*, rozdíly mezi nejdelší a nejkratší inspekční cestou jednotlivých robotů nejsou příliš veliké. Dále je z tabulek vidět, že s roustoucím počtem robotů výrazně roste celková délka tras. Naproti tomu, čím je nejdelší cesta kratší, tím je celá inspekce dokončena v kratším čase.

m	$l_{\min}$		$l_{\max}$		l_{sum}	
	$\bar{x}$	s_x	$\bar{x}$	s_x	$\bar{x}$	s_x
1	30 280,43	173,19	30 280,43	173,19	30 280,43	173,19
2	20 024,94	130,95	20 207,09	208,92	40 232,03	195,60
3	17 518,07	459,48	18 046,05	447,20	53 317,70	1 317,34
4	15 416,10	333,37	15 991,78	447,73	62 866,07	1 433,44
5	15 149,88	445,00	16 138,44	704,53	77 891,49	2 202,30
6	14 513,77	422,02	15 501,48	178,65	90 302,55	1 225,98
7	13 783,41	672,63	15 552,91	484,24	103 263,19	2 065,99
8	13 534,50	706,47	15 684,55	589,28	117 108,74	3 013,71
9	13 042,62	1 232,42	15 451,04	627,18	128 717,99	3 152,83
10	12 776,95	1 254,92	15 273,13	156,17	139 682,16	1 544,42

Tabulka 3.5: Statistiky pro řešení inspekční úlohy *mTSP* na mapě *TestArea*, průměry a standardní odchylky přes deset řešení pro počátek inspekce v místě 1. m je počet robotů, $\bar{x}$ je střední hodnota a s_x výběrová standardní odchylka příslušné veličiny. $l_{\min}$ je délka nejkratší trasy, $l_{\max}$ značí délku nejdelší trasy a l_{sum} je součet délek tras jednotlivých robotů.

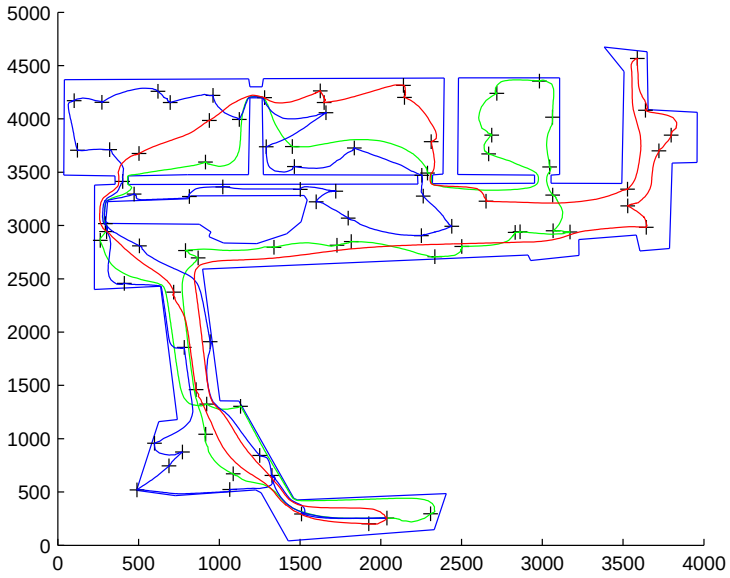
m	$l_{\min}$		$l_{\max}$		l_{sum}	
	$\bar{x}$	s_x	$\bar{x}$	s_x	$\bar{x}$	s_x
1	30 138,02	166,39	30 138,02	166,39	30 138,02	166,39
2	17 309,73	795,30	17 470,51	826,89	34 780,24	1 620,35
3	13 353,44	1 249,28	14 001,46	977,74	40 970,99	3 233,18
4	11 094,69	1 018,30	11 882,50	767,11	46 249,60	3 017,96
5	9 822,10	870,41	11 004,43	592,17	52 411,22	3 073,42
6	9 536,74	1 232,23	10 684,09	1018,35	61 253,21	5 753,67
7	8 404,79	1 355,54	10 674,18	441,08	68 772,17	2 985,61
8	6 745,07	1 860,57	10 223,26	337,94	74 156,56	2 899,73
9	7 863,28	909,88	10 078,34	251,38	83 075,55	2 353,59
10	6 632,15	1 788,84	10 058,10	137,77	91 377,27	4 023,46

Tabulka 3.6: Statistiky pro řešení inspekční úlohy *mTSP* na mapě *TestArea*, průměry a standardní odchylky přes deset řešení pro počátek inspekce v místě 17. m je počet robotů, $\bar{x}$ je střední hodnota a s_x výběrová standardní odchylka příslušné veličiny. $l_{\min}$ je délka nejkratší trasy, $l_{\max}$ značí délku nejdelší trasy a l_{sum} je součet délek tras jednotlivých robotů.

m	$l_{\min}$		$l_{\max}$		l_{sum}	
	$\bar{x}$	s_x	$\bar{x}$	s_x	$\bar{x}$	s_x
1	18 579,27	746,08	18 579,27	746,08	18 579,27	746,08
2	10 395,38	361,32	10 857,56	477,10	21 252,94	753,20
3	7 819,56	152,11	8 518,67	338,46	24 430,26	563,08
4	6 868,51	204,98	7 617,60	264,68	28 903,28	724,67
5	6 270,99	508,02	7 134,98	312,50	33 717,63	1 293,35
6	5 633,78	804,42	6 847,61	322,58	38 034,55	1 605,67
7	5 340,87	744,76	6 600,74	298,43	42 460,35	2 007,71
8	4 806,92	848,20	6 376,67	278,65	46 048,28	2 032,99
9	4 548,70	826,27	6 168,60	165,47	50 419,84	1 492,79
10	4 134,37	1 081,40	6 097,63	306,55	54 325,27	2 000,47
11	4 046,20	907,38	5 981,88	156,01	59 157,16	1 717,58

Tabulka 3.7: Statistiky pro řešení inspekční úlohy *mTSP* na mapě *jari-huge*, průměry a standardní odchylky pro počátek inspekce v místě 1, průměrováno přes 25 pokusů. m je počet robotů, $\bar{x}$ je střední hodnota a s_x výběrová standardní odchylka příslušné veličiny. $l_{\min}$ je délka nejkratší trasy, $l_{\max}$ značí délku nejdelší trasy a l_{sum} je součet délek tras jednotlivých robotů.

Na obrázku 3.8 je zobrazeno řešení inspekční úlohy jako *mTSP* pro tři roboty. Všechny tři cesty jsou téměř stejně dlouhé, což je důsledek *MinMax* kritéria. Narozdíl od řešení shlukováním, všechny cesty operují téměř nad celou oblastí, což zvyšuje riziko kolize robotů. Je však třeba říci, že ačkoli se kříží cesty jednotlivých robotů, nemusí se již křížit jejich trajektorie.



Obrázek 3.8: Řešení inspekce třech robotů získané použitím metody *mTSP*. Délky cest jsou 17 837,8, 17 826,1 a 17 803,7.

3.4.2 Řešení jako úloha *K-means* + *TSP*

Řeší-li se rozdělení míst mezi jednotlivé roboty shlukovacím algoritmem *K-means*, je zajímavé, jak se mění statistiky nalezených řešení v závislosti na počtu robotů (shluků). Pro $K = 1$, kde K je počet robotů, bude řešení stejné jako řešení získané vyřešením úlohy jednoho obchodního cestujícího. Vzhledem k tomu, že algoritmus *K-means* neprodukuje shluky se stejnou mohutností, dají se předpokládat s rostoucím počtem shluků (robotů) výrazné rozdíly mezi nejdelší a nejkratší cestou. Na obrázku 3.9(a), respektive 3.9(b) jsou znázorněny dvě možná řešení pro tři roboty lišící se pouze zadaným počátečním bodem. Jak je z obrázku patrné, volba tohoto bodu má na výsledek velký vliv.

V tabulkách 3.9 a 3.8 jsou uvedeny délky nejdelších a nejkratších inspekčních cest a jejich celkový součet. Srovnají-li se tyto hodnoty s řešením nalezeným předchozím přístupem, je vidět, že celkové délky cest jsou v tomto případě kratší, což znamená, že postup shlukováním a *TSP* povede na energeticky levnější řešení. Jelikož je ale velký rozdíl mezi

m	$l_{\min}$	$l_{\max}$	l_{sum}
1	30 123,95	30 123,95	30 123,95
2	14 007,30	19 791,09	33 798,40
3	11 343,15	12 645,09	35 571,54
4	7 937,63	11 251,62	37 722,65
5	8 588,54	11 343,15	47 135,73
6	4 732,72	10 762,45	49 122,93
7	2 954,42	9 706,64	50 647,25
8	3 916,25	9 706,64	55 967,76
9	3 916,25	9 651,11	66 272,19
10	3 730,82	9 651,11	75 219,89
11	2 536,59	9 651,11	77 145,41
12	2 567,49	9 651,11	82 771,10
13	2 567,49	9 651,11	87 640,23

Tabulka 3.8: Statistiky pro řešení inspekční úlohy shlukováním a *TSP*, jako počátek inspekce byl vybráno místo 17 – obrázek 3.9(a). m je počet robotů, $l_{\min}$ je délka nejkratší cesty, $l_{\max}$ délka nejdelší cesty, l_{sum} součet všech délek.

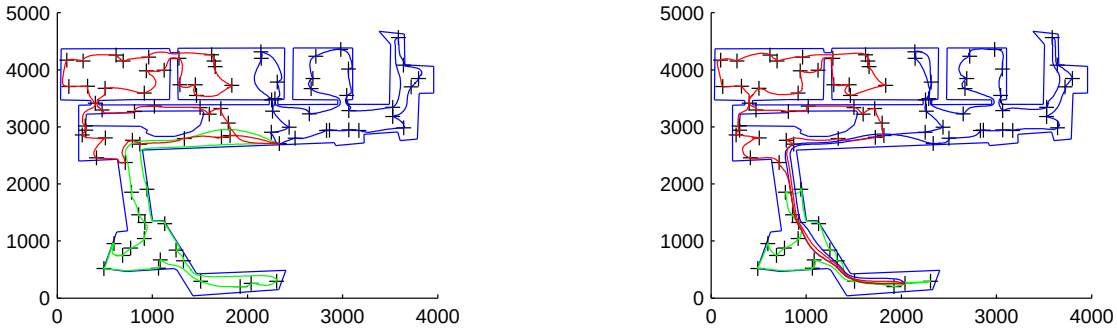
délkou nejkratší a nejdelší cesty – zejména v případě nevhodné konfigurace (tabulka 3.9) – nebude inspekční čas nejkratší možný, neboť pro čas inspekce platí

$$t_{insp} = \max\{t_k | k = 1, .., K\}.$$

Řešení začne být zajímavým pokud se zadání úlohy mírně pozmění a cílem nebude projet danou mapu jednorázově, ale lokalitu projíždět periodicky – hlídkovat. V takovém případě, pomine-li se samotný začátek inspekce – nájezd do počátečních bodů –, budou roboty začínat každý z jiného místa, jejich cesty se nebudou protínat a budou poměrně krátké, viz tabulka 3.10. Tato data již nejsou statisticky zpracována, neboť použitá implementace *K-means* je deterministická, stejně jako implementace hledání řešení *TSP* [2]. Z dané tabulky plyne, že celková délka všech cest se zvyšujícím se počtem robotů příliš neroste, což v důsledku znamená kratší periodu s jakou je celá mapa opakovaně projeta. Se zvyšujícím se počtem robotů také přibývá degenerovaných cest, kdy robot celou dobu stojí na jednom místě. V limitním případě, když bude robotů stejně jako měřicích míst, budou všechny roboty stát na místě a v praxi je pak možné v některých aplikacích mobilní roboty nahradit statickými senzory – což je mimochodem řešením úlohy *AGP*.

m	$l_{\min}$	$l_{\max}$	l_{sum}
1	30 123,95	30 123,95	30 123,95
2	18 672,41	22 142,73	40 815,14
3	7 055,52	20 239,11	44 485,38
4	7 027,79	17 746,85	53 699,44
5	1 698,73	20 121,14	56 362,32
6	1 698,73	20 175,86	67 942,93
7	1 698,73	17 062,24	77 862,61
8	1 698,73	16 695,09	87 199,03
9	585,96	16 695,09	86 898,34
10	272,12	15 802,57	93 326,14
11	272,12	15 754,36	101 752,02
12	272,12	15 754,36	107 460,33
13	272,12	15 754,36	116 226,53

Tabulka 3.9: Statistiky pro řešení inspekční úlohy shlukováním a *TSP*, jako počátek inspekce byl vybráno místo 1 – obrázek 3.9(b). m je počet robotů, $l_{\min}$ je délka nejkratší cesty, $l_{\max}$ délka nejdelší cesty, l_{sum} součet všech délek.

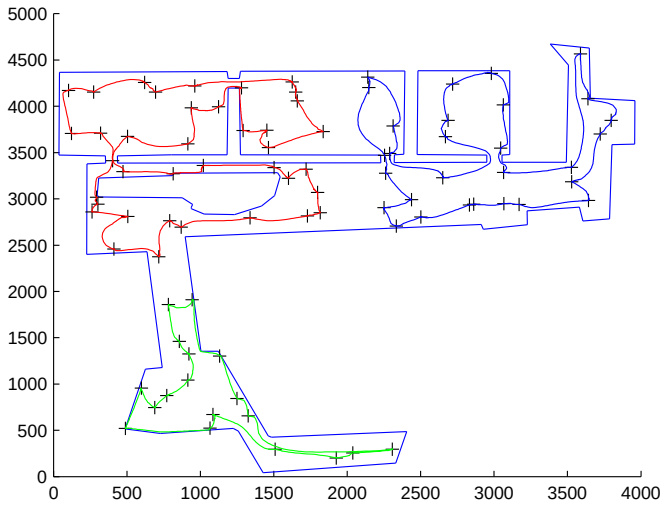


(a) (b)

Obrázek 3.9: Inspekce třech robotů, všechny cesty začínají ve stejném bodě. (a) Počátek v místě 17. Délky cest: 11 257,8, 11 398,4 a 12 848,8. (b) Počátek v místě 1, délky cest: 7 032,4, 20 299,7 a 17 274,8.

m	$l_{\min}$	$l_{\max}$	l_{sum}
1	30 037,20	30 037,20	30 037,20
2	14 112,55	18 552,27	32 664,82
3	7 094,66	11 675,83	30 168,88
4	7 094,66	8 050,01	30 919,91
5	1 718,04	11 398,39	31 860,39
6	1 718,04	9 727,11	32 144,42
7	1 718,04	7 061,04	32 368,95
8	1 718,04	6 158,52	33 718,18
9	0	6 088,12	33 062,98
10	0	7 747,26	36 631,16
11	0	7 000,31	36 785,63
12	0	7 000,31	37 043,76
15	0	6 217,10	35 216,78
20	0	6 018,42	34 151,19
25	0	5 593,94	34 443,05

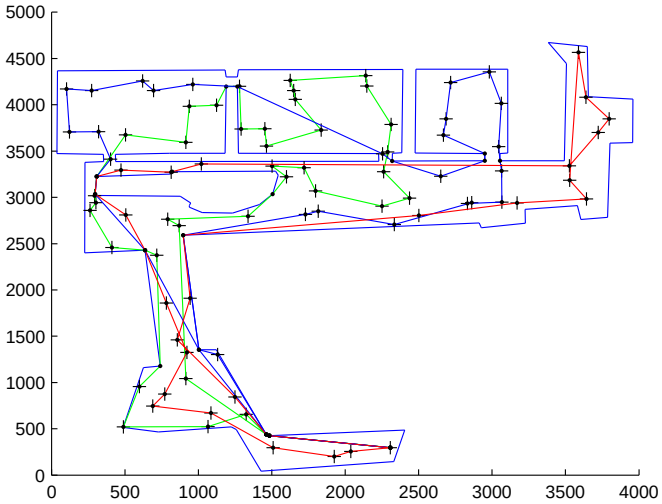
Tabulka 3.10: Statistiky pro řešení inspekční úlohy shlukováním a *TSP*, mapa *TestArea*. m je počet robotů, $l_{\min}$ je délka nejkratší cesty, $l_{\max}$ délka nejdelší cesty, l_{sum} součet všech délek.



Obrázek 3.10: Periodická inspekce s využitím třech robotů.

3.4.3 Srovnání délky a doby inspekce

Nejkratší možné inspekční cesty lze získat plánováním nad grafem viditelnosti, typické řešení je zobrazeno na obrázku 3.11. Srovnání celkové délky cest nalezených harmonickým potenciálem s celkovými délkami cest získaných využitím grafu viditelnosti prozradí, o kolik jsou cesty potenciálem delší než nejkratší možné. Pro jednotlivá získaná pořadí navštívení míst jsou vygenerovány cesty oběma přístupy a v tabulce 3.11 jsou uvedeny vypočtené průměrné hodnoty pro jeden až deset robotů. Z tabulky je patrný přibližně 6% rozdíl v celkové délce ve prospěch plánů nad grafem viditelnosti.



Obrázek 3.11: Řešení inspekční úlohy pro tři roboty, cesty mezi jednotlivými místy jsou plánovány nad grafem viditelnosti.

Cesty nalezené nad grafem viditelnosti jsou charakteristické svými dlouhými lineárními úseky a často ostrými změnami směru, a proto je nejvhodnější je projíždět přístupem *Turn-Move*. Naproti tomu cesty harmonickým potenciálem jsou tvořeny krátkými lineárními úseky s malými změnami směru mezi sousedními úseky a z tohoto důvodu se použití přístupu *Turn-Move* nehodí a je vhodnější cestu projet přístupem popsaným v podkapitole 3.2.2. V tabulce 3.11 jsou uvedeny celkové délky všech inspekčních cest a časy potřebné k dokončení inspekce pro případ plánování nad grafem viditelnosti a plánování nad harmonickým potenciálem. Časem inspekce je nejdelší z časů nutných k projetí jednotlivých cest. Srovnávaná data jsou pro identická řešení *mTSP* podproblému a liší se pouze způsobem naplánování a projetí cesty. Pro každý počet robotů jsou výsledky zprůměrovány pro 50 nezávisle nalezených řešení *mTSP*. Hodnoty uvedené v tabulce naznačují, že i když jsou celkově ujeté vzdálenosti pro cesty v harmonickém potenciálu delší, realizace inspekce je dokončena v mírně kratším čase než realizace plánu podle grafu viditelnosti projetém přístupem *Turn-Move*; rozdíl činí přibližně 3 %.

m	l_h		l_v		t_h		t_v	
	$\bar{x}$	s_x	$\bar{x}$	s_x	$\bar{x}$	s_x	$\bar{x}$	s_x
1	30 285,74	193,70	28 383,11	151,92	717,37	3,93	755,49	4,57
2	40 813,68	853,56	38 449,54	802,89	477,56	9,66	498,84	11,00
3	52 638,01	906,37	49 540,62	809,28	408,41	8,70	425,30	11,21
4	64 022,09	2 118,94	60 348,12	1 912,76	371,70	13,56	386,09	15,29
5	76 268,71	1 988,62	71 756,66	1 738,02	354,80	10,59	367,82	11,92
6	89 187,52	1 437,35	83 556,18	1 418,67	345,95	6,31	354,85	8,22
7	102 432,95	1 671,08	95 687,51	1 855,86	343,02	6,21	352,91	7,30
8	115 717,47	2 332,17	107 903,26	2 105,15	341,10	5,37	348,66	8,30
9	127 799,81	4 480,43	118 569,85	4 049,08	338,30	5,11	346,40	7,03
10	140 022,45	2 782,44	128 935,36	2 583,55	338,18	6,90	345,37	8,11
11	148 850,16	6 221,48	137 371,99	5 571,51	336,46	5,02	343,49	6,23

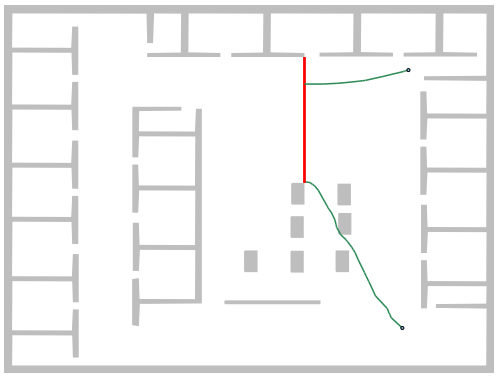
Tabulka 3.11: Srovnání celkové délky řešení inspekční úlohy *mTSP* a časů nutných na realizaci s cestami podle harmonického potenciálu a celkové délky řešení inspekční úlohy *mTSP* s cestami podle grafu viditelnosti na mapě *TestArea*, průměry a standardní odchylky přes padesát řešení. m je počet robotů, $\bar{x}$ je střední hodnota a s_x výběrová standardní odchylka příslušné veličiny. Veličina l_h, resp. t_h značí celkovou délku tras inspekce, resp. čas nutný k realizaci inspekce s cestami dle harmonického potenciálu, l_v, resp. t_v jsou tytéž veličiny pro inspekci s cestami dle grafu viditelnosti.

3.5 Řešení inspekce využitím zobecněných míst

V oddíle 2.5 je uvedena výhoda řešení nalezení cesty k cíli harmonickým potenciálovým polem i pro obecnější útvary než pouhý bod. V úloze inspekce tak lze nahradit jednotlivé měřicí místo množinou míst, při jejichž navštívení je prohledána ekvivalentní část prostředí. Nalezený harmonický potenciál pro takové zobecněné měřící místo poskytuje heuristiku pro nalezení navigační cesty z libovolného místa prostředí k měřicímu místu, přičemž cíl takové cesty může ležet libovolně na zobecněném měřicím místě.

Vlastní řešení nalezení vhodného pořadí navštívení zobecněných měřicích míst lze řešit podobně jako pro bodová místa, řešením úlohy obchodních cestujících. V případě kombinatorické formulace úlohy TSP je však nutné zohlednit libovolný cílový bod z jednoho města do jiného, který je navíc závislý na pořadí navštívení. Situaci ilustruje obrázek 3.12. Z tohoto důvodu je vhodné využít k nalezení řešení úlohy obchodního cestujícího samoorganizující se neuronovou síť (SOM), která již byla pro řešení inspekční úlohy úspěšně použita autory [16].

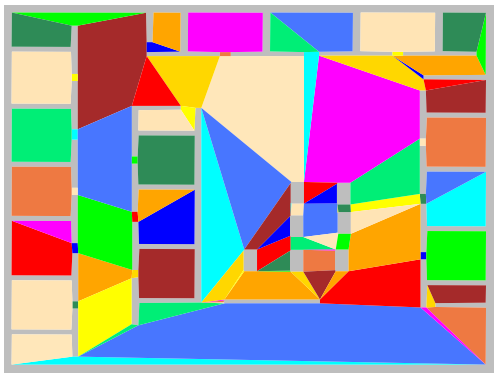
V následujících odstavcích je stručně popsán jednoduchý algoritmus nalezení zobecněných míst a výsledky řešení úlohy jediného a více obchodních cestujících algoritmem SOM.



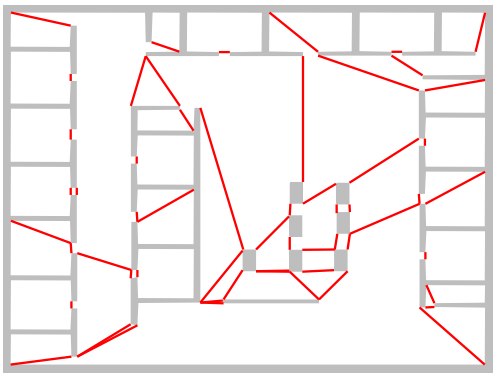
Obrázek 3.12: Příklad různých cílových bodů na zobecněném měřicím místě - úsečce.

Nalezení zobecněných měřicích míst

Nejjednodušším zobecněním bodových měřicích míst je rozšíření míst na úsečky. Podmínkou inspekční úlohy je prohledání celého prostředí, proto i navštívení zobecněného měřicího místa v kterémkoliv jeho bodě, musí tuto podmínku zajistit. Algorithms [20] využívá k rozmístění bodových strážců rozklad volného prostoru prostředí na konvexní polygony. Měřicí místa jako úsečky lze jednoduše získat jako hrany konvexních oblastí, které leží ve volném prostoru. Navštívením hrany v kterémkoliv jejím bodě zajistí prohledání obou konvexních oblastí příslušející úsečce. Na obrázku 3.13 je uvedeno jedno takové rozdělení (a) a nalezená měřicí místa jako úsečky (b).



(a)



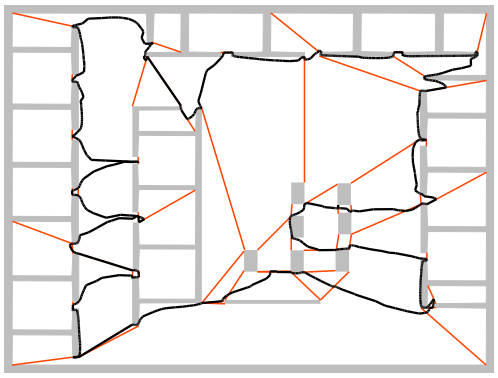
(b)

Obrázek 3.13: Nalezení zobecněných míst: (a) rozdělení prostředí na konvexní oblasti, (b) zobecněná místa.

Řešení úlohy jediného obchodního cestujícího

Základní algoritmus SOM je založen na adaptaci neuronové sítě, při které se jednotlivé neurony pohybují směrem k příslušnému městu, neboť váhy neuronové sítě přímo odpo-

vídají souřadnicím neuronu v prostředí. V případě využití harmonického potenciálu je při pohybu neuronu blíže k městu místo přímé nejkratší cesty využita cesta z harmonického potenciálu. Tento přístup má kromě možnosti adaptace neuronové sítě k libovolně složitému tvaru měřicího místa také tu výhodu, že při pohybu neuronu k městu (bodovému) není nutné uvažovat překážky, neboť ty jsou již zahrnuty v řešení harmonického potenciálového pole. Obrázek 3.14 reprezentuje příklad řešení inspekční úlohy pro jediný robot s měřicími místy jako úsečkami.



Obrázek 3.14: Řešení inspekční úlohy pro jediný robot algoritmem SOM a zobecněnými měřicími místy.

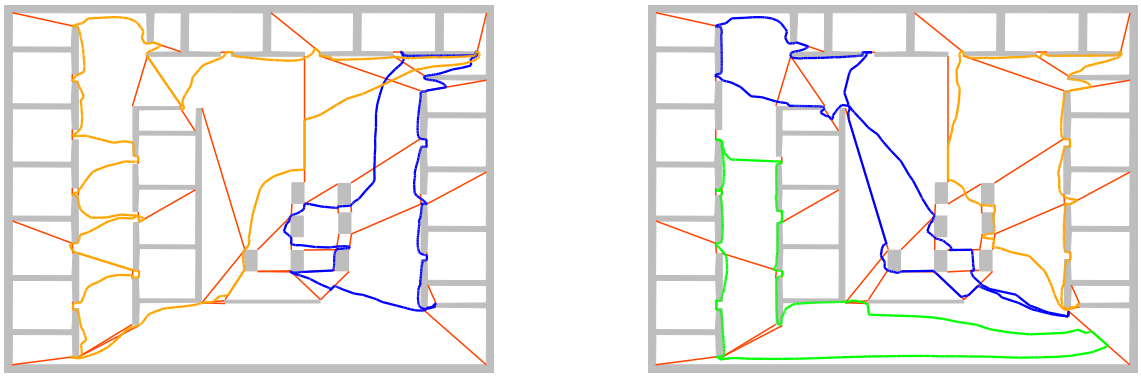
Řešení úlohy více obchodních cestujících

V úloze více obchodních cestujících s MinMax kritériem zohledňuje adaptace neuronové sítě délky jednotlivých cest [30]. Délka cesty je vypočtena jako aktuální délka řetízku[3] neuronů. Při takovém výpočtu je nutné určit vzdálenost dvou sousedních neuronů. V případě použití harmonického potenciálu to znamená nový výpočet pole, pro které je cílem jeden z neuronů. Tento výpočet je nutné provést pro každý neuron a po každé změně polohy byť jediného neuronu, což je velmi výpočetně náročné. Autoři [22] proto doporučují uvažovat délku řetízku jako délku cesty přes města, kterou řetízek v aktuálním kroku adaptace neuronové sítě reprezentuje. K tomuto výpočtu postačí již předem vypočtené hodnoty potenciálu pro každé zobecněné místo. Příklad řešení úlohy inspekce více roboty s uvažováním zobecněných míst je uveden na obrázcích 3.15(a) a 3.15(b).

3.6 Reálný experiment

Řešený problém inspekční úlohy vyžaduje známou mapu pracovního prostředí robotu. Z tohoto důvodu je prvním krokem pro ověření plánování trajektorie metodou harmonických potenciálů vytvoření polygonální mapy. Na získané mapě je následně nalezena cesta, pro kterou je určen rychlostní profil pohybu robotu. Pro reálný experiment byla využita

[3]Neuronů propojených do cyklu.

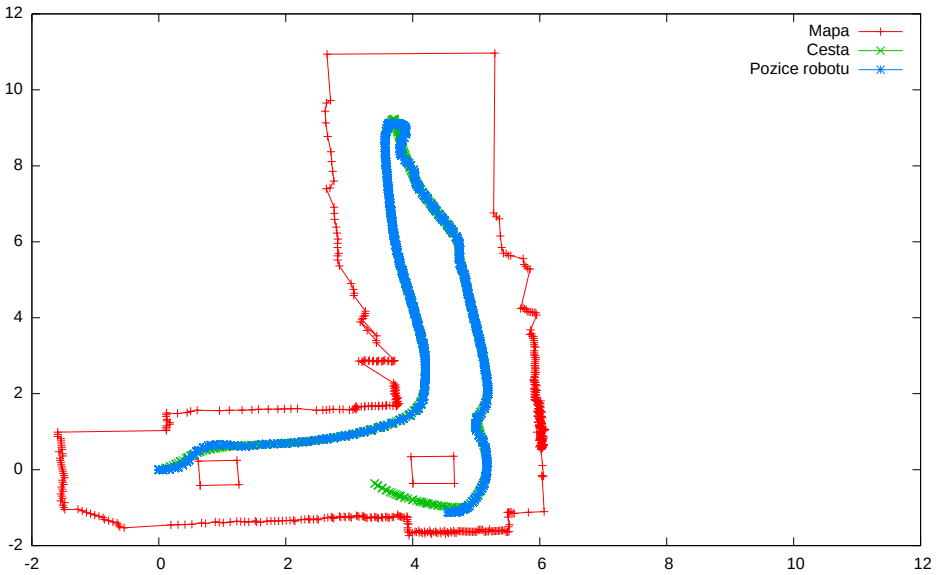


(a) (b)

Obrázek 3.15: Řešení inspekční úlohy pro tři roboty, algoritmus SOM a zobecněná měřicí místa.

mobilní robotická platforma G^{2}BOT. S ohledem na charakter práce byl reálný experiment zaměřen především na ověření schopnosti robotu projet vypočtenou trajektorii.

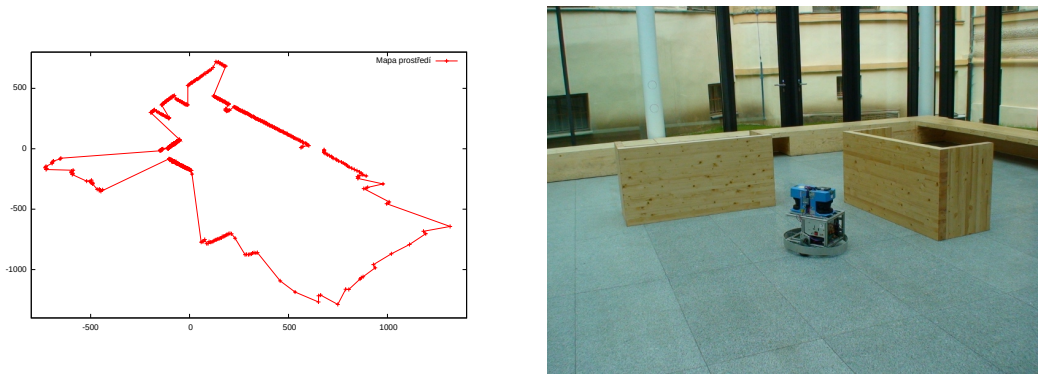
Obrázek 3.16 ilustruje polygonální mapu, naplánovanou a skutečně projetou trajektorii robotu. Z obrázku je patrné, že výpočtu harmonického potenciálu nevadí velmi složitá hranice oblasti složená z velkého množství krátkých lineárních úseků, což oproti grafu viditelnosti zvyšuje robustnost a použitelnost algoritmu v reálných situacích, kdy je mapa získána předchozí explorací neznámého prostředí a není následně nijak výrazně upravována. Z téhož obrázku je vidět, že robot je schopen věrně kopírovat požadovanou trajektorii.



Obrázek 3.16: Polygonální mapa prostředí naplánovaná a reálná pozice robotu.

Jiná reálná scéna je zobrazena na obrázku 3.17 spolu s odpovídající polygonální repre-

zentací okolního prostředí robotu.



Obrázek 3.17: Polygonální mapa prostředí (a) a reálná scéna.

Kapitola 4

Závěr

V této práci byla řešena multirobotická inspekční úloha statického prostředí skupinou identických robotů. Řešení bylo hledáno dekompozicí na nezávislé podproblémy – hledání množiny měřicích míst, hledání optimálního pořadí navštívení míst a plánování pohybu. Ke hledání množiny měřících míst byl použit *RDS* algoritmus a přístup hledající měřicí místa postupným rozkladem volného prostoru na konvexní polygony. Hledání rozdělení a pořadí navštívení míst jednotlivými roboty je formulováno jako řešení problému jednoho či více obchodních cestujících. K řešení problému obchodních cestujících bylo využito existujících algoritmů – heuristiky GENIUS a exaktního řešení aplikací Concorde a samoorganizující se neuronové sítě.

Tato práce se zaměřovala na možnosti využití harmonických funkcí při řešení multirobotické inspekce, zejména jejich využití při plánování pohybu mezi jednotlivými měřicími místy. Byly diskutovány vlastnosti harmonických funkcí a bylo navrženo několik způsobů, jak na triangulovaném volném prostoru, jehož uzly jsou ohodnoceny harmonickým potenciálem generovat relativně hladké cesty, což je jedním z hlavních přínosů harmonického potenciálu. Byly popsány jak klasické gradientní metody postupující po vrcholech triangulace, tak metody aproximující gradient z nejbližších bodů okolí a postupujících po tomto gradientu po celém volném prostoru (nikoliv jen pro vrcholech triangulace).

Dále byly identifikovány další specifické přínosy ohodnocování uzlů triangulace harmonickým potenciálem. Jedná se o možnost specifikovat na mapě více nezávislých cílů nebo možnost definovat zobecněná místa – nejen bodová, ale plošná s libovolným tvarem hranice. Na základě těchto zobecněných míst byla navržena metoda, jak dělit rozsáhlou mapu na oblasti, nad kterými se dá plánovat nezávisle, což může přinést jak kvalitnější cesty, tak úsporu výpočetního času.

Navržené postupy a teoretické předpoklady byly ověřeny a vyhodnoceny řadou experimentů, byl sledován vliv hustoty triangulace na čas nutný k výpočtu potenciálu, byla srovnána kvalita cest získaných jednotlivými navrženými postupy, byla srovnána celková řešení inspekční úlohy. Řešení, jehož pořadí navštívení měst bylo získáno řešením problému více obchodních cestujících, bylo srovnáno s řešením nalezeným explicitním rozdělením míst mezi roboty shlukovacím algoritmem *K-means* a následným exaktním vyřešením problému jednoho obchodního cestujícího nezávisle nad každým shlukem. Byla porovnána celková

délka cest a čas nutný k dokončení inspekce, pokud je cesta pro dané rozdělení a pořadí míst generována na základě grafu viditelnosti a použitím harmonického potenciálu. Byl proveden úspěšný pokus využít harmonického potenciálu a zobecněných míst k hledání inspekční cesty. Reálný experiment prokázal, že cestu nalezenou harmonickým potenciálem lze bez obtíží projet.

Ukázalo se, že cesty podle harmonického potenciálu jsou zhruba o 10-15 % delší než cesty podle Dijkstrova ohodnocení grafu – nejkratších cest v grafu. Analýza rychlostního profilu dále ukázala, že cesty podle Dijkstrova ohodnocení lze projet v kratším čase, naopak cesty podle harmonického potenciálu by mohly vést na energeticky úspornější trajektorii. Při srovnání přístupu *mTSP* a shlukování a následného *TSP* vyšlo najevo, že druhá zmíněná metoda je velmi citlivá na volbu výchozího bodu inspekce, ale je vhodná pro plánování periodické inspekce – vede na výrazně kratší celkovou délku tras a je tedy energeticky úspornější. Srovnání celkové délky cesty a času inspekce pro cesty podle grafu viditelnosti a podle harmonického potenciálu ukázalo, že plánování v grafu viditelnosti vede na euklidovsky kratší cesty – přibližně o 6 %, naopak plánování v harmonickém potenciálu přibližně o 3 % redukuje čas nutný k provedení inspekci. Tyto hodnoty platí pro použitý diferenciální model robotu.

V práci by bylo možno dále pokračovat v oblasti hledání řešení inspekce využitím zobecněných míst, neboť tato problematika nebyla důkladně rozebrána.

Literatura

[1] Ali, A. I.; Kennington, J. L.: The asymmetric m-traveling salesman problem: A duality based branch-and-bound algorithm. *Discrete Appl. Math.*, ročník 13, č. 2-3, 1986: s. 259–276, ISSN 0166-218X, doi:http://dx.doi.org/10.1016/0166-218X(86)90087-9.

[2] Applegate, D. L.; Bixby, R. E.; Chvatal, V.; aj.: Concorde TSP Solver, http://www.tsp.gatech.edu/concorde.html. 2005.
URL `http://www.tsp.gatech.edu/concorde.html`

[3] Applegate, D. L.; Bixby, R. E.; Chvatal, V.; aj.: *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, January 2007, ISBN 0691129932.

[4] Axler, S.; Bourdon, P.; Ramey, W.: *Harmonic Function Theory*. číslo 137 v Springer Graduate Texts in Mathematics, Springer, 2001, ISBN 978-0-387-95218-5, 259 s.

[5] Bajracharya, M.; Maimone, M. W.; Helmick, D.: Autonomy for Mars Rovers: Past, Present, and Future. *Computer*, ročník 41, č. 12, 2008: s. 44–50, ISSN 0018-9162, doi:http://doi.ieeecomputersociety.org/10.1109/MC.2008.515.

[6] Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, ročník 34, č. 3, June 2006: s. 209–219.

[7] Bentley, J. L.: Experiments on traveling salesman heuristics. V *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990, ISBN 0-89871-251-3, s. 91–99.

[8] Bjorling-Sachs, I.; Souvaine, D.: A tight bound for guarding general polygons with holes. Technická Zpráva LCSR-TR-165, Rutgers University, Dept. of Computer Science, 1991.

[9] Board, C. E.: *CGAL-3.2 User and Reference Manual*. 2006.
URL `http://www.cgal.org/Manual/3.2/doc_html/cgal_manual/index.html`

[10] Bochev, P.; Lehoucq, R. B.: On the finite element solution of the pure Neumann problem. *SIAM Rev*, ročník 47, 2005: s. 50–66.

[11] Boissonnat, J.-D.; Devillers, O.; Pion, S.; aj.: Triangulations in CGAL. *Comput. Geom. Theory Appl.*, ročník 22, 2002: s. 5–19.
URL `ftp://ftp-sop.inria.fr/geometrica/pion/publis/triangulations_in_cgal_cgta.pdf`

[12] Choset, H.; Lynch, K. M.; Hutchinson, S.; aj.: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[13] Chvatal, V.: A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, ročník 18, 1975: s. 39–41.

[14] Deshpande, A.; Kim, T.; Demaine, E. D.; aj.: A Pseudopolynomial Time (log)-Approximation Algorithm for Art Gallery Problems. V *WADS*, *Lecture Notes in Computer Science*, ročník 4619, editace F. K. H. A. Dehne; J.-R. Sack; N. Zeh, Springer, 2007, ISBN 978-3-540-73948-7, s. 163–174.

[15] Faigl, J.; Kulich, M.: Sensing Locations Positioning for Multi-robot Inspection Planning. *Distributed Intelligent Systems: Collective Intelligence and Its Applications, IEEE Workshop on*, ročník 0, 2006: s. 79–84, doi:http://doi.ieeecomputersociety.org/10.1109/DIS.2006.66.

[16] Faigl, J.; Kulich, M.; Přeučil, L.: Multiple traveling salesmen problem with hierarchy of cities in inspection task with limited visibility. V *5th Workshop on Self-Organizing Maps*, 2005, s. 91–98.

[17] Gendreau, M.; Hertz, A.; Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.*, ročník 40, č. 6, 1992: s. 1086–1094, ISSN 0030-364X.

[18] Gonzalez-Banos, H.; Latombe, J.-C.: Planning Robot Motions for Range-Image Acquisition and Automatic 3D Model Construction. 1998.

[19] Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; aj.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. V *IN IEEE INTERNATIONAL CONFERENCE ON METHODSAND MODELS IN AUTOMATION AND ROBOTICS, 2005*, MorganKaufmann, 1997, s. 566–580.

[20] Kazazakis, G. D.; Argyros, A. A.: Fast Positioning of Limited-Visibility Guards for the Inspection of 2D Workspaces. 2002.

[21] Krafft, M. F.; Harris, P.; Bougerel, S.: libkdtre++. 2008.
URL `http://libkdtree.alioth.debian.org/`

[22] Kulich, M.; Faigl, J.; Kléma, J.; aj.: Rescue Operation Planning by Soft Computing Techniques. V *IEEE 4th International Conference on Intelligent Systems Design and Application*, Piscataway: IEEE, 2004, s. 103–108.

[23] Latombe, J.-C.: *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991, ISBN 079239206X.

[24] LaValle, S. M.: *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, ISBN 0521862051.

[25] Mackay, D. J. C.: *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, June 2002, ISBN 0521642981.

[26] Míka, S.; Přikryl, P.: *Numerické metody řešení parciálních diferenciálních rovnic*. Západočeská univerzita Plzeň, 1995, ISBN 80-7082-204-X, 96 s.

[27] Nilsson, B. J.: *Guarding Art Galleries - Methods for Mobile Guards*. Dizertační práce, Lund University, 1995.

[28] Sibson, R.: Locally equiangular triangulations. *The Computer Journal*, ročník 21, č. 3, Srpen 1978: s. 243–245, ISSN 0010-4620.

[29] Siciliano, B.; Khatib, O. (editoři): *Springer Handbook of Robotics*. Springer, 2008, ISBN 978-3-540-23957-4.

[30] Somhom, S.; Modares, A.; Enkawa, T.: Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers and Operations Research*, ročník 26, č. 4, 1999: s. 395–407.

[31] Vitásek, E.: *Numerické metody*. Praha: SNTL, 1987, 516 s.

[32] Vitásek, E.: *Základy teorie numerických metod pro řešení diferenciálních rovnic*. Praha: Academia, 1994, ISBN 80-200-0281-2, 409 s.

[33] Vonásek, V.: *Úloha více obchodních cestujících s MinMax kritériem*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2006.

Obsah CD

Přiložené CD obsahuje zdrojové kódy pro řešení multirobotické inspekční úlohy s cestami plánovanými harmonickým potenciálem, text diplomové práce ve formátu PDF a zdrojové kódy celého textu pro systém LaTeX. V následující tabulce je popsána struktura CD.

Adresář	Popis
`src`	zdrojové kódy pro řešení inspekční úlohy
`doc`	zdrojové kódy textu diplomové práce
`thesis.pdf`	text diplomové práce

Tabulka 1: Adresářová struktura na CD.

Popis implementace

Výpočet harmonického potenciálu

Jádrem výpočtu harmonického potenciálu metodou konečných prvků je funkce

```
[u,vertices,fe,time] =
SolveProblem(constraints, seedpoint, params, outfile, draw).
```

Prvním vstupním argumentem je množina omezení popisujících volný prostor, druhým argumentem je bod specifikující, která v případě disjunktních oblastí má být triangulována, struktura `params` obsahuje parametry pro triangulaci a řešení – omezení na délku strany, minimální úhel triangulace, apod. Poslední dva argumenty jsou volitelné, prvním může být cesta k souboru, kam se výsledný potenciál zapíše, a druhý parametr určuje, zda se bude výsledek vykreslovat do grafu. Výstupem funkce je seznam hodnot potenciálu v uzlech triangulace, seznam vrcholů triangulace – tyto dva seznamy jsou vzájemně svázány pořadím – dále pak seznam všech konečných prvků a pro statistické účely seznam mezičasů trvání jednotlivých částí výpočtu.

Uvnitř funkce se volá *mex* funkce

```
[triang_indices, vertices, tttr,time] =
triangulate(constraints,seedpoint,params).
```

Vstupní argumenty jsou stejné jako u funkce `SolveProblem`, prvním výstupem je seznam konečných prvků – šestice čísel, první tři jsou indexy vrcholů, druhé tři 0 nebo 1 v závislosti na tom, zda daná strana leží na hranici. Druhým výstupem je seznam vrcholů triangulace, třetím výstupem je seznam konečných prvků s napočítanými koeficienty a posledním výstupem jsou opět časy jednotlivých mezivýpočtů. Následuje volání funkce pro sestavení globální matice a vyřešení soustavy lineárních rovnic. Funkce `SolveProblem` dokáže zjistit, zda je globální matice regulární a dle toho zvolit metodu řešení. Vzhledem k tomu, že detekce regularity zabírá velké množství času, je lepší tuto možnost nepoužívat a volit smíšený popis problému s vždy regulární maticí.

K pohodlnému načítání parametrizace volného prostoru slouží funkce

```
[constraints] =
ReadMap(filename,constrainttype, value,draw),
```

která zpracuje soubor `filename` a hranicím a překážkám nastaví typ a hodnotu okrajové podmínky. Možné typy okrajových podmínek jsou *'NONE'*, *'DIRICHLET'*, *'NEUMANN'*. Posledním parametrem je volba, zda má funkce situaci vykreslit do grafu.

K tvorbě jednotlivých lineárních částí hranic a omezení je určena funkce

```
[constraint] =
CreateConstraint(startp, endp, ctype, value),
```

zde jsou vstupní parametry poměrně transparentní – souřadnice počátku a konce, typ podmínky a její hodnota. Pro typ podmínky platí totéž, co pro výše popsanou funkci `ReadMap`. Příklad výpočtu harmonického potenciálu pro mapu *BugTrap* je

```
%%%%%%% bugtrap1.map %%%%%%%%%%%%%%%%%%%%%%%%%%%%
lines = ReadMap('bugtrap1.map','NEUMANN',1,0);
lines = [lines,CreateConstraint([600 600],[850 350],'DIRICHLET'
   ,-1)];
seedpoint = [850 400];
params.maxedgelen = 20;
params.smoothing = 0;
params.minedgeangle = 0.1;
params.includearea = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% using mex
[u,vertices,triangs] = SolveProblem(lines, seedpoint, params,'
   output.txt', 0);
%%% using mex
DrawAllRoutes('output.txt',vertices, lines, triangs);
```

Implementace hledání cesty

Funkce pro hledání cesty je realizována opět jako *mex* v *C++*. Jedná se o funkci

```
[path,length,time] =
findPath(fileName,coords,method),
```

vstupem funkce je název souboru s uloženou triangulací a jejím ohodnocením – vedlejší výstup funkce `SolveProblem` –, souřadnice výchozího bodu a metoda vyhledání cesty. Výstupem funkce je nalezená cesta, její eukleidovská délka a čas spotřebovaný pro její nalezení. Bližší popis se vypíše po volání funkce z MATLABu bez parametrů.

Funkci pro hledání cesty je možno použít i mimo MATLAB, k vyhledání cesty stačí vytvořit instanci třídy `PathQuery` a metodou `status findPath(Path* p, VPoint* start)` se dotazovat na cestu z vybrané počátenční polohy. Cesta se uloží do instance třídy `Path`. Možné návratové hodnoty jsou definovány výčtem `status`, který nabývá hodnot

- `OK` - cesta nalezena,

- `FILE_NOT_FOUND` - nepodařilo se otevřít soubor s uloženým potenciálem,
- `PATH_NOT_FOUND`, `TRIANG_NOT_FOUND`, `SUITABLE_TRIANG_NOT_FOUND` - cestu se nepodařilo nalézt.

Jednoduchý vzorový kód, který vytvoří instanci dotazovače cesty a nalezne a vypíše cestu z bodu (100,100) do cíle, je

```
#include "PathQuery.h"

PathQuery p("./soubor_s_potencialem");
double x=100,y=100;
VPoint start(x,y);
Path path;
status s = p.findPath(&path,&start);
path.printPath(0);
```