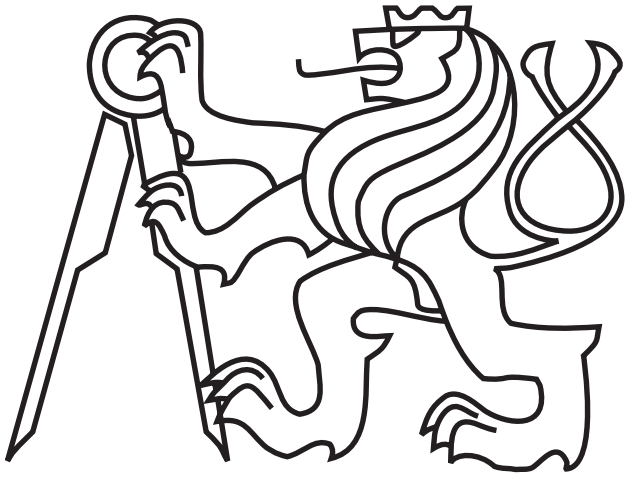
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

BAKALÁŘSKÁ PRÁCE



Roman Bedroš

Modul lokalizace mobilního robotu pro systém Player

Katedra kybernetiky
Vedoucí bakalářské práce: **RNDr. Miroslav Kulich, Ph.D.**

Praha, 2011

Prohlášení

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

V Praze dne 6.1.2011 .....

podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Roman B e d r o š

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Modul lokalizace mobilního robotu pro systém Player

Pokyny pro vypracování:

1. Seznamte se s algoritmem Iterative Closest Point (ICP) a jeho odvozeninami (viz. [1],[2]).
2. Naimplementujte nastudované algoritmy jako moduly pro systém Player.
3. Implementované algoritmy otestujte na datech z repozitáře OpenSLAM.org a na reálném robotu. Výsledky experimentů popište.

Seznam odborné literatury:

[1] Besl, P. and McKay, N.: A Method for Registration of 3-D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 14(2):239-256, February 1992.
[2] Lu, F.; Milios, E.: Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. Journal of Intelligent and Robotic Systems, vol.18, pp.249-275, 1997.

Vedoucí bakalářské práce: RNDr. Miroslav Kulich, Ph.D.

Platnost zadání: do konce zimního semestru 2010/2011

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

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 12. 1. 2010

Abstrakt

Cílem této práce bylo vytvořit modul pro systém Player/Stage, který by průběžně určoval polohu robotu ve dvoudimenzionálním prostoru bez pohybujících se objektů. S využitím algoritmů ICP, IMRP a IDC byl popsaný modul implementován a jeho vlastnosti byly ověřeny pomocí experimentů, které potvrdily, že (z reálných dat) lze pozici robotu určit s chybou v řádu centimetrů na jeden krok lokalizace.

Abstract

The goal of this work was to make a module for Player/Stage system which would continuously count position of the robot in two dimensional space without moving objects. Described module was implemented using ICP, IMRP and IDC algorithms and it's properties were checked through experiments which confirmed that position of the robot can be counted with an error in centimetres within one step of the localization process.

Na tomto místě bych chtěl poděkovat především vedoucímu práce, RNDr. Miroslavu Kulichovi, Ph.D., za mnoho dobrých rad, které jsem od něho dostal, a za jeho vstřícnost a trpělivost, kterou se mnou měl.

Poděkování také patří mé rodině, která mě po celou dobu studia obětavě podporovala.

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Lokalizace je jednou ze základních úloh mobilní robotiky. Autonomní mobilní robot, tedy zařízení, které je schopné pohybovat se samostatně v zadaném prostředí, obvykle potřebuje znát svoji polohu v něm. K jejímu určení je možné použít například informaci z odometrických senzorů, systém GPS, kompasy, gyroskopy, kamery atd. Ne všechny tyto metody jsou ale pro danou úlohu vhodné, hlavně co se týče přesnosti, doby potřebné pro získání údajů ze senzorů či výše nákladů na použitý hardware. Může se také stát, že budeme postaveni před úlohu lokalizace robotu s pevně danou konfigurací, kde tedy nemůžeme výběrem kvalitnějšího senzoru zlepšit jeho vlastnosti. V některých případech je proto vhodné získané údaje o poloze upravovat kombinováním údajů z více senzorů. A právě tento postup je základem algoritmů implementovaných v rámci této práce.

Využity budou konkrétně informace z odometrických senzorů, které budou zpřesněny pomocí dat získaných z laserového scanneru.

Samotný algoritmus bude implementován jako modul (driver) pro systém Player, který umožní jeho využití jak při nasazení na reálných robotech, tak při simulacích (v kombinaci se simulátorem Stage), což usnadní testování algoritmů.

Kapitola 1

Definice úlohy

Předpokládejme robot pohybující se v rovině, který umožňuje získání odometrické informace o své poloze, a který je vybaven dostatečně rychlým laserovým scannerem (t.j. kvalitu naměřeného scanu nesmí výrazně ovlivnit rychlost pohybu robotu). Dále předpokládejme statické okolní prostředí (tedy bez pohyblivých objektů), které se skládá pouze z objektů, jejichž vzdálenost může laserový scanner bez velké chyby změřit (neuplatní se tedy zrcadlení ani nebude laserový paprsek pohlcen). Cílem této práce je implementovat algoritmus, který bude pomocí údajů z laserového scanneru průběžně upravovat údaj o poloze robotu (tedy jeho souřadnice x, y a natočení φ), a to jak se známou mapou okolního prostředí, tak bez ní.

1.1 Odometrie

Odometrický systém umožňuje robotu získat informaci o své relativní poloze buď z příkazů posílaných akčním členům zajišťujícím pohyb (např. motory kol) anebo přímo z pohybu těchto akčních členů. První způsob (odhad na základě povelů) je méně přesný, protože předpokládá, že povel (např. „otoč dvakrát pravým kolem“) bude splněn dokonale.

Druhá metoda, odhad pozice z pohybu akčních členů, se v praxi realizuje typicky použitím optických enkodérů spojených s osou kol. Z informace o otočení kol je potom vypočítána změna polohy robotu. Odometrie poskytuje pouze relativní informaci o poloze, tedy její změnu od posledního měření. Pro určení absolutní polohy robotu je nutné tyto změny integrovat, což vede ke vzniku aditivní chyby. Největším problémem této metody jsou prokluzy a smýkání. I při sebemenším proklouznutí kol dochází při určení polohy k chybě, protože skutečně ujetá vzdálenost určitým kolem neodpovídá dráze určené ze změny otočení kola a jeho poloměru. Odometrický systém v tomto případě nemá možnost korekce ani detekce takovýchto případů. V prostředí s rovným podkladem, jehož součinitel tření s koly je neměnný[1], je chyba odometrie výrazně menší než např. při pohybu robotu v členitém terénu nebo při použití pásového podvozku, založeného na smýkání.

[1]Prostředí s těmito vlastnostmi budeme dále nazývat *kancelářským prostředím.*

Polohu robotu s diferenciálním podvozkem (tedy dvě kola se společnou osou) lze podle [6] určit na základě vztahů

$$\varphi(k) = \varphi(k-1) + \arctan\left(\frac{\Delta s_r - \Delta s_l}{l}\right)$$
$$x(k) = x(k-1) + \frac{\Delta s_r + \Delta s_l}{2}\cos(\varphi(k) - \varphi(k-1))$$
$$y(k) = y(k-1) + \frac{\Delta s_r + \Delta s_l}{2}\sin(\varphi(k) - \varphi(k-1)),$$

kde l je rozchod kol, $[x(k), y(k), \varphi(k)]$ a $[x(k-1), y(k-1), \varphi(k-1)]$ jsou souřadnice robotu a jeho natočení v čase k a $k-1$. Δs_r a Δs_l jsou vzdálenosti ujeté pravým a levým kolem v časovém intervalu $\langle k-1; k\rangle$.

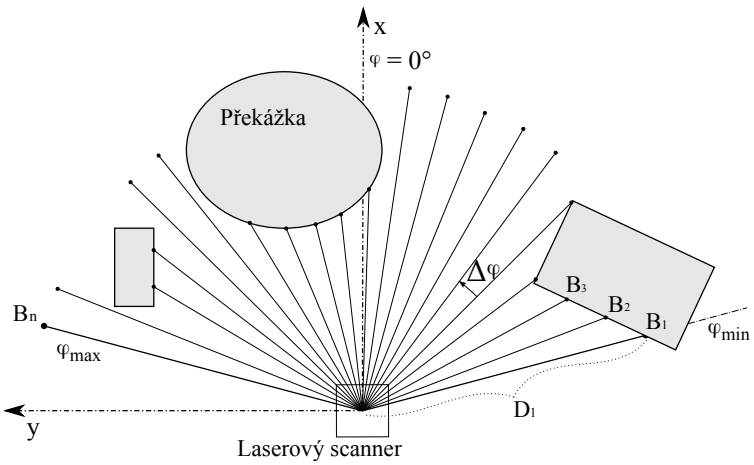
Obrázek 1.1: Optický enkodér MotionControl HS30 (převzato z http://www.motioncontrol.com)

1.2 Laserový dálkoměr

Určení relativní polohy robotu pomocí odometrie je v praxi často dosti nepřesné, pokud chceme určit polohu absolutní, musíme počítat ještě s aditivní chybou. Údaj o poloze robotu můžeme zpřesnit, máme-li k dispozici data z jiného vhodného senzoru. Tímto senzorem může být například laserový dálkoměr umístěný na robotu.

Laserový dálkoměr je zařízení, které pomocí rozmítaného laserového paprsku určuje vzdálenosti nejbližších překážek v kruhové výseči se středem v optice scanneru. Obvykle pracuje v rovině rovnoběžné s podkladem, existují sice možnosti, jak získat prostorový scan, takovéto typy scannerů však v této práci nebudeme využívat.

Laserový paprsek postupuje od jedné hranice výseče ke druhé s krokem v řádu desetin stupňů a v každé pozici změří vzdálenost k nejbližší překážce. Výstupem ze scanneru je potom posloupnost $\mathcal{D}$ vzdáleností naměřených paprskem v jednotlivých směrech.



Obrázek 1.2: Princip laserového scanneru, pohled shora

Posloupnost vzdáleností $\mathcal{D} = (D_i)_{i=1}^n$ lze podle vztahu

$$B_i = \begin{bmatrix} D_i \cdot \cos\left(i\left(\frac{\varphi_{max}-\varphi_{min}}{n-1}\right)+\varphi_{min}\right) \\ D_i \cdot \sin\left(i\left(\frac{\varphi_{max}-\varphi_{min}}{n-1}\right)+\varphi_{min}\right) \end{bmatrix} = \begin{bmatrix} D_i \cdot \cos(i \cdot \Delta\varphi + \varphi_{min}) \\ D_i \cdot \sin(i \cdot \Delta\varphi + \varphi_{min}) \end{bmatrix}$$

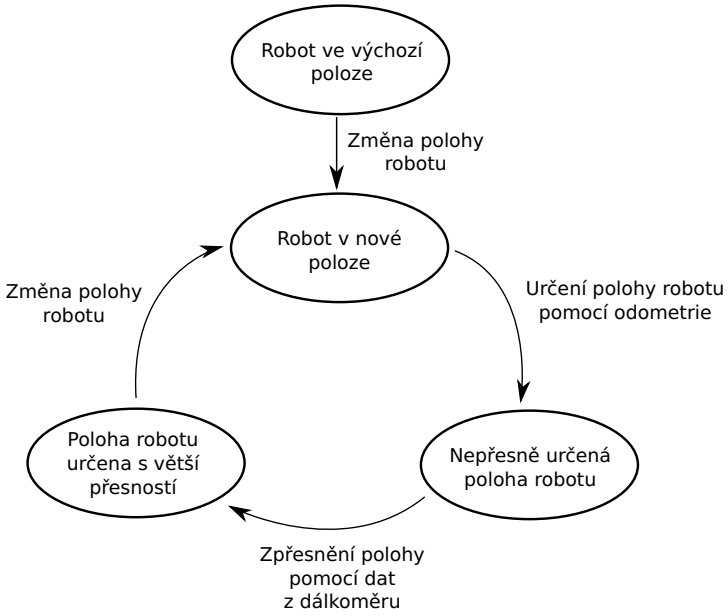
převést na posloupnost bodů $\mathcal{B} = (B_i)_{i=1}^n$ v souřadné soustavě spojené s laserovým dálkoměrem. Význam úhlů φ_{min}, φ_{max} a $\Delta\varphi$ je patrný z obr. 1.2.

Obrázek 1.3: Laserový scanner SICK LMS291-S14 (převzato z http://www.sick.com)

1.3 Kontinuální lokalizace

Úloha kontinuální lokalizace robotu spočívá v průběžném určování polohy robotu (tedy jeho pozice a natočení) v zadaném souřadném systému. Cílem této práce je najít řešení

úlohy kontinuální lokalizace robotu pro zadanou konfiguraci robotu (známy jsou údaje poskytované odometrickými senzory a laserovým dálkoměrem).



Obrázek 1.4: Stavový diagram kontinuální lokalizace pro zadanou konfiguraci robotu.

Tato práce se bude zabývat dvěma variantami popsané úlohy:

- kontinuální lokalizací bez známé mapy okolního prostředí,
- kontinuální lokalizací se známou mapou okolního prostředí.

Úloha kontinuální lokalizace bez známé mapy prostředí spočívá v průběžném určování polohy robotu v předem určeném souřadném systému pevně spojeném s nějakým bodem roviny, po které se robot pohybuje. Kontinuální lokalizace se znalostí mapy prostředí spočívá v průběžném určování polohy robotu v souřadném systému mapy. Předem určený souřadný systém pro lokalizaci bez mapy, resp. souřadný systém mapy pro lokalizaci se známou mapou budeme nazývat *globálním souřadným systémem*.

Kapitola 2

Principy implementovaných algoritmů

Úloha popsaná v kapitole 1 je nejprve pomocí postupu vysvětleného v odstavci 2.1 převedena na úlohu nalezení parametrů transformace, která minimalizuje hodnotu penalizační funkce (popsané vzorcem 2.2) pro dvě zadané množiny bodů. K nalezení těchto parametrů jsou poté využity algoritmy *iterative closest point* (ICP), *iterative matching range point* (IMRP) a *iterative dual correspondence* (IDC) popsané v kapitolách 2.2, 2.3 a 2.4.

2.1 Řešení úlohy kontinuální lokalizace

Při svém pohybu v okolním prostředí získává robot průběžně údaje z odometrie a z laserového scanneru. Pokud každému scanu přiřadíme poslední údaj o poloze, který robot obdržel z odometrie, dostaneme posloupnost přibližně lokalizovaných scanů, které budou postupně přicházet na vstup lokalizačního modulu. Po přijetí každého z těchto přibližně lokalizovaných scanů musíme zjistit polohu robotu, ze které byl scan skutečně pořízen.

2.1.1 Lokalizace bez známé mapy prostředí

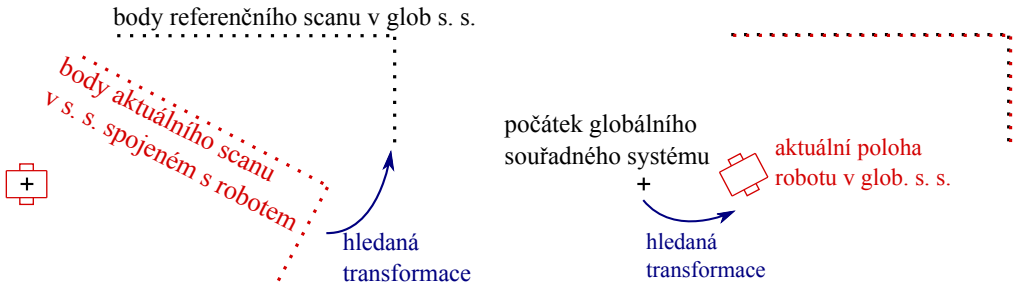
Při lokalizaci bez známé mapy budeme vždy využívat toho, že máme k dispozici *referenční scan*, tedy posloupnost naměřených vzdáleností a dobrý odhad souřadnic (v globálním souřadném systému) místa, ze kterého byly tyto vzdálenosti naměřeny. Jako referenční scan bude použit předchozí scan (z posloupnosti přibližně lokalizovaných scanů) spolu s pozicí robotu získanou lokalizováním robotu v okamžiku pořízení tohoto scanu. Budeme vycházet z toho, že podaří-li se registrovat body aktuálního scanu (v souřadnicích spojených s tělem robotu) vůči bodům scanu referenčního (v globálních souřadnicích), t.j. najít transformaci, která body aktuálního scanu co nejvíce „přiblíží“ (minimalizuje penalizační funkci) bodům scanu referenčního, pak budou parametry nalezené transformace odpovídat hledané poloze robotu v globálních souřadnicích (viz obr. 2.1).

Postup po přijetí nového přibližně lokalizovaného scanu (na obr. 1.4 je tento krok označen jako „Zpřesnění polohy pomocí dat z dálkoměru“) je následující:

1. Z rozdílu odometrií určených souřadnic místa pořízení referenčního scanu a místa pořízení aktuálního scanu je vypočtena přibližná změna polohy robotu.

2. Ze skutečných souřadnic místa pořízení referenčního scanu a vypočtené změny polohy je určena přibližná poloha robotu v globálních souřadnicích.

3. Pomocí jednoho z níže popsaných algoritmů (ICP, IMRP, IDC) jsou body aktuálního scanu v souřadném systému s počátkem v optice dálkoměru registrovány vůči bodům scanu referenčního v globálním souřadném systému. K tomu je mimo jiné využita přibližná poloha robotu v globálních souřadnicích vypočtená v kroku 2. Algoritmem určená transformace je pak hledanou polohou robotu v globálních souřadnicích (viz obr. 2.1).



Obrázek 2.1: Kontinuální lokalizace bez známé mapy - ilustrace rovnosti parametrů transformace a hledané polohy robotu

Podaří-li se tedy registrovat body aktuálního scanu vůči bodům referenčního scanu (tedy jednu množinu bodů vůči jiné), je možné určit polohu robotu v globálním souřadném systému.

2.1.2 Lokalizace se známou mapou prostředí

Postup při lokalizaci ve známé polygonální mapě je obdobný, jediný rozdíl je v tom, že scan není registrován vůči poslednímu lokalizovanému scanu, ale vůči posloupnosti úseček, ze kterých se skládají polygony mapy. Podaří-li se tedy registrovat body aktuálního přibližně lokalizovaného scanu (v souřadnicích spojených s tělem robotu) vůči mapě, pak parametry získané transformace odpovídají přímo poloze robotu v mapě.

2.2 Algoritmus ICP (Iterative closest point)

Algoritmus ICP, popsaný v [2], umožňuje registrovat posloupnost bodů $\mathcal{Q} = (Q_i)_{i=1}^{n}$ vůči posloupnosti $\mathcal{R}$ referenčních geometrických objektů. Registrace spočívá v nalezení

parametrů x_t,y_t,φ_t transformace definované předpisem

$$R'_i = \begin{bmatrix} cos(\varphi_t) & -sin(\varphi_t) \\ sin(\varphi_t) & cos(\varphi_t) \end{bmatrix} Q_i + \begin{bmatrix} x_t \\ y_t \end{bmatrix} = UQ_i + T,$$

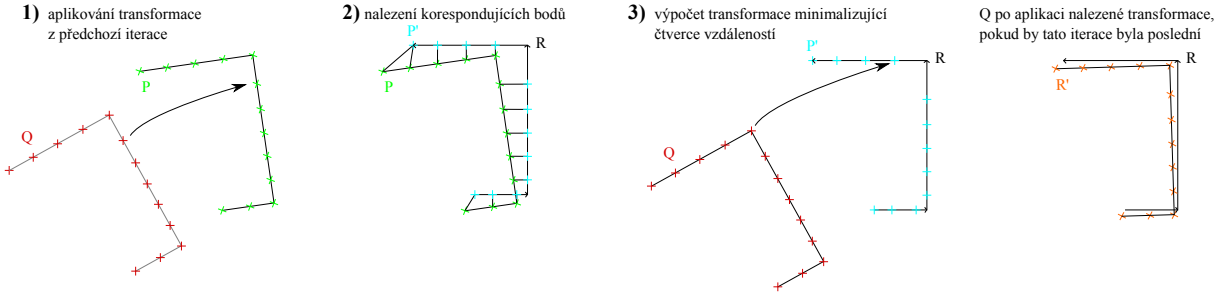
která převede $\mathcal{Q}$ na posloupnost bodů $\mathcal{R}'$ co nejméně se lišící od $\mathcal{R}$ (míra odlišnosti je určena penalizační funkcí 2.2 popsanou níže). Matici U budeme nazývat *rotační maticí*, matici T *translační maticí*.

ICP je iterativní algoritmus, který v každé iteraci provede tyto kroky:

1. Na body posloupnosti $\mathcal{Q}$ je aplikována transformace vypočtená v předchozí iteraci (v první iteraci ji můžeme považovat za nulovou nebo použít její odhad získaný např. z odometrie), získáme tak posloupnost $\mathcal{P}$.

2. Ke každému bodu P_i z posloupnosti $\mathcal{P}$ je nalezen bod P'_i tak, že ze všech bodů, které jsou prvky některého z referenčních geometrických objektů R_i, je vybrán ten s nejmenší euklidovskou vzdáleností od P_i.

3. Pro body posloupností $\mathcal{P}$ a $\mathcal{P}'$ jsou minimalizací penalizační funkce $E_{dist}(\varphi_t, x_t, y_t)$ určeny nové parametry transformace.



Obrázek 2.2: Průběh jedné iterace algoritmu ICP (referenčními útvary jsou zde 3 orientované úsečky)

Iterace jsou prováděny do té doby, než

$$|x_t - x_{t-1}| < \Delta x_t \quad \wedge \quad |y_t - y_{t-1}| < \Delta y_t \quad \wedge \quad |\varphi_t - \varphi_{t-1}| < \Delta\varphi_t, \tag{2.1}$$

kde x_t, y_t, φ_t jsou parametry transformace získané v aktuální iteraci, x_{t-1}, y_{t-1}, φ_{t-1} jsou parametry transformace získané v předchozí iteraci a Δx_t, Δy_t, $\Delta\varphi_t$ jsou předem určené konstanty.

Penalizační funkce, určující míru odlišnosti dvou scanů, je určena tímto předpisem:

$$E_{dist}(\varphi_t, x_t, y_t) = \sum_{i=1}^{n} \left| \begin{bmatrix} cos(\varphi_t) & -sin(\varphi_t) \\ sin(\varphi_t) & cos(\varphi_t) \end{bmatrix} P_i + \begin{bmatrix} x_t \\ y_t \end{bmatrix} - P'_i \right|^2 = \sum_{i=1}^{n} |U\ P_i + T - P'_i|^2 . \tag{2.2}$$

Vyjadřuje součet čtverců euklidovských vzdáleností bodů posloupnosti $\mathcal{P}$, které byly orotovány dle matice U a posunuty dle matice T, od bodů posloupnosti $\mathcal{P}'$. Úloha minimalizace E_{dist} má analytické řešení, které bylo odvozeno v [1]:

$$\varphi_t = \arctan \frac{S_{xy'} - S_{yx'}}{S_{xx`} + S_{yy'}} \tag{2.3}$$

$$x_t = \bar{x}' - (\bar{x} \cos \varphi_t - \bar{y} \sin \varphi_t) \tag{2.4}$$

$$y_t = \bar{y}' - (\bar{x} \sin \varphi_t + \bar{y} \cos \varphi_t), \tag{2.5}$$

kde

$$P_i = \left[\begin{smallmatrix} x_i \\ y_i \end{smallmatrix} \right] \qquad P_i' = \left[\begin{smallmatrix} x_i' \\ y_i' \end{smallmatrix} \right]$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} y_i \qquad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\bar{x}' = \frac{1}{n} \sum_{i=1}^{n} x_i' \qquad \bar{y}' = \frac{1}{n} \sum_{i=1}^{n} y_i'$$

$$S_{xx'} = \sum_{i=1}^{n} (x_i - \bar{x})(x_i' - \bar{x}') \qquad S_{yy'} = \sum_{i=1}^{n} (y_i - \bar{y})(y_i' - \bar{y}')$$

$$S_{xy'} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i' - \bar{y}') \qquad S_{yx'} = \sum_{i=1}^{n} (y_i - \bar{y})(x_i' - \bar{x}').$$

Algoritmus ICP je univerzální v tom, že je schopen registrovat posloupnost bodů $\mathcal{Q}$ vůči referenční posloupnosti $\mathcal{R}$ libovolných geometrických objektů s jediným omezením, a to že musíme být schopni určit bod tohoto objektu euklidovsky nejbližší k zadanému bodu. V našem případě budou referenčními geometrickými objekty body a úsečky.

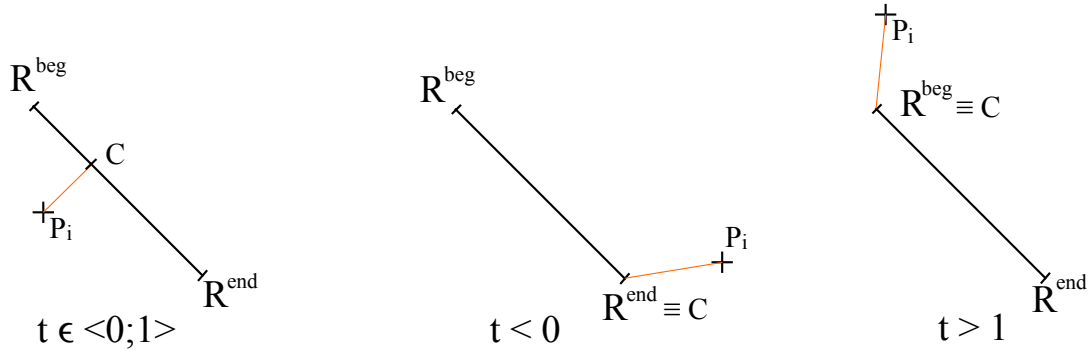
Pokud jsou prvky posloupnosti $\mathcal{R}$ body, pak spočívá nalezení bodu P_i' ze 2. kroku algoritmu ICP ve výběru toho bodu z posloupnosti $\mathcal{R}$, který má od bodu P_i nejmenší euklidovskou vzdálenost.

Pokud jsou referenčními geometrickými objekty úsečky, pak je třeba najít na každé úsečce z $\mathcal{R} = (R)_{j=1}^{m}$ bod euklidovsky nejbližší bodu P_i. Takto získáme m bodů, z nichž opět vybereme ten, který má nejmenší euklidovskou vzdálenost od P_i. Tento bod je hledaný bod P_i'.

Hledáme-li bod

$$C = uR^{beg} + (1 - u)R^{end}$$

na úsečce R_j s počátečním bodem R^{beg} a koncovým bodem R^{end}, který má nejmenší euklidovskou vzdálenost od P_i, stačí, když určíme hodnotu parametru $u \in \langle 0; 1 \rangle$. Mohou nastat 3 případy konfigurace úsečky a bodu P_i, které ilustruje obrázek 2.3.



Obrázek 2.3: Konfigurace bodu a úsečky pro různá t

Pokud $R^{beg} \neq R^{end}$, z podmínky pro vzdálenost $|CP_i|$ hledaného bodu C a bodu $P_i = \begin{pmatrix} P_{i,x} \\ P_{i,y} \end{pmatrix}$

$$|CP_i| = \min_u |uR^{beg} + (1-u)R^{end} - P_i| \tag{2.6}$$

lze pro výpočet parametru u odvodit následující vzorec

$$t = \frac{(P_{i,x} - R_x^{end})(R_x^{beg} - R_x^{end}) + (P_{i,y} - R_y^{end})(R_y^{beg} - R_y^{end})}{|R^{beg}R^{end}|^2} \tag{2.7}$$

$$u = t \quad \Leftrightarrow \quad t \in \langle 0; 1\rangle \tag{2.8}$$

$$u = 0 \quad \Leftrightarrow \quad t < 0 \tag{2.9}$$

$$u = 1 \quad \Leftrightarrow \quad t > 1. \tag{2.10}$$

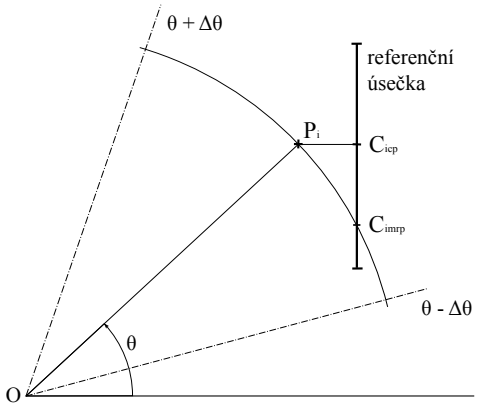
Důležitou vlastností algoritmu ICP je to, že byla dokázána jeho konvergence do lokálního minima, viz [2]. Pokud tedy odhad transformace (získaný z odometrie) nedělí od skutečné transformace lokální maximum penalizační funkce, algoritmus bude schopen skutečnou transformaci s libovolnou konečnou přesností najít.

2.3 Algoritmus IMRP (Iterative matching range point)

Algoritmus ICP určuje korespondující body ($\mathcal{P}'$ z kroku 2) tak, že hledá body euklidovsky nejbližší. Tím však dochází k preferování translace před rotací, což vede ke zpomalení konvergence rotační složky transformace. Algoritmus IMRP (popsaný v [1]) se snaží tento nedostatek odstranit použitím jiného postupu pro hledání odpovídajících si bodů, přičemž ostatní části algoritmu jsou stejné jako u ICP.

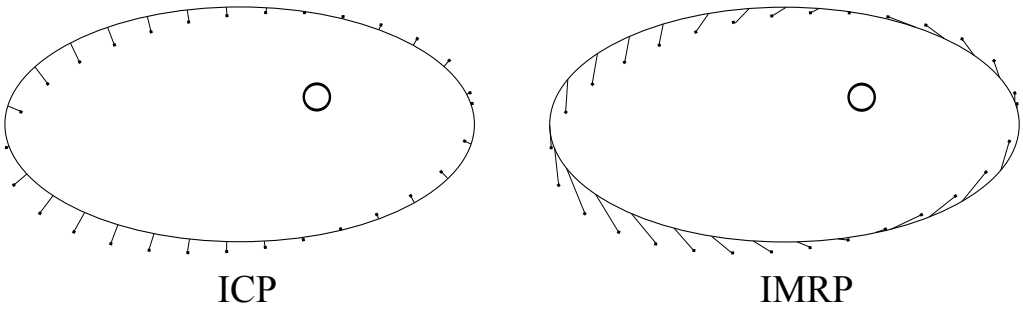
Aby bylo dosaženo upřednostnění rotace před translací, při hledání korespondujících bodů je translace zcela zanedbána. Platí tedy $|P_i'| \approx |P_i|$ a $\measuredangle P_i' \approx \measuredangle P_i + \varphi$, kde $\measuredangle P_i'$, resp. $\measuredangle P_i$ označuje úhlovou polární souřadnici bodu P_i', resp. P_i a φ je úhlová složka transformace, kterou musíme provést pro registraci bodu P_i' vůči bodu P_i.

Hledání korespondujícího bodu P_i' k bodu P_i tedy probíhá tak, že se hledá bod z referenční množiny R, který má shodnou vzdálenost od počátku (O) jako bod P_i. Tedy $|OP_i| = |OP_i'|$. Pokud bod s přesně stejnou vzdáleností neexistuje, použije se bod se vzdáleností co nejméně se lišící od vzdálenosti $|OP_i|$. Až v případě shody je preferován bod, který leží euklidovsky blíže k P_i.



Obrázek 2.4: Rozdíl mezi algoritmy ICP a IMRP I

Obrázek 2.4 ilustruje základní rozdíl mezi algoritmy ICP a IMRP. Při použití algoritmu ICP bude nejbližším bodem k bodu P_i bod C_{icp}, při použití algoritmu IMRP bude nejbližším bodem k bodu P_i bod C_{imrp}.



Obrázek 2.5: Rozdíl mezi algoritmy ICP a IMRP II

Obrázek 2.5 ukazuje korespondující body nalezené algoritmy ICP a IMRP, pokud je referenčním objektem křivka tvaru elipsy.

Aby nedocházelo k nežádoucímu vyhledávání bodů příliš vzdálených od P_i, vyhledávání probíhá pouze v kruhové výseči se středem v počátku a s úhly hraničních přímek $\theta - \Delta\theta$ a $\theta + \Delta\theta$ (na obr. 2.4 vyznačeny čerchovaně). Úhel $\Delta\theta$ se s každou další iterací zmenšuje, protože se s každou iterací zmenšuje i chyba v určení rotace, kterou je nutné odstranit. V [1] byl experimentálně určen vztah pro výpočet úhlu $\Delta\theta(t)$ v iteraci t takto

$$\Delta\theta(t) = \Delta\theta(0)e^{-\alpha t}, \tag{2.11}$$

kde $\alpha \in \mathbb{R}^+$, $\Delta\theta(0)$ je předem stanovená konstanta určená např. z maximální rotační chyby odometrie.

2.4 Algoritmus IDC (Iterative dual correspondence)

Pro registraci scanu vůči referenční posloupnosti máme zatím k dispozici dva algoritmy: ICP, který umožňuje rychlou konvergenci translační složky transformace, a IMRP, který naopak umožňuje rychlou konvergenci rotační složky transformace. Algoritmus IDC (popsaný v [1]) spojuje výhody obou těchto algoritmů. V každé iteraci algoritmu jsou provedeny následující kroky:

1. Na body posloupnosti $\mathcal{Q}$ je aplikována transformace (s parametry x_t, y_t a φ_t) vypočtená v předchozí iteraci (v první iteraci ji můžeme považovat za nulovou nebo použít její odhad získaný např. z odometrie), získáme tak posloupnost $\mathcal{P}$.
2. Pro každý bod P_i
   (a) je na $\mathcal{R}$ nalezen bod $P'_{i,icp}$ s nejmenší euklidovskou vzdáleností od P_i.
   (b) je na $\mathcal{R}$ nalezen bod $P'_{i,imrp}$ se stejnou vzdáleností od počátku O, jakou má bod P_i.
3. Pro body posloupností
   (a) $\mathcal{P}$ a $\mathcal{P}'_{icp}$ jsou minimalizací penalizační funkce $E_{dist}(\varphi_{icp}, x_{icp}, y_{icp})$ určeny nové parametry transformace x_{icp}, y_{icp} a φ_{icp}.
   (b) $\mathcal{P}$ a $\mathcal{P}'_{imrp}$ jsou minimalizací penalizační funkce $E_{dist}(\varphi_{imrp}, x_{imrp}, y_{imrp})$ určeny nové parametry transformace x_{imrp}, y_{imrp} a φ_{imrp}.
4. Parametry výsledné transformace jsou

$$x_t = x_{icp} \qquad y_t = y_{icp} \qquad \varphi_t = \varphi_{imrp}.$$

Kapitola 3

Popis implementace

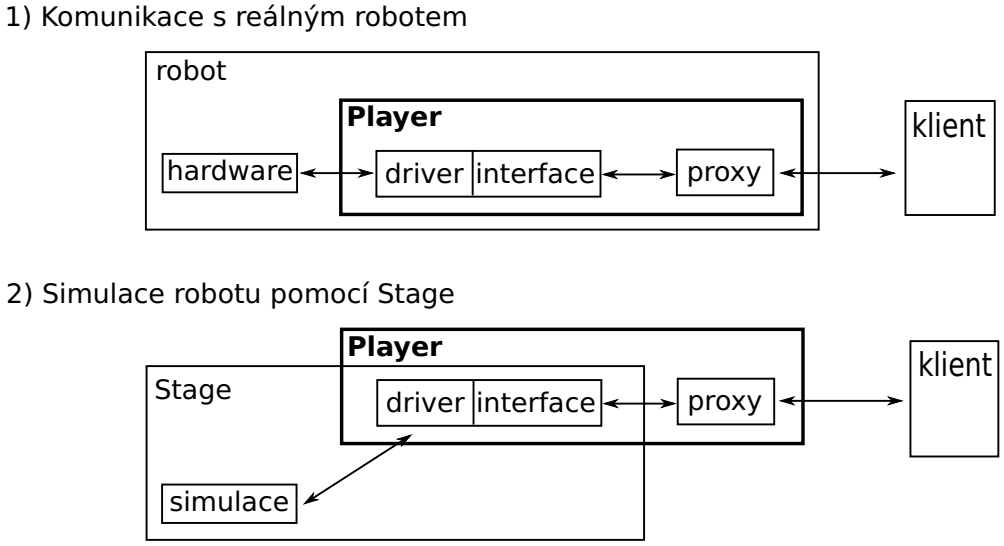
V rámci práce byl implementován driver pro systém Player/Stage [3], který umožňuje provádět kontinuální lokalizaci robotu v neznámé mapě algoritmy ICP, IMRP a IDC. Dále dokáže lokalizovat robot ve známé polygonální mapě s využitím algoritmu ICP.

3.1 Systém Player 3.0.2

Systém Player (součást projektu Player/Stage) je serverová aplikace, která vytváří jednotný interface pro komunikaci s robotem pomocí protokolu TCP. Může sloužit například ke čtení údajů ze senzorů robotu nebo k ovládání jeho akčních členů. Komunikaci klientské aplikace se serverem obvykle zprostředkovávají připravené knihovní funkce, s robotem server komunikuje pomocí zásuvných modulů, tzv. *driverů*. Funkce driveru však není omezena jen na komunikaci serveru s hardwarem. Drivery mohou také zpracovávat výstupní data jiného modulu a poskytovat je ostatním driverům.

Komunikačním rozhraním driverů jsou tzv. *interface*. Jedná se o definice formátů zpráv zasílaných klientem serveru a naopak. Úkolem driveru komunikujícího s robotem je tedy, mimo vlastní komunikaci s hardwarem, také převod dat získaných z robotu do správného formátu, který vyžaduje výstupní interface driveru a naopak převod požadavků zaslaných driveru ve standardizovaném tvaru na povely pro hardware. Například driver, který se stará o čtení dat z optických enkodérů kol v podvozku robotu, může z natočení kol vypočítávat polohu robotu a klientské aplikaci posílat už jen vlastní odometrickou informaci ve formátu, který vyžaduje např. interface `position2d`. Systém Player/Stage umožňuje jednak nová rozhraní vytvářet, jednak využít některá z již připravených (např. `laser`, `ranger`, `position2d`, `simulation`, ...), což urychluje vývoj aplikací.

Úkolem samotného serveru je potom zprostředkovat spojení mezi klientem a zvoleným interface, k čemž jsou využívány tzv. *proxies* (zástupné moduly).



Obrázek 3.1: Konfigurace serveru Player při použití na skutečném a simulovaném robotu

3.2 Simulátor Stage 4.0.0

Aby bylo možné využít možnosti systému Player bez nutnosti stavět skutečný robot, byl vyvinut simulátor Stage, který dokáže robot (případně roboty) simulovat. Místo fyzického, hardwarového robotu, který by se pohyboval v reálném prostředí, dokáže Stage celou situaci vykreslit na monitor s tím, že systému Player se každý simulovaný robot jeví stejně jako skutečný, hardwarový, a to včetně senzorů a akčních členů.

Stage se používá jako driver, který poskytuje data o simulovaných robotech pomocí zvolených rozhraní (interface) a zároveň tato data v reálném čase zobrazuje.

3.3 Implementovaný driver

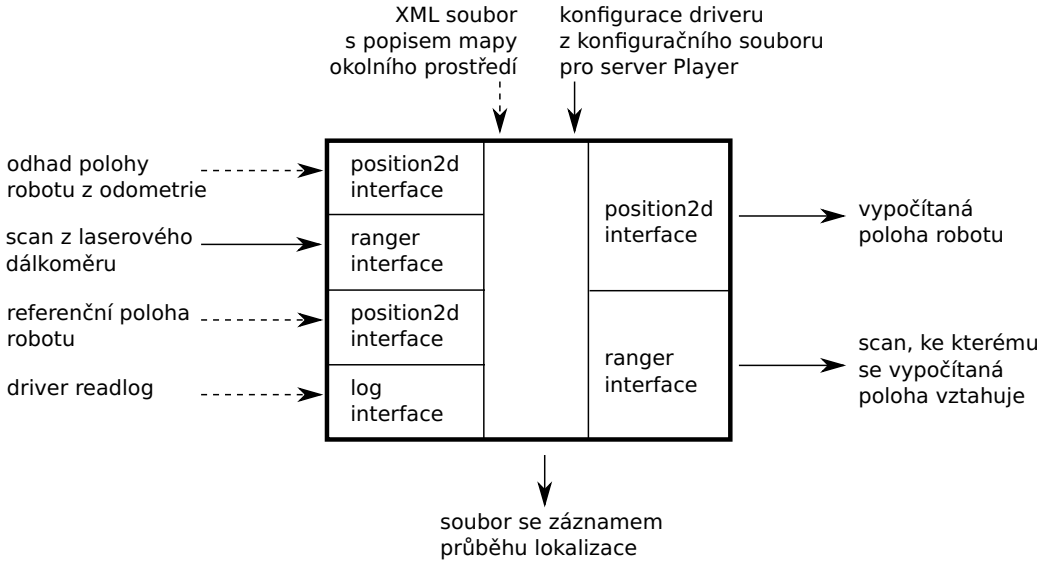
V rámci této práce byl v jazyce C++ vytvořen driver, který bude v této kapitole podrobně popsán.

3.3.1 Rozhraní

Vstupem implementovaného driveru jsou data z dálkoměru (`ranger` interface) a volitelně také pozice robotu určená odometrií (interface `position2d`). Pokud má být lokalizace prováděna vůči zadané mapě, je vstupem driveru také XML soubor, který tuto mapu popisuje. Dalším volitelným vstupem je referenční pozice (`position2d` interface), která je ale pouze ukládána do výstupního souboru spolu s dalšími daty o průběhu lokalizace, a jejíž hlavní využití je při testování. K testování slouží i volitelný interface `log`, který, pokud jsou vstupní data pro implementovaný driver generována pomocí driveru readlog, umožňuje dotazovat se tohoto driveru, zda simulace stále probíhá. V případě, že readlog dosáhl konce

souboru se zaznamenanými daty, je server Player (včetně lokalizačního driveru) ukončen. Posledním vstupem driveru je uživatelské nastavení uložené jako součást konfiguračního souboru pro server Player (viz 3.3.2).

Na výstupu driveru je potom pozice robotu (`position2d` interface) vypočtená zvoleným lokalizačním algoritmem (ICP, IMRP nebo IDC) a dále scan (`ranger` interface), ke kterému se tato pozice vztahuje (viz 3.4.2). Výstupem driveru je také textový soubor se záznamem průběhu lokalizace.



Obrázek 3.2: Interface implementovaného driveru

3.3.2 Nastavení parametrů lokalizace

Nastavení nezbytné pro správnou funkci jednotlivých driverů je uloženo v konfiguračním souboru pro server Player, který ho při svém startu jednotlivým modulům předá.

Chování implementovaného driveru lze ovlivnit následujícími parametry:

general_verbose
: Určuje, zda má driver kromě varování a chybových hlášení vypisovat také informace, které jsou pouze informativní a nevyžadují od uživatele žádný zásah (např. počty iterací v jednotlivých krocích algoritmu).

general_logFile
: Udává název a umístění textového souboru, do kterého budou uloženy údaje o průběhu lokalizace (viz 3.3.3). Pokud je cesta zadána relativně, vztahuje se k aktuálnímu umístění konfiguračního souboru.

general_method
: Udává název metody, která bude použita pro lokalizaci. Povolené hodnoty jsou: ICP, IMRP, IDC.

general_localizeToKnownMap
Určuje, zda má být příchozí scan registrován vůči zadané polygonální mapě (*true*) nebo vůči předchozímu lokalizovanému scanu (*false*).

general_mapfile
Udává cestu k souboru obsahujícímu popis polynomiální mapy prostředí, ve kterém se bude robot pohybovat. Pokud je cesta zadána relativně, vztahuje se k aktuálnímu umístění konfiguračního souboru. Tento parametr je nezbytný pouze pokud je parametr **general_localizeToKnownMap** nastaven na hodnotu *true*.

general_maxLaserDistance
Určuje maximální dosah laserového scanneru. Pokud driver obdrží od scanneru naměřenou vzdálenost, která je větší nebo rovna **general_maxLaserDistance**, bude tuto vzdálenost ignorovat.

general_minPtsNrAfterMaxLaserDistance
Jde o minimální počet bodů, které musí zbýt po odebrání bodů, které byly za hranicí dosahu laserového scanneru. Jedná se pouze o parametr, který slouží k detekci špatně nastavené hodnoty parametru **general_maxLaserDistance**, a který by se běžně neměl projevit.

general_acceptedPtsRatio
Určuje, jaká část z nalezených bodových párů bude brána v potaz.

general_maxNrIterations
Udává maximální počet iterací, které mohou proběhnout při lokalizaci jednoho scanu.

general_toleranceX, general_toleranceY, general_toleranceA
Pokud se poloha robotu změní oproti poloze vypočítané v předchozí iteraci o hodnoty menší než jsou hodnoty těchto parametrů, bude scan považován za lokalizovaný a provádění dalších iterací bude zastaveno.

icp_connectPrevPtsWithLines
Určuje, zda bude lokalizace metodou ICP prováděna vůči úsečkám spojujícím referenční body (true) nebo jen vůči samostatným referenčním bodům (false). Při lokalizaci vůči polygonální mapě (t.j. když je **general_localizeToKnownMap** nastaveno na *true*) je tento parametr ignorován.

imrp_connectPrevPtsWithLines
Určuje, zda bude lokalizace metodou IMRP prováděna vůči úsečkám spojujícím referenční body (true) nebo jen vůči samostatným referenčním bodům (false). Při lokalizaci vůči polygonální mapě (t.j. když je **general_localizeToKnownMap** nastaveno na *true*) je tento parametr ignorován.

imrp_initialSectorSize
: Udává výchozí velikost výseče (v radiánech), ve které budou hledány nejbližší body při první iteraci algoritmu IMRP.

imrp_sectorDecreaseConstant
: Ovlivňuje rychlost zmenšování výseče, ve které se hledá nejbližší bod ve druhé a následujících iteracích algoritmu IMRP. Jedná se o konstantu α ze vztahu (2.11).

imrp_minClosestPtsFoundRatio
: Určuje, k jaké části platných bodových párů (t.j. po odebrání párů s příliš vzdálenými body) musí při každé iteraci existovat po prohledání výseče blízký bod. Pokud je platných párů méně, znamená to, že je výseč již příliš malá, v iterování se proto nepokračuje.

3.3.3 Popis výstupního souboru se záznamem průběhu lokalizace

Soubor se záznamem průběhu lokalizace je textový soubor, který se skládá z bloků oddělených prázdným řádkem. Každý blok odpovídá jednomu měření dálkoměru (t.j. jedné množině naměřených bodů), které bylo registrováno vůči měření předchozímu. Blok může vypadat například takto:

```
RANGER_TIMESTAMP 35.5785
POSITION_REF 6.44783 4.66382 0.978863
POSITION_ODO 6.64547 4.30054 0.921067
POSITION_SUPP 6.47598 4.59771 0.966529
SCAN 0.019 -2.249  0.058 -2.249  0.099 -2.277  0.138 -2.275 ...
TRANSIENT 6.49 4.58 0.97  8.38 3.33  8.38 3.37  8.38 3.37 ...
TRANSIENT 6.50 4.58 0.97  8.38 3.33  8.38 3.37  8.44 3.41 ...
TRANSIENT 6.50 4.58 0.97  8.38 3.33  8.38 3.36  8.44 3.41 ...
TRANSIENT 6.51 4.58 0.97  8.38 3.33  8.38 3.36  8.44 3.41 ...
TRANSIENT 6.51 4.58 0.97  8.38 3.33  8.38 3.36  8.44 3.41 ...
POSITION_ALG 6.51 4.58 0.97
```

Na řádku začínajícím klíčovým slovem *RANGER_TIMESTAMP* je uveden čas, ve kterém bylo měření dálkoměru provedeno.

Řádek označený *POSITION_REF* udává referenční polohu robotu v globálním souřadném systému (souřadnice x, y a natočení) předanou driveru pomocí `position2d` interface.

Na řádku začínajícím klíčovým slovem *POSITION_ODO* je uvedena poloha robotu přibližně určená odometrickými senzory.

Řádek *POSITION_SUPP* udává odhadovanou pozici robotu v době pořízení scanu dálkoměrem. Tato pozice je získána přičtením z odometrie získané změny polohy robotu mezi místem pořízení aktuálního a předchozího lokalizovaného scanu k algoritmem určenému místu pořízení předchozího lokalizovaného scanu.

Na řádku *SCAN* jsou uvedeny body naměřené dálkoměrem ve formátu $x␣y␣␣x␣y␣␣$ atd. Jedná se již o body v souřadném systému s počátkem v optice dálkoměru, ne pouze

o naměřené vzdálenosti.

V každém bloku je dále obsaženo tolik řádků začínajících slovem *TRANSIENT*, kolik iterací algoritmu proběhlo. Na každém takovém řádku je uvedena pozice robotu vypočítaná v dané iteraci (první 3 čísla po klíčovém slově). Za vypočtenou pozicí jsou uvedeny nejbližší body nalezené pomocí zadaného lokalizačního algoritmu (ve formátu $x \sqcup y \sqcup \sqcup x \sqcup y \sqcup \sqcup$ atd.). Pokud k některému bodu nebyl odpovídající nejbližší protějšek nalezen, je místo souřadnic uvedeno „null null“.

Na řádku uvozeném klíčovým slovem *POSITION_ALG* je uvedena poloha robotu vypočítaná lokalizačním driverem (ve formátu x, y, natočení).

Pokud některý z těchto údajů nemůže být zjištěn (např. pokud na vstupu driveru není referenční poloha robotu), pak je na příslušném řádku za klíčovým slovem pouze vypsáno „null“.

3.3.4 Popis mapy okolního prostředí

Při implementaci algoritmu pro lokalizaci robotu ve známé mapě bylo nutné zajistit předání informací o umístění a tvaru jednotlivých překážek lokalizačnímu driveru. Kvůli srozumitelnému formátu a možnosti snadného rozšíření o nové funkce byl k tomuto účelu použit standard XML.

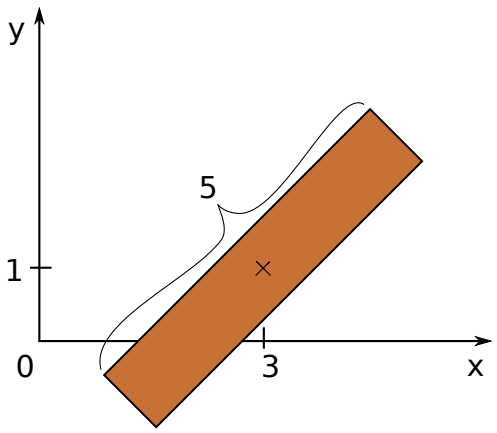
Jednoduchý soubor s definicí jedné obdélníkové překážky může vypadat např. takto:

```
<?xml version="1.0" encoding="UTF-8" ?>
<arena>
        <obstacle id='1'>
                <center x='3' y='1' phi='45' />
                <size x='5' y='1' z='0.5' />
        </obstacle>
</arena>
```

Tag *obstacle* představuje překážku obdélníkového tvaru, jeho parametr *id* udává její jednoznačné identifikační číslo. Parametry tagu *center* určují souřadnice středu překážky a její natočení (ve stupních), parametry tagu *size* udávají velikost překážky ve směru jednotlivých os souřadného systému (před natočením překážky dle parametru *phi* tagu *center*).

Překážka s id 1 z uvedeného ukázkového souboru tedy bude mít střed v bodě $[3, 1]$ a její rozměry budou 5, 1 a 0.5 (ve směru os x, y a z). Celá situace je znázorněna na obr. 3.3.

Ke zpracování tohoto XML souboru je využita knihovna *Property Tree* z balíku *Boost* [5], která umožňuje snadný přístup k jednotlivým částem dokumentu ve formátu XML. Pro správný běh driveru je nutné používat *Boost* minimálně ve verzi 1.41.0, což je verze, ve které byla knihovna *Property Tree* do tohoto balíku poprvé přidána.



Obrázek 3.3: Překážka definovaná v ukázkovém souboru (pohled shora)

3.3.5 Architektura

Lokalizační driver se skládá z deseti základních částí (implementovaných jako třídy OOP), jejichž účel a základní princip je následující:

matchingDriver
: Základním úkolem této třídy je zprostředkovat komunikaci mezi serverem a jádrem lokalizačního modulu, které zajišťuje výpočet pozice. Tato třída se tedy stará o správné zavedení driveru do serveru, načtení nastavení z konfiguračního souboru, získávání dat od ostatních modulů a poskytování dat vypočítaných. Také obsluhuje všechny přijaté zprávy a stará se o ukončení činnosti driveru.

Localizer
: Třída Localizer je komunikačním rozhraním jádra lokalizačního modulu. Cílem při jeho návrhu bylo jednak umožnit třídám, které ho budou používat, snadno a přehledně komunikovat s výpočetním jádrem, jednak odstínit specifické požadavky těchto tříd. Rozhraní proto obsahuje pouze konstruktor a 5 základních metod, které zajišťují veškerou komunikaci mezi jádrem a jeho „uživateli“. Je proto možné místo třídy **matchingDriver** vytvořit modul, který bude informace ze senzorů robotu načítat např. z textového souboru, a žádná část výpočetního jádra včetně třídy Localizer se nebude muset měnit.

  Veřejné metody této třídy (kromě konstruktoru a destruktoru) jsou

  storeSettings Metoda, která jako parametr obdrží strukturu obsahující veškeré parametry ovlivňující chování výpočetního jádra. Tyto parametry zkontroluje a uloží tak, aby k nim měla přístup každá část jádra, která je vyžaduje.

  initializePosition Metoda, jejímž parametrem je výchozí poloha robotu. Volána je pouze jednou před prvním voláním metody **countPosition**.

updatePosition Tato metoda oznamuje jádru, že robot obdržel z odometrických senzorů aktuální informaci o své pozici. Nová pozice je funkci **updatePosition** předána jako parametr, díky čemuž může být jádrem dále zpracována.

updateLaser Jedná se o obdobu funkce updatePosition, ale v tomto případě předává funkce jádru data naměřená laserovým scannerem.

countPosition Tato metoda bez parametrů z dat získaných pomocí metod **updatePosition** a **updateLaser** vypočítá jednou z metod popsaných v kapitole 2 pozici robotu a předá ji tazateli jako svou návratovou hodnotu.

ILocMethod
Jedná se o čistě virtuální třídu, jejímiž potomky jsou **Icp**, **Imrp** a **Idc**. Jedinou povinnou součástí, kterou tato třída od potomků vyžaduje, je implementace metody matchScan, která vrací parametry transformace vypočítané pomocí zvolené metody.

Icp, Imrp, Idc
Třídy reprezentující jednotlivé metody výpočtu. Každá z nich musí být schopna vypočítat transformaci způsobem, který je definován v konfiguračním souboru (t.j. pro případ samostatných referenčních bodů, úseček z nich vytvořených, i pro případ zadané referenční mapy).

Scan
Tato třída reprezentuje pole bodů se souřadnicemi x,y, které zná svoji velikost a počet aktuálně uchovávaných bodů.

Closest
Třída Closest představuje pole bodových párů spolu s dalšími pomocnými údaji.

Globals
Soubory globals.h a globals.cpp umožňují snadný přístup k některým proměnným, konstantám a strukturám, které musí být přístupné z většiny výše uvedených tříd.

LocalizationException
Výjimka sloužící pro signalizaci chyby lokalizačního driveru.

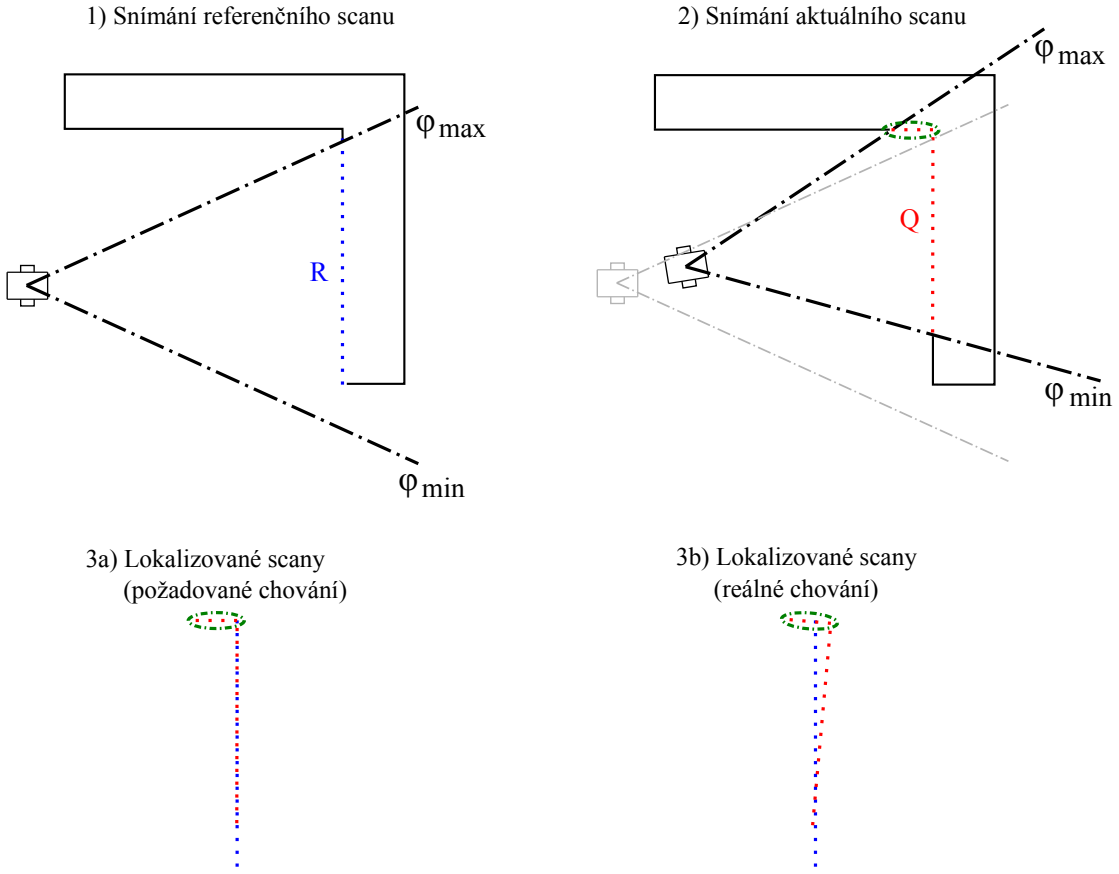
3.4 Problémy, které se při implementaci objevily

Při implementaci driveru bylo nutné vypořádat se s několika problémy způsobenými převodem teoretického řešení problému do praxe. V této kapitole budou problémy popsány spolu se zvoleným postupem jejich řešení.

3.4.1 Odebírání falešných bodových párů

V některých situacích se může stát, že bod z aktuálního scanu nemá ve scanu referenčním svůj protějšek. Jde především o případy, kdy aktuální scan pokryje oblast, která

v referenčním scanu nebyla viditelná (viz obr. 3.4), nebo kdy dálkoměr změří (např. vlivem odrazu) špatnou vzdálenost. K této situaci může také dojít, vyskytne-li se v některém ze scanů pohyblivý předmět. Poslední dva případy sice porušují podmínky uvedené v definici úlohy, v praxi se jim ale vyhnout nemůžeme. Lokalizační algoritmy tak, jak byly popsány v kapitole 2 však najdou protějšek i pro tyto body, což způsobí vznik tzv. *falešných párů*, které zanáší do výpočtu transformace nežádoucí chybu. Proto je třeba tyto případy detekovat a falešné páry nezahrnovat do výpočtu parametrů transformace (vztahy 2.3 - 2.5).



Obrázek 3.4: Ilustrace vzniku falešných bodových párů. Zeleně označené body z Q nemají v referenčním scanu R korespondující bod.

Implementovaný driver řeší tento problém metodou ignorování konstantního množství nejvzdálenějších bodů v každé iteraci.

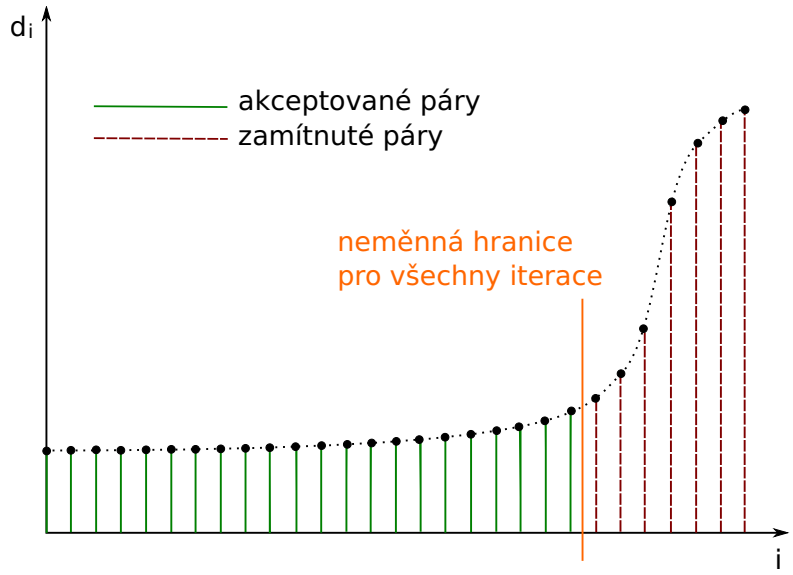
Tato metoda, kterou ilustruje obr. 3.5, spočívá v následujícím postupu:

1. v každé iteraci jsou po nalezení korespondujících bodů vzniklé páry seřazeny podle vzdálenosti bodů od sebe,

2. je vybrán pouze určitý počet párů s nejmenší vzdáleností bodů (procentuální zastoupení akceptovaných párů se nastavuje v konfiguračním souboru pomocí parametru

general_acceptedPtsRatio),

3. parametry transformace jsou vypočítány pouze z těchto vybraných párů, ostatní jsou ignorovány.



Obrázek 3.5: Ignorování konstantního množství nejvzdálenějších bodů

3.4.2 Zpoždění způsobená dobou výpočtu

Výpočet pozice robotu může trvat relativně dlouhou dobu (až cca 2s pro 400 iterací IMRP na core i5). Při testování driveru se stávalo, že během této doby dálkoměr odměřil další scan a poslal ho klientské aplikaci ke zpracování. Po dokončení výpočtu polohy se tato také odeslala do klientské aplikace, bohužel kvůli délce trvání výpočtu již nebyla aktuální. Když se tedy klientská aplikace pokoušela nejaktuálnější scan lokalizovat pomocí nejaktuálnějšího údaje o poloze, který měla k dispozici (ale který se vztahoval k jednomu z předchozích scanů), nebyly scany lokalizovány správně.

Tento problém byl vyřešen tak, že driver zároveň s informací o poloze robotu odesílá také body scanu, ke kterému se tato informace vztahuje. Klientská aplikace pak může pracovat se scanem i s jemu odpovídající polohou robotu zároveň.

3.4.3 Velké množství parametrů, na kterých závisí vlastnosti lokalizace

Při implementaci driveru bylo nutno vyřešit kompromis mezi jednoduchostí uživatelského rozhraní (t.j. snahou o to, aby uživatel musel v konfiguračním souboru nastavovat co nejméně parametrů) a univerzálností algoritmů.

Ve finální podobě driveru je třeba nastavovat pro některé algoritmy poměrně velké množství parametrů (např. pro IMRP až 8), což ztěžuje jak práci s driverem, tak testování. Samotné množství parametrů však není takovým problémem, jako značná závislost kvality lokalizace na nich. Konkrétně parametry **imrp_sectorDecreaseConstant** a **imrp_initialSectorSize** se při použití algoritmu IMRP ukázaly jako velice podstatné.

Před případným praktickým využitím driveru je proto nutné pomocí testovacích dat nejprve všechny parametry vhodně zvolit.

Kapitola 4

Experimenty

Kvůli ověření funkčnosti driveru a vlastností implementovaných algoritmů byla provedena řada experimentů. Testování bylo ve všech případech provedeno na notebooku HP ProBook 6550b s procesorem core i5 a 4 GB RAM a operačním systémem Ubuntu 10.04 LTS - Lucid Lynx. Na tomto notebooku byl nainstalován server Player ve verzi 3.0.2, simulátor Stage ve verzi 4.0.0, knihovna boost ve verzi 1.44.0 a soubor kompilátorů gcc ve verzi 4.4.3. Ke generování grafů byl použit program gnuplot ve verzi 4.2.6-1.

Testy byly rozděleny do čtyř kategorií:

testování konvergence algoritmů
: sledování závislosti hodnoty penalizační funkce na počtu iterací jednotlivých algoritmů při registrování vybraných dvojic scanů

testování lokalizace bez známé mapy na datech z repozitáře OpenSLAM.org
: určování kvality lokalizace pomocí jednotlivých algoritmů na datech naměřených skutečnými (nesimulovanými) roboty

testování kontinuální lokalizace se známou mapou prostředí
: porovnání kvality lokalizace bez známé mapy okolního prostředí a kvality lokalizace se znalostí této mapy

praktické experimenty
: ověření schopnosti lokalizačního driveru zpracovat data naměřená robotem S1R, který byl na ČVUT vyvinut jako součást projektu SyRoTek.

4.1 Testování konvergence algoritmů

Při tomto testu bude sledována závislost hodnoty penalizační funkce 2.2 na počtu iterací lokalizačního algoritmu. Vykresleno bude celkem 8 průběhů penalizační funkce:

- pro algoritmus ICP
- pro algoritmus IMRP

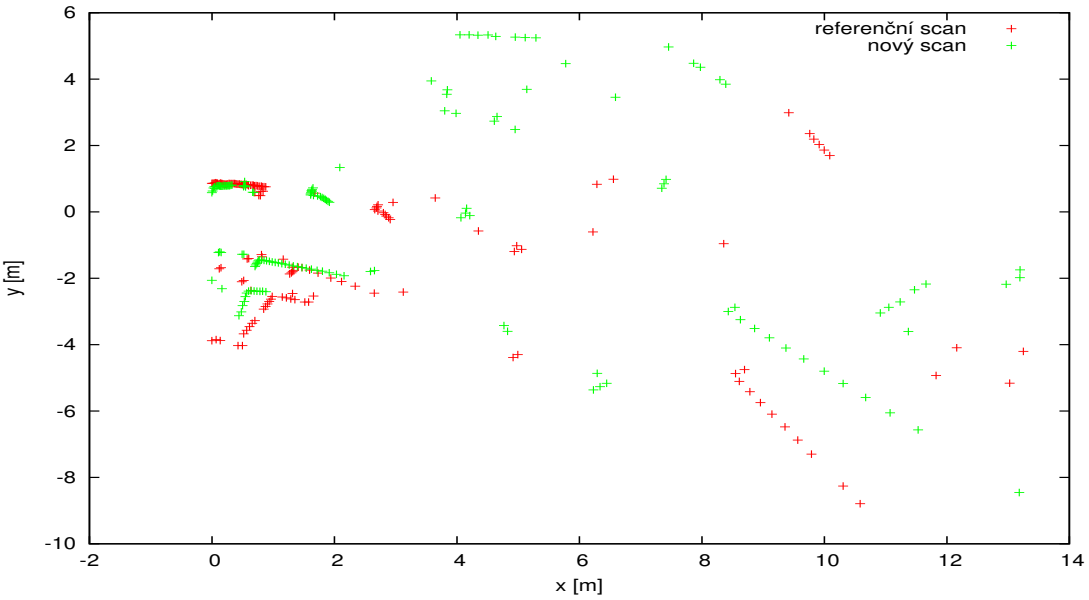
- pro body nalezené algoritmem ICP při lokalizaci pomocí IDC
- pro body nalezené algoritmem IMRP při lokalizaci pomocí IDC
- pro algoritmus ICP s propojováním referenčního scanu
- pro algoritmus IMRP s propojováním referenčního scanu
- pro body nalezené algoritmem ICP při lokalizaci pomocí IDC s propojováním referenčního scanu
- pro body nalezené algoritmem IMRP při lokalizaci pomocí IDC s propojováním referenčního scanu

Tyto průběhy budou vykresleny pro registraci tří dvojic scanů bez znalosti odometrie. Jedna dvojice z těchto scanů je typickým příkladem hodnot naměřených v kancelářském prostředí, jedna v málo členitém prostředí a jedna dvojice umožňuje vznik většího množství falešných bodových párů. Tyto tři dvojice scanů jsou znázorněny na obrázcích 4.1, 4.2 a 4.3. Provedeno bude vždy maximálně 400 iterací každého algoritmu, pokud výsledek nezkonverguje dříve (t.j. nebudou nalezeny dvě téměř stejné transformace po sobě).

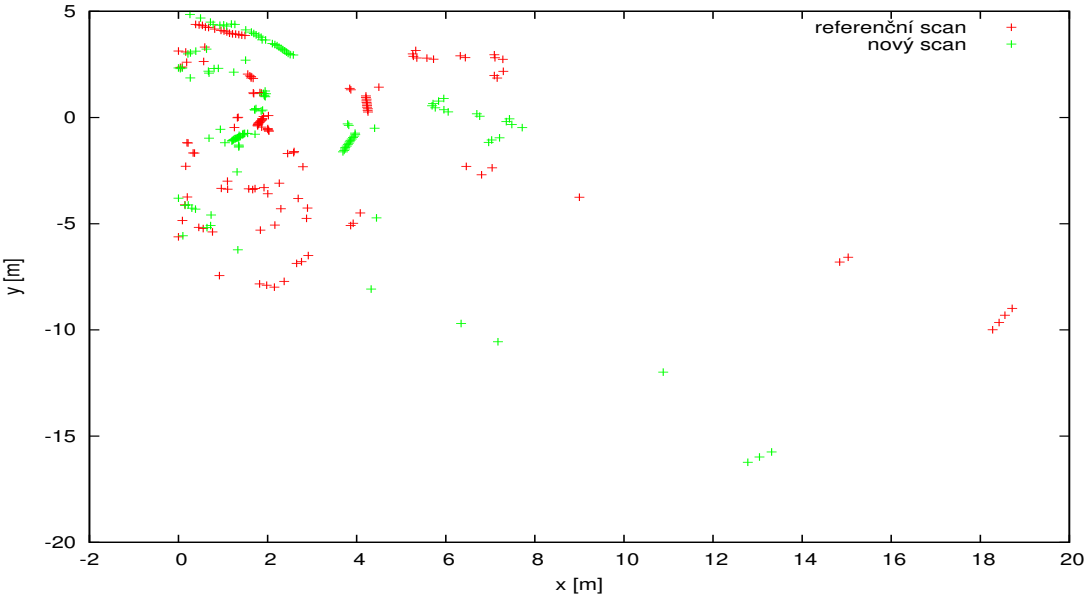
Parametry nastavení driveru společné pro všechny testované algoritmy a jejich varianty jsou:

název parametru	hodnota
general_verbose	true
general_logFile	"./prubehLok.log"
general_localizeToKnownMap	false
general_maxLaserDistance	75
general_minPtsNrAfterMaxLaserDistance	5
general_acceptedPtsRatio	0.75
general_maxNrIterations	400
general_toleranceX,Y,A	10^{-12}
imrp_initialSectorSize	0.7
imrp_sectorDecreaseConstant	0.003
imrp_minClosestPtsFoundRatio	0.3

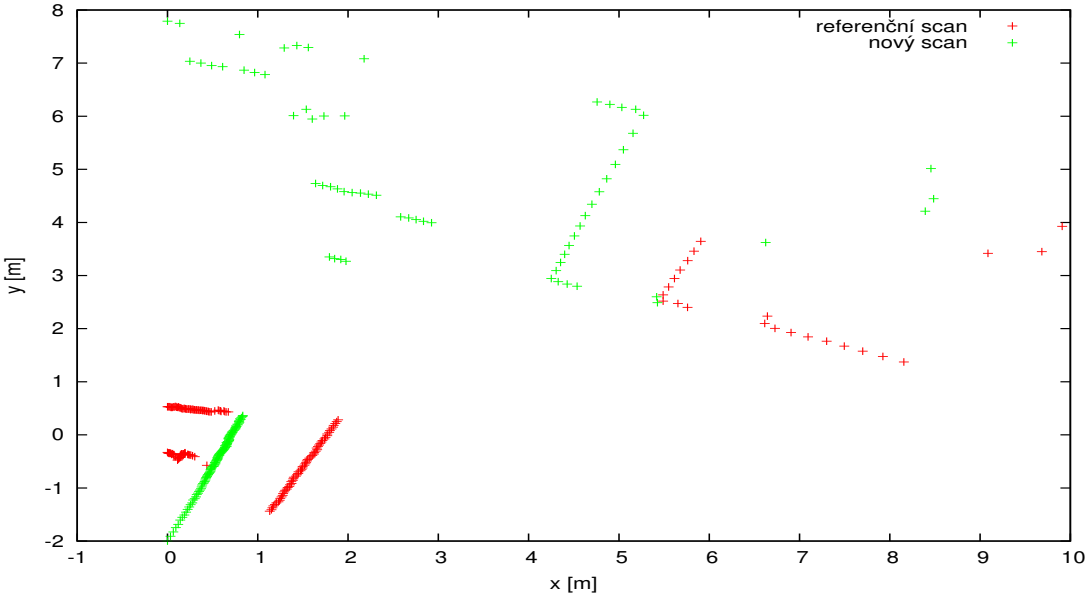
Výsledné grafy závislosti hodnoty penalizační funkce na počtu iterací jsou na obrázcích 4.4, 4.5 a 4.6.



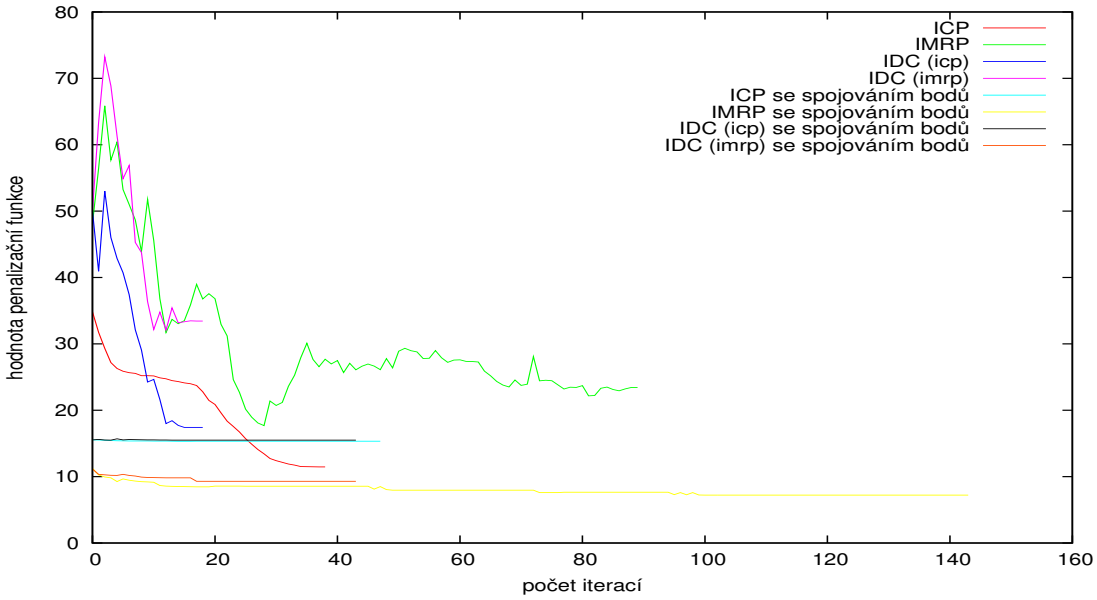
Obrázek 4.1: Dvojice scanů změřená v málo členitém prostředí.



Obrázek 4.2: Dvojice scanů změřená v členitém prostředí.



Obrázek 4.3: Dvojice scanů umožňující vznik velkého množství falešných bodových párů.

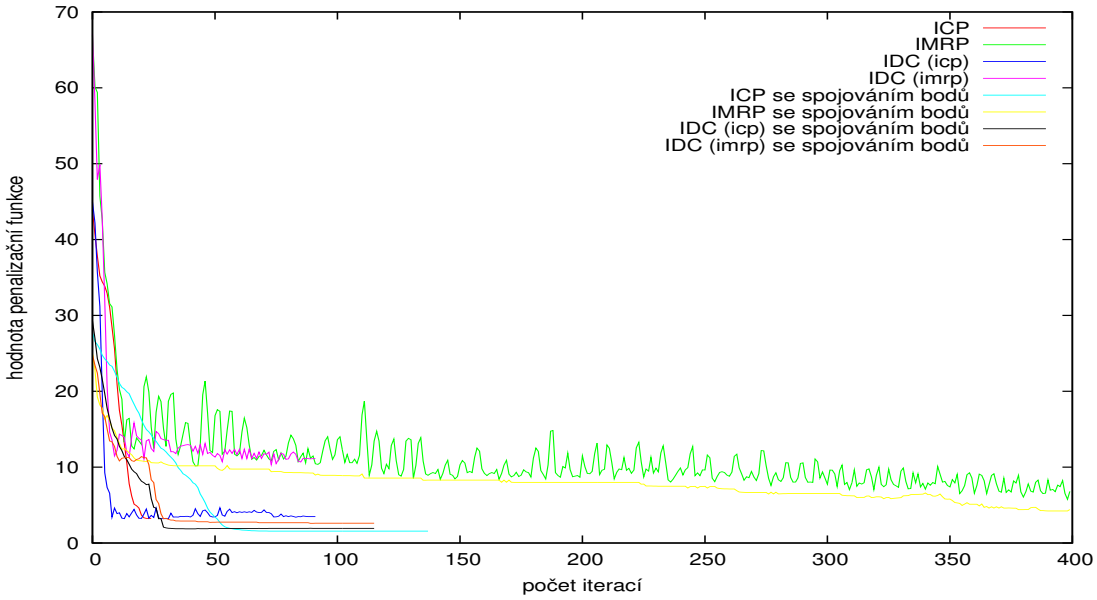


Obrázek 4.4: Závislost hodnoty penalizační funkce na počtu iterací (málo členité prostředí).

Z naměřených grafů je vidět, že varianty algoritmů ICP a IMRP, které registrují body nového scanu vůči úsečkám propojujícím body referenčního scanu, umožňují dosáhnout nižší hodnoty penalizační funkce než odpovídající algoritmy, které referenční body nepropojují. To je dáno tím, že algoritmus, který body nepropojuje, má možnost vybrat korespondující body pouze z konečné množiny bodů referenčních, zatímco algoritmus, který body propojuje, vybírá z nekonečného množství bodů ležících na úsečkách.

Dále si lze všimnout, že grafy pro algoritmus ICP (jak s propojováním, tak bez něj) jsou nerostoucí funkce. Pokud by nebyla použita metoda odebírání bodových párů, pak by tato vlastnost musela platit pro každou zvolenou dvojici scanů, neboť v [2] bylo teoreticky dokázáno, že algoritmus ICP konverguje do lokálního minima penalizační funkce. Odebírání falešných bodových párů, které není součástí algoritmu ICP, však způsobí, že penalizační funkce může být v určitých případech i rostoucí. Pro ostatní algoritmy (IMRP a IDC) může být penalizační funkce rostoucí už jen proto, že se pokaždé nevypočítává ze stejného množství bodových párů (pokud ve výseči popsané v 2.3 nebude nalezen referenční bod, počet párů se sníží a naopak).

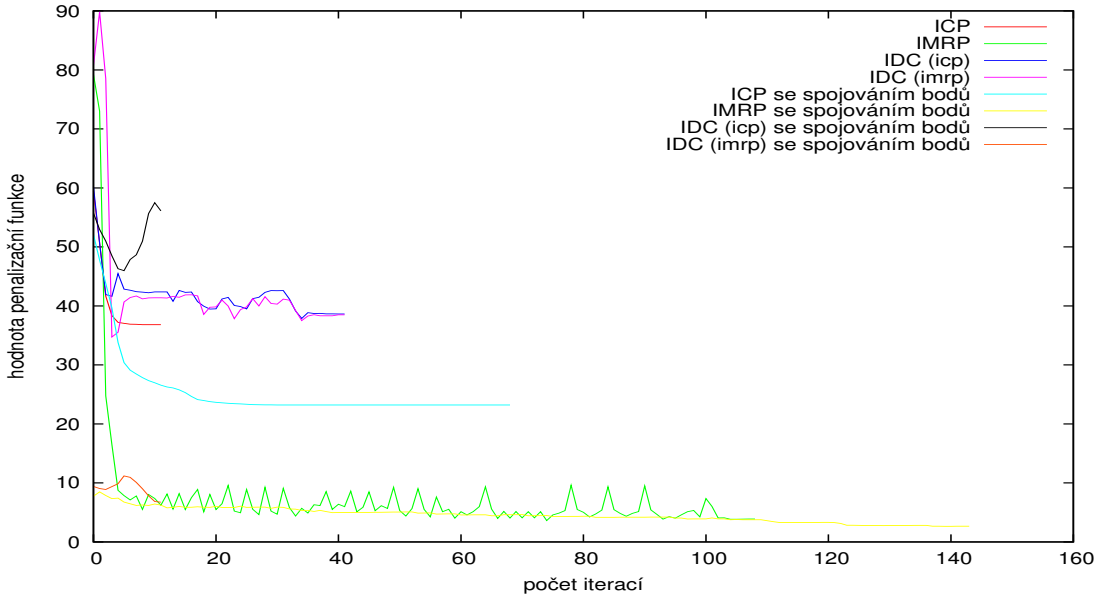
Na všech grafech je také patrné, že hodnoty penalizační funkce pro algoritmus IMRP a pro body nalezené algoritmem IMRP při lokalizaci pomocí IDC jsou v počátečních iteracích vyšší než pro ostatní algoritmy. To je způsobeno implementací algoritmu IMRP pro lokalizaci bez propojování referenčního scanu, která je odlišná od ostatních algoritmů. U ostatních algoritmů jsou totiž při odstraňování falešných bodových párů vybírány ty s největší euklidovskou vzdáleností, zatímco u IMRP jsou vybrány ty, jejichž rozdíl vzdáleností od počátku O je největší. Snadno se pak může stát, že budou za platné považovány páry s malým rozdílem vzdáleností od počátku, ale s velkou vzdáleností euklidovskou.



Obrázek 4.5: Závislost hodnoty penalizační funkce na počtu iterací (členité prostředí).

Při porovnání rychlosti konvergence algoritmů je patrné, že ve většině případů konvergují složky algoritmů IDC rychleji, než pokud by byly algoritmy použity samostatně (výjimkou je pouze algoritmus ICP bez propojování referenčního scanu).

Z naměřených grafů je dále vidět, že pokud bude maximální počet iterací omezen na cca 70 (pro IDC na 20), nebude výsledek lokalizace příliš ovlivněn, protože hodnota penalizační funkce po této hranici již výrazněji neklesá.



Obrázek 4.6: Závislost hodnoty penalizační funkce na počtu iterací (scany s velkým množstvím falešných bodových párů).

4.2 Testování lokalizace bez známé mapy na datech z repozitáře OpenSLAM.org

Cílem tohoto testu je ověřit funkčnost algoritmů na reálných datech a porovnat úspěšnost jednotlivých algoritmů při lokalizaci robotu.

Pro testování bylo použito 8 souborů se záznamem naměřených dat z repozitáře OpenSLAM.org (konkrétně ze stránky [7]). OpenSLAM.org je repozitář sloužící k podpoře výzkumu algoritmů řešících úlohu současné lokalizace a mapování (**s**imultaneous **l**ocalization **a**nd **m**apping). Lze do něj uložit vlastní implementaci algoritmů, prezentaci výsledků výzkumu či roboty nahraná data a poskytnout je tak ostatním. V této práci využijeme toho, že některé poskytované záznamy dat ze senzorů na robotu byly již kvalitně lokalizovány, pozici určenou autorem dat tedy budeme považovat za referenční.

Testování probíhalo tak, že získané soubory byly nejprve upraveny do formátu, který je schopen zpracovat driver readlog. Vybraná data z OpenSLAM.org byla totiž uložena ve formátu, který používá systém Carmen (více viz [4]), navíc všechny datové soubory nebyly uloženy ve formátu přesně stejném, bylo tedy nutné vytvořit pro každý soubor speciální transformační skript. Transformované soubory byly dále zkráceny tak, aby simulace trvala vždy cca 10 minut. Poté byly tímto driverem postupně „přehrávány“ a simulovaná senzorická měření byla lokalizačním driverem zpracována. Data uložená v driverem vytvořeném souboru se záznamem průběhu lokalizace byla po skončení simulace analyzována.

Pro získání prvotního odhadu o kvalitě lokalizace byla vygenerována mapa okolního prostředí robotu tak, že každý dálkoměrem naměřený scan byl umístěn do polohy vypočítané algoritmem. Čím méně se takto získaná mapa liší od mapy získané umístěním scanů do referenční polohy, tím kvalitnější lokalizace je. Bohužel tato metoda může sloužit jen jako prvotní odhad (grafické porovnání), protože možností, jak definovat míru odlišnosti map, je mnoho.

Další metoda určení kvality lokalizace spočívá ve výpočtu vzdálenosti algoritmem určené pozice od pozice referenční. Tato metoda již může sloužit jako ukazatel kvality lokalizace, má však podstatnou nevýhodu. Pokud totiž v jediném kroku lokalizace určí algoritmus chybně pozici nebo úhel robotu, tato chyba se projeví i v dalších odhadovaných polohách a záleží na parametrech mapy a tvaru dráhy robotu, jak moc se tato chyba projeví. Tato metoda tedy pro porovnávání kvality lokalizace na různých mapách není vhodná.

Použitá metoda určení kvality lokalizace z předchozí metody vychází, ale neporovnává polohu robotu v každém kroku lokalizace s polohou referenční, ale změnu polohy robotu vzhledem k jeho poloze v předchozím kroku lokalizace určenou algoritmem s odpovídající změnou vypočítanou z referenčních poloh robotu. Každá algoritmem nově určená poloha je tedy přepočítána do souřadného systému s počátkem v předchozí algoritmem určené poloze a výsledek je porovnán s odpovídající referenční polohou robotu přepočítanou do souřadného systému s počátkem v odpovídající předchozí referenční poloze robotu. Čím menší je potom rozdíl vypočítaných změn polohy, tím je kvalita lokalizace větší. Co je ale hlavní, nepřesnost určení polohy v jednom kroku lokalizace se neprojeví v krocích následujících.

Použití této metody určení kvality lokalizace má smysl pouze pro algoritmy, u nichž je registrace nového scanu prováděna pouze vůči jednomu předchozímu scanu, t.j. bez vlivu scanů naměřených před předchozím scanem. Tuto podmínku ale všechny implementované algoritmy splňují, proto může být popsaná metoda použita.

Upravené soubory z OpenSLAM.org byly lokalizovány postupně algoritmy ICP, ICP s propojováním referenčních bodů, IMRP, IMRP s propojováním referenčních bodů, IDC a IDC s propojováním referenčních bodů. Ostatní parametry lokalizace byly následující:

název parametru	hodnota
general_verbose	true
general_localizeToKnownMap	false
general_maxLaserDistance	(různý pro jednotlivé mapy)
general_minPtsNrAfterMaxLaserDistance	5
general_acceptedPtsRatio	0.75
general_maxNrIterations	400
general_toleranceX,Y,A	10^{-12}
imrp_initialSectorSize	0.3
imrp_sectorDecreaseConstant	0.003
imrp_minClosestPtsFoundRatio	0.3.

Výsledky tohoto testu jsou uspořádány do šesti tabulek (4.3-4.8), každá z těchto tabulek obsahuje následující sloupce

č.m.
: Číslo mapy tak, jak je uvedeno v tab. 4.1.

mapa
: Mapa získaná umístěním každého dálkoměrem naměřeného scanu do polohy vypočítané algoritmem. Mapy pro srovnání (získané umístěním každého dálkoměrem naměřeného scanu do referenční polohy) jsou v tabulce 4.1.

rozdíly pozice
: Rozdíl mezi změnou pozice robotu určenou algoritmem a odpovídající změnou pozice robotu vypočítanou z referenčních poloh robotu. Změna pozice určená algoritmem (Δx_a,Δy_a,$\Delta \varphi_a$) je reprezentována bodem o souřadnicích $[\Delta x_a, \Delta y_a]$, referenční změna pozice (Δx_r,Δy_r,$\Delta \varphi_r$) je reprezentována bodem o souřadnicích $[\Delta x_r, \Delta y_r]$. Graf v tomto sloupci zobrazuje euklidovskou vzdálenost těchto bodů pro každou dvojici lokalizovaných scanů.

rozdíly úhlu
: Rozdíl mezi změnou úhlu robotu určenou algoritmem a odpovídající změnou úhlu robotu vypočítanou z referenčních poloh robotu. Graf v tomto sloupci zobrazuje hodnotu $\Delta \varphi_a - \Delta \varphi_r$ pro každou dvojici lokalizovaných scanů.

Kvalita lokalizace pro jednotlivé algoritmy je shrnuta v tabulce 4.2, která má následující sloupce

č.m.
: Číslo mapy tak, jak je uvedeno v tab. 4.1.

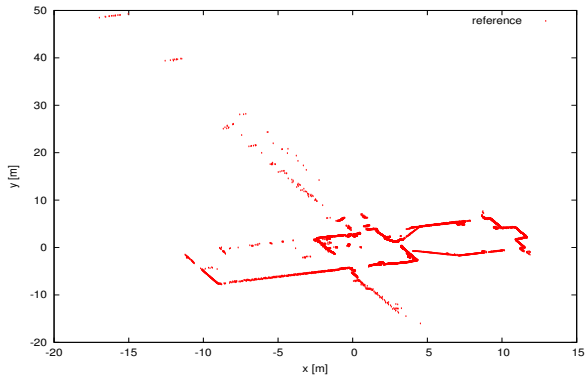
metoda
: Zkratka názvu použité metody (*propoj.* znamená propojování referenčních bodů úsečkami).

x
: Průměr absolutních hodnot výrazu $\Delta x_a - \Delta x_r$ přes všechny dvojice lokalizovaných scanů.

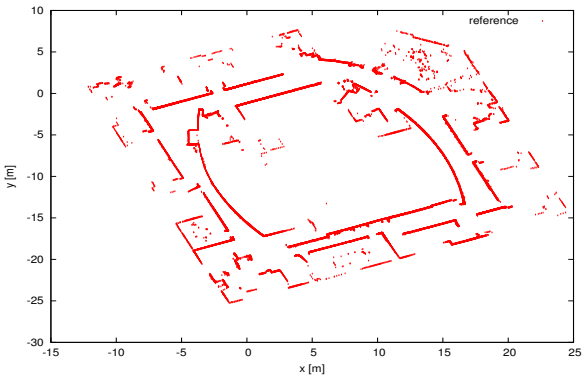
y
: Průměr absolutních hodnot výrazu $\Delta y_a - \Delta y_r$ přes všechny dvojice lokalizovaných scanů.

pozice
: Průměr hodnot výrazu $(\Delta x_a - \Delta x_r)^2 + (\Delta y_a - \Delta y_r)^2$ přes všechny dvojice lokalizovaných scanů.

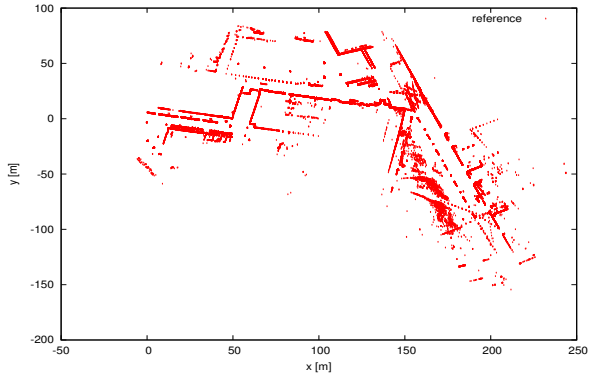
φ
: Průměr absolutních hodnot výrazu $\Delta\varphi_a - \Delta\varphi_r$ přes všechny dvojice lokalizovaných scanů.



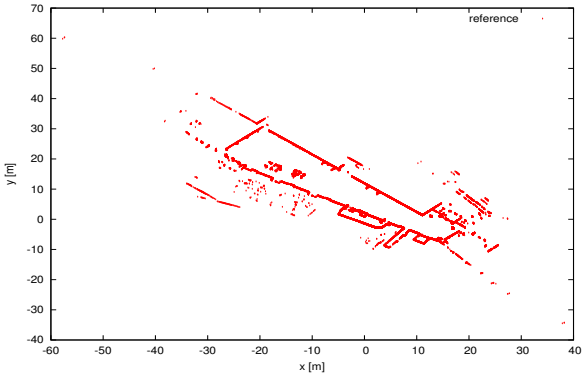
1) Belgioioso Castle.



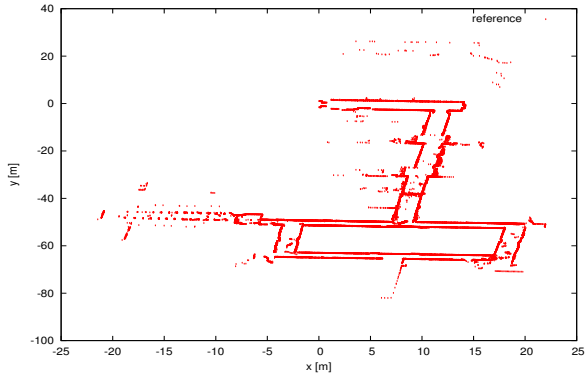
2) Intel Research Lab.



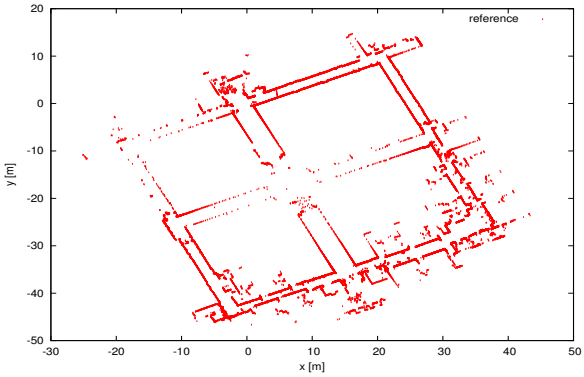
3) Freiburg Campus.



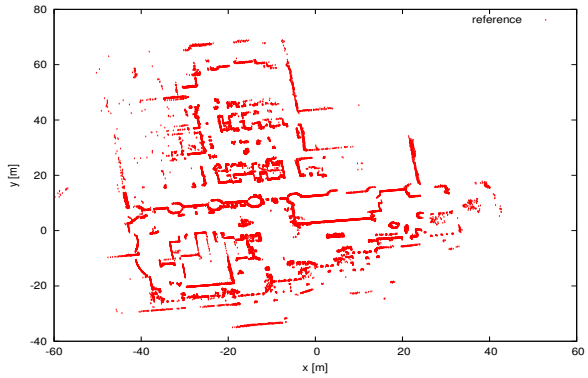
4) Freiburg Building 101.



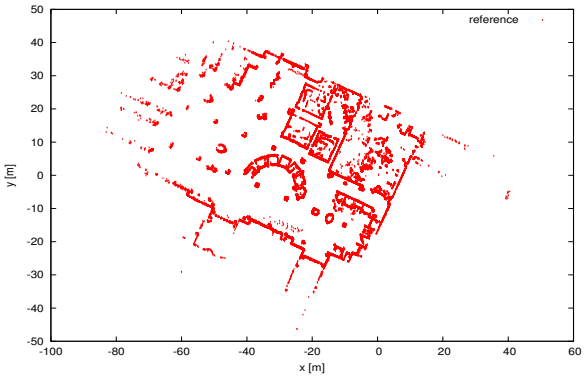
5) MIT Infinite Corridor Dataset.



6) ACES3, Austin.



7) Edmonton.

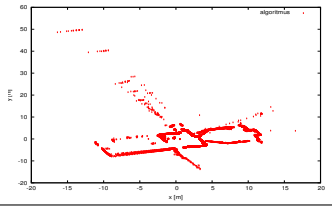
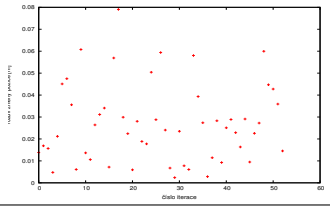
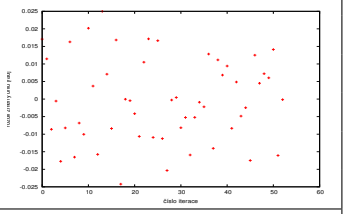
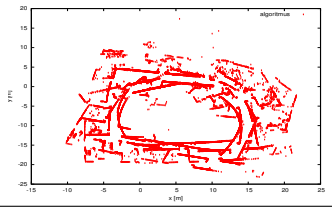
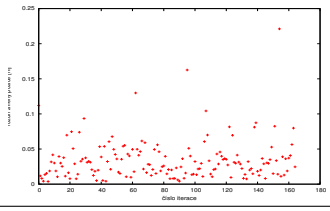
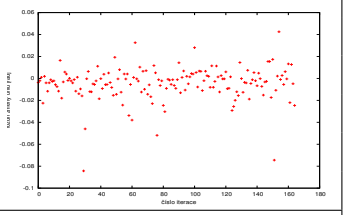
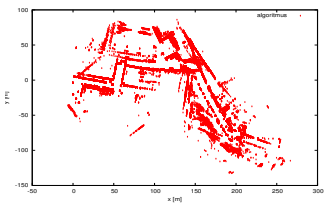
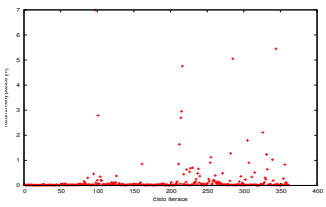
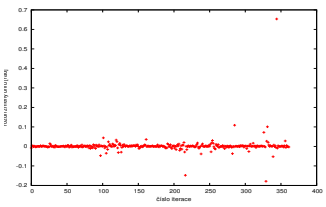
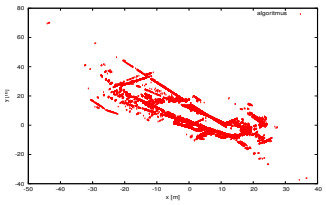
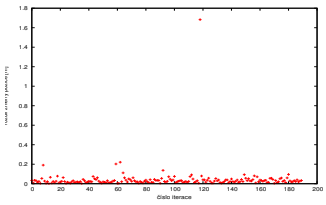
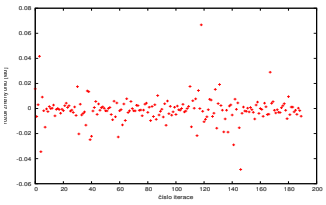
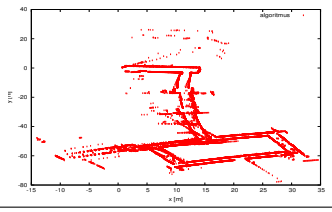
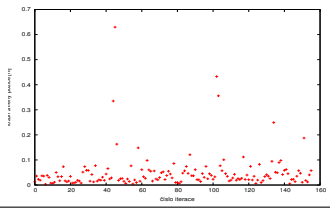
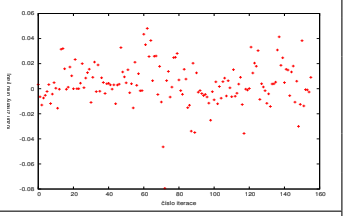
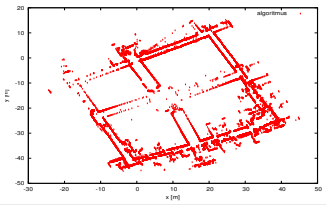
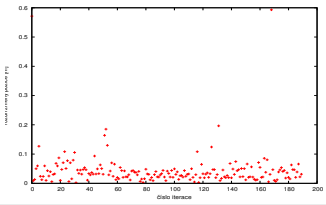
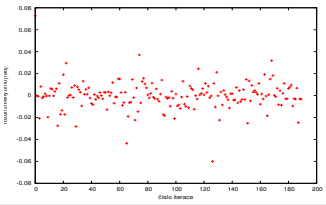
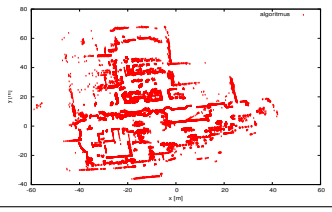
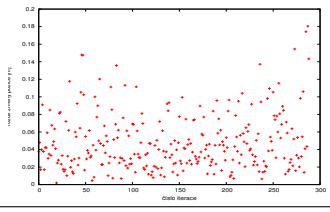
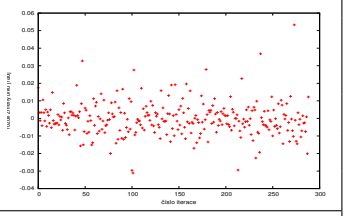
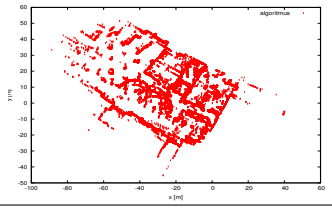
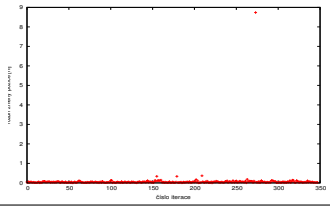
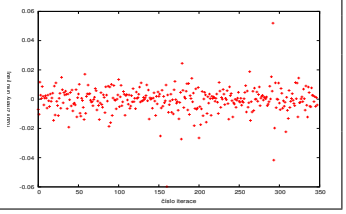


8) Acapulco Convention Center, Mexico.

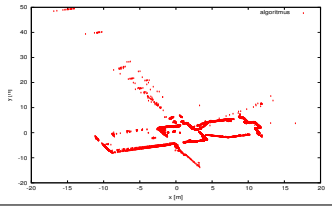
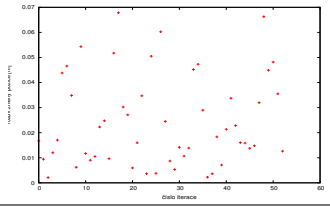
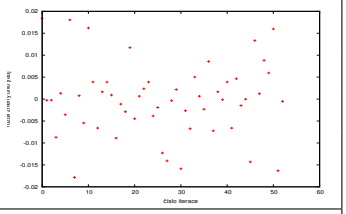
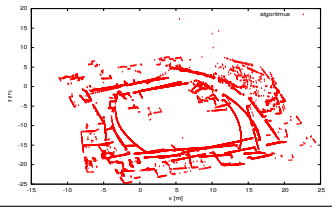
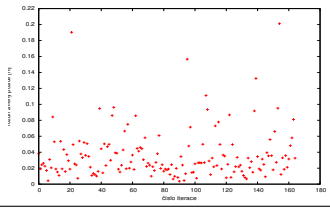
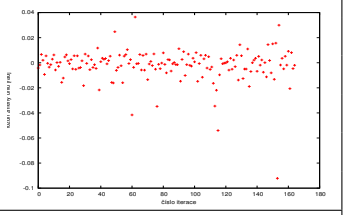
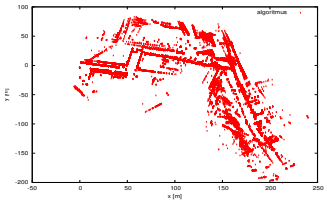
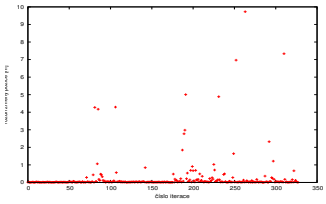
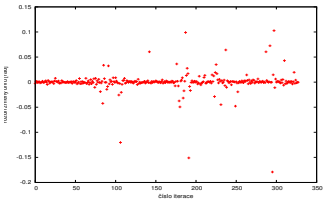
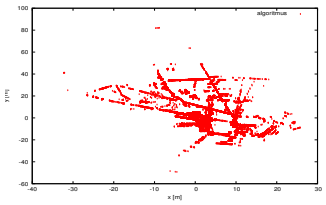
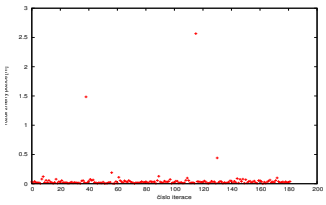
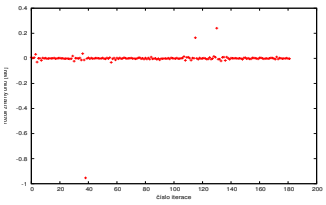
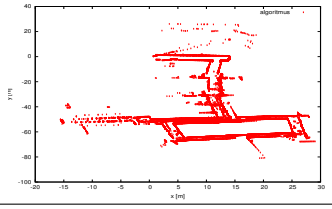
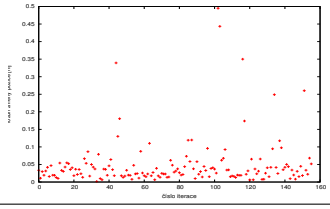
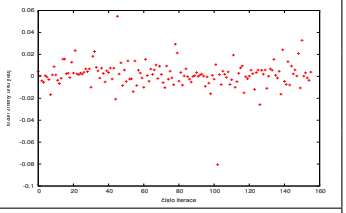
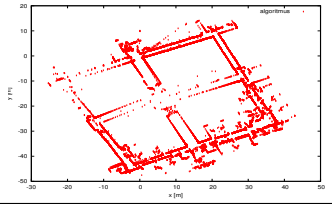
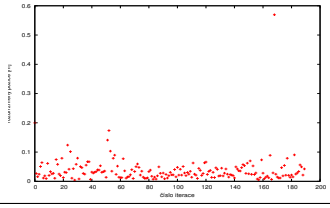
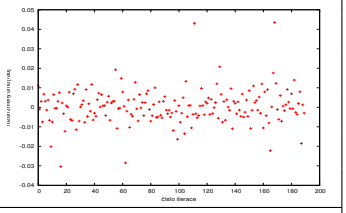
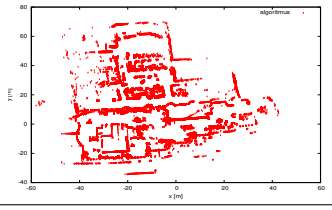
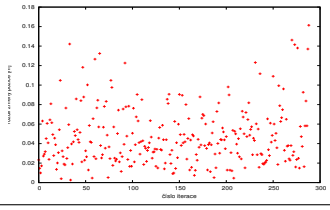
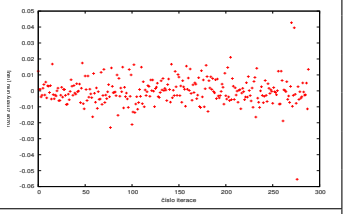
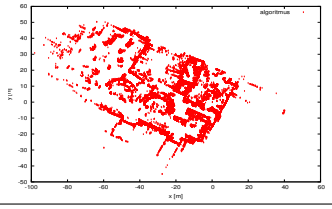
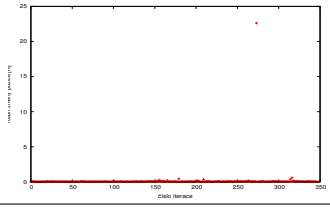
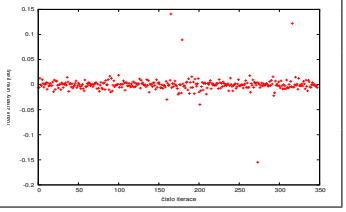
Tabulka 4.1: Mapy získané posunutím každého scanu do referenční polohy. U každé mapy je uvedeno i její číslo použité v tabulkách 4.3 - 4.8.

č.m.	**metoda**	**x**	**y**	**pozice**	φ
1	ICP	0.015101	0.017899	0.026687	0.009959
	ICP propoj.	0.015293	0.015221	0.024350	0.006082
	IDC	0.056913	0.047792	0.080796	0.016238
	IDC propoj.	0.065562	0.046425	0.091894	0.023472
	IMRP	0.082210	0.106677	0.143506	0.025326
	IMRP propoj.	0.096930	0.128030	0.181811	0.043318
2	ICP	0.022611	0.020168	0.034210	0.010046
	ICP propoj.	0.025887	0.019590	0.036264	0.007565
	IDC	0.024933	0.022751	0.038102	0.013719
	IDC propoj.	0.053843	0.154416	0.180786	0.032701
	IMRP	0.155432	0.059695	0.182679	0.024177
	IMRP propoj.	0.352539	0.122038	0.400855	0.015542
3	ICP	0.149779	0.081463	0.187836	0.008659
	ICP propoj.	0.205343	0.104141	0.256224	0.007040
	IDC	0.255832	0.115113	0.303495	0.025759
	IDC propoj.	1.115929	0.749969	1.440563	0.052663
	IMRP	0.645247	0.299517	0.762211	0.028153
	IMRP propoj.	0.562732	0.308955	0.704023	0.026156
4	ICP	0.021042	0.030518	0.042123	0.006110
	ICP propoj.	0.033445	0.038905	0.056417	0.012003
	IDC	0.103520	0.119666	0.177831	0.025665
	IDC propoj.	0.233835	0.178215	0.340390	0.073844
	IMRP	0.127443	0.085052	0.168460	0.021977
	IMRP propoj.	0.149339	0.119572	0.213757	0.022212
5	ICP	0.037307	0.023548	0.050148	0.012243
	ICP propoj.	0.035664	0.025873	0.049866	0.007418
	IDC	0.046388	0.026340	0.061075	0.025463
	IDC propoj.	0.630283	0.510455	0.920274	0.082587
	IMRP	0.254581	0.152533	0.350049	0.046441
	IMRP propoj.	0.293373	0.055054	0.316623	0.035747
6	ICP	0.030286	0.024974	0.044750	0.008310
	ICP propoj.	0.028010	0.019507	0.038192	0.006072
	IDC	0.037806	0.028127	0.053567	0.013498
	IDC propoj.	0.120066	0.152906	0.202102	0.014952
	IMRP	0.141823	0.040841	0.160315	0.016353
	IMRP propoj.	0.213024	0.071509	0.242753	0.011838
7	ICP	0.028577	0.032974	0.048228	0.006748
	ICP propoj.	0.030321	0.029807	0.047213	0.005847
	IDC	0.032804	0.035802	0.053557	0.010934
	IDC propoj.	0.037607	0.035962	0.058090	0.007851
	IMRP	0.113317	0.109106	0.177171	0.015661
	IMRP propoj.	0.174810	0.126635	0.243142	0.012895
8	ICP	0.032907	0.053279	0.072152	0.005741
	ICP propoj.	0.055861	0.090885	0.115093	0.006508
	IDC	0.042507	0.071607	0.093678	0.010233
	IDC propoj.	0.060077	0.078546	0.109338	0.008156
	IMRP	0.094491	0.090794	0.145601	0.016178
	IMRP propoj.	0.132674	0.087852	0.180833	0.013895

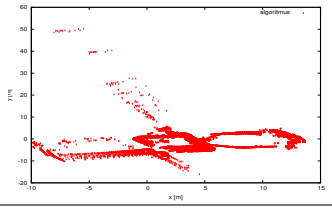
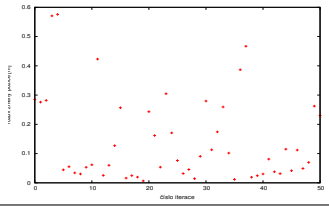
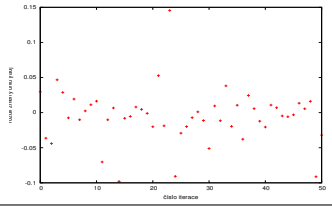
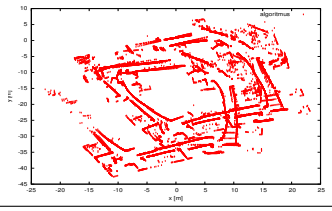
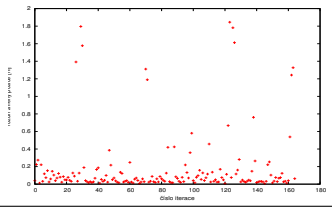
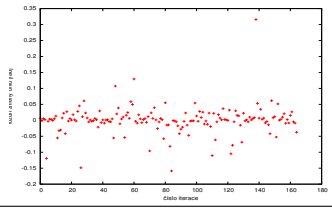
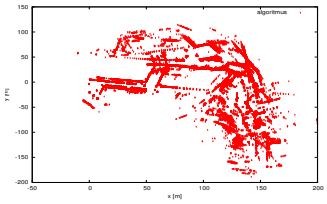
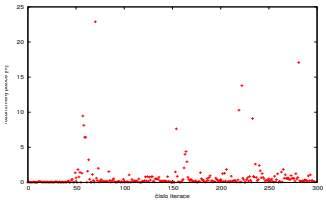
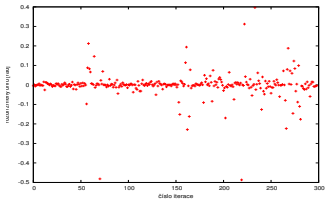
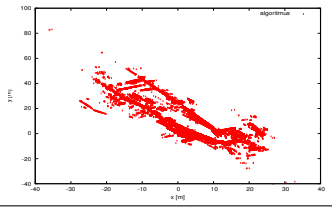
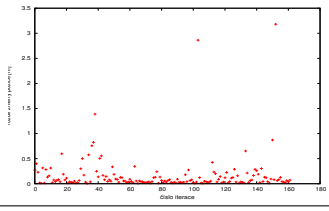
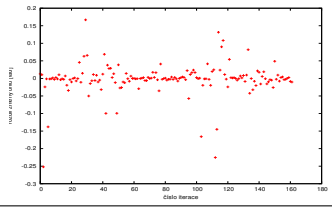
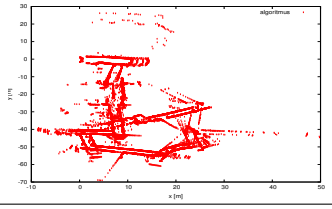
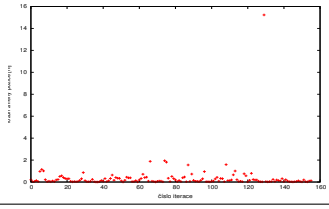
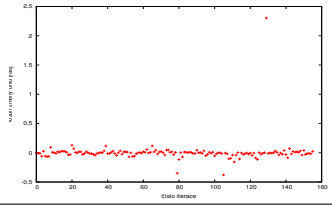
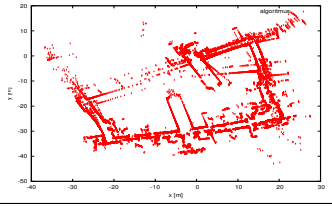
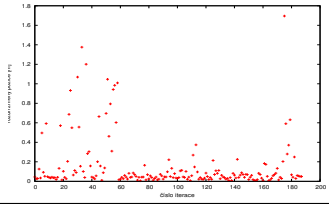
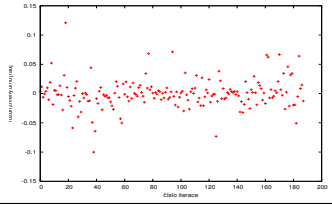
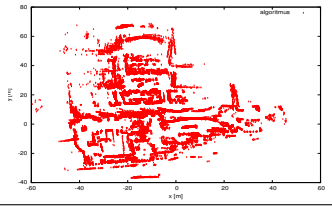
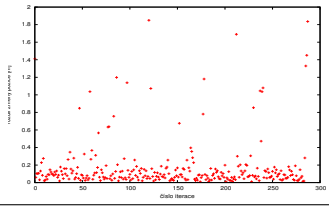
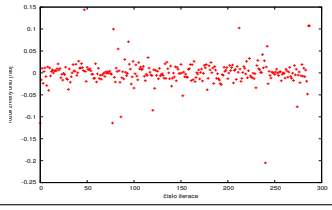
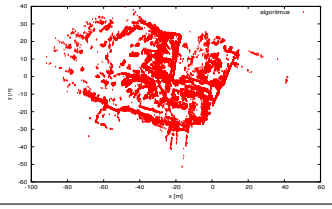
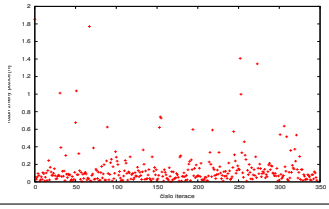
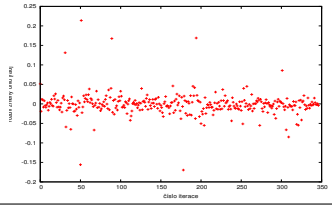
Tabulka 4.2: Parametry vypovídající o kvalitě lokalizace pomocí jednotlivých algoritmů.

č.m.	mapa	rozdíly pozice	rozdíly úhlu
1			
2			
3			
4			
5			
6			
7			
8			

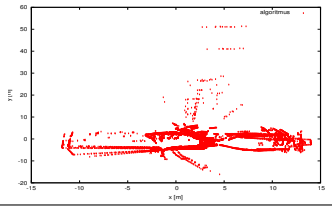
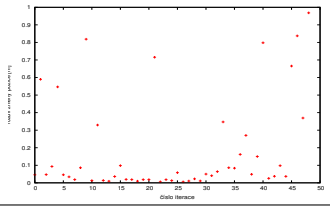
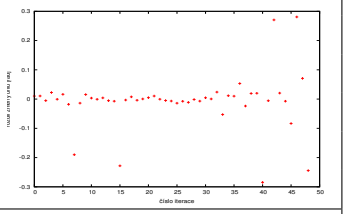
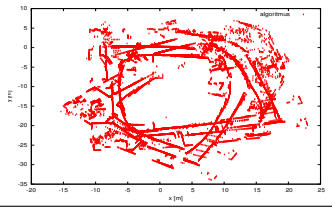
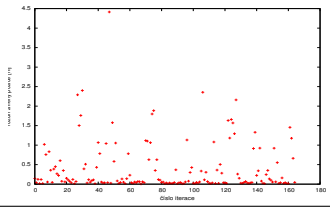
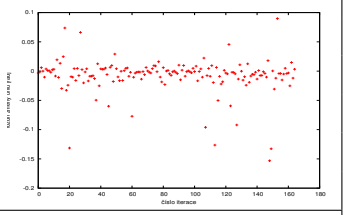
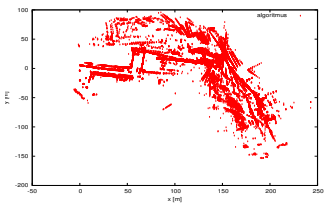
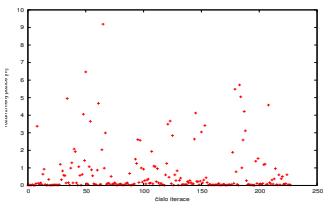
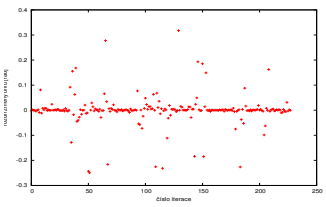
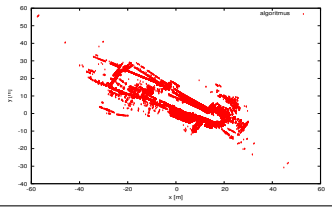
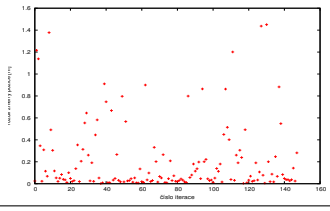
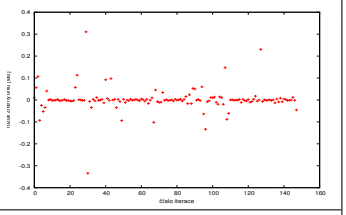
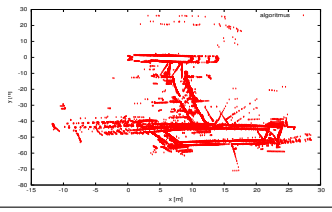
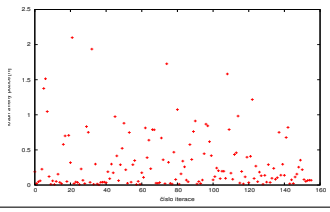
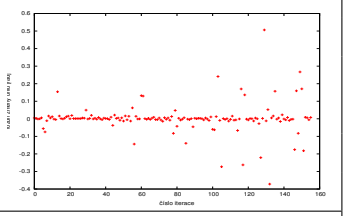
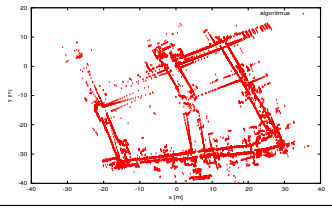
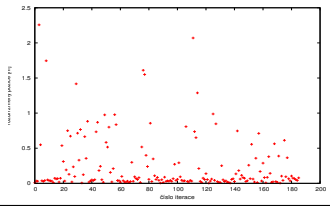
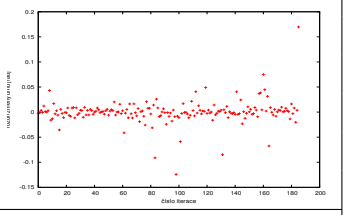
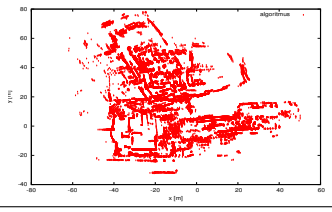
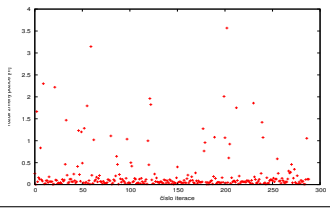
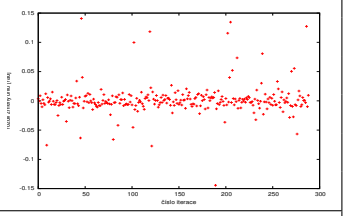
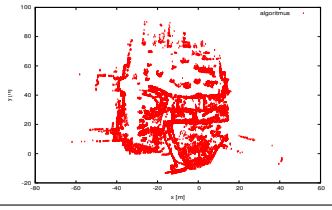
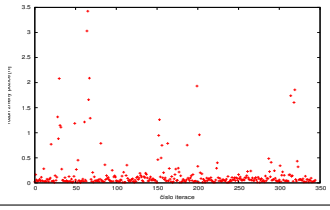
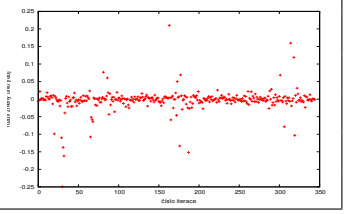
Tabulka 4.3: Výsledky lokalizace pomocí algoritmu ICP bez propojování referenčních bodů.

č.m.	mapa	rozdíly pozice	rozdíly úhlu
1			
2			
3			
4			
5			
6			
7			
8			

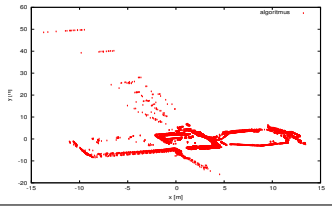
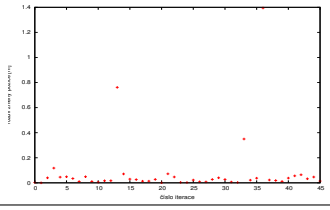
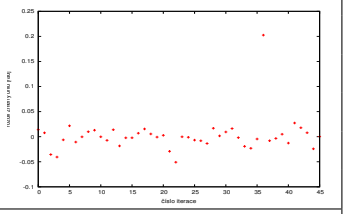
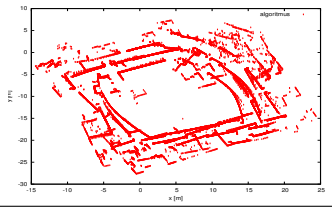
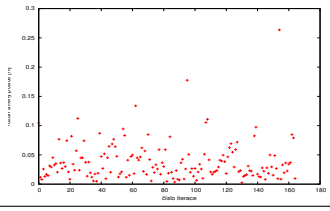
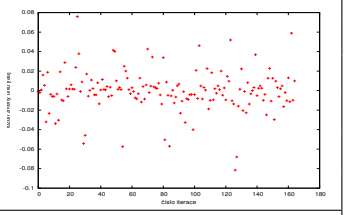
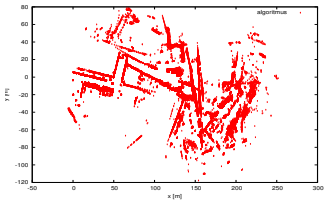
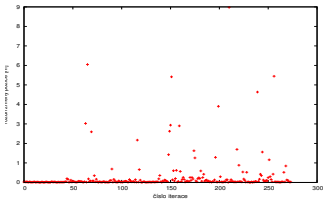
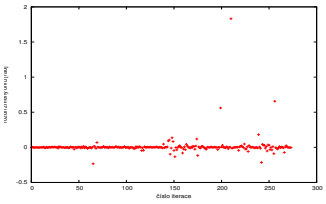
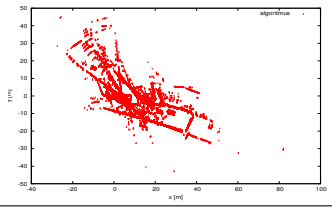
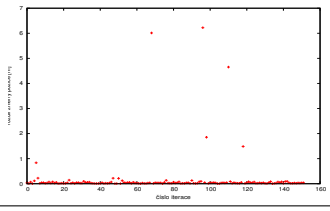
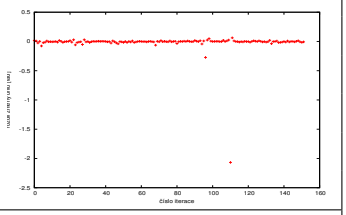
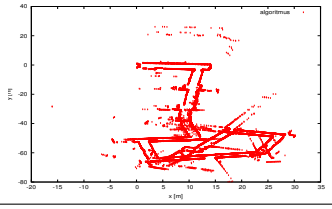
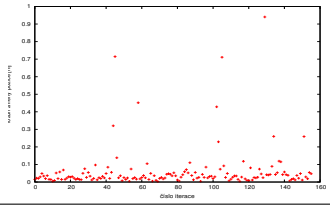
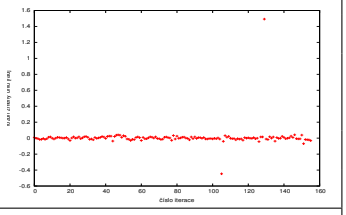
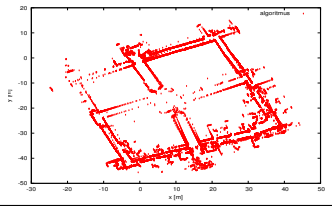
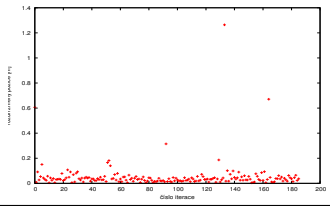
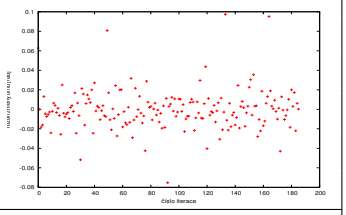
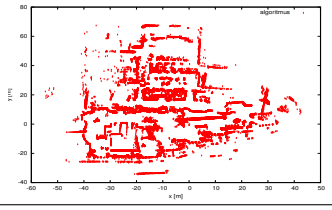
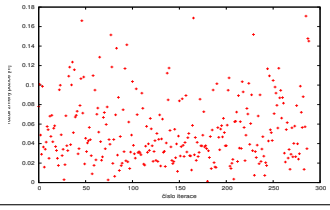
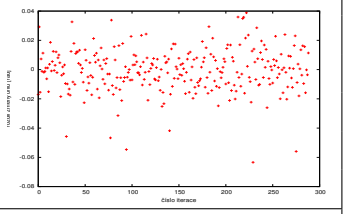
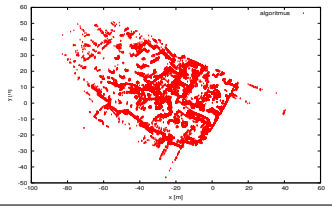
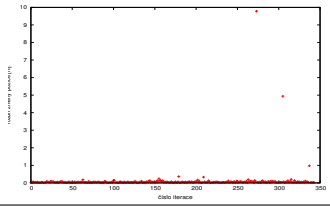
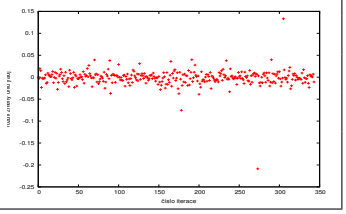
Tabulka 4.4: Výsledky lokalizace pomocí algoritmu ICP s propojováním referenčních bodů.

č.m.	mapa	rozdíly pozice	rozdíly úhlu
1			
2			
3			
4			
5			
6			
7			
8			

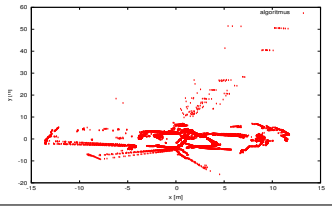
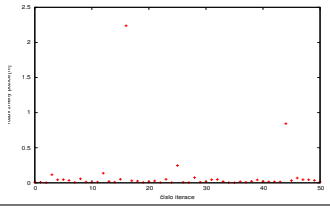
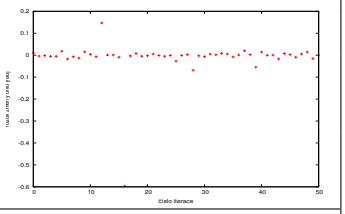
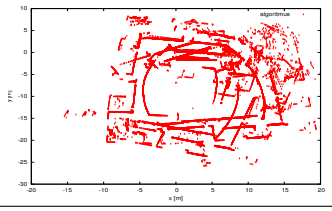
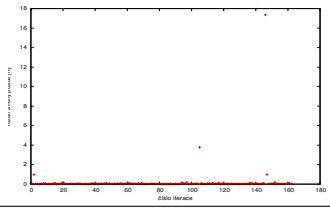
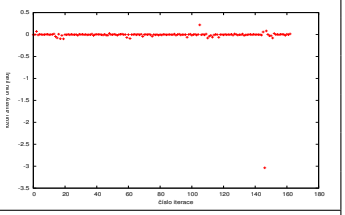
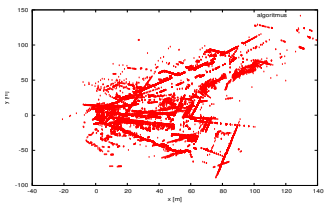
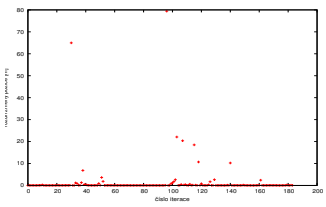
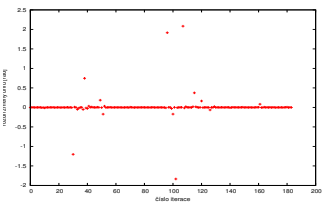
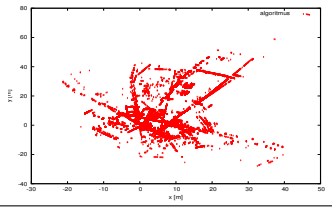
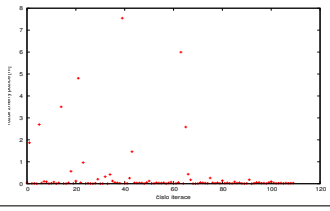
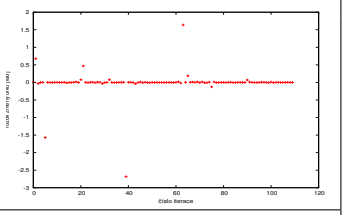
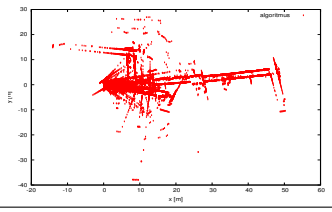
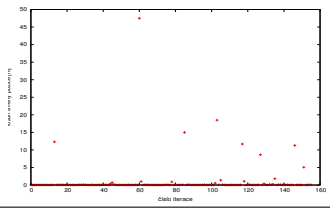
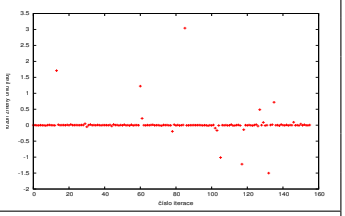
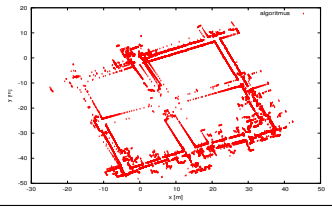
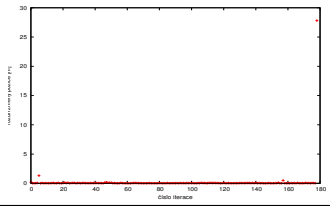
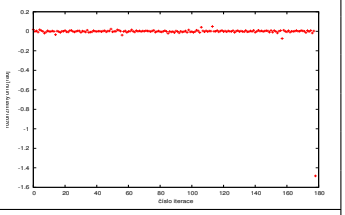
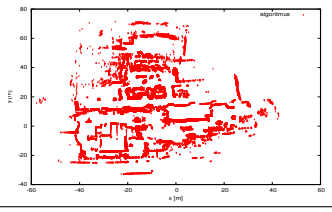
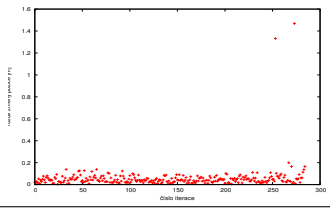
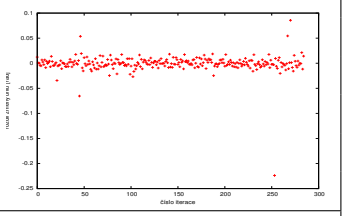
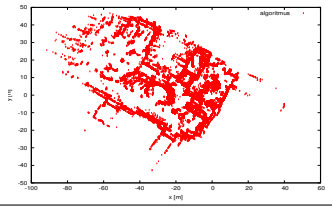
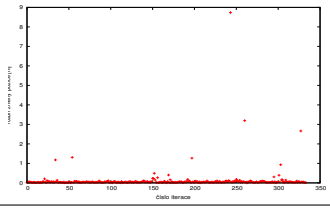
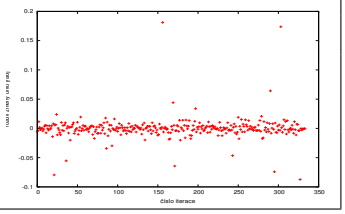
Tabulka 4.5: Výsledky lokalizace pomocí algoritmu IMRP bez propojování referenčních bodů.

č.m.	mapa	rozdíly pozice	rozdíly úhlu
1			
2			
3			
4			
5			
6			
7			
8			

Tabulka 4.6: Výsledky lokalizace pomocí algoritmu IMRP s propojováním referenčních bodů.

č.m.	mapa	rozdíly pozice	rozdíly úhlu
1			
2			
3			
4			
5			
6			
7			
8			

Tabulka 4.7: Výsledky lokalizace pomocí algoritmu IDC bez propojování referenčních bodů.

č.m.	mapa	rozdíly pozice	rozdíly úhlu
1			
2			
3			
4			
5			
6			
7			
8			

Tabulka 4.8: Výsledky lokalizace pomocí algoritmu IDC s propojováním referenčních bodů.

Pokud porovnáme úspěšnost určení pozice robotu pro jednotlivé mapy (t.j. průměr hodnot výrazu $(\Delta x_a - \Delta x_r)^2 + (\Delta y_a - \Delta y_r)^2$ v tab. 4.2 označený jako pozice), můžeme si všimnout, že u mapy „Freiburg Campus“ je tato hodnota pro všechny algoritmy vyšší než u ostatních map. To je způsobeno tím, že se na rozdíl od ostatních záznamů při pořizování těchto dat robot pohyboval mimo budovu, v prostředí, které bylo nestatické, navíc s nerovným a nehomogenním podkladem, jenž způsobil větší odchylky v odometrickém odhadu polohy robotu.

Při porovnání jednotlivých algoritmů mezi sebou se pro lokalizaci robotu ze zvolených datových záznamů jako nejlepší jeví algoritmus ICP s propojováním referenčních bodů úsečkami. Tento algoritmus dosahuje v určení pozice robotu na krok iterace odchylky v řádu setin metru. Tato hodnota je přibližně stejná i pro algoritmus ICP bez propojování referenčních bodů úsečkami, ovšem propojování referenčních bodů u většiny map zlepšilo přesnost určení úhlu robotu, která je při propojování v řádu tisícin radiánu.

Algoritmus IMRP bez propojování referenčních bodů dosahoval v určení pozice robotu ve statickém prostředí odchylky do 0.34 m na krok iterace, což jej činí samostatně prakticky nepoužitelným. Přesnost v určení úhlu robotu je oproti algoritmu ICP bez propojování referenčních bodů přibližně dvakrát horší. Propojováním referenčních bodů došlo sice k mírnému zlepšení odhadu v určení úhlu natočení robotu, ovšem za cenu dalšího zvýšení nepřesnosti v určení polohy.

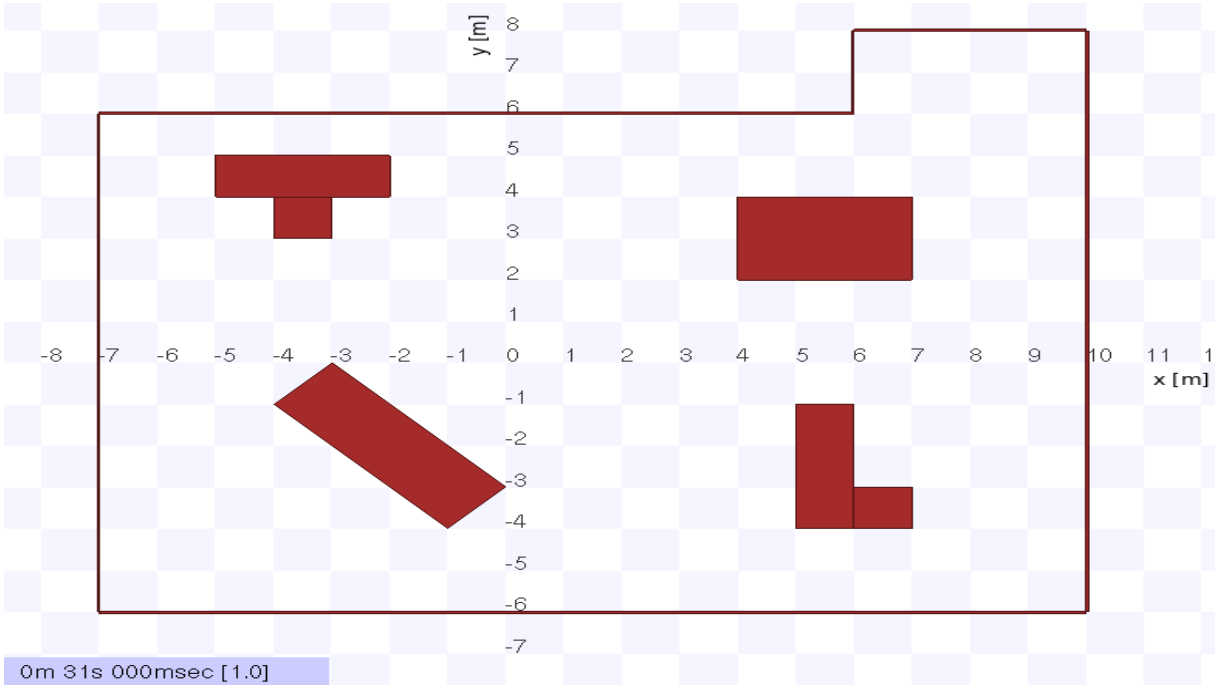
Odchylka algoritmu IDC při určování pozice robotu ve statickém prostředí je v řádu setin metru, odchylka v určení úhlu natočení robotu je ale v řádu setin radiánu, což je horší výsledek než u algoritmu ICP. Lepší výsledky algoritmu ICP jsou však také způsobeny poměrně vysokým maximálním povoleným počtem iterací (general_maxNrIterations = 400). Při nastavení nižšího maximálního počtu iterací by při porovnání dosáhl lepších výsledků algoritmus IDC, protože, jak bylo ukázáno v kapitole 4.1, algoritmus IDC konverguje rychleji než algoritmus ICP. Z tohoto důvodu je při praktickém nasazení driveru, kdy je frekvence příchodu nových scanů v řádu jednotek Hz, vhodnější použít algoritmus IDC.

4.3 Testování kontinuální lokalizace se známou mapou prostředí

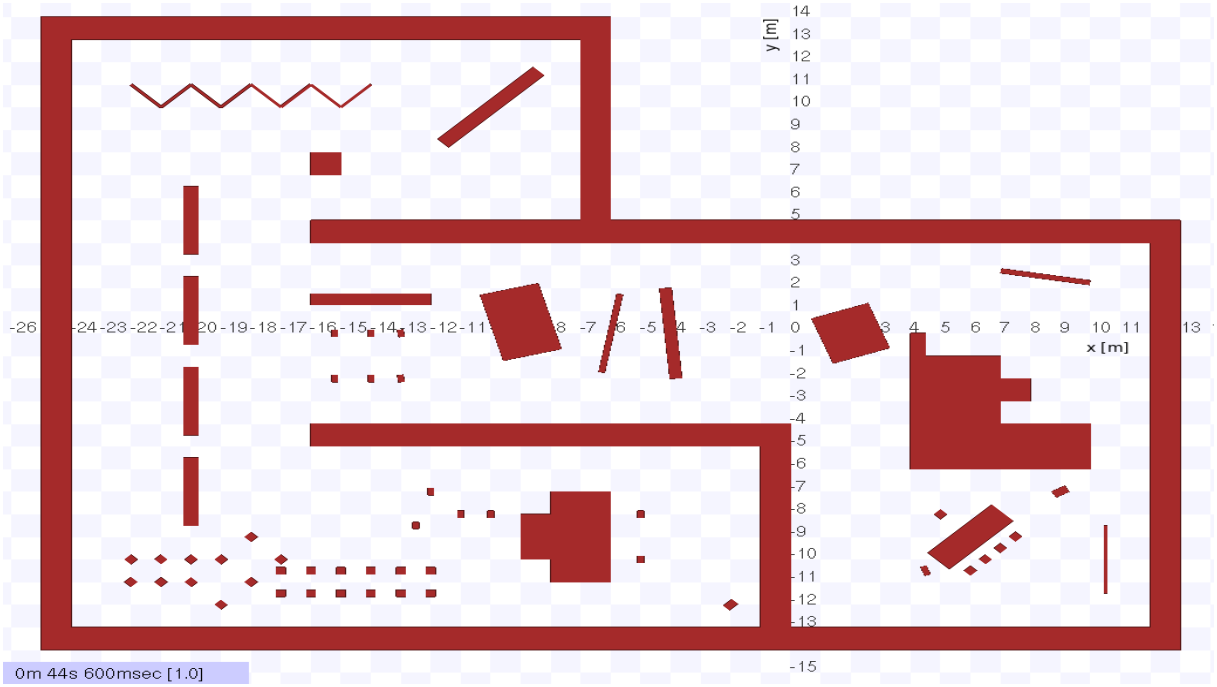
V této kapitole budou srovnány výsledky lokalizace se znalostí mapy okolního prostředí a bez ní. K lokalizaci bude použit algoritmus ICP s propojováním referenčních bodů (nejlepší dle předchozí série testů). Jako vstupní data budou použity záznamy vytvořené pomocí driveru writelog při pohybu simulovaných robotů v prostředích zobrazených na obr. 4.7 a 4.8. Parametry lokalizace jsou tyto:

název parametru	hodnota
general_verbose	true
general_method	ICP
general_localizeToKnownMap	false/true
general_maxLaserDistance	8
general_minPtsNrAfterMaxLaserDistance	5
general_acceptedPtsRatio	0.75/0.95
general_maxNrIterations	400
general_toleranceX,Y,A	10^{-12}
icp_connectPrevPtsWithLines	true

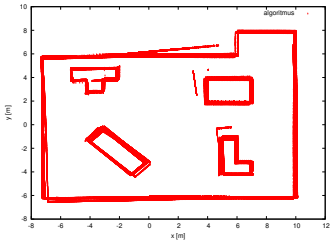
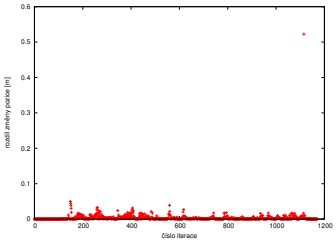
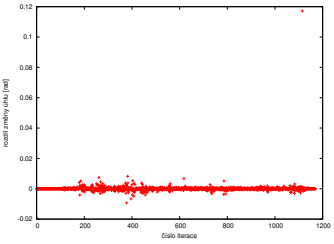
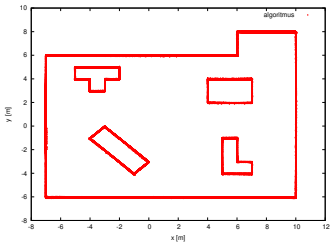
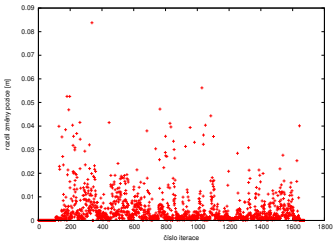
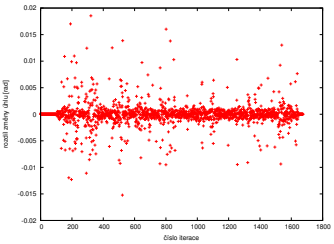
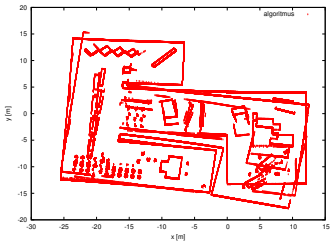
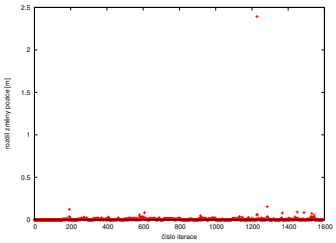
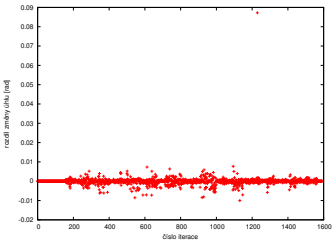
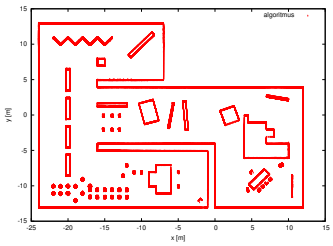
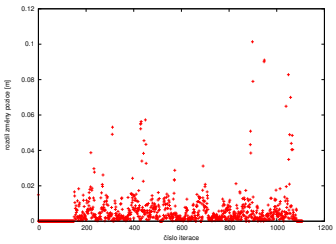
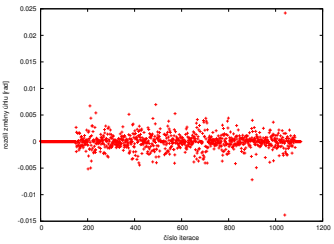
Výsledky popsaného testu jsou zobrazeny v tabulkách 4.9 a 4.10.



Obrázek 4.7: Mapa okolního prostředí č.1.



Obrázek 4.8: Mapa okolního prostředí č.2.

experiment	mapa	rozdíly pozice	rozdíly úhlu
Prostředí 1 bez mapy			
Prostředí 1 s mapou			
Prostředí 2 bez mapy			
Prostředí 2 s mapou			

Tabulka 4.9: Porovnání výsledků lokalizace se známou mapou prostředí a bez ní.

číslo prostředí	známá mapa	x	y	pozice	φ
1	ne	0.001995	0.002472	0.003495	0.000713
	ano	0.001869	0.002363	0.003592	0.000629
2	ne	0.004204	0.004271	0.006558	0.000838
	ano	0.003607	0.003349	0.005546	0.000825

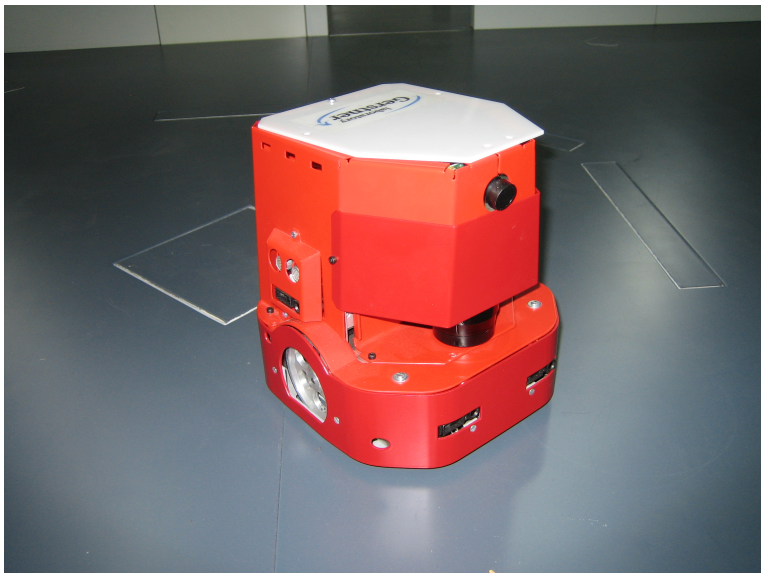
Tabulka 4.10: Parametry vypovídající o kvalitě lokalizace se známou mapou a bez ní.

Z tabulek 4.9 a 4.10 je vidět, že znalost mapy prostředí, ve kterém se robot pohyboval, vedla v obou testovaných případech ke zlepšení výsledku lokalizace, což je očekávaný závěr. Při lokalizaci se známou mapou okolního prostředí však došlo hlavně k odstranění aditivní chyby, což je nejlépe vidět z map získaných umístěním scanů do poloh určených algoritmem (obrysy překážek na těchto mapách pro lokalizaci vůči mapě nejsou rozmazané tolik jako při lokalizaci bez mapy).

4.4 Praktické experimenty

Cílem tohoto testu bylo ověřit, že implementovaný lokalizační driver je schopen zpracovávat data naměřená robotem S1R (viz obr. 4.9), který byl na ČVUT vyvinut jako součást projektu SyRoTek. Data byla při pohybu robotu zaznamenána pomocí driveru writelog, pro jejich využití bylo pouze nutné upravit časy příchodů jednotlivých zpráv (aby první zpráva přišla v čase 0.0), seřadit zprávy dle času příchodu a zjistit maximální dosah laseru.

Provedený experiment ověřil, že lokalizačním driverem takto získaná data skutečně zpracovat lze.



Obrázek 4.9: Robot S1R.

Kapitola 5

Závěr

Dle zadání byly nastudovány algoritmy ICP, IMRP a IDC, a to jak ve variantě s propojováním referenčních bodů, tak bez něj. Tyto algoritmy jsou podrobně popsány v kapitole 2.

Dále byl v rámci této práce implementován lokalizační driver pro systém Player, který umožňuje provádět kontinuální lokalizaci mobilního robotu v kancelářském prostředí pomocí dat z laserového dálkoměru a odometrických senzorů, a který je popsán v kapitole 3. Pro lokalizaci robotu je možné použít libovolný ze zmíněných algoritmů.

I přes problémy popsané v 3.4 (vznik falešných bodových párů, zpoždění způsobená dobou výpočtu, velké množství parametrů, na kterých závisí vlastnosti lokalizace) se pomocí provedených experimentů (viz 4.2) podařilo ukázat, že implementovaný driver je schopen pomocí algoritmu ICP s propojováním referenčních bodů úsečkami určit polohu robotu s chybou v řádu centimetrů a úhel natočení robotu s chybou v řádu tisícin radiánu na jeden krok lokalizace. Naopak, algoritmus IMRP se kvůli poměrně velké chybě v odhadu pozice robotu ukázal pro lokalizaci jako ne příliš vhodný. Dalšími experimenty (viz 4.3) bylo ověřeno, že při použití mapy okolního prostředí lze odstranit aditivní charakter vzniklých chyb, což dále zlepšuje kvalitu lokalizace.

Pomocí dalších experimentů bylo ověřeno, že implementovaný driver je schopen zpracovávat data naměřená robotem S1R (viz obr. 4.9), který byl na ČVUT vyvinut jako součást projektu SyRoTek.

5.1 Možnosti dalšího rozšíření práce

Při implementaci driveru byla jako struktura pro uložení referenčních měření dálkoměru použita dvě pole bodů (v jednom byly uloženy x-ové souřadnice, ve druhém y-ové souřadnice přijatých bodů). Popsané řešení bylo použito z důvodu snadné implementace, má ale nevýhodu v tom, že vyhledávání referenčního bodu, který má nejmenší vzdálenost od jiného zadaného bodu, je poměrně pomalé. Možným rozšířením této práce by proto bylo využití K-D stromů (popsaných např. v [6]) jako struktury pro ukládání referenčních scanů.

Literatura

[1] F. Lu a E. Milios. Robot pose estimation in unknown environments by matching 2d range scans, 1997.

[2] P. J. Besl a N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, únor 1992.

[3] A. Howard B. Gerkey, R. T. Vaughan. „The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems“ proceedings of the 11th international conference on advanced robotics, červen 2003.

[4] CARMEN-Team. Carmen (carnegie mellon robot navigation toolkit). `http://carmen.sourceforge.net/intro.html`.

[5] Boost community. Boost c++ libraries. `http://www.boost.org/`.

[6] R. Mázl. *Lokalizace pro autonomní systémy.* PhD thesis, České vysoké učení technické v Praze, 2007.

[7] Cyrill Stachniss. Robotics datasets. `http://www.informatik.uni-freiburg.de/~stachnis/datasets.html`.

Obsah CD

Přiložené CD obsahuje zdrojové kódy lokalizačního driveru, roboty naměřená data, která byla použita při experimentech, text bakalářské práce ve formátu PDF a zdrojové kódy celého textu pro systém LaTeX. V následující tabulce je popsána struktura CD.

Adresář	Popis
`text`	zdrojové kódy textu bakalářské práce
`thesis.pdf`	text bakalářské práce
`matchingDriver`	
`src`	zdrojové kódy lokalizačního driveru
`doc`	složka, do které bude při překladu vygenerována dokumentace ke driveru pomocí nástroje Doxygen
`test`	složka s daty pro jednoduché otestování funkčnosti driveru
`experiments`	data, která byla použita při experimentech

Tabulka 1: Adresářová struktura na CD.