EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vybraná témata z mobilní robotiky

Učební texty k semináři

Autoři:

RNDr. Miroslav Kulich Ph.D. (ČVUT v Praze)
Dr.rer.nat. Martin Saska (ČVUT v Praze)

Datum:

17. 2. 2011

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

Tento dokument byl vytvořen na základě disertačních prací Miroslava Kulicha, Martina Sasky a Romana Mázla.

Obsah

Kapitola 1

Úvod

V tomto dokumentu jsou představeny dva z fundamentálních problémů dnešní mobilní robotiky. Kapitola 2 se věnuje problému určení pozice robotu (lokalizaci). Nejprve je problém lokalizace definován a popsány jeho základní varianty. Kapitola 2.1 je věnována podrobnějšímu představení problému kontinuální lokalizace, který spočívá v periodickém opravování pozice robotu na základě údajů z dálkoměrných senzorů. Rovněž budou představeny základní metody řešící tento problém, včetně algoritmu ICP (Iterative Closest Point) a jeho derivací. V kapitola 2.2 je popsán problém globální lokalizace a použití pravděpodobnostních metod. Bude odvozen Bayesův filtr jako obecná lokalizační metoda a popsány jeho konkrétní specializace: Kalmanův filtr a částicový filtr.
Kapitola 3 se věnuje problému plánování pohybu mobilního robotu a skupiny mobilních robotů. V této kapitole budou posluchači seznámeni se základními přístupy k plánování. Kromě nejjednodušších algoritmů, jako jsou potenciálové pole, graf viditelnosti, Voroného diagram či VFH (Vector Field Histogram), budou prezentovány i pokročilejší metody, mezi které lze řadit například RRT (Rapidly-exploring Random Tree), případně i RHC (Receding Horizon Control) dotýkající se ve své podstatě i oblasti prediktivního řízení mobilních robotů. Dále bude problematika plánování rozšířena o koordinaci multi-robotických systémů. Zde se budeme soustředit na problém stabilizace formací robotů a robotických rojů. Opět budou zmíněny základní architektury řízení formací i jejich modifikace. Kromě formací pozemních mobilních robotů se také krátce dotkneme tématu rojů bezpilotních letounů a satelitů. Na závěr se čtenář seznámí s praktickými ukázkami aktuálních multi-robotických projektů.

Kapitola 2

Metody lokalizace v mobilní robotice

Jedním z hlavních požadavků kladených na inteligentní mobilní roboty je schopnost poznávat prostředí, ve kterém se pohybují a vytvářet si o tomto prostředí vlastní představu (model). Nutným předpokladem budování takového modelu je lokalizace robotu v prostředí. Lokalizací je myšlen proces odhadu aktuálních souřadnic (polohy) robotu na základě informací z dostupných senzorů. Vhodnými senzory pro řešení úlohy lokalizace jsou senzory, které umožňují získat informaci buď o vzájemné poloze robotu vůči prostředí nebo znalost o vlastním pohybu robotu v prostředí. Pro získání této informace se používají inerciální a odometrické senzory, které poskytují přibližný výchozí odhad polohy robotu. Inerciální senzory odvozují polohu robotu od jeho zrychlení a rotace, odometrické senzory měří přímo délku trajektorie, kterou mobilní robot projel. Oba měřící principy jsou však zatíženy vysokou chybou měření. Jako hlavní senzory pro přesné určení pozice jsou nejčastěji používány laserové dálkoměry měřící přímou vzdálenost senzoru od překážek. V poslední době se vedle laserových dálkoměrů začínají prosazovat i řešení využívající kamerových systémů.
Problém lokalizace zahrnuje širokou škálu úloh lišících se použitými senzory, množstvím a kvalitou informací o okolním prostředí, charakterem prostředí a způsobem reprezentace polohy robotu. My se budeme omezíme pouze na některé nejzajímavější úlohy, které lze omezit následujícími předpoklady:

- Uvažujme terestriální robot operující v běžném kancelářském prostředí; a pohybující se na rovné, horizontální ploše (tj. vylučujeme např. schody, nakloněnou rovinu).
- Většina objektů je statická a detekovatelná senzory, kterými je robot vybaven.
- Robot je vybaven senzory, které měří vzdálenost robotu od překážek (laserové či ultrazvukové dálkoměry), přičemž je snímána jedna rovina prostředí.

Souřadnice robotu v pracovním prostředí mohou být vztaženy buď k existujícímu modelu prostředí (mapě) nebo ke zvolené vztažné referenci. Referencí často bývá například počáteční poloha robotu. Uvědomte si, že pozice (poloha) robotu zahrnuje rovněž jeho natočení. Polohou ve 2D pak rozumíme

trojici (x, y, ϕ), kde $(x, y) \in R^2$ jsou souřadnice robotu v kartézské soustavě souřadnic a $\phi \in \langle 0, 2\pi)$ jeho natočení. Pozici robotu ve 3D lze definovat několika různými způsoby, jedním z nich je např. $(x, y, z, \phi, \theta, \psi)$, kde kde $(x, y, z) \in R^3$ jsou souřadnice robotu v kartézské soustavě souřadnic a ϕ, θ a ψ jsou tzv. Eulerovy úhly.
Důležitým aspektem lokalizace je to, zda je známa pozici robotu alespoň přibližně či nikoliv. Je-li výchozí pozice robotu známa nebo pokud jsou důležité pouze relativní souřadnice mobilního robotu vzhledem k výchozí pozici, mluvíme o úloze sledování polohy (position tracking). Úloha sledování polohy je někdy nazývaná též úlohou **kontinuální lokalizace**. Tyto lokalizační úlohy korigují odometrické chyby vznikající během jízdy, čímž zpřesňují aktuální pozice robotu.
V opačné situaci, kdy apriorní informace o poloze robotu není k dispozici (např. pokud je robot poprvé nastartován nebo po kompletním selhání navigačního systému), mluvíme o úloze **globální lokalizace**, někdy též absolutní lokalizace či „lost robot problem“. Jako jeden se základních rozdílů mezi úlohou globální lokalizace a sledování polohy lze považovat požadavek na existenci předem známého modelu světa. Zatímco globální lokalizace model světa potřebuje, pro sledování polohy není model prostředí nezbytně nutný, případně lze využít průběžně vytvářeného modelu během pohybu.
Nejobecnější lokalizační úlohou je tzv. **problém uneseného robotu** (kidnapped robot problem), což je v principu zobecněná úloha sledování polohy a globální lokalizace. V této úloze jde o velmi robustní sledování polohy, kdy robot vedle sledování své pozice musí být schopný rozpoznat situaci, kdy byl násilně přemístěn do neznámé polohy tak, aniž by to byl schopen jeho senzorický systém přímo podchytit. Lokalizační metoda musí být tedy schopna detekovat, že se robot nachází na zcela jiném než předpokládaném místě.
V dalších odstavcích popíšeme problémy lokální a globální lokalizace podrobněji.

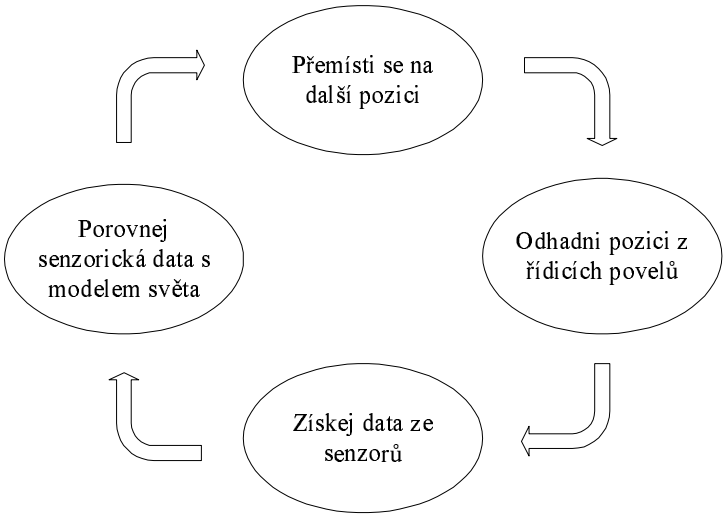
2.1 Metody kontinuální lokalizace

Cílem algoritmů kontinuální lokalizace je korekce polohy během jízdy robotu. Proces kontinuální lokalizace se typicky skládá ze čtyř akcí, které se neustále cyklicky opakují:

1. Robot se přemístí z pozice A do pozice B.
2. Na základě znalosti pozice A, řídicích povelů pro motorický systém a případně informací z čidel měřících ujetou vzdálenost jsme schopni odhadnout novou polohu.

3. Robot provede senzorická měření.
4. Porovnáním naměřených senzorických dat s modelem světa, případně s předchozími senzorickými měřeními se koriguje odhad polohy.

Z předchozího odstavce mimo jiné vyplývá, že během kontinuální lokalizace se poloha robotu nehledá v celém stavovém prostoru, ale pouze v nejbližším okolí pozice odhadnuté v kroku 2. Tím se stává tento problém lehčím než globální lokalizace, na druhou stranu je požadována mnohem větší přesnost, neboť chyby v určení polohy v jednotlivých iteracích se sčítají.



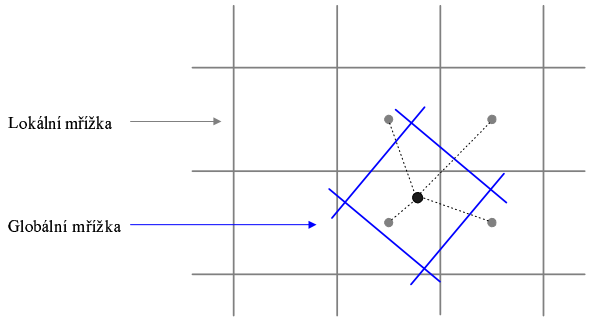
Obrázek 2.1: Proces kontinuální lokalizace

V uplynulých dvou desetiletích bylo prezentováno množství metod kontinuální lokalizace lišících se zejména reprezentací a zpracováním aktuálního senzorického měření a předchozích měření. Všechny metody ale postupně vedou na optimalizaci míry nesouladu zpracovaného aktuálního senzorického měření a předchozích měření.

2.1.1 Lokalizace na pravděpodobnostních mřížkách obsazenosti

Mnoho postupů lokalizace využívá reprezentaci prostředí ve formě tzv. **mřížky obsazenosti** [6], [4]. Prostředí je v takovém případě dekomponováno na stejně velké části typicky čtvercového tvaru, přičemž mřížka obsazenosti je matice, jejíž každý prvek nese informaci o pravděpodobnosti obsazení dané části prostředí objektem. Tato informace je budována postupně z měření získaných z různých pozic robotu, případně i z více senzorů.
Lokalizace pak spočívá v hledání optimální korespondence mezi lokální a globální mřížkou. Zatímco lokální mřížka je generována pouze z aktuálního senzorického měření, globální mřížka je tvořena během jízdy robotu integrací

všech měření. Úloha lokalizace pak spočívá v nalezení takové transformace lokální mřížky skládající se z posunutí o vektor $(\Delta x, \Delta y)$ a otočení $\Delta\phi$ takové, aby si odpovídající si buňky mřížek byly co nejvíce podobné.



Obrázek 2.2: Protože souřadné systémy lokální a globální mřížky se liší, každá buňka jedné mřížky překrývá buňky mřížky druhé.

Formálně lze lokalizaci na mřížkách definovat jako optimalizační úlohu:

$$\max_t corr(t(\mathcal{L}), \mathcal{G}), \tag{2.1}$$

kde $\mathcal{L}$ a $\mathcal{G}$ označuje lokální, respektive globální mřížku mřížku obsazenosti, *corr* korelační funkci a t transformaci.
Vzhledem k tomu, že po transformaci nemají lokální a globální mřížka stejný souřadný systém, dochází k tomu, že se buňky obou mřížek překrývají (viz obr. 2.2) a nelze tedy jednoznačně určit, které buňky si navzájem odpovídají. V [27] se pro každou buňku $\mathcal{L}_{xy}$ v lokální mřížce vypočítá její geometrický střed c_{xy}. Korespodující buňka v globální mřížce je potom taková, která obsahuje střed c_{xy}.
Robustnější technika pro určení korespondencí mezi buňkami je popsána v [24]. Jak ukazuje obrázek 2.2, každá buňka $\mathcal{G}_{xy}$ globální mřížky je překrytá několika buňkami transformované lokální mřížky - $\mathcal{L}^T_{x_i y_i}$. Globální buňka $\mathcal{G}_{xy}$ je potom porovnávána s fiktivní buňkou $\mathcal{L}^F_{xy}$ vzniklou interpolací hodnot čtyř nejbližších buněk v lokální mřížce:

$$\mathcal{L}^F_{xy} = \frac{\sum_{i=1}^{4} |x - x_i||y - y_i|\mathcal{L}^T_{x_i y_i}}{\sum_{i=1}^{4} |x - x_i||y - y_i|}.$$

Klíčovou výhodou takto definované interpolace oproti výběru pouze nejbližší hodnoty je to, že interpolační funkce je hladká a diferencovatelná. Díky tomu lze pro výpočet maxima ve vztahu 2.1 použít některou sofistikovanější optimalizační metodu.
Poslední otázkou zůstává výpočet podobnosti dvou mřížek. Yamauchi a Langley [27] počítají pro každý pár odpovídajících si buněk následující metriku:

$$corr_{Yam}(\mathcal{L}^F_{xy}, \mathcal{G}_{xy}) = \begin{cases} 1 & \text{jestliže } \mathcal{L}^F_{xy} > p_0 \text{ a } \mathcal{G}_{xy} > p_0 \\ 1 & \text{jestliže } \mathcal{L}^F_{xy} < p_0 \text{ a } \mathcal{G}_{xy} < p_0 \\ 1 & \text{jestliže } \mathcal{L}^F_{xy} = p_0 \text{ a } \mathcal{G}_{xy} = p_0 \\ 0 & \text{jinak,} \end{cases}$$

kde p_0 je apriorní pravděpodobnost, že buňka odpovídá prostoru obsazenému překážkou. Jinými slovy, dvě buňky si odpovídají, jestliže pravděpodobnost obsazení překážkou je u obou větší než apriorní nebo u obou současně menší, případně do obou buněk nezasáhlo žádné senzorické měření. Celková korespondence dvou mřížek je sumací přes všechny odpovídající si buňky:

$$corr_{Yam}(\mathcal{L}^F_{,}\mathcal{G}) = \sum_{x=1}^{n}\sum_{y=1}^{n} corr_{Yam}(\mathcal{L}^F_{xy}, \mathcal{G}_{xy})$$

Moravec a Blackwell [19] vycházejí ze skutečnosti, že pravděpodobnost obsazení obou korespodujících buněk je dána jejich součinem a pravděpodobnost, že jsou obě volné součinem jejich doplňků. Pravděpodobnost, že obě mřížky reprezentují to samé, je pak dána vztahem:

$$corr^*{}_{Mor}(\mathcal{L}^F, \mathcal{G}) = \prod_{x=1}^{n}\prod_{y=1}^{n}\left(\mathcal{L}^F_{xy}\mathcal{G}_{xy} + (1-\mathcal{L}^F_{xy})(1-\mathcal{G}_{xy})\right) \tag{2.2}$$

Obecně má hodnota $corr^*{}_{Mor}$ definovaná vztahem 2.2 velmi malou hodnotu, proto se tento výraz ještě logaritmuje. Navíc přidáním jedničky ke každému činiteli dosáhneme toho, že korelace buňek s největší jistotou (tedy mající hodnotu 0 nebo 1) bude 1, zatímco pro buňku s neznámým stavem (0.5) bude korelace při porovnání s libovolnou buňkou nulová:

$$\begin{aligned} corr_{Mor}(\mathcal{L}^F, \mathcal{G}) &= n + \log_2 \prod_{x=1}^{n}\prod_{y=1}^{n}(\mathcal{L}^F_{xy}\mathcal{G}_{xy} + (1-\mathcal{L}^F_{xy})(1-\mathcal{G}_{xy})) = \\ &= \sum_{x=1}^{n}\sum_{y=1}^{n}(1 + \log_2(\mathcal{L}^F_{xy}\mathcal{G}_{xy} + (1-\mathcal{L}^F_{xy})(1-\mathcal{G}_{xy}))) \end{aligned}$$

Hlavní nevýhodou lokalizace na základě porovnávání mřížek je velká výpočetní složitost (pro každý krok optimalizačního algoritmu $O(n^2)$, kde n je velikost mřížky), kterou lze sice snížit zvětšením rozměrů jedné buňky mřížky a tím i zmenšením velikosti celé mřížky, ale pouze na úkor přesnosti určení polohy. Proto se někteří autoři snaží nalézt vylepšení výše uvedených metod, které by byly méně výpočetně náročné.

2.1.2 Metody založené na porovnávání sad nezpracovaných senzorických měření

Další třídou metod kontinuální lokalizace jsou postupy pracující přímo s hrubými senzorickými daty. Vstupem algoritmu je tedy dvojice scanů - referenční S^{ref} a aktuální S^{akt}. Problém lokalizace v tomto případě spočívá v hledání nejlepšího umístění aktuálního scanu vzhledem k referenčnímu tak, aby se minimalizovala suma kvadrátů vzdáleností odpovídajících si bodů v referenčním a aktuálním scanu.
Lu a Milios presentují v [17] iterativní algoritmus, jehož idea je následující:

1. Referenční scan S^{ref} se aproximuje křivkou K^{ref} tak, že sousední body ve scanu se spojí úsečkou.
2. Pro každý bod $P_i \in S^{akt}$ se nalezne korespondující bod $P'_i \in K^{ref}$.
3. Ze všech nalezených dvojic odpovídajících si bodů se metodou nejmenších čtverců určí relativní otočení $\Delta\phi$ a posunutí T, které se získají minimalizací funkce E:

$$E(\phi, T) = \sum_{i=1}^{n} |P_i M_{\Delta\phi} + T - P'_i|^2, \tag{2.3}$$

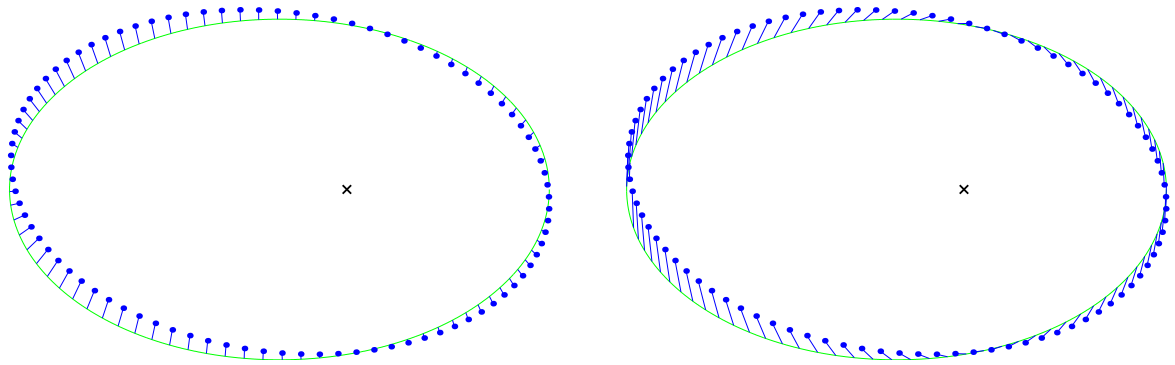
   kde $M_{\Delta\phi}$ je transformační matice popisující otočení o úhel $\Delta\phi$:

$$M_{\Delta\phi} = \begin{bmatrix} \cos(\Delta\phi) & \sin(\Delta\phi) \\ -\sin(\Delta\phi) & \cos(\Delta\phi) \end{bmatrix}.$$

4. Tento postup se iterativně opakuje tak dlouho, dokud nekonverguje.

Klíčovým problémem porovnávání dvou scanů je nalezení korespondencí mezi jednotlivými body obou scanů. Obecně používané pravidlo k danému bodu z jednoho scanu vybere nejbližšího souseda z druhého scanu. Toto pravidlo nazývané **pravidlo nejbližšího bodu** je ilustrováno na obrázku 2.3(a).
Výše popsaný postup je speciálním případem algoritmu **Iterative Closest Point** (ICP) vyvinutým Beslem a McKayem. Dá se dokázat [1], že pro pravidlo nejbližšího bodu a distanční funkci definovanou v 2.3 konverguje ICP monotónně k lokálnímu minimu. Jak ukázaly experimenty, pokud je rotace $\Delta\phi$ malá, tento algoritmus pracuje velmi dobře.
Nevýhodou ICP je to, že konverguje velmi pomalu. To je způsobeno tím, že páry generované pravidlem nejbližšího bodu mají nekonzistentní směry (viz obr. 2.3(a)), což vede k tomu, že je informace o rotaci téměř nulová. Další problém nastává, pokud se aktuální řešení přiblíží k lokálnímu minimu. I v tomto případě je rychlost konvergence velmi malá.

Malou rychlost konvergence v blízkosti lokálního minima lze zvýšit [1], ale problém s rotací jednoduše odstranit nelze. Proto Lu a Milios navrhli další kritérium a nazvali ho **pravidlo podobného dosahu** (*Iterative Matching Range Point*). Toto kritérium upřednostňuje rotační složku oproti translační.



(a) (b)

Obrázek 2.3: Pravidla pro hledání korespondencí. Vlevo pravidlo nejbližšího bodu, vpravo pravidlo podobného dosahu. Zelená elipsa reprezentuje referenční scan proložený úsečkami, modré body aktuální scan a křížek polohu robotu.

Předpokládejme bod $P = (\alpha, r) \in S^{akt}$ a jemu odpovídající bod $P' = (\alpha', r') = PM_{\Delta\phi} + T \in K^{ref}$. Pokud zanedbáme posunutí, pak $r \approx r'$. Na druhou stranu, úhly α a α' jsou svázány vztahem $\alpha = \alpha' + \Delta\phi$. To znamená, že odpovídající si body mají přibližně stejnou naměřenou vzdálenost od senzoru a jejich úhly reprezentující tyto body v polárních souřadnicích se liší o $\Delta\phi$. Lze předpokládat, že i v případě, kdy mezi scany bude malé posunutí, bod P' koresponduje s P, pokud mají stejnou vzdálenost od senzoru.
Aby bylo zajištěno, že pravidlo najde správnou korespondenci, musíme uvažovat pouze body P', jejichž úhel α' se od α liší o hodnotu $\pm\Phi$. Φ je konstanta definující toleranci. Pravidlo podobného dosahu lze tedy definovat takto: Pro bod $P = (\alpha, r) \in S^{akt}$ je korespondující bod $P' = (\alpha', r') \in K^{ref}$, pro který platí, že $|r - r'|$ je minimální pro všechny body splňující $|\alpha - \alpha'| \leq \Phi$.
Obrázek 2.3(b) ukazuje příklad použití pravidla podobného dosahu pro scany ve tvaru elipsy. Je vidět, že vektory spojující korespondující body poměrně přesně indikují rotační složku transformace, což vede k rychlé konvergenci rotace. Naopak translační složka konverguje pomaleji než u algoritmu ICP.
Protože pravidlo nejbližšího bodu hledá velmi rychle posunutí mezi scany, zatímco pravidlo podobného dosahu jejich otočení, navrhli Lu a Milios algoritmus **Iterative Dual Correspondence**, který obě pravidla kombinuje. V každém kroku iteračního cyklu se počítají korespondující body pomocí

obou pravidel, přičemž výsledná transformace je složením translace nalezené prvním pravidlem a rotace nalezené druhým.
Efektivní lokalizační metoda byla vyvinuta Hinkelem a Knieriemenem [9] a o několik let později zdokonalena Weißem a spol. [26]. Tato metoda počítá transformaci mezi dvěma scany korelováním dvojic histogramů vytvořených z referenčního a aktuálního scanu.
V prvním kroku algoritmu se vytvoří pro každý scan tzv. úhlový histogram, což je histogram úhlů, které svírají vždy dvě sousední měření jednoho scanu:

$$H = (H_i)_{i=0}^{n},$$

kde H_i je počet úhlů v intervalu $\langle i\frac{\pi}{n}, (i+1)\frac{\pi}{n}\rangle$.
Lokální maxima úhlového histogramu odpovídají hlavním směrům překážek ve scanu, tj. v kancelářském prostředí směrům zdí. Navíc, pro dva scany stejného prostředí získané ze dvou různých (ale blízkých) pozic a s jiným natočením, budou úhlové histogramy vůči sobě posunuté, ale jinak velice podobné. Tato vlastnost je použita pro získání rotační diference dvou scanů minimalizací korelační funkce odpovídajících úhlových histogramů:

$$corr_{Hin}(H^{act}, H^{ref}, p) = \sum_{i=1}^{n} H_i^{act} H_{i\oplus p}^{ref}.$$

Znak $\oplus$ značí sčítání modulo π. Použití této operace je smysluplné, neboť úhlové histogramy jsou periodické.
Pro výpočet posunutí se používají dvě dvojice histogramů - jedna pro určení x-ové složky posunutí a druhá pro určení y-ové složky. Při výpočtu těchto histogramů může často nastat problém, že x-ové a y-ové souřadnice bodů scanu jsou rozmístěny téměř rovnoměrně (a tedy histogramy neobsahují výrazné extrémy), ačkoliv v samotném scanu jsou zřetelné obrysy překážek. To lze obejít tím, že se scany natočí tak, aby x-ová i y-ová osa byla paralelní s nejvýraznějšími překážkami, které odpovídají největším lokálním maximům v úhlovém histogramu.

2.2 Pravděpodobnostní metody lokalizace

V předchozích odstavcích byly představeny metody, které pracovaly s jednou hypotézou polohy robotu a tuto hypotézu považovaly za správnou. Vzhledem k tomu, že senzorická měření i realizace naplánovaných akcí jsou ze své podstaty nepřesné, byly navrženy metody, které s těmito neurčitostmi umí pracovat a odhadovat tak i neurčitost v poloze robotu.
Tyto metody jsou obecně více robustní, lépe se dokáží vyrovnat se šumy měření, ovšem jsou často výpočetně náročnější než metody pracující bez ní.

Zavedení neurčitosti s sebou přináší nutnost práce s pravděpodobnostním popisem modelu světa i senzorických dat a tím souvisejícím výpočtem složitých posteriorních hustot pravděpodobností, které jsou v idealizovaném teoretickém pojetí spojité a mnohadimenzionální. V praktickém řešení je nutnost přistoupit z důvodů výpočetní složitosti vždy na určitou míru aproximace reálného světa a výpočetních postupů. Pravděpodobnostní algoritmy se mohou zdát méně efektivní, je to ovšem přirozená cena za schopnost práce s nepřesnými modely, nepřesnými daty a možnosti samozotavení se z případných hrubých chyb měření či modelu.
Pravděpodobnostní lokalizační algoritmy patří do skupiny globálních lokalizačních technik, provádí odhad pozice robotu ve známém prostředí a jsou založeny na Bayesově filtru:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)},$$

který vyjadřuje pravděpodobnost podmíněného jevu pomocí „opačné“ podmíněné pravděpodobnosti, tj. umožňuje zaměnit příčinu a důsledek.
Základním předpokladem pravděpodobnostních metod je tzv. **markovský předpoklad**, který předpokládá, že následující stav závisí pouze na aktuálním stavu a nezávisí na stavech předchozích. Pravděpodobnostní metody lokalizace jsou z tohoto důvodu označovány jako **Markovská lokalizace**.
Odhad polohy robotu je v Markovských lokalizačních metodách popsán jako statistická veličina včetně nejistoty určení této polohy. Místo udržování jedné hypotézy o tom, kde se robot může nacházet, Markovská lokalizace používá teorie pravděpodobnosti k udržování odhadů pravděpodobnostního rozložení výskytu robotu v celém prostoru přes celou množinu všech možných poloh robotu.
Problém lokalizace robotu je v podstatě převeden na problém aktualizace tohoto rozložení pravděpodobnosti na základě informací o pohybu robotu, dostupných senzorických datech a existující mapy prostředí. Aktuální (nejvěrohodnější) poloha robotu v prostředí je pak určena jako maximum v rozložení hustoty pravděpodobnosti.
Základní myšlenka Markovské lokalizace je shrnuta na obr. 2.4, na kterém je uvažován zjednodušený případ jednorozměrného světa, ve kterém se pohybuje robot bez výchozí znalosti své aktuální pozice. Markovská lokalizace tento stav reprezentuje uniformě rozdělenou hustotou pravděpodobnosti přes všechny možné pozice (první diagram z obr. 2.4). Robot je schopen měřit přibližnou změnu své polohy a vnímat prostředí na úrovni rozhodování, zda stojí před dveřmi nebo před zdí.
Jakmile robot zpracuje senzorická data a zjistí, že se nachází před některými

Obrázek 2.4: Myšlenka Markovské lokalizace - mobilní robot v úloze globální lokalizace, obrázek převzat z [7]

dveřmi, Markovská lokalizace upraví na základě známé mapy a senzorického měření rozložení pravděpodobnosti. Pravděpodobnost, že se robot nachází v některé z pozic před dveřmi se zvýší, naopak v ostatních místech prostoru se sníží (druhý diagram obr. 2.4). Snížení hodnot hustoty pravděpodobnosti však není nikde až na nulovou hodnotu vzhledem k šumu měření a možnosti, že robot se nachází i jinde než před dveřmi, přestože senzorická měření dávají jiný výsledek. Výsledná hustota pravděpodobnosti upravená na základě senzorického modelu je multimodální, což reprezentuje neurčitost znalosti aktuální polohy robotu.

Robot se posléze přesune do jiné pozice o přibližně známou vzdálenost, což je v Markovské lokalizaci reprezentováno posunem celé distribuce pravděpodobnosti ve směru a vzdálenosti odpovídající předpokládanému pohybu robotu (třetí diagram z obr. 2.4). Možné nepřesnosti a nejistoty v pohybu robotu jsou v distribuci zohledněny zploštěním a větším rozprostřením distribuce (zvýšení rozptylů dříve vzniklých lokálních extrémů v rozložení). Tento krok se nazývá aplikace modelu pohybu robotu.

V další fázi robot získá nová senzorická měření, která odpovídají tomu, že se robot opět nachází před některými dveřmi. Toto pozorování je zkombinováno se současnou mírou důvěry v určité polohy robotu, což vede k získání nového rozložení pravděpodobnosti, ve kterém je již velmi výrazné maximum odpovídající skutečné poloze robotu. Robot je tak lokalizován. V reálné situaci je proces pochopitelně složitější, zejména je třeba více kroků k jednoznačnému určení polohy robotu.

Formálně lze metodu popsat následovně. Pozice robotu je chápána jako vektor $x = (pos_x, pos_y, \varphi)$ obsahující souřadnice robotu v kartézských souřadni-

cích spolu s jeho natočením, x_t je pak reálná pozice robotu v čase t. Budeme-li s polohou zacházet jako s náhodnou veličinou, označíme ji jako x_t. Markovská lokalizace pracuje s pojmem *důvěra v pozici robotu* v čase t, která je označovaná jako $Bel(x_t)$, což je v principu distribuce pravděpodobnosti přes prostor všech pozic robotu. $Bel(x_t = x)$ je pak hustota pravděpodobnosti, přiřazující robotu pravděpodobnost, že jeho pozice v čase t je právě x.
Důvěra $Bel(x_t)$ je aktualizována následkem dvou událostí, příchodem senzorických dat z_t nebo informací o pohybu u_t (odometrie). Robot pak zpracovává sekvenci měření:

$$d = d_0, d_1, ..., d_t,$$

kde každé d_t (s $0 \leq t \leq t$) reprezentuje buď jedno senzorické z_t nebo odometrické měření u_t.
Markovská lokalizace odhaduje posteriorní distribuci pravděpodobnosti náhodné veličiny x_t, která je podmíněna příchozími daty:

$$p(x_t = x \mid d) = p(x_t = x \mid d_0, d_1, ..., d_t),$$

V odvození inkrementálních aktualizací posteriorních pravděpodobností se používá markovský předpoklad. Platí tedy:

$$p(d_{t+1}, d_{t+2}... \mid x_t = x, d_0, ..., d_t) = p(d_{t+1}, d_{t+2}... \mid x_t = x) \quad \forall\, t.$$

Během aktualizace $p(x_t = x \mid d)$ se rozlišují případy, podle typu aktuálně příchozích dat:

Případ 1: Data jsou senzorická měření $d_t = z_t$

Základní vztah $P(x_t = x \mid d) = p(x_t = x \mid d_0, d_1, \ldots, z_T)$ je možné aplikací Bayesova pravidla upravit do podoby

$$p(x_t = x \mid d) = \frac{p(z_t \mid d_0, \ldots, d_{t-1}, x_t = x) \cdot p(x_t = x \mid d_0, \ldots, d_{t-1})}{p(z_t \mid d_0, \ldots, d_{t-1})},$$

který může být s využitím markovského předpokladu a zavedením α_T za jmenovatele (na x_t nezávislým) $p(z_t \mid d_0, \ldots, d_{t-1})$ upraven na:

$$p(x_t = x \mid d) \;=\; \alpha_t \cdot p(z_t \mid x_t = x) \cdot p(x_t = x \mid d_0, ..., d_{t-1}),$$

Vzhledem k tomu, že důvěra v danou pozici robotu je ekvivalentní posteriorní pravděpodobnosti

$$Bel(x_t = x) = p(x_t = x \mid d_0, \ldots, d_t),$$

vztah 2.2 lze za předpokladu $p(z_t \mid x_t = x) \simeq P(z_t \mid x)$ (nezávislost na čase) a minulých datech $\{d_0, \ldots, d_{t-1}\}$ vyjádřit v rekurzivní podobě:

$$Bel(x_t = x) \;=\; \alpha_t \cdot p(z_t \mid x) \cdot Bel(x_{t-1} = x),$$

Zde stojí za zmínku člen $p(z_z \mid x)$, který představuje **senzorický model** a v praxi popisuje pravděpodobnost daného senzorického měření (předpokládaného podle modelu prostředí), pokud se robot nachází v určité pozici v prostoru.

Případ 2: Data jsou informace z odometrie $d_t = u_t$

V tomto případě budeme počítat $P(X_T = x \mid d)$ na základě teorému o úplné pravděpodobnosti:

$$p(x_t = x \mid d) = \int p(x_t = x \mid d, x_{t-1} = x') \cdot p(x_{t-1} = x' \mid d)\; dl' \qquad (2.4)$$

Pro první součinitel na pravé straně opět použijeme markovský předpoklad, čímž dostaneme:

$$\begin{aligned} P(x_t = x \mid d, x_{t-1} = x') &= p(x_t = x \mid d_0, \ldots, d_{t-1}, u_t, x_{t-1} = x') \\ &= p(x_t = x \mid u_t, x_{t-1} = x') \end{aligned}$$

Druhý součinitel na pravé straně může být též zjednodušen na základě toho, že u_t nepřináší žádnou novou informaci o minulé poloze x_{t-1}:

$$\begin{aligned} p(x_{t-1} = x' \mid d) &= p(x_{t-1} = x' \mid d_0, \ldots, d_{t-1}, u_t) \\ &= p(x_{t-1} = x' \mid d_0, \ldots, d_{t-1}) \end{aligned}$$

Dosadíme-li do výrazu 2.4 součinitele 2.5 a 2.5 získáme výsledek

$$p(x_t = x \mid d) = \int p(x_t = x \mid u_t, x_{t-1} = x') \cdot p(x_{t-1} = x' \mid d_0, \ldots, d_{t-1})\; dl',$$

který lze dále upravit za použití definované důvěry 2.2 a předpokladu $p(x \mid u_t, x') \simeq p(x_t = x \mid u_t, x_{t-1} = x')$ (předpokládáme nezávislost na čase) do inkrementální podoby:

$$Bel(x_t = x) = \int p(x \mid u_t, x') \cdot Bel(x_{t-1} = x') dx'$$

Rozložení pravděpodobnosti $p(x \mid u_t, x')$ je označeno jako **model pohybu**. Model pohybu popisuje, jak se zvýší neurčitost v pozici robotu po aplikaci pohybu popsaného odometrií u_t. Lze tedy shrnout, že pro Markovskou lokalizaci jsou podstatné reprezentace důvěry $Bel(x)$, resp. reprezentace distribuce pravděpodobnosti výskytu robotu v daném místě v prostoru a dále distribuce

podmíněných pravděpodobností $p(x \mid u, x')$ (reprezentují pohybový model robotu) a $p(z \mid x)$ (reprezentující senzorický model robotu, někdy nazývaný též percepční model).

Uvážíme-li, že oba uvedené případy se během jízdy robotu střídají, důvěru v příslušnou pozici x robotu v čase t označíme jako $Bel_t(x_t)$, pak dostaneme rekurzivní vztah popisující princip Markovské lokalizace:

$$Bel(x_t) = \alpha_t \, p(z \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \, Bel(x_{t-1} = x') dx_{t-1}, \qquad (2.5)$$

kde α_t je normalizační koeficient zajišťující, aby výsledná suma důvěry přes všechny možné pozice robotu byla rovna jedné. Zde je nutno poznamenat, že pohybový a zejména senzorický model reálně závisí na modelu prostředí m. Po zavedení modelu prostředí m je nutno výše uvedené hustoty pravděpodobnosti označovat jako $p(x_t \mid u, x_{t-1}, m)$ a $P(z \mid x_t, m)$.

2.2.1 Postupy využívající Kalmanova filtru

Mezi velmi populární pravděpodobnostní přístupy Markovské lokalizace patří Kalmanovy filtry. Základní princip byl představen Rudolphem Kalmanem v roce 1960 [11]. Lokalizační metody založené na Kalmanově fitru reprezentují odhad pozice robotu pomocí unimodálního gaussovského rozložení. Reprezentace pozice robotu je tedy popsána střední hodnotou této distribuce a jejím rozptylem, což umožňuje pracovat s příslušnou neurčitostí pozice robotu.

Proces lokalizace se opírá o aktualizace gaussovského rozložení následujícím způsobem. Jakmile se robot přemístí, gaussovské rozložení je posunuto podle předpokládané velikosti a směru pohybu, která je zpravidla měřena odometrickými senzory, nebo je určena přímo z akčních zásahů do motorů robotu (model pohybu). Současně s tím je upraven rozptyl rozložení podle modelu odometrického subsystému, resp. podle jeho předpokládaných chyb. Následně se do odhadu pozice, tedy do aktualizace gaussovského rozložení, zahrnou informace ze senzorů, které přinášejí informaci o reálné poloze ve vztahu k modelu světa (model senzoru). Využití senzorické informace dříve zvýšenou neurčitost (rozptyl) gaussovského rozložení opět sníží a cyklus se může zopakovat.

Formálně lze činnost Kalmanova filtru, tedy aktualizaci parametrů distribuce hustoty rozložení pravděpodobnosti výskytu robotu v příslušné poloze a čase

t, popsat následujícími vztahy:

$$\begin{aligned}
\mu_{t-1}^{'} &= \mu_{t-1} + Bu_t \\
\Sigma_{t-1}^{'} &= \Sigma_{t-1} + \Sigma_{control} \\
K_t &= \Sigma_{t-1}^{'} C^T (C\Sigma_{t-1}^{'} C^T + \Sigma_{measure})^{-1} \\
\mu_t &= \mu_{t-1}^{'} + K_t(z - C\mu_{t-1}^{'}) \\
\Sigma_t &= (I - K_t C)\Sigma_{t-1}^{'},
\end{aligned}$$

kde μ_t a Σ_t jsou momenty distribuce pravděpodobnosti, B je převodní matice řízení, C je převodní matice měření a z je vektor naměřených dat. Předpokládaná přesnost pohybového modelu robotu je parametrizována maticí $\Sigma_{control}$, zatímco důvěra v senzorická data je popisována kovarianční maticí $\Sigma_{measure}$. Kalmanovo zesílení K_t v principu udává jakou měrou bude budoucí odhad stavu (poloha robotu) ovlivněn nově změřenými daty, jeho velikost závisí na odhadované přesnosti minulého odhadu stavu Σ_{t-1} a přesnosti měření $\Sigma_{measure}$.
Je nutno uvést, že výše uvedené vztahy platí pouze pro lineární systémy, pro praktické realizace se musí využívat rozšířeného Kalmanova filtru, který celý problém linearizuje vůči aktuální poloze robotu.
Výhoda metod založených na Kalmanově filtraci spočívá v jejich efektivitě a velmi vysoké přesnosti za předpokladu, že je známa výchozí pozice. Bohužel, významnou principiální nevýhodou Kalmanových filtrů jsou omezení kladená na distribuce hustot pravděpodobností. Teoretické základy metody předpokládají, že všechny neurčitosti je možné modelovat procesy s normálním rozložením. Tyto podmínky je v praxi často obtížné splnit, navíc díky reprezentaci polohy robotu jedním gaussovským rozložením nejsou metody tohoto typu schopny řešit úlohy globální lokalizace a nejsou schopny se zotavit z větších pozičních chyb.

2.2.2 Monte-Carlo lokalizace

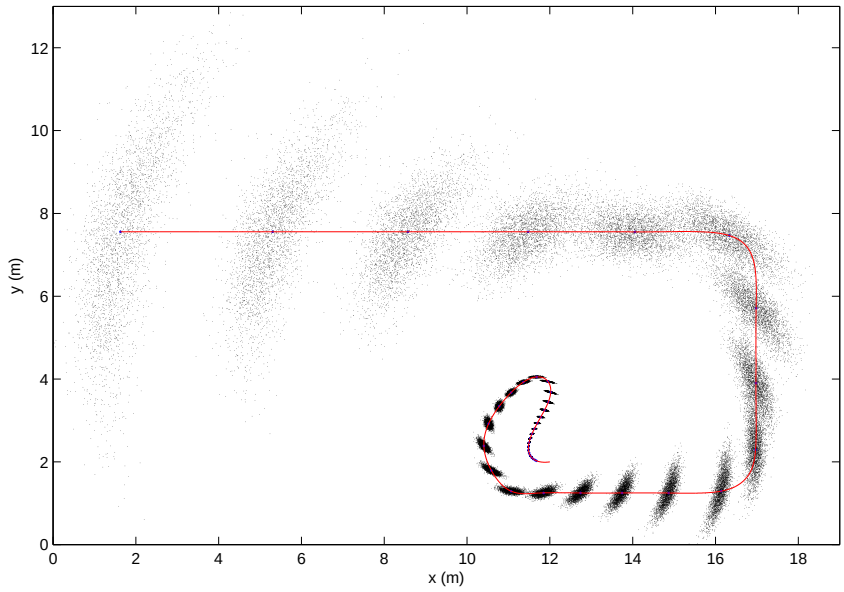
Další metoda Markovské lokalizace je založená na metodě Monte-Carlo [8], která reprezentuje distribuci pravděpodobnosti polohy množinou vzorků. Tento typ reprezentace je poměrně často používaný princip v mnoha různých oborech. V robotické a statistické literatuře jsou Monte-Carlo přístupy známy pod pojmem *částicové filtry*(particle filters) [21], v oblasti počítačového vidění se vyskytuje pojmem kondenzační algoritmus [10], v ostatních oborech lze nalézt též názvy *bootstrap filters*, *interacting particle approximations* nebo *survival of the fittest*.
Výhody použití částicových filtrů lze spatřit v následujících bodech:

- Částicové filtry jsou univerzálním nástrojem pro aproximaci hustot pravděpodobností, které nekladou omezení na tvar posteriorních hustot pravděpodobnosti.
- Částicové filtry se mohou přizpůsobit téměř libovolné charakteristice senzorů, dynamice pohybu či rozložení šumu.
- Částicové filtry rozprostírají výpočetní výkon uvnitř stavového prostoru v poměrech odpovídající příslušným posteriorním pravděpodobnostem, čímž je výkon soustředěn zejména do okolí vysokých pravděpodobností.
- Výpočetní náročnost algoritmu může být jednoduše přizpůsobena aktuální výpočetní kapacitě i za běhu přizpůsobením počtu vzorků bez zásahu do samotného algoritmu.
- Implementace algoritmu není příliš komplikovaná a může být poměrně snadno paralelizována.

Základní myšlenka Monte-Carlo lokalizace (MCL) je vhodná aproximace důvěry $Bel(x)$ váženou množinou vzorků tak, aby diskrétní distribuce definovaná vzorky skutečně odpovídala výchozí spojité distribuci.
Počáteční rozložení důvěry je reprezentováno uniformním rozdělením množiny vzorků o velikosti n, což znamená, přičemž každý vzorek má váhu n^{-1}.
MCL algoritmus ve své podstatě implementuje aktualizaci vztahu 2.5 vytvořením nové množiny vzorků ze stávající množiny jako odezvu na příchozí informaci o pohybu robotu u_{t-1} a senzorické měření z_t (pozorování světa) následovně:

1. Náhodně vyber vzorek x_{t-1} z aktuálního popisu důvěru $Bel_{t-1}(x_{t-1})$ s pravděpodobností danou importance faktory příslušných vzorků důvěry $Bel_{t-1}(x_{t-1})$.
2. Pro tento vybraný vzorek x_{t-1} odhadni jeho možnou pozici x_t podle rozložení pravděpodobnosti $P(x_t \mid u_{t-1}, x_{t-1}, m)$ (model pohybu), viz obr. 2.5.
3. Vzorku x_t přiřaď výchozí hodnotu váhy podle rozložení modelu senzoru $p(z_t \mid x_t, m)$ a přidej jej do množiny vzorků reprezentující $Bel_t(x_t)$.
4. Opakuj kroky 1-3 n krát a nakonec normalizuj váhy všech vzorků množiny $Bel_t(x_t)$, aby jejich součet byl roven jedné.

Shrneme-li výše uvedená fakta, MCL pracuje rekurzivně ve dvou základních krocích:



Obrázek 2.5: Vzorkovaná aproximace pravděpodobnosti výskytu robotu za předpokladu, že robot měří pouze odometrii. Červená plná čára představuje předpokládaný pohyb robotu a tečky reprezentují rozložení důvěry robotu v různých bodech v čase.

- *Predikce* - posun všech vzorků na základě informací o změně pozice robota např. z odometrie.
- *Korekce* - úprava množiny jednotlivých vzorků a jejich vah na základě shody či neshody naměřených dat s očekáváními, která by odpovídala pozici reprezentované příslušným vzorkem.

V kontextu praktického použití MCL algoritmu je třeba poznamenat, že díky vzorkovaným aproximacím pravděpodobnostních rozložení algoritmus nemůže pracovat s idealizovanými daty (distribuce $p(z_t \mid x_t, m)$ mají velmi ostrá maxima) a bezchybným modelem světa. Data musí obsahovat určité množství šumu, které více rozprostře distribuce $P(z_t \mid x_t, m)$. Z tohoto důvodu je přímé využití metody pro přesné senzory s nízkým šumem omezeno.

Kapitola 3

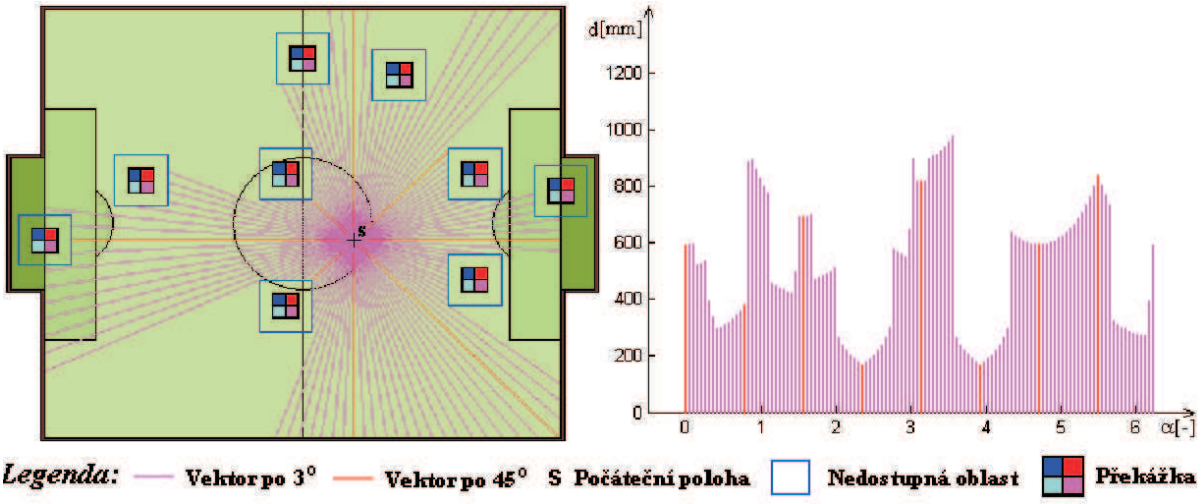
Plánování pohybu mobilního robotu a skupiny mobilních robotů

3.1 Základní algoritmy pro plánování pohybu mobilního robotu

V této kapitole bude čtenář seznámen se základními principy používanými v robotice pro plánování cesty případně trajektorie mobilního robotu. V klasické literatuře se můžeme setkat s několika pohledy na taxonomii plánování mobilního robotu. Z praktického pohledu se jako vhodné jeví dělit metody plánování cesty na lokální a globální. Lokální algoritmy uvažují pouze bezprostřední okolí robotu, často omezené dosahem robotických senzorů. Tyto metody jako výstup určí pouze směr pohybu robotu z jeho aktuální pozice. Zatímco takzvané globální metody pracují s celým pracovním prostorem robotu, jehož mapa je vstupem těchto algoritmů, a jako výstup poskytují kompletní předpokládanou cestu pohybu robotu z aktuálního do požadovaného stavu.

3.1.1 Lokální metody

Jak již bylo řečeno, lokální algoritmy pro plánování pohybu robotu pracují pouze s jeho bezprostředním okolím. Tyto metody jsou často přímo přizpůsobeny konkrétnímu typu senzoru používaného pro vnímání prostředí robotem. Obecně jsou tyto přístupy výpočetně méně náročné a umožňují tak rychlou odezvu na změny v prostředí a dynamické překážky. Určitou nevýhodou v některých aplikací je neznalost celé předpokládané cesty, která například znemožní odhadnout dobu, za kterou se robot dostane do požadované pozice. Tato informace může být klíčovým vstupem pro rozhodovací algoritmy, typicky vybírající nejvhodnější z množiny možných lokací, které by robot měl dosáhnout při plnění jeho úkolů. Další z technických problémů některých jednodušších verzí těchto algoritmů je možnost výskytu nežádoucích oscilací pohybu robotu, případně i jeho uváznutí, v důsledku lokálních extrémů plánovacích funkcí.



Obrázek 3.1: Vlevo: aplikace VFH v robotickém fotbalu. Vpravo: výsledný graf.

Histogram vektorového pole

Přestože většina monografií začíná svůj výčet plánovacích metod algoritmem potenciálového pole, který je v robotice pravděpodobně nejrozšířenější, v tomto skriptu se nejprve seznámíme s metodou založenou na histogramu vektorového pole. Tento jednoduchý algoritmus totiž používaly již prvotní robotické platformy a lze jej doporučit i nyní začínajícím zájemcům o robotiku.

Histogram vektorového pole (dále jen VFH, z anglického Vector Field Histogram) původně vznikl jako navigační mechanismus pro roboty opatřené sonarem pohybující se v prostředí s překážkami. Chceme-li použít robot vybavený libovolným dálkoměrným senzorem, umožňujícím získat informaci o vzdálenosti k nejbližším překážkám v pracovní prostoru robotu v rozsahu nejlépe 360 stupňů, lze takovýto senzorový výstup použít přímo jako hledaný histogram. V článku [2] lze nalézt podrobný popis tvorby VFH a jeho následné použití v úloze navigace mobilního robotu a vyhýbání se překážkám.

My zde ukážeme jak VFH použít i v aplikacích, které nevyužívají dálkoměrné senzory, což umožní pochopit daný princip obecněji. Zaměříme se na aplikace, ve kterých je poloha a tvar jednotlivých překážek poskytnuta jinými senzory nebo je dána mapou. Výsledný histogram potom tvoří vzdálenosti od výchozí pozice robota k hranicím nejbližší překážky ve směru jednotlivých úhlů, stejně jako u otočného sonaru, který zkoumá okolní pracovní prostor postupně v rozsahu 360 stupňů. Dá se říci, že histogram je tvořen velikostmi maximálního vektoru, který lze do daného směru vepsat a pro který platí, že má počátek ve středu robotu a neprotíná žádnou z překážek.

Pro zjednodušení popisu algoritmu jsme vybrali aplikaci robotického fotbalu, konkrétně implementaci, která se stala součástí řídícího systému týmu ČVUT G-Bot. Na obrázku 3.1 je sestaven VFH pro konkrétní situaci v robotickém fotbalu. Z obrázku je patrné rozdělení pracovního prostoru robotu pomocí vektorů tak, že dva sousední vždy svírají úhel tři stupně. Tímto způsobem byly informace z mapy prostředí transformovány do podoby odpovídající výstupu sonaru. Pro získání optimálního směru pohybu robotu je takto nalezený histogram zpracován algoritmem popsaným v článku [2]. Zde je histogram naprahován funkcí s adaptivním prahem, jehož velikost závisí na vzdálenosti od cíle. Daný práh odpovídá vzdálenosti, za kterou se již neuvažují překážky, přestože je senzory robotu mohou zaznamenat. Výsledným směrem pohybu robotu je určen takový vektor, který splňuje následující podmínky.

- Délka daného vektoru je větší než zvolený práh.
- Úhel, který svírá s přímkou procházející aktuální a požadovanou polohou robotu, je ze všech vektorů splňujících první podmínku nejmenší.

Potenciálové pole

Jakožto nejznámější a nejpoužívanější zástupce zmíněné třídy lokálních algoritmů plánování pohybu mobilního robotu je uváděn algoritmus potenciálového pole. Tato metoda, původně inspirovaná fyzikálními jevy v teorii elektromagnetických polí, je velmi jednoduchá na pochopení i implementaci. Též díky její výpočetní nenáročnosti našla široké uplatnění v oblasti mobilní robotiky již v jejích počátcích.
Základní myšlenkou tohoto algoritmu, popsaného například v [12], je nalézt vhodný matematický popis pracovního prostoru robotu funkcí, jejíž záporný gradient v libovolném bodě prostoru určí požadovaný směr pohybu robota v tomto bodě. Tato funkce by měla ve své nejednodušší podobě uvažovat dva často protichůdné požadavky:

- zajištění bezkolizního pohybu v dostatečné vzdálenosti od překážek,
- dosažení požadované pozice robotu v pokud možno nejkratším čase.

Pro dosažení zmíněných kritérií je vhodné v analogii s elektromagnetickými poli zapsat hledanou funkci jako součet dvou složek, atraktivní a repulsivní. My budeme značit atraktivní složku výsledné funkce jako Z_1 a repulsivní část jako Z_2. Vlastní směr pohybu robotu, zde značený jako $\varphi(x, y)$, při plánování v 2D prostoru lze potom jednoduše určit jako záporný gradient součtu obou příspěvků.

$$\varphi(x,y) = -grad(Z_1(x,y) + Z_2(x,y)) = \\ = -\left(\frac{\delta(Z_1(x,y) + Z_2(x,y))}{\delta x}; \frac{\delta(Z_1(x,y) + Z_2(x,y))}{\delta y}\right) \tag{3.1}$$

Obecně funkce Z_1 by měla mít jediné minimum v bodě, který odpovídá požadované poloze robota C a její záporný gradient by měl v libovolném bodě pracovního prostoru robotu směřovat do bodu C. Naopak funkce Z_2 musí zabránit kolizi robota s překážkami a umožnit jejich plynulé objetí. Z_2 by proto měla mít lokální maxima například ve středu jednotlivých překážek. Konkrétní podoba funkcí Z_1 a Z_2 se bude mírně lišit pro jednotlivé robotické aplikace a použité typy map. My vám zde ukážeme pro ilustraci návrh potenciálového pole pro potřeby robotického fotbalu.

- **Atraktivní část potenciálového pole**.

Gradient funkce potenciálového pole, která má robot „přitahovat“ do požadovaného stavu, by zároveň měl mít konstantní velikost v celém pracovním prostoru robotu. To zajistí, že jednou definované konstanty, které nastavují vliv jednotlivých složek potenciálového pole, budou platit po celou dobu pohybu robotu. V takovém případě nebude chování robotu ovlivněno vzdáleností od cílového stavu. Například při vyhýbání se překážek bude stále dodržena přibližně stejná bezpečnostní vzdálenost mezi robotem a překážkou.
Pro robotický fotbal se jako vhodná jeví funkce Z_1 ve tvaru

$$Z_1(x,y) = \sqrt{\left((x - C_x)^2 + (y - C_y)^2\right)}. \tag{3.2}$$

- **Repulsivní část potenciálového pole**.

Funkci Z_2 je v robotickém fotbalu vhodné dekomponovat na součet dvou částí. První složka Z_{2M} popisuje vliv mantinelu na pohyb robotu, zatímco druhá složka Z_{2R} zajišťuje vyhýbání se ostatním robotům na hřišti.
Celý mantinel hracího pole a tedy pracovní prostor robotu, lze aproximovat obdélníkem. Funkce Z_{2M} se tedy bude skládat ze složek odpovídajících vlivu přímek, které byly získány protažením hran zmíněného obdélníku. Protože je nutné, aby funkce Z_{2M} rostla, bude-li se robot blížit k těmto hraničním přímkám, jako vhodný se jeví tvar

$$Z_{2M} = c_1 \left(\frac{1}{x - A_x} - \frac{1}{x - B_x} + \frac{1}{y - A_y} - \frac{1}{y - B_y}\right), \tag{3.3}$$

kde c_1 je konstanta udávající vliv složky Z_{2M} na funkci Z. Bod A je levý dolní roh pracovního prostoru a B je pravý horní roh pracovního prostoru.

Funkce Z_{2R}, která má popisovat vliv ostatních robotů, se bude skládat z příspěvků Z_{2R_j} konkrétních hráčů pohybujících se po hrací ploše a bude tedy mít tvar

$$Z_{2R} = \sum_{j=1}^{N} Z_{2R_j}. \tag{3.4}$$

Směr záporného gradientu funkce Z_{2R_j} musí být orientován od středu P překážky R_j a vliv každé překážky musí být od určité vzdálenosti zanedbatelný. Jako vhodná funkce splňující všechny výše uvedené předpoklady se ukázala exponenciální funkce ve tvaru

$$Z_{2R_j} = c_2 \exp\left(-c_3\left((x-P_x)^2 + (y-P_y)^2\right)\right). \tag{3.5}$$

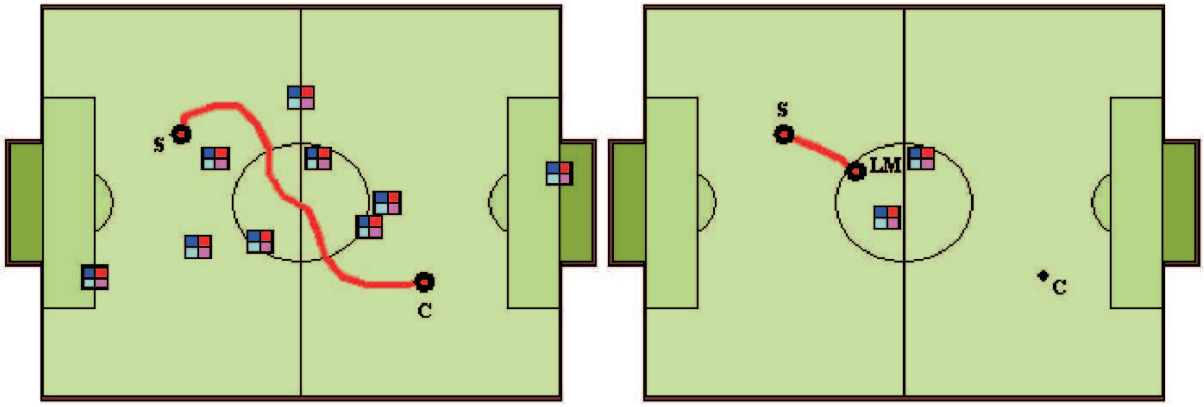
Sestavíme-li všechny příspěvky tvořící výsledné potenciálové pole a aplikujeme-li rovnici 3.1, získáme požadovaný směr pohybu robotu jednoduchým dosazením aktuální pozice robotu do následující rovnice.

$$\begin{aligned}\varphi(x,y) = -grad(Z_1(x,y) + Z_2(x,y)) = \frac{(x-C_x; y-C_y)}{\sqrt{\left((x-C_x)^2 + (y-C_y)^2\right)}} + \\ c_1\left(-\frac{1}{(x-A_x)^2} + \frac{1}{(x-B_x)^2} - \frac{1}{(y-A_y)^2} + \frac{1}{(y-B_y)^2}\right) - \\ 2c_2c_3\sum_{j=1}^{N}\exp\left(-c_3\left((x-P_x)^2 + (y-P_y)^2\right)\right)(x-P_x; y-P_y)\end{aligned} \tag{3.6}$$

Na obrázku 3.2 vlevo je zobrazena trajektorie, kterou by robot jel, pokud by dané překážky zůstaly statické. Tato trajektorie byla získána simulací pohybu vždy o konstantní skok ve směru daném funkcí 3.6, dokud se robot takto nedostal do blízkosti požadovaného bodu C nebo do lokálního minima. Situace, při které robot uvízne v lokálním minimu, je na obrázku 3.6 vpravo.

3.1.2 Globální metody

Mezi globální metody řadíme ty algoritmy, které při plánování uvažují celý pracovní prostor robotu. Nalezená cesta by měla spojovat aktuální pozici robotu a požadovanou pozici, případně region kolem této pozice. Znalost kompletního plánu robotu je esenciální pro některé vyšší rozhodovací algoritmy. Zakomponování globální informace o podobě mapy, ve které se robot

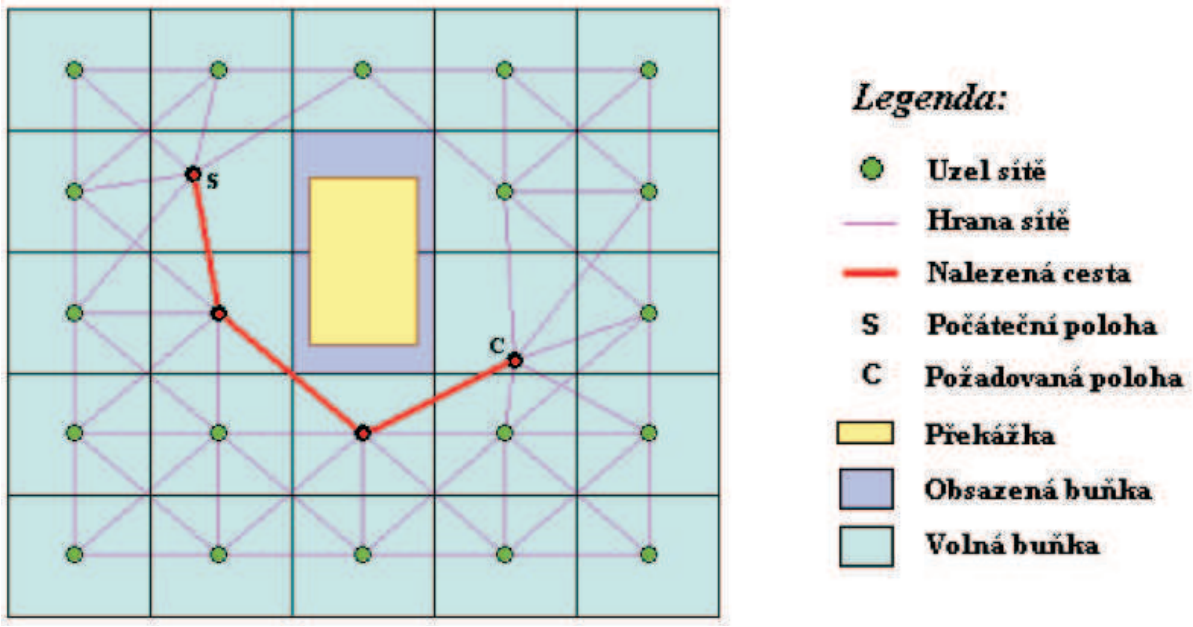


Obrázek 3.2: Vlevo: potenciálové pole v situaci s 9-ti překážkami. Vpravo: problém lokálního minima.

pohybuje, obecně vede k lepšímu výsledku, ale za cenu delšího výpočetního času nutného pro nalezení řešení.

Mřížka obsazenosti

Jako jeden z nejjednodušších algoritmů, které lze řadit do třídy globálních plánovacích metod, je uváděna takzvaná mřížka obsazenosti (dále jen OG, z anglického Occupancy Grid). Základem tohoto algoritmu, který je popsán například v [3], je vhodně rozdělit celý pracovní prostor robotu na disjunktní oblasti (buňky). Tvar těchto buněk bývá nejčastěji čtverec či trojúhelník. Každé buňce je přiřazena hodnota 0, pokud se v ní nalézá překážka, nebo hodnota 1, pokud je taková buňka volná. Hodnota 0 je přiřazena také buňkám, které obsahují body odpovídající počáteční poloze robotu S a jeho požadované poloze C. Dále jsou nalezeny středy těch oblastí, jež mají hodnotu 1. Tyto body tvoří uzly grafu společně s body S a C. Hrany grafu jsou získány spojením středů buněk, pro něž platí, že mají společný alespoň jeden bod hranice a zároveň jím byla přiřazena hodnota 1. Navíc jsou součástí grafu hrany mezi bodem S (resp. C) a středy buněk, které sousedí s buňkou obsahující bod S (resp. C). Počáteční uzel grafu OG tedy odpovídá středu robotu, pro kterého je cesta plánovaná. Obdobně konečný uzel grafu OG představuje bod, do něhož se robot má dostat, jak je vidět na obrázku 3.3. Nejkratší cestu v takto získaném grafu lze jednoduše nalézt pomocí libovolného algoritmu pro hledání nejkratší cesty v grafu.
Nevýhodou tohoto přístupu je možnost existence situace, ve které nebude nalezeno řešení, přestože existuje. Toto riziko klesá, bude-li zmenšena velikost jednotlivých buněk, jak ukazuje obrázek 3.4, který opět demonstruje použití algoritmu v aplikaci robotického fotbalu. Samozřejmě při rostoucí hustotě mřížky roste počet uzlů a tedy i hran, což má za následek růst výpočetní

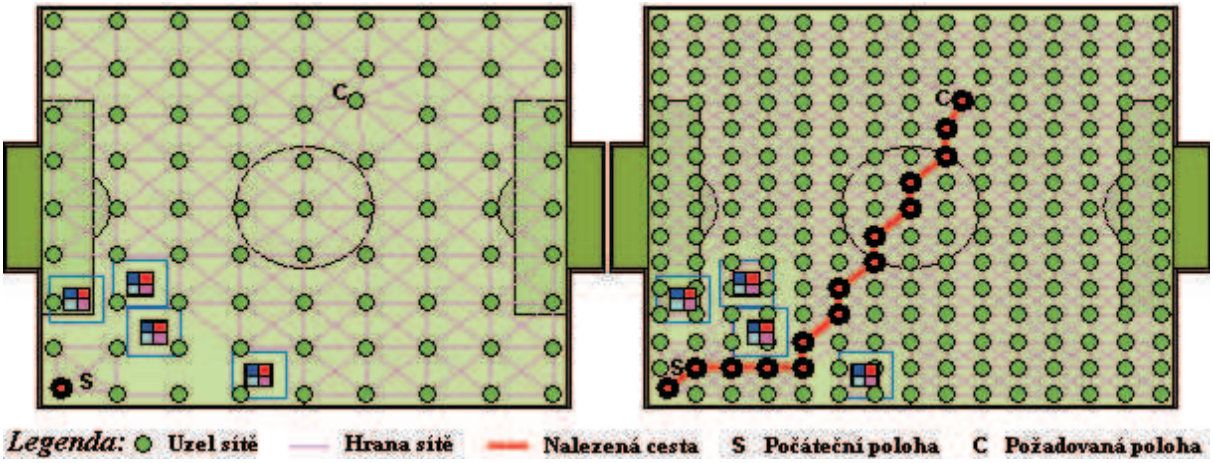


Obrázek 3.3: Mřížka obsazenosti využita při řešení problému s jednou obdélníkovou překážkou.

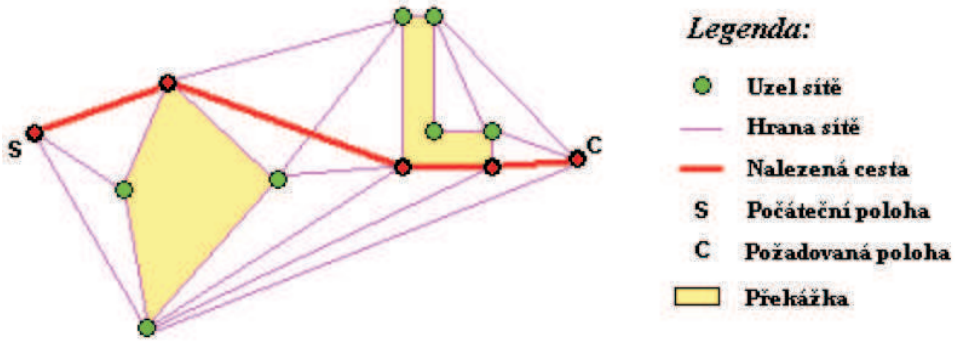
složitosti algoritmu. Aby algoritmus fungoval optimálně, měla by být velikost jednotlivých buněk alespoň srovnatelná s velikostí překážek, což odpovídá obrázku 3.4 vpravo.

Graf viditelnosti

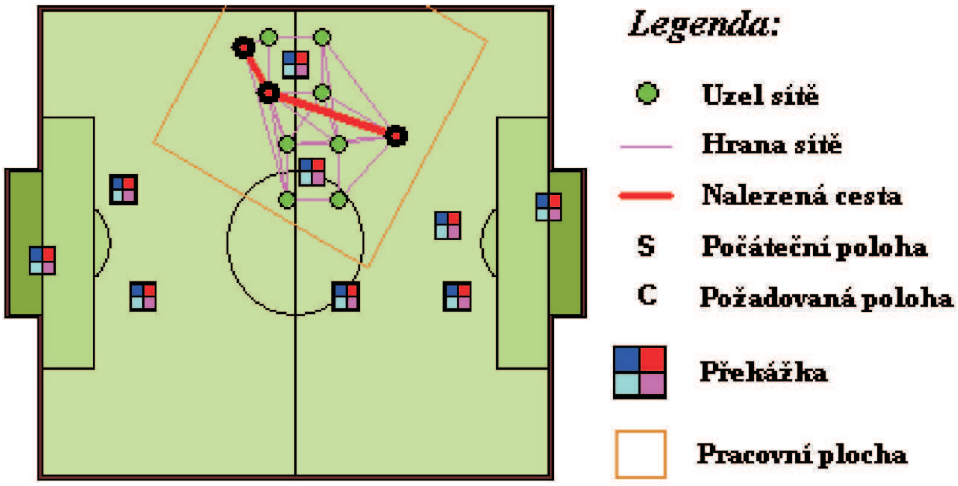
Nevýhodu mřížky obsazenosti ilustrovanou v obrázku 3.4 dokáže odstranit graf viditelnosti (dále jen VG, z anglického Visibility Graph). Tento algoritmus, uvedený například v [13], řešení nalezne vždy, pokud existuje. Překážky při použití VG mohou mít tvar libovolného polygonu. V reálném prostředí je tedy nutné mapu pracovního prostoru robotu předzpracovat a obecný tvar překážek aproximovat polygony. Toto omezení přináší výpočetní krok navíc a tedy i jisté zpoždění při použití složitých map, nicméně neomezuje aplikaci algoritmu jako takového. VG je podobně jako mřížka obsazenosti založena na hledání nejkratší cesty v grafu. Uzly grafu viditelnosti tvoří počáteční bod S, požadovaný bod C a vrcholy všech polygonů aproximujících jednotlivé překážky. Hrany grafu jsou všechny takové, které spojují jednotlivé uzly a pro které platí, že žádná jejich část není vnitřním bodem výše zmíněných polygonů, jak je vidět na obrázku 3.5. Nejkratší cesta spojující počáteční a konečný uzel, tvořená hranami grafu, může být opět nalezena pomocí některého ze známých grafových algoritmů.
Podobně jako u předchozích metod nyní ukážeme aplikaci VG v robotickém fotbalu. Jelikož robotický fotbal je vysoce dynamická aplikace vyžadující



Obrázek 3.4: Vlevo: příliš malá hustota mřížky – řešení nenalezeno. Vpravo: dostatečná hustota mřížky.



Obrázek 3.5: Graf viditelnosti při řešení problému se dvěma obecnými překážkami.

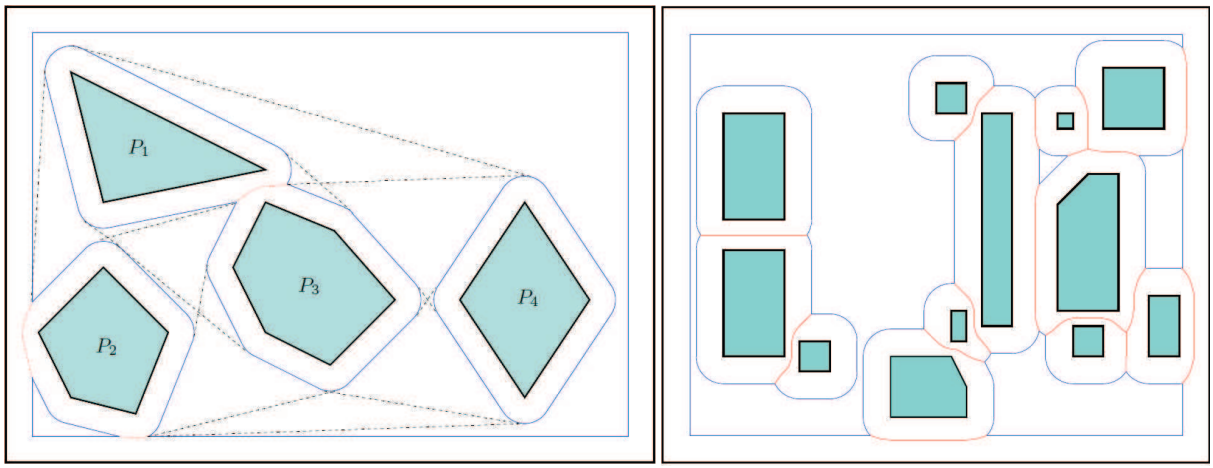


Obrázek 3.6: Aplikace grafu viditelnosti pro situaci v robotickém fotbale.

velmi časté a rychlé přeplánování, v řídícím systému týmu G-Bot byla použita modifikace VG redukující velikost výsledného grafu. Pracovní plocha robotu je v této modifikaci zmenšena pomocí obdélníku opisujícího elipsu, v jejichž ohniscích bude ležet počáteční a konečný bod pohybu robotu, jak je vidět na obrázku 3.6. Všechny uzly ležící mimo tuto plochu nebudou do algoritmu uvažovány, protože je malá pravděpodobnost, že pohyb robotu v robotickém fotbale ovlivní. Počet hran sítě a tedy i potřebný výpočetní čas se tak výrazně redukuje.
Největší nevýhodou tohoto algoritmu je jeho snaha vést trajektorii co nejblíže k jednotlivým překážkám. To je v případě reálných aplikací nevýhodné, protože pohyb robotů z principu nemůže být úplně přesný díky chybám senzorů i aktuátorů a roste tak nebezpečí kolize s překážkami. Řešením by bylo „nafouknout“ jednotlivé překážky zvětšením oblasti kolem nich, která je pro roboty uvažována jako zakázaná. Cesta by potom mohla vést ve větší vzdálenosti, což by ale opět vedlo k možnosti nenalezení řešení i v případech, kdy existuje. Tento problém řeší takzvaný „Visibility-Voronoi complex“ (VVC) [25] kombinující výhody VG diagramu a algoritmu založeného na Voroného diagramu, který bude popsán v následující kapitole. Podrobnější popis algoritmu VVC přesahuje rozsah tohoto učebního textu a čtenáře tak musíme odkázat na článek [25], kde byla metoda publikována. Zde pro motivaci uvedeme pouze obrázek 3.7 převzatý z této publikace.

Voroného diagram

Při popisu tohoto, v mobilní robotice často používaného, algoritmu vyjdeme z jeho definice v [29]. Voroného diagram je síť, která rozděluje plochu na regiony vlivu množiny bodů $P = p_1, p_2, \cdots, p_n$. Pro všechny body x takového

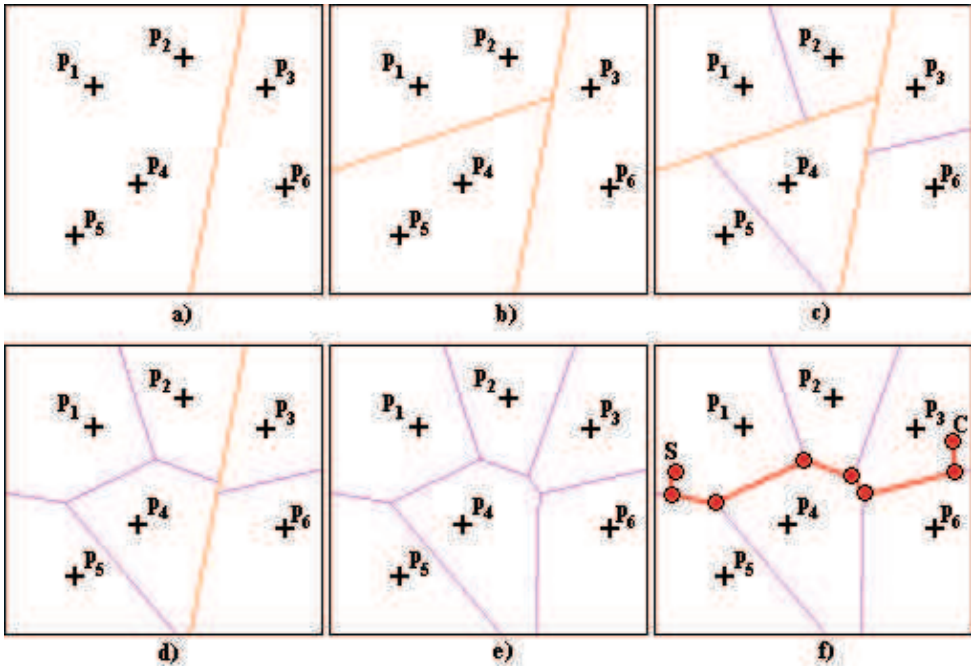


Obrázek 3.7: Algoritmus kombinující výhody grafu viditelnosti a Voroného diagramu. Zdroj: [25].

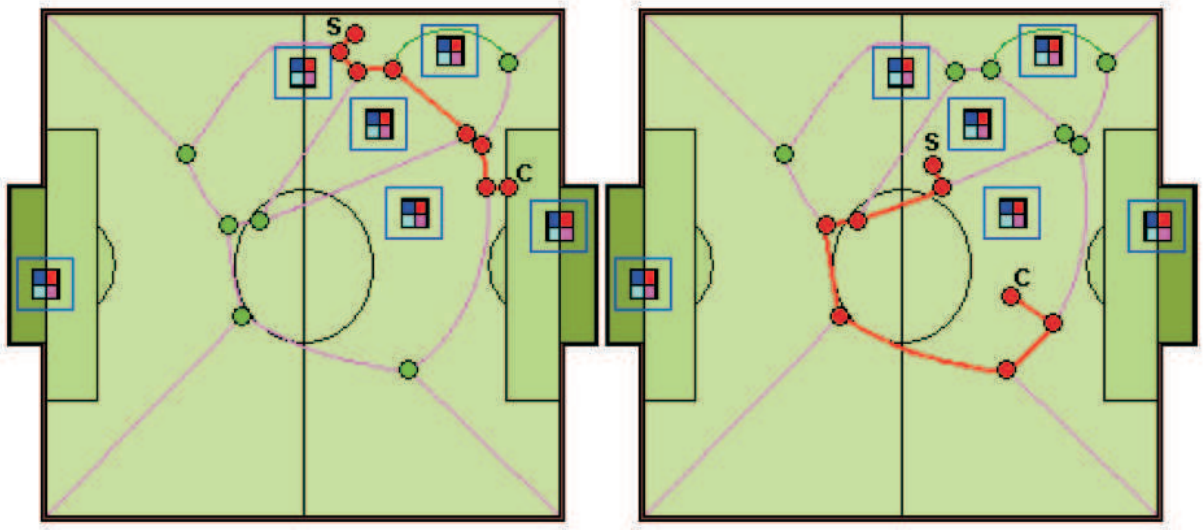
regionu $R(p_i)$ potom platí

$$V(p_i) = x : |x - p_i| \leq |x - p_j|, \tag{3.7}$$

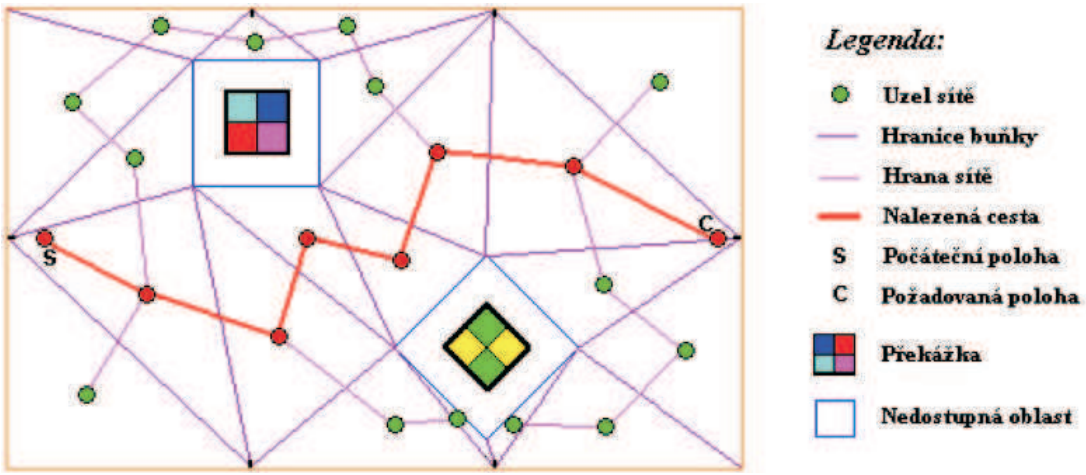
kde p_j jsou všechny body množiny $\{P - p_i\}$.
Existuje spousta metod konstrukce Voroného diagramu. Zde bude popsána metoda rozděl a panuj, jejíž detailnější popis lze nalézt například v [5]. V prvním kroku algoritmu, který odpovídá obrázku 3.8 a), je množina P rozdělena na lineárně separabilní podmnožiny. Tento krok je opakován, dokud existuje nerozdělená podmnožina s počtem prvků větším než dva. Tato situace je znázorněna na obrázku 3.8 b). Pro tyto maximálně dvouprvkové podmnožiny se sestrojí Voroného diagram, který je redukován na jednu přímku, obrázek 3.8 c). Nyní jsou podmnožiny opět skládány. Jednotlivé diagramy jsou spojeny pomocí přímek odpovídajících osám spojnic, které propojují body z obou podmnožin. Tento krok je ilustrován na obrázku 3.8 d) a e). V posledním kroku algoritmu, který je znázorněn na obrázku 3.8 f), je prohledán výsledný graf. Jeho uzly odpovídají společnému bodu tří sousedních regionů vlivu, počátečnímu bodu pohybu robota S, cílovému bodu C a bodům diagramu, které leží nejblíže bodům S a C. Hrany tohoto grafu jsou hranice sousedních regionů vlivu a nejkratší možné úsečky připojující body S a C k diagramu.
Z definice Voroného diagramu je zřejmé, že tento přístup se snaží nalézt cestu co nejvíce vzdálenou od překážek, za cenu jejího prodloužení. Je tedy vhodný převážně pro aplikace, kde je vzdálenost mezi jednotlivými překážkami jen o málo větší než je velikost robota, který má takovým prostorem projet. V robotickém fotbalu je tato podmínka splněna jen velmi zřídka a proto se algoritmus pro tuto aplikaci ukázal jako nevhodný (viz. obrázek 3.9).



Obrázek 3.8: Metoda rozděl a panuj použitá při tvorbě Voroného diagramu.



Obrázek 3.9: Voroného diagram použitý na dvě situace v robotickém fotbale.



Obrázek 3.10: Trojúhelníková dekompozice s uzly grafu, které odpovídají těžišti trojúhelníků.

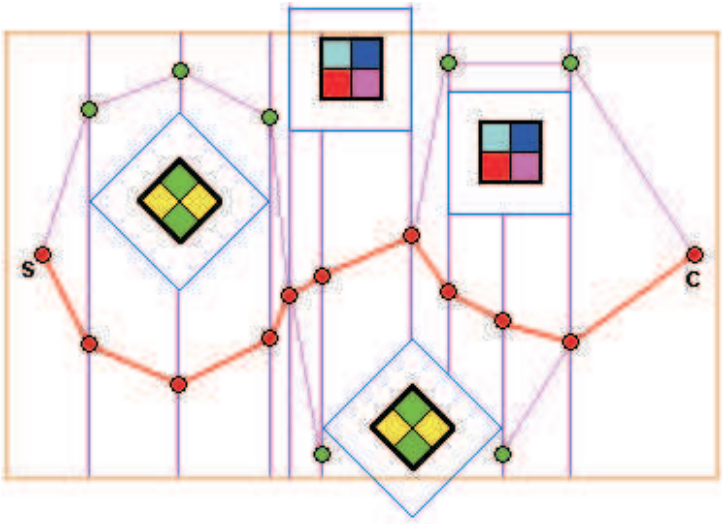
Buněčná dekompozice

Do této skupiny, v anglické literatuře citované jako „Cell Decomposition“, patří velké množství algoritmů, jejichž základní myšlenkou je rozdělit část pracovního prostoru robotu, která neobsahuje překážky, na množinu disjunktních oblastí a výslednou cestu hledat v nich. Na rozdíl od algoritmu mřížka obsazenosti, kde byl disjunktně rozdělený celý pracovní prostor robotu nezávisle na existenci překážek, zde tvar překážek ovlivňuje vlastní rozdělení volného prostoru, ve kterém se robot může pohybovat.

- **Trojúhelníková dekompozice**

Jak napovídá název, pracovní prostor robotu je při běhu tohoto algoritmu rozdělen na trojúhelníky. Prvním krokem algoritmu je nalezení množiny všech vrcholů jednotlivých trojúhelníků. Tato množina, kterou budeme dále značit V, se skládá z vrcholů polygonů aproximujících jednotlivé překážky. Jak je možné vyčíst z obrázku 3.10, jednotlivé trojúhelníky vznikají spojováním bodů množiny V následovně. Nejprve je pro každou hranu polygonu překážky nalezen bod z množiny V, pro který platí, že je jeho vzdálenost od středu této hrany minimální. Trojúhelník je potom tvořen tímto bodem a krajními body hrany. Takto vzniklé trojúhelníky rozdělí pracovní prostor na množinu polygonů. V dalším kroku jsou polygony stupně většího než tři dále děleny vždy pomocí nejkratší dosud nepoužité spojnice vrcholů tohoto polygonu. Uzly sítě, ve kterých bude hledána nejkratší cesta, odpovídají těžišti jednotlivých elementů rozdělení volného prostoru. Hrany jsou takové spojení dvou sousedních elementů, které neobsahují bod překážky.

- **Lichoběžníková dekompozice**



Obrázek 3.11: Cesta nalezená pomocí lichoběžníková dekompozice.

Při sestavování tohoto algoritmu, jsou nejprve nalezeny vrcholy všech polygonů, které aproximují jednotlivé překážky. Každým takovým bodem je vedena přímka kolmá na spojnici počátečního bodu pohybu S a požadované polohy robota C, jak je patrné na obrázku 3.11. Body překážek rozdělí úsečku, kterou z přímky vytnou hranice překážek, na části, které žádné body překážky neobsahují. Jednotlivé oblasti, na které je pracovní plocha rozdělena, tvoří lichoběžníky s hranami, které odpovídají částem přímek vymezených překážkami. Nejkratší cesta z S do C je hledána v grafu, jehož vrcholy tvoří středy hran těchto oblastí.

Rychle náhodně rostoucí stromy

Vznik algoritmu rychle náhodně rostoucích stromů (dále jen RRT, z anglického Rapidly-exploring Random Tree) byl motivován potřebou *kinodynamického* plánování trajektorií *neholonomních* robotů, který by navíc umožňoval i plánování v prostorech vyšších dimenzí. To je důležité, chceme-li například získat plán pro manipulátor s vysokým stupněm volnosti. Kinodynamické plánování se snaží uvažovat kinematická omezení, kterými mohou být například mechanická omezení kloubů u manipulátoru nebo poloměr otáčení u *car-like* robotu, a dynamická omezení, jako jsou omezení rychlosti nebo zrychlení. Základním problémem je tedy najít takovou trajektorii robotu, která začíná v počátečním stavu, končí ve vzdálenosti menší než zvolený parametr ε od cílového stavu a splňuje všechna požadovaná kinodynamická omezení.
Algoritmus RRT je nejnovější z metod uváděných v našem výběru a patří k nejnovějším šířeji používaným metodám v robotice vůbec. Jeho podrobný popis spolu s výčtem možných aplikací je uveden v monografii [14]. V této

knize navíc čtenář může najít ucelený popis dalších plánovacích algoritmů a lze ji doporučit jako vhodný doplňkový materiál k tomuto textu. Zde se omezíme na základní popis principu RRT a krátký souhrn jeho výhod a nevýhod.

Technika RRT vytváří stromovou datovou strukturu reprezentující jednotlivé stavy robotu (u mobilního robotu lze stavem chápat jeho polohu v souřadném systému a natočení). Základním principem techniky RRT je vygenerování náhodného stavu x_{rand}, k němuž je snahou rozšířit strom. Je-li nalezen stav, kterým je možné strom rozšířit, je po přidání do stromu změřena jeho vzdálenost od cílového stavu x_{goal}. V případě, že je tato vzdálenost menší než zvolené ε, podařilo se nalézt výslednou trajektorii, která je ze stromu snadno rekonstruovatelná. Pokud je tato vzdálenost větší než zvolené ε, následuje opakování generování náhodného stavu a rozšiřování stromu. Stavy ve stromu musí ležet ve volném prostoru mimo překážky. Zároveň přechod z jednoho stavu do následujícího stavu stromu musí respektovat příslušná omezení pohybu robotu (rychlosti, zrychlení). U této techniky je obtížné, aby výsledná trajektorie vedla z počátečního stavu x_{init} přesně do cílového x_{goal}, proto vhodná volba tolerančního pásma ε zvyšuje rychlost nalezení cesty.

Algoritmus 3.12 naznačuje základní princip RRT techniky. Kořen stromu je vytvořen z počátečního stavu x_{init}. Po této inicializaci následuje hlavní cyklus algoritmu RRT. Počet iterací tohoto cyklu může být omezen jeho maximální hodnotou nebo časovým limitem, ve kterém musí být plánovací úloha vyřešena. Druhý případ bývá součástí plánování v reálném čase, kde musí být plánování provedeno v určitém časovém intervalu. Funkce *growTree* vybere takový stav, který je možný expandovat z existujícího stromu a zároveň je nejblíž k náhodně generovanému stavu x_{rand} a připojí jej ke stromu.

Největší výhoda RRT algoritmu spočívá v možnosti nalézt relativně rychle platné řešení úlohy plánování pohybu tělesa v prostoru vysoké dimenze. RRT metoda je dobře uplatnitelná při hledání plánu v prostoru s vysokým výskytem lokálních minim, jelikož postupně pokrývá celou prohledávanou oblast, dokud nedosáhne hledaného cíle. Nevýhodou RRT je, že nalezení řešení nemůže být ze své podstaty optimální a ve většině praktických příkladů se od optimálního řešení velmi liší. Při použití RRT v mobilní robotice je často nutné výsledek dále upravit vhodnou optimalizační metodou, jinak nelze očekávat dostatečně hladký pohyb robotu.

Algoritmus 1: RRT

Vstup: Počáteční stav x_{init}, cílový stav x_{goal}.
Výstup: Výsledná trajektorie pokud je nalezena.

```
T = treeInit(x_init)   // inicializace stromu; z x_init je vytvořen kořen stromu;
for i = 0 to max do
    x_rand = getRandomState()          // generování náhodného stavu;
    x_new = growTree(T, x_rand);
    if x_new ∧ δ(x_new, x_goal) < ε then // δ vypočítá vzdálenost mezi dvěma stavy v používané metrice
        return extractSolution(x_new)   // extrahuje ze stromu výslednou trajektorii;
return failed;
```

Obrázek 3.12: Základní myšlenka funkce RRT algoritmu.

3.2 Multi-robotické systémy

Systémy několika spolupracujících robotů se staly předmětem intenzivního výzkumu v uplynulých desetiletích a je tomu tak dodnes. Obecně větší počet robotů je vhodné použít v aplikacích jejichž řešení pouze jedním robotem by bylo nemožné nebo příliš nákladné. Mimo to, několikanásobný distribuovaný systém zvyšuje robustnost použití díky možné redundanci. Konečně, většinu robotických úkolů lze vhodně dekomponovat tak, že použití více robotů sníží celkový čas řešení. Navíc byla ukázána třída aplikací, kde celkový čas plnění úkolu klesá víc než lineárně s rostoucím počtem nezávisle se pohybujících robotických platforem.

Výzkum inteligentních mobilních robotů zahrnuje robotické systémy stovek autonomních vozidel, ale i například malé skupiny robotů pohybujících se ve formacích přesně definovaných tvarů. Jako jeden z prvních reálných systémů umožňujících studium chování sta úzce kooperujících robotů je uváděn projekt ALLIANCE [20], který vznikal ke konci dvacátého století v USA. Ale ani Evropský výzkum v této oblasti nezůstává pozadu. V současné době týmy předních Evropských univerzit, zabývajících se výzkumem inteligentní robotiky (součástí konsorcia je i ČVUT v Praze) , studují principy autonomního vzniku složitějších robotických struktur ze stovek samostatně se pohybujících jednoduchých robotů v rámci projektů REPLIKATOR & SYMBRION [16]. Tyto dva projekty lze považovat za vlajkové lodě výzkumu mobilní robotiky v Evropské Unii. Navíc oba projekty svým zaměřením na výzkum evolučních principů přesahují oblast mobilní robotiky a dají se považovat za mezidisciplinární. Jako další z oblastí zkoumajících interakce ve velkých

skupinách robotů můžeme jmenovat například vznikající systémy pro řízení autonomních vozidel na dálnici. My se v tomto úvodu do mobilní robotiky zaměříme na segment multi-robotických systémů řešících koordinaci pohybu autonomních robotů ve formaci.

3.2.1 Formace inteligentních robotů

Zkusme si nejdříve položit otázku, proč vlastně udržovat mobilní roboty ve formaci. Pomineme-li skutečnost, že roboti tvořící během svého pohybu předem zadané obrazce působí esteticky a pro nezasvěcené i inteligentně, koordinovaný pohyb robotů umožní provádět úkony, které jsou samostatně se pohybujícím robotem neuskutečnitelné. Jako klasická aplikace formací mobilních robotů se udává takzvané „cooperative box-pushing“. V této úloze se dva nebo více robotů koordinovaně pohybují ve formaci s cílem společně dotlačit daný předmět z místa A do místa B. Hmotnost předmětu a typ robotů jsou zvoleny tak, aby jeden robot nebyl schopen zadaný úkol provést a naopak, aby společně pracující roboty měly pro daný úkol dostatečný výkon. Podobnou aplikací je i kooperativní transport nákladu, jehož rozměry nebo hmotnost přesahují přepravní kapacitu robotu. V obou těchto případech rozměry přepravovaného předmětu určují tvar formace a nezbytný počet robotů. Další velkou skupinou aplikací autonomních formací je takzvané „cooperative sweeping“. Tento název slučuje všechny typy úloh, ve kterých má formace robotů za úkol pokrýt svými efektory kompletně celý pracovní prostor. Kromě všech typů uklízecích robotů do této kategorie řadíme také autonomní zemědělské stroje, sekačky trávníků nebo vozidla pro údržbu pozemních komunikací. Použití formací při řešení těchto úkolů je motivováno nutností zrychlit prováděný proces, případně sekvenčně použít různě vybavené roboty (například první robot sbírá objemnější odpadky, druhý robot zametá, třetí vytírá a čtvrtý leští podlahu). Tvar formace je zde ovlivněný tvarem pracovního prostředí a prováděným úkolem. Počet robotů je určen požadovanou dobou řešení úlohy.
Z dalších možností použití formací autonomních robotů nesmíme opomenout vojenské aplikace. Kromě konvojů transportních vozidel, kde robotické transportéry autonomně následují vedoucí vůz řízený lidskou posádkou, se formace robotů využívají k obraně. Dobře pancéřované autonomní stroje v takové úloze obklopují zranitelnější vozidla ve středu formace. Podobné principy, které se používají pro řízení transportů vojenských vozidel, byly úspěšně odzkoušeny i v civilní praxi. Již v roce 1991 byla v rámci projektu PATH v San Diegu zkonstruována dálnice umožňující autonomní pohyb vozidel ve formaci. V poslední době se nicméně upouští od stavby speciální in-

frastruktury nutné pro autonomní pohyb automobilů a přední automobilky se zaměřují na vývoj senzorů umístěných přímo na palubě sériově vyráběných vozů.
Formace robotů nicméně nejsou omezeny pouze na pozemní operace. Například bezpilotní letadla létají ve formaci kvůli snížení spotřeby paliva nebo během tankování za letu. Podobné principy řešící let dvou objektů v těsné blízkosti se úspěšně používají i ve vesmíru za účelem spojování kosmických těles. Další typy formací, které byly studovány a některé i otestovány ve vesmíru, jsou „trailing formations“ a „cluster formations“. Prvně zmiňovaný typ formací je tvořen několika satelity, které se pohybují po stejné orbitě a mapují jedno místo povrchu v různém čase, případně pod jiným úhlem. Tento přístup umožňuje sledovat dynamické procesy (lesní požáry, hurikány apod.) s frekvencí vyšší než je frekvence oběhu satelitu. Mapování pod různým úhlem navíc umožňuje získat 3D informaci skenovaného povrchu. Druhý typ formací tvoří skupina satelitů v kompaktnějším uspořádání a používá se například pro přesnou interferometrii.
V našem stručném výčtu nejrůznějších aplikací jsme již zmínili formace na zemi, ve vzduchu i ve vesmíru a zbývá nám tedy ještě vodní hladina a podvodní prostor. V těchto oblastech se formace mobilních robotů používají především pro monitoring. Například v rámci projektu „Autonomous oceanographic sampling networks“ byla formace autonomních miniponorek použita pro studium vodních proudů. Z dalších projektů můžeme jmenovat monitorování znečištění v Monterey Bay pomocí formace autonomních plavidel, případně detekci hranic ropných skvrn pomocí dvou kooperujících robotických lodí.
Pokusme se nyní podívat na formace mobilních robotů ne z pohledu možných aplikací, ale z pohledu metod, které jsou použité pro stabilizaci robotů ve formaci. V literatuře se můžeme setkat se třemi hlavními skupinami takových metod: „leader-follower“, „behavior based“ a „virtual structure“.

3.2.2 Formace používající metodu Leader-Follower

V této skupině metod jsou jeden případně i několik robotů vybrány jako lídři a ostatní roboty jsou přiřazeny jako jejich následovníci [23]. V průběhu pohybu formace tito následovníci udržují předem definovanou pozici vzhledem k jejich lídrovi, což zajistí požadovaný tvar formace. V literatuře se můžeme setkat se dvěma přístupy udržování takto definovaných formací. První přístup předpokládá znalost polohy jednotlivých robotů v mapě, která je sdílená v rámci formace pomocí bezdrátové komunikace. Na základě této informace jednotliví následovníci upravují svůj pohyb tak, aby byl dosažen a udržován požadovaný tvar formace. Druhý z přístupů nevyžaduje nutnost

komunikace a následovníci získají znalost o relativní poloze svého lídra pomocí senzorů. Často se například používá detekce a trekování identifikačního obrazce na vedoucím robotu pomocí kamery na následovníkovi nebo detekce obrysů robotu dálkoměrným senzorem. U více než dvoučlených formací se lze setkat s přístupy využívajících pouze jednoho lídra, k jehož pozici je dopočítáván pohyb všech ostatních členů formace. Tento centralizovaný přístup je však náchylný k poruše lídra, což vede k selhání celé formace. Proto se častěji používá takzvaný virtuální lídr. Virtuálním lídrem je myšlený bod v prostoru, jehož pohyb je simulován a který je použit pro koordinaci všech robotů v týmu. Další z možností koordinace rozsáhlých formací je použití několika lídrů sekvenčně řazených za sebou. Některé z robotů jsou v takových metodách uvažovány zároveň jako lídři i následovníci a informace o poloze a pohybu vůdčích robotů je propagována formací postupně.
Aplikace metod založených na leader-follower principu jsou často motivovány použitím heterogenní skupiny vozidel sestávající se z jednoho či několika dobře vybavených robotů a z několika jednodušších (a tedy i levnějších) robotů, kteří nemají dostatek senzorů umožňujících samostatný pohyb. Aplikace těchto metod pro koordinaci homogenních formací stejně vybavených robotů často předurčuje podstata úlohy vyžadující kopírování pohybu vozidel jedoucích vpředu. Zde lze zmínit například konvoje vozidel, ale i formace letounů pohybujících se v zákrytu, který šetří palivo díky nižšímu odporu vzduchu, nebo páry satelitů mapujících dané území sekvenčně v čase. Na závěr bychom chtěli zdůraznit výhody a nevýhody použití leader-follower principu.

+ Možnost analyzovat stabilitu pohybu formace a kovergenci robotů do požadovaného tvaru.

+ Vysoká přesnost udržování formace.

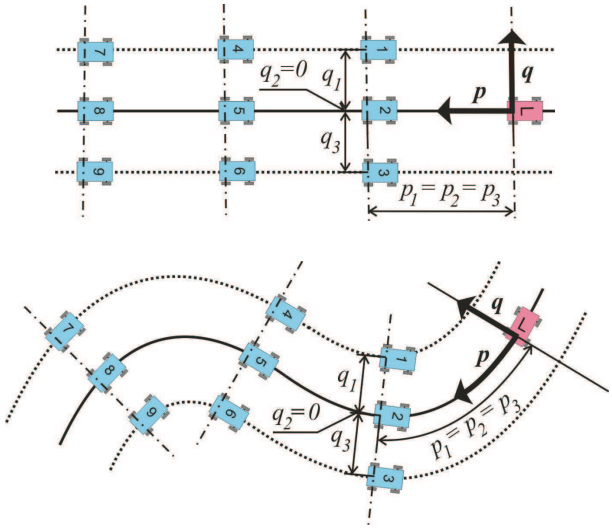
+ Snadná predikovatelnost pohybu jednotlivých robotů dodržujících daný rámec pohybu.

- Nižší robustnost daná v případě použití reálného vůdčího robotu.

- Obtížně použitelné pro koordinaci velkých skupin robotů.

3.2.3 Formace řízené behaviorálními pravidly pohybu jejich členů

Druhá skupina metod používaných pro pohyb formací je založena na slučování různých vzorů chování, které vedou ke splnění jednotlivých dílčích úkolů,



Obrázek 3.13: Ekvivalentní formace, které jsou definované pomocí křivočarých souřadnic p a q a následují různé cesty pohybu lídra.

do výsledného řízení robotů [15]. Hledané řešení je většinou získáváno jako součet několika pravidel chování, které jsou váženy podle jejich důležitosti. Chování robotu, které má za úkol jeho ochranu před zničením, má většinou vyšší prioritu než chování vedoucí k úspoře paliva a podobně. Tyto metody jsou nezřídka motivovány přírodou, například pohybem hejn ptáků, školek ryb, rojů hmyzu nebo molekul formujících krystal. Výhodou je jejich snadná implementace, ale je obtížné provádět matematické analýzy pohybu takto řízené formace a predikovat její výsledné chování. Jelikož je výsledný pohyb formace formován dílčími příspěvky pohybu jednotlivých robotů, je zpětná vazba od členů skupiny promítnuta do jejího chování okamžitě a formace tak dokáže rychle reagovat na změny v okolním prostředí.
Tyto předpoklady předurčují aplikace zmíněných principů převážně na řízení velkých skupin (rojů) jednoduchých robotů, které jsou vybaveny jednoduchými senzory a komunikačními prostředky omezeného dosahu a pohybují se v dynamickém prostředí. Behaviorální pravidla aplikovaná pro stabilizaci formace přináší následující výhody a nevýhody.

+ Velmi jednoduchá pravidla stabilizují i poměrně složité struktury.

+ Složitost řídících algoritmů neroste s počtem robotů.

+ Vhodné pro skupiny s vysokým počtem robotů.

+ Komunikační kanál mezi všemi roboty není nutný.

- Obtížná či nemožná schopnost predikce pohybu jednotlivých robotů.

- Často chybí teorie popisující stabilitu pohybu roje a konvergenci do požadovaného tvaru.

Obrázek 3.14: Behaviorální pravidla pro stabilizaci formací jsou často inspirovány přírodou; hmyzími roji, rybími školkami nebo hejny ptáků.

- Formulace pravidel pohybu jednotlivých entit pro dosažení složitějších vzorců chování formace může být obtížná.

3.2.4 Formace založené na principu virtuálních struktur

Metody používající koncept virtuálních struktur uvažují celou skupinu robotů jako jedno pevné virtuální těleso [22]. Řídící vstupy pro jednotlivé roboty jsou počítány tak, aby formace jako celek sledovala požadovanou trajektorii. Tento proces je ve většině případů prováděn centrálně a výsledek je následně distribuován ke všem členům skupiny. Obecně se tyto metody vyznačují vysokou přesností udržování tvaru formace, a proto jsou typicky používány pro řízení satelitů a v úlohách typu „cooperative box-pushing“. Při transportu předmětů tvar přepravovaného tělesa přímo určuje požadovaný tvar formace. Výhody či nevýhody metod založených na principu virtuálních struktur jsou následující.

+ Možnost dosáhnout požadovaného tvaru formace velmi přesně.

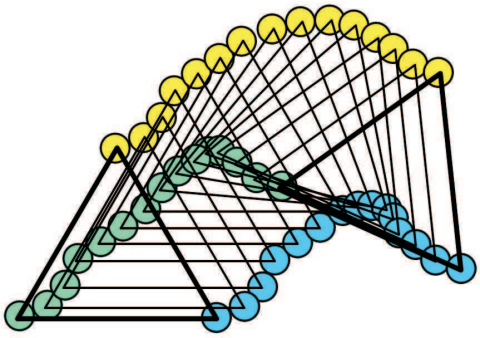
+ Pohyb robotů i formace jako celku je snadno predikovatelný.

- Obtížně formulovatelný pohyb ve formaci pro některé typy neholonomických robotů.

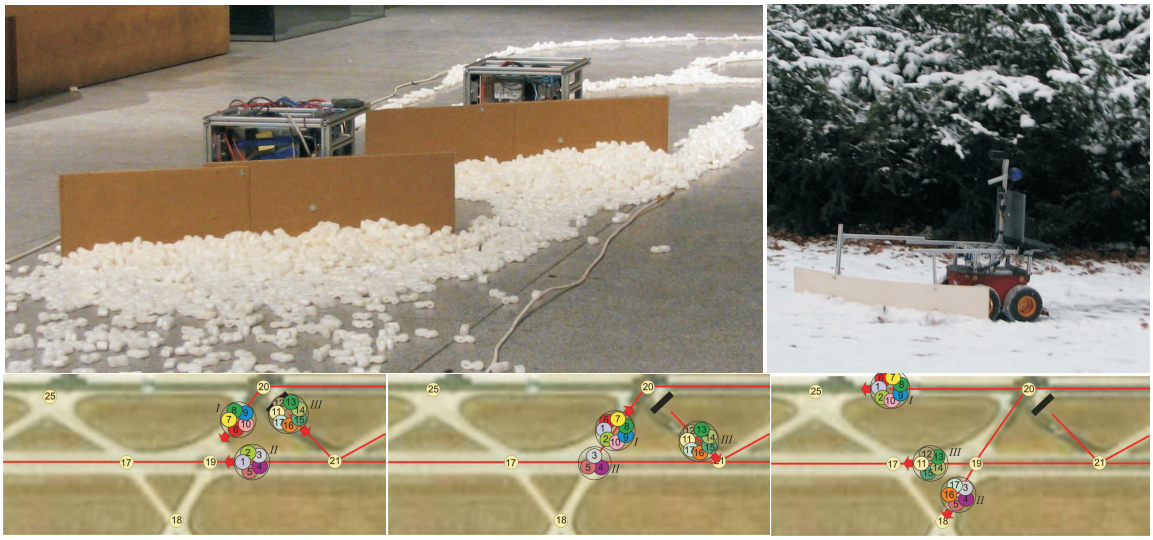
- Nutnost rekonfigurace řídicích schémat při selhání některého z robotů a s tím související nižší robustnost daná použitím centralizovaného konceptu řízení.

- Požadavek na přesnou znalost stavu ostatních robotů, což často vyžaduje komunikaci mezi všemi roboty.

Kromě těchto tří principů používaných pro řízení formací se v mobilní robotice lze setkat s nepřeberným množství jejich modifikací a vzájemných



Obrázek 3.15: Formace založené na principu virtuálních struktur.



Obrázek 3.16: Nahoře - roboty v laboratorním prostředí simulujícím letiště a v reálném prostředí parku. Dole - simulace reakce plánovacího algoritmu na nově detekovanou překážku na letišti ve Frankfurtu.

kombinací. Populární je například kombinovat metodu využívající pohybu lídra vedoucího formaci a principů virtuálních struktur zpětně ovlivňujících pohyb lídra v případě opoždění některého z následovníků [28]. Takto lze do procesu stabilizace formace zahrnout zpětnou vazbu od všech robotů, což klasické leader-follower metody neumožňují.

3.2.5 Formace inteligentních pluhů na letišti

V poslední části naší exkurze do problematiky formací mobilních robotů podrobněji popíšeme aplikaci *uklízení sněhu na letišti pomocí formací autonomních pluhů*, která je v současné době řešena na pracovišti Inteligentní a mobilní robotiky ČVUT v Praze. Na konkrétním příkladě chceme podrobněji ukázat výhody použití formací a dát čtenáři představu o nejdůležitějších komponentech, ze kterých se takový systém skládá.

Myšlenka použití formace pluhů na úklid ranveje letiště není nová. Skoro každou zimu nám televizní stanice předkládají záběry formací velkých letištních pluhů uklízejících zasněženou ranvej. Jak již bylo řečeno v úvodu, formace spolupracujících vozidel vypadá atraktivně, ale hlavní důvod pro puožití formací lze nalézt ve snaze maximalizovat rychlost úklidu a bezpečnost. Formace pluhů pokrývající celou šířku ranveje zvládne její úklid v minimálním možném čase. To umožní pokud možno co nejdříve obnovit provoz letiště. Navíc zasněžená ranvej může být mnohem bezpečnější pro případné nouzové přistání, než částečně uklizená ranvej, která by vznikla při postupném odklízení sněhu nezávisle se pohybujícími pluhy. Nestejnorodá vrstva sněhu a případné sněhové bariéry, které by během postupného odklízení sněhu na povrchu letiště zůstaly, by zvýšily riziko smyku přistávajícího letadla.

Vezmeme-li v úvahu zmíněné požadavky na úklid letiště, jeví se tato aplikace jako ideální pro použití formací autonomních robotů. Letiště jsou již nyní vybavena systémy pro určování přesné polohy vozidel na ploše, což je pro udržování požadované polohy robotů ve formaci nezbytné. Další důležitý prvek mluvící pro použití autonomních robotů je skutečnost, že plocha letiště je během uklízení uzavřena pro běžný provoz. Tím je sníženo množství potenciálních překážek v pracovním prostoru robotů, kterým by se formace musela vyhýbat, což celý systém zjednodušuje. Použití robotů umožní snížit náklady na posádky pluhů, které musí být v pohotovosti 24 hodin denně celou zimu. Navíc, díky možné optimalizaci pohybu robotů po ploše, robotický systém umožní snížit dobu, po kterou je letiště uzavřené.

Při vlastním vývoji takového robotického systému je třeba si uvědomit, že úklidem hlavní ranveje letiště práce robotů nekončí. Jakmile formace dosáhne konce ranveje, pluhy musí být rozděleny do menších formací pro úklid ostatních cest, které obklopují ranvej. Velikost těchto formací i zde musí odpovídat šířce uklízené cesty. Abychom dosáhli optimálního postupu při uklízení sněhu, velikost formace by se měla postupně přizpůsobovat každému novému úkolu. Nadbytečné roboty by měly být přiřazeny ostatním formacím v případě potřeby a naopak nedostatečně velké skupiny by měly být doplněny o volné pluhy. Systém by měl být navíc schopen reagovat na možné změny v pracovním prostoru robotů (obr. 3.16 dole). Jako vhodné pro efektivní a flexibilní alokování pluhů do formací a pro plánování v jakém pořadí budou jednotlivé cesty a ranveje uklízeny se ukázalo použití principů multiagentních systémů, jejichž popis však již překračuje rámec tohoto textu.

Máme-li informaci o požadovaném složení skupiny, lze přistoupit k implementaci vlastního algoritmu pro stabilizaci robotů ve formaci. Pro řízení autonomních pluhů na letišti byla aplikována metoda „leader-follower“, konkrétně

její modifikace využívající takzvané virtuální lídry. V této modifikaci jsou všechny roboty uvažovány jako následovníci a ve formaci není žádný fyzický lídr. Virtuálním lídrem může být zvolen libovolný bod v prostoru a pohyb všech robotů ve formaci je odvozován od pohybu tohoto bodu. Při použití letištních pluhů se jako vhodné jeví určit tvar formace vzdáleností robotů od virtuálního lídra v křivkových souřadnicích p a q, jejichž význam je definován v obrázku 3.13. Známe-li aktuální pozici a orientaci virtuálního lídra, požadovaná pozice a orientace každého z robotů může být získána pomocí následujícího pravidla:

$$\begin{aligned} x_i(t) &= x_L(\tau_i) - q_i \sin(\theta_L(\tau_i)) \\ y_i(t) &= y_L(\tau_i) + q_i \cos(\theta_L(\tau_i)) \\ \theta_i(t) &= \theta_L(\tau_i), \end{aligned} \tag{3.8}$$

Souřadnice p_i a q_i udávají polohu i-tého robota ve formaci. Karteziánské souřadnice $x_L(\tau_i)$ a $y_L(\tau_i)$ a orientace $\theta_L(\tau_i)$určují stav virtuálního lídra v momentě τ_i, kdy byl ve vzdálenosti p_i od své současné pozice.
Pro řízení pohybu virtuálního lídra byla vybrána metoda založená na technice zvané „Model Predictive Control“ (dále jen MPC). Tato metoda umožňuje optimální pohyb formace po ranveji, v zatáčkách i na křižovatkách. Navíc umožňuje optimálně se vyhýbat statickým i dynamický překážkám, ale také plynule slučovat a opět rozdělovat jednotlivé formace. Metoda založená na MPC byla navíc použita i pro řízení jednotlivých robotů ve formaci a jejich stabilizaci ve stavu určeném transformací stavu lídra podle výše uvedeného pravidla (3.8). MPC je vhodný regulátor, který je navíc schopen zabránit kolizím s ostatními roboty formace i s překážkami, kterým se nešlo vyhnout při plánování pohybu lídra. Popis MPC, zvané též jako „receding horizon control“, přesahuje rámec tohoto textu a můžete ho nalézt například v [18].

Literatura

[1] Besl, P., McKay, N. D. A method for registration of 3-d shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, sv. 14, č. 2, s. 239–256, 1992.

[2] Borenstein, J., Koren, Y. The vector field histogram: Fast obstacle avoidance for mobile robots.. *IEEE Journal of Robotics and Automation*, s. 278–288, 1991.

[3] Braunl, T., Tay, N. Combining configuration space and occupancy grid for robot navigation.. *Industrial Robot*, sv. 28(3), s. 233–41, 2001.

[4] Abidi, M. A., Gonzalez, R. C., editors *Data Fusion in Robotics and Machine Intelligence.*, vol. I Academic Press, Inc. (ISBN 0-12-042120-8), Oval Road, London, 1992.

[5] Dwyer, R. A. A faster divide-and-conquer algorithm for constructing delaunay triangulations. *Algorithmica 2*, s. 137–151, 1987.

[6] Elfes, A. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*, 2929 Campus Drive, San Mateo, CA 94403, September 1990. Morgan Kaufmann Publishers, Inc.

[7] Fox, D., Burgard, W., Thrun, S. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, sv. 11, 1999.

[8] Fox, D., Burgard, W., Dellaert, F., Thrun, S. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, s. 343–349, 1999.

[9] Hinkel, R., Knieriemen, T. Environment perception with a laser radar in a fast moving robot. In *Symposium on Robot Control 1988 (SYROCO '88)*, s. 68.1–68.7, Karlsruhe, Germany, October 1988.

[10] Isard, M., Blake, A. Condensation Â— conditional density propagation for visual tracking. *International Journal of Computer Vision*, sv. 29, č. 1, s. 5–28, 1998.

[11] Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, sv. 82, č. Series D, s. 35–45, 1960.

[12] Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, sv. 5, s. 90–98, 1986.

[13] Kunigahalli, R., Russell, J. Visibility graph approach to detailed path planning in cnc concrete placement. *Automation and Robotics in Construction XI*, 1994.

[14] LaValle, S. M. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006.

[15] Lawton, J., Beard, R., Young, B. A decentralized approach to formation maneuvers.. *IEEE Transactions on Robotics and Automation*, sv. 19, č. 6, s. 933—-941, December 2003.

[16] Levi, P., Kernbach, S. *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution.* Springer, 2010.

[17] Lu, F., Milios, E. Robot pose estimation in unknown environments by matching 2d range scans. In *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, s. 935–938, Seattle, WA, 1994.

[18] Mayne, D. Q., Rawlings, J. B., Rao, C. V., Scokaert, P. O. M. Constrained model predictive control: Stability and optimality. *Automatica*, sv. 36, č. 6, s. 789–814, 2000.

[19] Moravec, H. P., Blackwell, M. *Learning sensor models for evidence grids.* Robotics Institute Research Review, Pittsburgh, PA, 1992.

[20] Parker, L. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, sv. 14, č. 2, s. 220–240, April 1998.

[21] Pitt, M. K., Shephard, N. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, sv. 94, č. 446, s. 590Â–631, 1999.

[22] Ren, W., Beard, R. Virtual structure based spacecraft formation control with formation feedback. In *Proc. of AIAA Guidance, Navigation, and Control Conference*, 2002.

[23] Tanner, H., Pappas, G., Kumar, V. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, sv. 20, č. 3, s. 443—-455, June 2004.

[24] Thrun, S. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, sv. 99, č. 1, s. 21–71, 1998.

[25] Wein, R., van den Berg, J. P., Halperin, D. The visibility–voronoi complex and its applications. In *Proceedings of the twenty-first annual symposium on Computational geometry*, SCG '05, s. 63–72, New York, NY, USA, 2005. ACM.

[26] Weiss G., Wetzler, C., von Puttkamer, E. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the International Conference on Intelligent Robots and Systems*, s. 595–601, 1994.

[27] Yamauchi, B., Langley, P. Place recognition in dynamic environments. *Journal of Robotic Systems*, sv. 14, s. 107–120, 1997.

[28] Young, B., Beard, R., Kelsey, J. A control scheme for improving multi-vehicle formation maneuvers. In *Proc. of American Control Conference*, vol. 2, s. 704–709, Arlington, VA, 2002.

[29] Zwynsvoorde, D. V., Simeon, T., Alami, R. Incremental topological modeling using local voronoi-like graphs. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and System (IROS 2000)*, 2000.

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz
info@crr.vutbtr.cz