Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Návrh a testování metod vizuální simultánní lokalizace a mapování

PLZEŇ, 2013 Petr Neduchal

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

PLZEŇ, 2013 _____

Petr Neduchal

Poděkování

Rád bych poděkoval Ing. Miroslavu Flídrovi Ph.D za vedení diplomové práce a za cenné rady během jejího zpracování.

Abstrakt

Práce se zabývá analýzou a návrhem metod vizuální simultánní lokalizace a mapování. Popisuje matematické základy celé úlohy a zaměřuje se na speciální případ monokulární simultánní lokalizace a mapování. V práci je popsáno testování metod pro detekci významných bodů a implementace navrženého algoritmu. V závěru prezentuje výsledky z provedených experimentů a další možný postup při výzkumu této úlohy.

Klíčová slova : SLAM, FAST, SURF, SIFT, Harrisův operátor, detekční metody, mapování, lokalizace, Kalmanův filtr, C++, OpenCV, kamera

Abstract

This thesis deals with analysis and design of visual simultaneous localization and mapping methods. It describes mathematical principles of the problem and it focuses on the special case called monocular simultaneous localization and mapping. The paper contains description of testing landmark detection methods and implementation of designed algorithm. In conclusion paper presents experimental outcomes and the way to future research of this task.

Keywords : SLAM, FAST, SURF, SIFT, Harris, detection methods, mapping, localization, Kalman filter, C++, OpenCV, camera

Obsah

Seznam obrázků

Kapitola 1

Úvod

Stále častěji se do popředí zájmu vědců a firem dostávají systémy pro automatické rozpoznání polohy zařízení a jeho bezprostředního okolí tak, aby na tyto informace mohlo zařízení v reálném čase reagovat. Jako dobrý příklad může posloužit brzdící systém automobilové společnosti Volvo, plně automatické řízení vozidla vyvinuté vědci ve firmě Google a v neposlední řadě také částečně autonomní vozidlo Curiosity, které v současné době zkoumá povrch planety Mars. Není ovšem nezbytné hledat jen takto extrémně složité systémy, jako příklad jednoduššího zařízení, které se snaží monitorovat okolí za současného určování vlastní polohy, poslouží robotický vysavač, kterých se na trhu objevila již celá řada od různých výrobců. Tyto přístroje obsahují jen nejjednodušší algoritmy a v mnohých případech se řídí pouze dotykovými senzory a reagují tak jen v případě najetí do pevné překážky, ovšem i takovéto aplikace je možné započítat do předchozího výčtu příkladů.

Obecně existuje více přístupů k tomuto problému, který se souhrně označuje jako simultánní lokalizace a mapování zkráceně SLAM[1]. Z předchozího je patrné, že cílem SLAMu je vytvoření dostatečně přesné mapy okolí a současný odhad polohy zařízení, které toto okolí snímá pomocí jednoho či více možných snímačů jejichž seznam bude uveden dále v této práci. Důležitým prvkem v problému SLAM je běh algoritmů v reálném čase. To ovšem přináší jeden velmi zajímavý a nepříznivý efekt, který je znám jako "slepice a vejce". Zařízení totiž v reálném čase potřebuje zjistit svoji polohu, ovšem k té potřebuje znalost mapy. Ovšem k vytvoření mapy je na druhou stranu potřeba znát polohu zařízení, ze kterého jsou data pořizována. V jedné z dalších částí této práce bude uvedeno jak v takovém případě postupovat, ovšem obecně se dá říci, že je potřebné zavést určitý druh inicializace, které do problému vnese určitou prvotní informaci. V tuto chvíli by bylo na místě ocitovat zajímavý postřeh z článku [1]. Text je záměrně uveden v originále.

A solution of the SLAM problem has been as a "holy grail" for the mobile robotics comunity as it would provide the means to make robot truly autonomous.

[1] Dále bude v textu až na výjimky používána pouze zkratka SLAM.

Jediným problémem, na který výše uvedená citace naráží je ten, že dokonalé řešení problému SLAM je zatím opravdu jen snem. Celý problém je v současné době řešen v teoretické rovině v mnoha různých podobách. Existuje také mnoho reálných aplikací, které se ovšem vždy zaměřují na určitý druh zařízení, určitý druh robota, nebo v neposlední řadě také určitý druh snímacích prvků. Nemůže být ovšem řeč o aplikaci, která by bez problémů zvládala všechny situace. Takové řešení existuje pouze v teoretické a konceptuální rovině, které je zatím bohužel od realizace velmi daleko.

Tato práce se kromě základního přehledu problému SLAM zabývá zejména podmnožinou používající optických senzorů jako snímače okolí. Tato podmnožina je označována jako Vizuální SLAM a v praktických aplikacích je jako snímač použita jedna či více kamer. Není samozřejmě vyloučené ani doplnění jiného druhu snímače jakým je například laserový nebo ultrazvukový snímač vzdálenosti. V dalších částech bude zmíněn problém monokulárního SLAMu, který používá právě jednu kameru. Díky tomu se do celého procesu dostávají další problémové faktory, které je potřeba řešit.

Práce je organizována následujícím způsobem. V druhé kapitole nazvané Teorie budou uvedeny základní pojmy potřebné k formulaci problému SLAM dále pak možnosti přístupu k celému problému a možných řešení. V neposlední řadě bude součástí této kapitoly také stručná historie problému datující se od 80. let 20. století. V rámci teorie budou také zmíněny metody pro detekci a popis významných bodů z obrazových dat získaných pomocí kamery. Ve třetí a čtvrté kapitole bude uvedena řada informací o existujících řešeních SLAM úlohy, o implementacích a knihovnách, které je možné při studiu problému zkoumat, testovat a v neposlední řadě se v nich inspirovat a přiučit nad možnostmi implementace jednotlivých částí SLAM aplikace. V následující kapitole budou popsány testy jednotlivých aplikací a metod včetně výsledků a ukázek jejich fungování. Bude zde brán zřetel zejména na metody popsané v teorii a dále na základní možnosti řešení problému SLAM. V poslední kapitolách pak bude popsáno zhodnocení celé práce a aktuálního stavu a možné budoucnosti výzkumu problému SLAM.

Kapitola 2

Úloha simultánní lokalizace a mapování

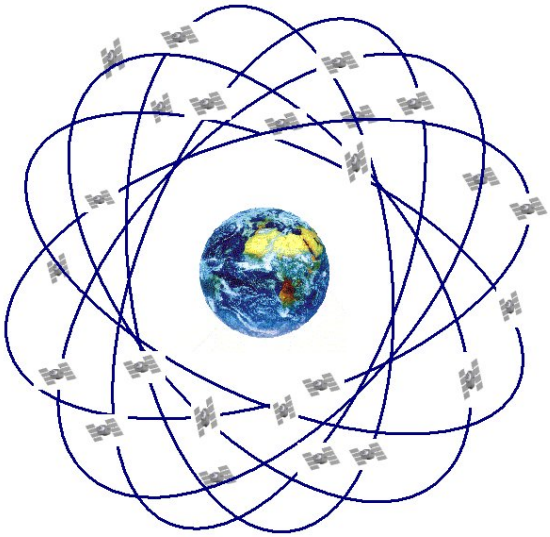
Úloha simultánní lokalizace a mapování spočívá v propojení problému lokalizace a problému mapování. Řešením úlohy SLAM je pak vytvoření vhodné reprezentace mapy okolí zařízení (například mobilního robota) a současné určení polohy tohoto zařízení ve vytvořené mapě. Důležitým aspektem celé úlohy je provádění obou dvou částí současně v reálném čase. Což může přinášet mnoho problémů.

V této části práce bude uvedena veškerá teorie vztahující se Simultánní lokalizaci a mapování. Zvláštní důraz pak bude zaměřen na metody, na kterých je založena úloha vizuálního SLAMu. Teorií jsou myšleny zejména metody detekce významných bodů a porovnávání těchto bodů mezi snímky. Součástí kapitoly bude také část věnovaná stručné historii problému SLAM.

2.1 Problém lokalizace

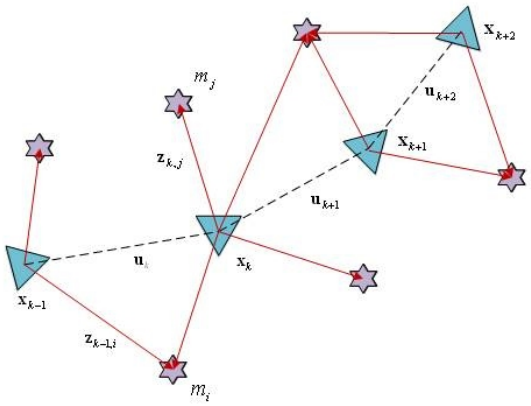
Úloha lokalizace je v současné době chápána zejména v globálním měřítku. Konkrétně je řeč o globální pozičním systému (GPS), který je založen na datech satelitů rozmístěných kolem celé planety Země. S pomocí této technologie je možné určit polohu zařízení komunikujícího se zmíněnými satelity s přesností do deseti metrů[1]. Je nutné zmínit, že tento systém spravuje ministerstvo obrany USA. Z toho vyplývá, že pro civilní použití nejsou k dispozici veškeré vlastnosti tohoto systému. Armáda má pak k dispozici GPS zařízení, která dokážou ze satelitních dat určit polohu s přesností až na jednotky centimetrů.

[1] Dle informací na *http://cs.wikipedia.org/wiki/Global_Positioning_System*



Obrázek 2.1: Síť satelitů pro lokalizaci pomocí GPS. Zdroj : http://www.techquark.com

V souvislosti s úlohou SLAM je však lokalizace chápána jiným způsobem. Zatímco při použití GPS zařízení se poloha určuje z dat satelitů, v případě SLAMu je lokalizace určována z dat získaných přímo na zařízení, které se snažíme lokalizovat. Může se jednat o vzdálenostní data získaná pomocí laserového či ultrazvukového snímače, vizuální data z kamery nebo případně kombinaci obou zmíněných.



Obrázek 2.2: Ukázka určování polohy pomocí významných bodů. Převzato z článku [1]

Další důležitou vlastností této lokalizace je nutnost znalosti okolí zařízení. Toto okolí může být určeno pomocí speciálních předem známých a daným snímačem rozpoznatelných bodů, nebo pomocí vytvářené mapy. Z toho je také zřejmé, že je problém lokalizace do značné míry závislý na problému mapování.

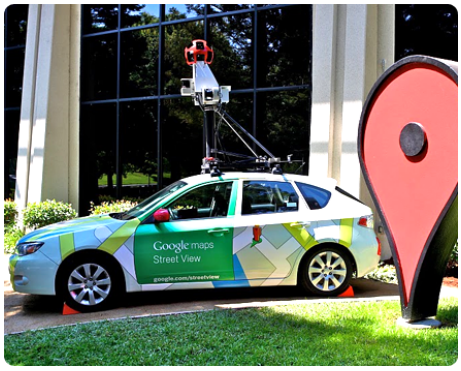
2.2 Problém mapování

Mapováním je v tomto smyslu myšleno online vytváření mapy okolí. Jinými slovy se jedná o vybrání vhodné reprezentace, do které jsou následně ukládány významné body. Základní rozdělení těchto reprezentací je následující

- **Metrická mapa**, ve které je důležité přesné určení bodů v prostoru pomocí vektoru souřadnic. Nevýhodou je náchylnost na šum ve snímači a také fakt, že přesná poloha jednotlivých významných bodů se těžko určuje.
- **Topologická mapa**, která může být určena například grafem označujícím vzájemné závislosti jednotlivých významných bodů. Většinou se vztahem mezi dvěma významnými body myslí jejich vzájemná vzdálenost.

Velmi dobrým příkladem mapování jsou aplikace Google Maps a Google Steetview. První z nich je služba obsahující mapy s různými druhy podkladů. Tyto podklady je potřeba určitým způsobem pospojovat, aby se pro stejné místo nezobrazovala u každého podkladu jiná oblast. Toho je docíleno známou zeměpisnou polohou jednotlivých částí mapy. Na tuto mapu se můžeme dívat buď z hlediska metrického přístupu, kde má každý bod svojí polohu. A nebo z pohledu topologické mapy, kde jsou jednotlivé části mapy propojeny do grafu.

Druhá aplikace obsahuje ryze metrický přístup k problému mapování. Zde je významným bodem myšlen jakýkoliv výrazný bod na fotkách vyfocených speciálním zařízením. Z těchto fotek se později vytvoří navázaná panoramata, kterými lze procházet. Je pravděpodobné, že se používají velice podobné algoritmy jako v případě metod Visual SLAMu.



Obrázek 2.3: Automobil se speciálním zařízením pro snímání okolí do služby Google Streetview

2.3 Stručná historie problému

Historie simultánní lokalizace a mapování se začala psát v 80. letech 20. století. Přesněji řečeno v roce 1986 na konferenci IEEE Robotics and Automation Conference, která se konala v San Franciscu v Kalifornii. V této době se do popředí zájmu vědců zajímajících se o umělou inteligenci dostávaly metody založené na pravděpodobnostních principech. Jednou z diskutovaných úloh byla aplikace teoretických odhadových metod na mapovací a lokalizační problémy. První ucelenou práci na toto téma představuje článek z téhož roku pojmenovaný On the Representation and Estimation of the Spatial Uncertainly [4], ve kterém se autoři věnují odhadům pozice pomocí skládání transformací od jednoho bohu k druhému.

Je důležité zejména zmínit jaký výsledek tato konference přinesla. Vědci po mnoha diskuzích zjistili, že se jedná o velmi zajímavý problém, který však obsahuje mnoho teoretických, implementačních i výpočetních problémů a je opravdu zapotřebí tyto problémy vyřešit.

Velmi významný problém vyvstal také díky mylné představě, že by body v okolí mobilního robota měly být nekorelované. Později díky článkům věnujícím se této problematice [4] bylo ukázáno, že korelovanost mezi významnými body patří k velice důležité části lokalizačních a mapovacích problémů. Při praktickém řešení jsou to právě body v okolí zařízení, které určují podobu mapy a pokud jsou tyto body nepohyblivé, pak se proporce této mapy nemění a z toho vyplývají dva poznatky:

1. Body musejí být korelované.

2. Je možné s vysokou přesností určovat vzdálenosti mezi jednotlivými významnými body.

I přes tato zjištění se však řada článků věnovala aproximacím, kterými by se dala korelace mezi jednotlivými body snížit.

Je s podivem, že do roku 1995 pro tuto úlohu neexistovala lehce zapamatovatelná zkratka. To se změnilo až na mezinárodním symposiu robotiky, kde byl poprvé použit akronym SLAM a problém lokalizace a mapování se začal řešit jako jedna úloha.

Od chvíle kdy byl tento akronym přijat bylo vydáno již mnoho článků a úloha se začala řešit z mnoha různých úhlů. V dnešní době se používá mnoho druhů snímačů. Zvláště se pak skupiny vědců zaměřují na vizuální SLAM hlavně kvůli nízké ceně snímačů. Na druhou stranu je zřejmé, že se tyto snímače nehodí pro všechny typy okolí. Například pro tvorbu mapy mořského dna. Pro tuto úlohu se mnohem více hodí například sonar. V dalších kapitolách budou popsány metody vizuálního a zvláště pak monokulárního SLAMu včetně příkladů konkrétních úloh. Nejdříve je však potřeba představit obecné principy simultánní lokalizace a mapování.

2.4 Formulace a struktura úlohy

Před samotnou formulací úlohy je potřeba uvést styl zápisu jednotlivých částí SLAMu, aby čtenář později v textu nebyl zmaten. Jedná se zejména o značení důležitých vektorů.

2.4.1 Definice a značení jednotlivých částí SLAM úlohy

Pro potřeby formulace úlohy bude uvažován mobilní robot pohybující se v neznámém prostředí. V tomto prostředí bude v zadaném časovém intervalu pořizovat snímky okolí pomocí snímače[2] umístěného přímo na robotu jak je vidět na obrázku 2.2. Nyní je možné přistoupit k jednotlivým definicím zápisů:

- x_k - Stavový vektor polohy a orientace mobilního robota
- u_k - Vektor řízení aplikovaný v čase $k-1$ pro přivedení robota do stavu x_k
- m_i - Vektor polohy i-tého významného bodu. Je předpokládána jeho neměnná poloha.
- z_{ik} - Pozorování polohy i-tého významného bodu pořízené robotem v čase k.

Kromě těchto vlastností definovaných v jednom časovém okamžiku je nutné definovat také tyto vlastnosti v čase od započetí snímání okolí až po aktuální časový krok. Vektory lze také označovat pojmem historie.

- $X_{0:k} = \{x_0, x_1, \ldots, x_k\} = \{X_{0:k-1}, x_k\}$ - Historie polohy vozidla.
- $U_{0:k} = \{u_0, u_1, \ldots, u_k\} = \{U_{0:k-1}, u_k\}$ - Historie řízení.
- $m = \{m_0, m_1, \ldots, m_n\}$ - Vektor významných bodů.
- $Z_{0:k} = \{z_0, z_1, \ldots, z_k\} = \{Z_{0:k-1}, z_k\}$ - Vektor všech pozorování do času k.

2.4.2 Pravděpodobnostní SLAM

Po uvedení výše uvedené definice je možné formulovat úlohu SLAM pomocí pravděpodobnosti. Přesněji řečeno je potřeba určit sdruženou aposteriorní hustotu pozic jednotlivých významných bodů a pozice mobilního robotu. Pro tuto hustotu se používá následující zápis :

$$p(x_k, m \mid Z_{0:k}, U_{0:k}, x_0)$$

Ze zápisu je patrné, že se výpočet této hustoty provádí v každém časovém kroku k a vždy zahrnuje celou historii polohy robotu a použitého řízení. Nutností k výpočtu je vektor pozorování, řízení a počáteční poloha mobilního robota, od které se celý proces simultánní lokalizace a mapování odvíjí.

[2] Obecně může být více snímačů pro zpřesnění informace o okolí.

Výpočet výše uvedené hustoty je nutné provádět v každém časovém kroku. Proto je na místě vycházet z hustoty vypočítané pro časový okamžik $k-1$:

$$p(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0)$$

Pro získání hustoty v kroce k je třeba určit řízení u_k a pozorování z_k pomocí Bayesova vztahu. Proto bude výhodné zavést pojmy pozorovacího a pohybového modelu, které popisují efekt pozorování respektive řídícího vstupu.

Model pozorování popisuje hustotu pravděpodobnosti pozorování z_k za předpokladu, že je známa poloha mobilního robota i všech významných bodů v jeho okolí. Matematicky je model možné zapsat následovně :

$$p(z_k \mid x_k, m).$$

Pohybový model může být popsán jako hustota pravděpodobnosti stavu (polohy) robota x_k v závislosti na stavu předchozím a řízení u_k. Matematicky :

$$p(x_k \mid x_{k-1}, u_k).$$

Z této definice je patrné, že se jedná o Markovský proces závislý na x_{k-1} a u_k a nezávislý na jednotlivých pozorováních ani na vytvořené mapě.

Celý rekurzivní algoritmus simultánní lokalizace a mapování má pak následující tvar:

Časový krok (predikce)

$$p(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0) =$$

$$= \int p(x_k \mid x_{k-1}, u_k) \times p(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1}$$

Filtrace

$$p(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) = \frac{p(z_k \mid x_k, m) p(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0)}{p(z_k \mid Z_{0:k-1}, U_{0:k})}$$

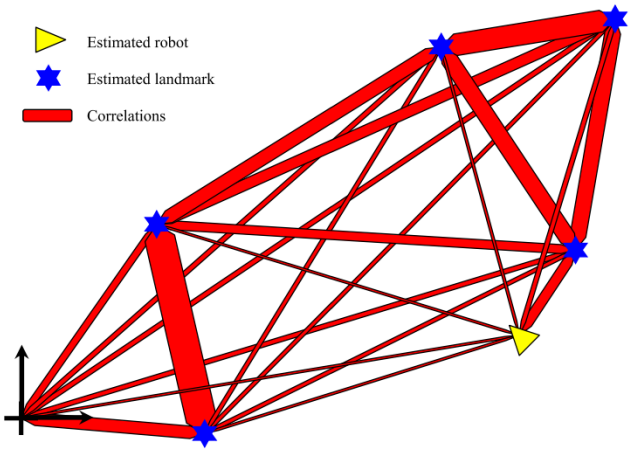
Jednotlivé modely použité v tomto algoritmu si zaslouží více pozornosti. Zejména pak pozorovací model. Ten je přímo závislý na poloze robota i na mapě významných bodů v okolí. Z toho vyplývá i tvar aposteriorní pravděpodobnosti[3] a zejména pak neoddělitelnost problémů lokalizace a mapování. Matematicky to lze zapsat následujícím způsobem

$$p(x_k, m \mid z_k) \neq p(x_k \mid z_k) p(m \mid z_k).$$

Důsledkem takového rozdělení je zejména nekonzistentnost odhadů a tedy naprosté znehodnocení výsledků. Na tomto místě je také příhodné opět zmínit korelovanost jednotlivých výsledných bodů. Tato vlastnost prakticky znamená, že relativní poloha mezi dvěma

[3] V dalším textu budou použity zkrácené zápisy sdružené hustoty ve tvarech $p(x_k, m \mid z_k)$ *a* $p(x_k, m)$, dle možností aktuálního kontextu okolního textu.

body $m_i,\ m_j$ může být určena s velkou přesností. To i přes fakt, že samotná poloha bodu m_i není přesně určena. V pravděpodobnostní formulaci to znamená, že sdružená hustota dvojice bodů $p(m_i, m_j)$ je výrazně špičatá s malým rozptylem. To i v případě, že hustota bodu m_i je hodně rozptýlená. Důležité také je, že tato korelace s časem monotónně roste a tím roste přesnost relativních poloh mezi jednotlivými body a to dokonce nezávisle na pohybu robota. Monotónní růst korelace je dokázán pro Gaussovský případ SLAM úlohy. Obecný důkaz je prozatím otevřeným problémem. Celý princip korelace je pak možné vizualizovat pomocí bodů propojených pružinami, jak je uvedeno na obrázku 2.4. Čím větší pružina, tím vyšší korelace mezi body .
Po každém pozorování se celá mapa vytvořená z pružin pohne a jen na síle pružin závisí velikost a přesnost tohoto posunu. S počtem pozorování tuhosti pružin rostou. V limitním případě se pak celá mapa změní na soustavu pevných bodů.



Obrázek 2.4: Princip korelace ukázaný na modelu pružin mezi body

2.5 Přístupy k řešení SLAM úlohy

V předchozí kapitole byla představena formulace SLAM úlohy. Nyní bude popsáno řešení úlohy respektive možnosti řešení úlohy.

Principem řešení je nalezení vhodné reprezentace pro modely pozorování a pohybu uvedené výše. Nejpoužívanější reprezentací je stavový model s Gaussovským šumem, jehož použití vede na algoritmus rozšířeného Kalmanova filtru. Druhou možnou reprezentací je potom popis pohybového modelu pomocí sady vzorků s ne-Gaussovskou distribucí. Tato varianta využívá takzvaného Rao-Blackwellizovaného částicového filtru. Jiné označení pro dvě zmíněné reprezentace jsou EKF SLAM[1] [3] a FAST SLAM [1]. Principy obou budou vyloženy v samostatných odstavcích.

2.5.1 EKF-SLAM

Pohybový model je v případě použití EKF-SLAMu přepsán do následující podoby

$$p(x_k \mid x_{k-1}, u_k) \Longrightarrow x_k = f(x_{k-1}, u_k) + w_k,$$

kde $f(.)$ modeluje pohyb robota a $w_k \sim N(0, \mathbf{Q_k})$ je aditivní nekorelovaný Gaussovský šum. Pozorovací model je přepsán podobným způsobem na tvar

$$p(u_k \mid x_k, m) \Longrightarrow z_k = h(x_k, m) + v_k,$$

kde $h(.)$ modeluje geometrii pozorování a $v_k \sim N(0, \mathbf{R_k})$ je aditivní nekorelovaný Gaussovský šum.

Pomocí těchto modelů může být standardní EKF algoritmus použit k výpočtu střední hodnoty

$$\begin{bmatrix} \hat{x}_{k|k} \\ \hat{m}_k \end{bmatrix} = E \begin{bmatrix} \hat{x}_k \\ \hat{m} \end{bmatrix} \mid Z_{0:k}$$

a kovariance

$$P_{k|k} = \begin{bmatrix} Pxx & Pxm \\ Pxm^T & Pmm \end{bmatrix}_{k|k} = E \left[\begin{pmatrix} x_k - \hat{x}_k \\ m - \hat{m}_k \end{pmatrix} \begin{pmatrix} x_k - \hat{x}_k \\ m - \hat{m}_k \end{pmatrix}^T \mid Z_{0:k} \right]$$

sdružené aposteriorní hustoty pravděpodobnosti $P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0)$. Celý algoritmus výpočtu se pak skládá z časového a filtračního kroku :

Časový krok (predikce)

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$$
$$\hat{P}_{xx,k|k-1} = \nabla f \hat{P}_{xx,k-1|k-1} \nabla f^T + Q_k,$$

kde ∇f je jakobián funkce pohybového modelu f vypočítaném v posledním filtračním kroku. Zajímavostí ovšem je, že v případě stacionárních významných bodů se teoreticky tento krok může úplně přeskočit. Bohužel v praktických aplikacích není možné předpokládat všechny významné body stacionární.

Pozorování (filtrace)

$$\begin{bmatrix} \hat{x}_{k|k} \\ \hat{m}_k \end{bmatrix} = \left[\hat{x}_{k|k-1} \hat{m}_{k-1} \right] + W_k \left[z_k - h(\hat{x}_{k|k-1}, \hat{m}_{k-1}) \right]$$

$$P_{k|k} = P_{k|k-1} - W_k S_k W_K^T,$$

kde S_k je kovarianční matice residuí

$$S_k = \nabla h P_{k|k-1} \nabla h^T + R_k,$$

a W_k je Kalmanův zisk

$$W_k = P_{k|k-1} \nabla h^T S_k^- 1.$$

Člen ∇h je jakobián funkce pozorovacího modelu h vypočítaný v posledním časovém kroku.

EKF-SLAM je nejpoužívanějším algoritmem pro simultánní lokalizaci a mapování a to i přesto, že trpí několika problémy, se kterými je potřeba dopředu počítat.

Konvergence : Konvergencí se v tomto případě myslí konvergence kovarianční matice P a všech jejích submatic směrem k nule. Každý významný bod má při přidání do algoritmu vysokou neurčitost polohy, která se s novými pozorováními snižuje až k dolní hodnotě dané omezením v podobě minimální neurčitosti a šumem v získávaných datech.

Výpočetní náročnost : V EKF-SLAMu je vyžadováno, aby s každým pozorováním byly všechny významné body aktualizovány ve vektoru x i v kovarianční matici P. V praxi to znamená, že výpočetní náročnost roste kvadraticky s počtem významných bodů. Nemluvě o růstu paměťové náročnosti stejnou rychlostí.

Správné asociování dat : Základní verze metody je velice náchylná na správné přiřazení nových pozorování k již existujícím významným bodům. Tento problém vyvstává hlavně ze dvou důvodů:

1. Robot se může pohybovat na větším území a nemusí rozpoznat již navštívené místo.
2. I při malém pohybu může být problém určit stejný významný bod, jelikož se může z jiného úhlu jevit odlišně.

Obecně se řeší zejména první problém. Tato úloha se nazývá loop-closure, což by se do češtiny dalo přeložit jako úloha uzavření smyčky.

Nelinearita : EKF-SLAM používá linearizovaný model nelineárního pohybu a pozorovacích modelů. Nelinearita se tak může stát problémem, který vede ke špatnému řešení a tedy špatné konzistenci dat. Konvergence a konzistence mohou být v případě EKF-SLAMu garantovány jedině v lineárním případě.

Přes všechny tyto nedostatky je stále SLAM využívající rozšířeného Kalmanova filtru nejpoužívanější ze známých řešení. Důvodem je jeho relativně snadná implementace.

2.5.2 Rao-blackwallizovaný filtr aneb FastSLAM

Mnoho vědců se soustředilo zejména na vylepšování EKF-SLAMu a zejména eliminaci jeho slabin. Bylo ovšem otázkou času než se objeví jiná metoda. Takovou metodou je v tomto případě takzvaný FastSLAM. Základem je rekurzivní Monte Carlo vzorkování, nebo částicové filtrování. Hlavním rozdílem oproti EKF-SLAMu je použití nelineárního pohybového modelu a ne-gaussovského rozdělení pravděpodobnosti. Dochází pouze k linearizaci pozorovacího modelu, což ovšem není na škodu ve chvíli, kdy je určena poloha robotu, na kterém probíhá celá úloha.

Přímá aplikace FastSLAM je naneštěstí díky vysoké dimenzi stavu výpočetně neproveditelný. Je však možné redukovat stavový prostor použitím Rao-Blackwellizace, která rozdělí sdružený stav podle následujícího pravidla

$$p(x_1, x_2) = p(x_2 \mid x_1)p(x_1).$$

Jestliže může být navíc $p(x_2 \mid x_1)$ vyjádřeno analyticky, pak stačí vzorkovat pouze $p(x_1)$:

$$x_1^{(i)} \sim p(x_1).$$

Sdružená hustota pravděpodobnosti je pak reprezentována sadou vzorků

$$\left\{x_1^{(i)}, \tilde{p}(x_2 \mid x_1)\right\}$$

a marginální hustota pravděpodobnosti

$$p(x_2) \approx \frac{1}{N} \sum_i^N p\left(x_2 \mid x_1^{(i)}\right)$$

je pak možné určit s větší přesností než při vzorkování celého sdruženého stavu.

Při aplikaci výše uvedené teorie na sdružený stav SLAM úlohy bude získána samostatná pravděpodobnost pro vozidlo a podmíněná část pro mapu.

$$p(X_{0:k}, m \mid Z_{0:k}, U_{0:k}, x_0) = p(m \mid X_{0:k}, Z_{0:k})p(X_{0:k} \mid Z_{0:k}, U_{0:k}, x_0)$$

Při pohledu na uvedenou hustotu je vidět, že je místo stavu x_k použita celá historie stavů $X_{0:k}$. Tato záměna je velmi výhodná, neboť díky závislosti na celé trajektorii se jednotlivé významné body stávají nezávislými. Jinými slovy se dá říct, že pro FastSLAM tvoří každá částice jinou hypotézu trajektorie robota. Právě tato vlastnost je pro metodu klíčová, jelikož je pak celá mapa modelována sadou nezávislých gaussovských rozdělení, které mají lineární složitost. Na obrázku 2.5 je zobrazena mapa obsahující jednu trajektorii $X_{0:k}^{(i)}$
Sdružená pravděpodobnost ve FastSLAMu je reprezentována sadou vážených vzorků

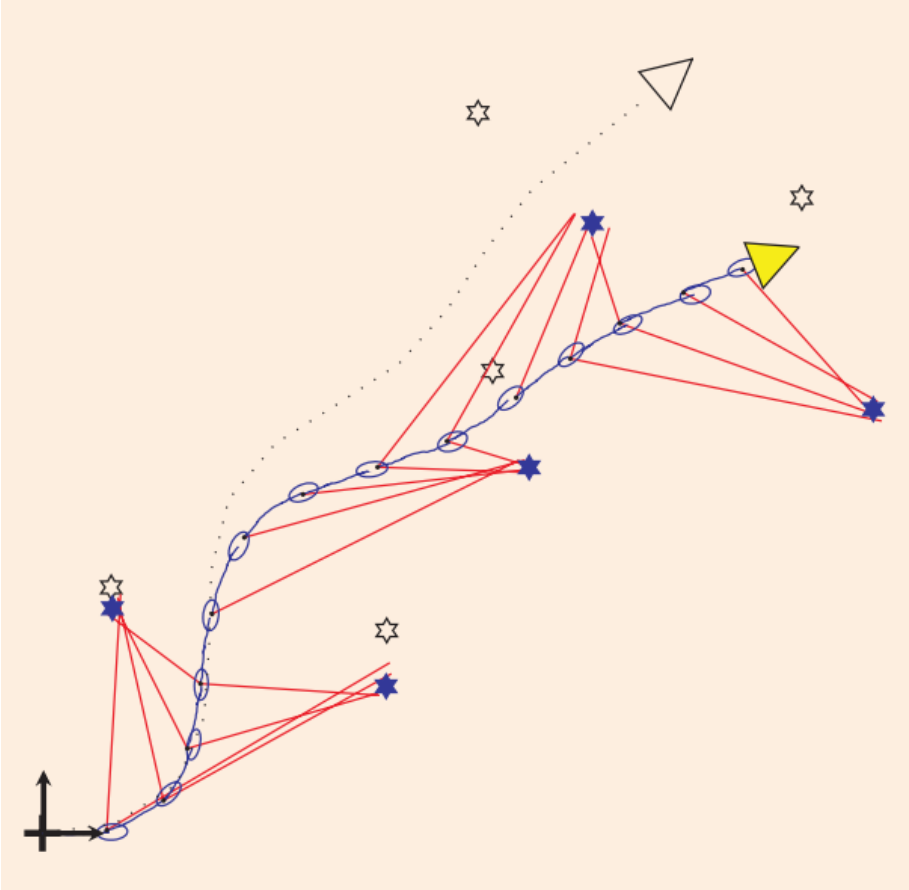
$$\left\{w_k^{(i)}, X_{0:k}^{(i)}, p(m \mid X_{0:k}^{(i)}, Z_{0:k})\right\}$$

a modelovaná mapa tvořená gaussovskými rozděleními pak má následující tvar

$$p\left(m \mid X_{0:k}, Z_{0:k}\right) = \prod_j^M p\left(m_j \mid X_{0:k}, Z_{0:k}\right).$$

Ve výsledku je tedy částicové filtrování použito na stav robota. Na stav mapy respektive významných bodů je použit EKF algoritmus jako v případě EKF-SLAMu uvedeném v předchozí části.

Při aktualizaci mapy pro danou trajektorii $X_{0:k}^{(i)}$ se na každý pozorovaný významný bod použije filtrační krok z EKF algoritmu. Významné body, které v daném kroku nejsou



Obrázek 2.5: Ukázka jedné realizace trajektorie mobilního robota

pozorovány se nezmění. Problém u částic reprezentujících pozici robota je o trochu složitější. Využívá se zde částicové filtrování, které má svůj základ v metodě *sequential important sampling*[4] *(SIS)*[20], která při vzorkováná používá celý sdružený stav obsahující celou historii respektive celou trajektorii. Díky součinovému pravidla je pak možné odvodit rekurzivní vztah. Součinové pravidlo vypadá následovně

$$p(x_0, x_1, \ldots, x_T \mid Z_{0:T}) = p(x_0 \mid Z_{0:T}) p(x_1 \mid x_0 Z_{0:T}) \ldots p(x_T \mid X_{0:T-1}, Z_{0:T})$$

V každém kroku *k* jsou částice brány z navržené hustoty $\pi(x_k \mid X_{0:k-1}, Z_{0:k})$, které aproximuje skutečnou hustotu pravděpodobnosti $p(x_k \mid X_{0:k-1}, Z_{0:k})$. Dalším krokem je přidání vah k jednotlivým částicím, čímž se kompenzují nesrovnalosti vzniklé aproximací. chyba aproximace pak s časem roste a je potřeba provést převzorkování, které tuto chybu redukuje. Nevýhodou převzorkování je ztráta informací o historii jednotlivých částic. Je to ovšem v souladu s vlastnostmi metody SIS, která může produkovat *rozumné* výsledky jen pro systémy, které *exponencielně zapomínají* svoji historii.

[4]Možný překlad : Sekvenční výběrové vzorkování

Obecná forma algoritmu Rao-Blackwallizovaného částicového filtru je následující[5]

1. Pro každou částici se vypočte hustota π závislé na historii této částice a z ní se vytvoří vzorek.
$$x_k^{(i)} \sim \pi(x_k \mid X_{0:k-1}^{(i)}, Z_{0:k}, u_k).$$
O tento vzorek je doplněna historie částice
$$X_{0:k}^{(i)} = \left\{ X_{0:k-1}^{(i)}, x_k \right\}$$

2. Následuje výpočet vah
$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(z_k \mid X_{0:k}^{(i)}, Z_{0:k-1}^{(i)}\right) p\left(x_k^{(i)} \mid x_{k-1}, u_k\right)}{\pi(x_k \mid X_{0:k-1}^{(i)}, Z_{0:k}, u_k)}.$$
Čitatel uvedeného zlomku obsahuje model pohybu a model pozorování.

3. Dalším krokem je převzorkování. V současné době se jedná o stále otevřený problém. Zejména není jasno v kterých časových okamžicích se má převzorkování provádět. Prozatím existuje několik postupů:
   - Převzorkování v každém kroku.
   - Převzorkování po zadaném počtu kroků.
   - Převzorkování při překročení určitého prahu.
   - Heuristický přístup.

   Samotné převzorkování začíná výběrem částic z $\left\{ X_{0:k}^{(i)} \right\}_i^N$ s pravděpodobností ovlivněnou vahou $w_k^{(i)}$. Vybraným částicím je přiřazena jednotná váha $\left\{ w_k^{(i)} = \frac{1}{N} \right\}$.

4. Pro každou částici je provedena EKF filtrace aplikovaná na pozorované významné body.

Kromě této obecné formy algoritmu jsou zkoumány a implementovány dvě verze FastSLAMu s označením FastSLAM 1.0 a FastSLAM 2.0 lišící se pouze ve formulaci navržené hustoty π.

Pro FastSLAM 1.0 je hustota používána v následujícím tvaru
$$x_k^{(i)} \sim p(x_k \mid x_{k-1}^{(i)}, u_k).$$

Výpočet vah se díky tomu zjednoduší následovně
$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(z_k \mid X_{0:k}^{(i)}, Z_{0:k-1}^{(i)}\right) p\left(x_k^{(i)} \mid x_{k-1}, u_k\right)}{\pi(x_k \mid X_{0:k-1}^{(i)}, Z_{0:k}, u_k)} = w_{k-1}^{(i)} p\left(z_k \mid X_{0:k}^{(i)}, Z_{0:k-1}^{(i)}\right).$$

[5]Předpoklad : Sdružený stav v čase k-1 má tvar $\left\{ w_{k-1}^{(i)}, X_{0:k-1}^{(i)}, p(m \mid X_{0:k-1}^{(i)}, Z_{0:k-1}) \right\}_i^N$

V případě FastSLAMu 2.0 je navržená hustota, která obsahuje také aktuální pozorování

$$x_k^{(i)} \sim p(x_k \mid X_{0:k-1}^{(i)}, Z_{0:k}, u_k),$$

kde

$$p\left(x_k \mid X_{0:k-1}^{(i)}, Z_{0:k}, u_k\right) = \frac{1}{C} p\left(z_k \mid x_k, X_{0:k-1}^{(i)}, Z_{0:k-1}\right) p\left(x_k \mid x_{k-1}^{(i)}, u_k\right).$$

a C je normalizační konstanta. Po dosazení do obecného vzorcep ro výpočet váhy dostaneme vztah :

$$w_k^{(i)} = w_{k-1}^{(i)} C.$$

Algoritmus FastSLAM 2.0 pak díky lokální optimalitě navržené hustoty zaručuje nejmenší možnou váhu $w_k^{(i)}$ v závislosti na dostupné informaci o trajektorii, pozorování a řízení.

Oba FastSLAM algoritmy trpí problémem spojeným se zapomínáním historie stavů v trajektorii. Z hlediska statistiky to znamená, že algoritmus může degenerovat a poskytovat špatné výsledky. Reálné nasazení těchto algoritmů však dokazuje, že je možné pomocí tohoto algoritmu generovat mapu s dostatečnou přesností.

2.6 Vizuální SLAM

V předchozích kapitolách byl probrán obecný algoritmus SLAM úlohy. Nyní se text zaměří na metody, které umožňují provádět simultánní lokalizaci a mapování s využitím snímačů obrazových dat. Algoritmus zahrnující tyto metody a snímače se nazývá *Visual SLAM*, nebo také zkráceně *V-SLAM*.

Velice důležitou vlastností dat získávaných ze snímačů používaných ve V-SLAMu je vysoké množství informace o okolí mobilního robota. Konkrétně je z obrazových dat možné získat následující informace:

- Informace o poloze (všechny tři souřadnice) jednotlivých významných bodů
- Barevná informace
- Jasová informace
- Informace o struktuře celé scény
- Texturní informace

Kromě těchto výhod mohou mít obrazová data i své nevýhody. Největšími nevýhodami jsou :

- Šum projevující se většinou jasové informaci [6]
- Rozmazání dat při pohybu.

[6] Záleží zejména na kvalitě snímače.

- Velké množství dat → Náročné na paměť a výkon počítače.

V následující kapitole budou uvedeny základní typy snímačů používané pro snímání obrazových dat.

2.6.1 Snímače

V následujících odstavcích budou popsány základní snímače používané v úloze V-SLAM. Je potřeba zmínit, že snímač Microsoft Kinect a RGB-D[7] kamera jsou téměř totožné a proto budou v práci popsány společně.

Microsoft Kinect (RGB-D kamera)

Tento snímač nebyl původně určen vývojářům aplikací, které nějakým způsobem pracují s obrazovými daty. Přesto si postupem času našel cestu z herní konzole XBOX360 na počítač. Před popsáním celého zařízení je vhodné prohlédnout si jeho podobu :

Obrázek 2.6: Snímač Microsoft Kinect - zdroj : Wikipedia

Na zařízení jsou viditelné celkem tři otvory se snímači. Krajní jsou určeny ke snímání informace o hloubce obrazu. Prostřední je snímač slouží jako RGB kamera, jejíž funkce je shodná s funkcí klasické kamery. Vlastnosti snímačů jsou následující:

Hloubkový snímač : 640×480 pixelů po 8 bitech

RGB kamera : 640×480 pixelů po 24 bitech

Oba dva snímače snímají s frekvencí 30 Hz.

K použití snímače v počítači je potřeba vybrat ovladače. V současné době existují dvě možnosti. První možností jsou oficiální ovladače a SDK[8] od společnosti Microsoft. S

[7] Kamera snímající informaci o jasu a hloubce scény (Red Green Blue Depth camera).
[8] Sada vývojářských nástrojů

oficiálním SDK lze pracovat v programovacím jazyku *C++* a rodině jazyků *.Net*. Duhou možností je použití framework OpenNi, který je určený pro jazyk *C++*. Práce s kinectem je v obou případech podobná a lze dosáhnout stejných výsledků.

V současnosti se cena Kinect snímače pohybuje kolem 3000 Kč. Jedná se tedy o dražší variantu než v případě klasické kamery, na druhou stranu je zde jako bonus přítomen snímač hloubkových dat, což může při implementaci a řešení SLAM úlohy velice usnadnit práci.

Kinect není jediným zařízením umožňujícím současné snímání obrazových a hloubkových dat. V této práci byl zvolen jako příklad takzvaných RGB-D kamer zejména z důvodu největšího rozšíření a stálé podpory od výrobce.

Webová kamera

Jedná se o výrazně levnější variantu předchozího snímače. Na druhou stranu je kamera schopna zaznamenávat pouze obrazová data, což může, jak bude probráno později, pro programátora představovat závažný problém. Výhodou je velké množství zařízení na trhu. Od nejlevnějších, které dovedou snímat obraz s rozlišením 640x480 pixelů. Nejdražší webové kamery dokážou snímat obraz až v HD rozlišení. V praktické části bude použita kamera uvedená na obrázku 2.7[9].

Obrázek 2.7: Webová kamera Canyon (použitá v praktické části)

Z uvedených faktů je patrné, že je možné sehnat levné zařízení, které má minimálně srovnatelné rozlišení obrazových dat jako v případě snímače Microsoft Kinect. Lepší rozlišení přispívá zejména k redukci šumu v obrazu. Což zvyšuje přesnost metod pro práci s obrazovými daty. Vzhledem k úloze SLAM se jedná zejména o metody detekující významné

[9]Zdroj : http://www.lan-shop.cz/img/kamery-pro-pc/canyon/50761/detail_1

body v obraze. Obecně se může jednat například také o metody pracující s texturou obrazu.

Pro práci s kamerou existuje mnoho různých knihoven. Pro jazyk C++ je velmi rozšířena sada knihoven OpenCV, která disponuje nejen funkcemi pro práci s kamerou, nýbrž také množstvím hotových funkcí pro práci s obrazovými daty. O OpenCV bude řeč v kapitole Implementace.

2.7 Monokulární SLAM

Práce se dále bude zabývat pouze monokulárním SLAMem. Tento typ přináší řadu nepříjemností, které je potřeba řešit. Největším problémem je inicializace a určení vzdálenosti významného bodu od kamery. Tyto metody budou vyloženy na konci kapitoly.

2.7.1 Základní metoda

Stejně jako u klasického SLAMu i zde existuje více možných způsobů jak k úloze přistupovat. Na dalších řádcích bude proto vyložena jedna z možných variant, která byla podrobně popsána v roce 2007 v článku [5]. Základem této metody je pravděpodobnostní přístup k mapě jako celku i k spravovaným významným bodům. Mapa je reprezentována odhadem pozice kamery, která snímá okolí a odhadem polohy významných bodů včetně nejistoty reprezentované elipsoidem. Celá reprezentace je průběžně aktualizována pomocí rozšířeného Kalmanova filtru. Z toho vyplývá, že jsou odhady poloh obnovovány během pohybu kamery i při pozorování okolí. Při jednom kroku můžou nastat tři stavy, které je třeba řešit. Může být objeven nový významný bod, který je přidán do mapy a rozšíří tak stav celého systému. Druhou možností je objevení již známého významného bodu a třetí možností je stav, kdy je některý významný bod nepotřebný, nebo se ho delší dobu nedaří detekovat. V takovém případě je stav celého systému zmenšen a tento bod je ze systému kompletně vymazán.

Mapu lze matematicky zapsat pomocí stavového vektoru $\hat{\mathbf{x}}$ a kovarianční matice $\mathbf{P}$. Stavový vektor má podobu

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_\mathbf{v} \\ \hat{\mathbf{y}}_\mathbf{1} \\ \hat{\mathbf{y}}_\mathbf{2} \\ \vdots \end{bmatrix},$$

kde $\hat{\mathbf{x}}_\mathbf{v}$ je odhad pozice kamery a $\hat{\mathbf{y}}_\mathbf{i}$ jsou odhady pozic významných bodů. Dále definujme kovarianční matici

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy_1} & \mathbf{P}_{xy_2} & \cdots \\ \mathbf{P}_{y_1x} & \mathbf{P}_{y_1y_1} & \mathbf{P}_{y_1y_2} & \cdots \\ \mathbf{P}_{y_2x} & \mathbf{P}_{y_2y_1} & \mathbf{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix},$$

kde podmatice $\mathbf{P}_{ab}$ představují kovarianční matice mezi prvkem a a b. Z uvedeného je dále patrné, že celková pravděpodobnost je v tomto případě aproximována jedním vícerozměrným Gaussovým rozdělením o dimenzi rovné velikosti vektoru stavu.

Pozice kamery v tří-dimenzionálním prostoru přirozeně není skalár. Jedná se o vektor obsahující globální pozici kamery, natočení kamery v prostoru, rychlost pohybu kamery a úhlovou její rychlost. zápis vektoru vypadá následovně

$$\hat{\mathbf{x}}_{\mathbf{v}} = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \\ \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix},$$

kde $\mathbf{r}$ je tříprvkový vektor pozice kamery, $\mathbf{q}$ je quaternion reprezentující natočení kamery v prostoru, $\mathbf{v}$ je tříprvkový vektor rychlosti pohybu kamery a $\boldsymbol{\omega}$ je tříprvkový vektor představující úhlovou rychlost kamery. Celková velikost vektoru $\mathbf{x}_{\mathbf{v}}$ pro tří-dimenzionální případ je 13 čísel reprezentujících uvedené informace.

2.7.2 Modelování pohybu

Pro větší obecnost bude modelování pohybu uvažováno pouze jako pohyb kamery bez použití mobilního robota, který by do systému přiváděl informace o aktuálním pohybu v prostoru. Z pohledu této úlohy se jedná o dva modely z celé spousty typů, které by bylo možné uvažovat. Navíc i v případě mobilního robota, který by systém o tyto informace obohacoval se může stát, že dojde k prokluzu kol, či jinému problému a apriorní informace bude znehodnocena. V obou případech tedy dochází k pohybům, které nejsou v systému přímo modelovány. Výhodou je možnost doplnit je do systému pomocí pravděpodobnostního modelování.

V případě tohoto konkrétního modelu bude předpokládána "konstantní rychlost a konstantní úhlová rychlost" pohyblivé kamery. To ovšem neznamená, že by se kamera během celého procesu pohybovala stále stejným způsobem. Pouze je počítáno s nedeterministickými změnami zrychlení kamery.

V každém časovém kroku je předpokládáno náhodné zrychlení $\alpha \sim N(0, \sigma_V^2)$ a náhodné úhlové zrychlení $\beta \sim N(0, \sigma_\Omega^2)$. Zrychlení pak způsobí impulz rychlosti $\mathbf{V}$ a úhlové rychlosti $\boldsymbol{\Omega}$ v nějakém předem neznámém směru. Impulzy jsou uspořádány do vektoru $\mathbf{n}$

$$\mathbf{n} = \begin{bmatrix} \mathbf{V} \\ \boldsymbol{\Omega} \end{bmatrix} = \begin{bmatrix} \alpha \Delta t \\ \beta \Delta t \end{bmatrix}$$

Model pohybu kamery pro výpočet vektoru $\mathbf{x}_{\mathbf{v}}$ v čase $k+1$ má následující tvar:

$$\mathbf{f}_{\mathbf{v}} = \begin{bmatrix} \mathbf{r_{k+1}} \\ \mathbf{q_{k+1}} \\ \mathbf{v_{k+1}} \\ \boldsymbol{\omega}_{\boldsymbol{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{r_k} + (\mathbf{v_k} + \mathbf{V})\Delta t \\ \mathbf{q_k} \times \mathbf{Q}((\boldsymbol{\omega}_{\boldsymbol{k}} + \boldsymbol{\Omega})\Delta t) \\ \mathbf{v_k} + \mathbf{V} \\ \boldsymbol{\omega}_{\boldsymbol{k}} + \boldsymbol{\Omega} \end{bmatrix}$$

kde označení $\mathbf{Q}((\boldsymbol{\omega_k}+\boldsymbol{\Omega})\Delta t)$ je quaternion vytvořený z úhlové rychlosti a impulzu úhlové rychlosti v kroku k. Po aplikaci odhadu $\mathbf{f_v}$ je potřeba vypočítat zvýšení nejistoty stavu ve formě kovarianční matice procesního šumu $\mathbf{Q_v}$ vypočítané pomocí výpočtu jakobiánu $\frac{\partial \mathbf{f_v}}{\partial \mathbf{n}}$ a kovarianční matice $\mathbf{P_n}$ příslušné vektoru $\mathbf{n}$.

$$\mathbf{Q_v} = \frac{\partial \mathbf{f_v}}{\partial \mathbf{n}} \mathbf{P_n} \frac{\partial \mathbf{f_v}}{\partial \mathbf{n}}^T,$$

Kovarianční matice $\mathbf{P_n}$ obsahuje informaci o hladkosti pohybu. Při malých hodnotách v matici $\mathbf{P_n}$ je očekáván hladký pohyb s malým zrychlením, ale systém si pak neporadí s rychlými změnami pohybu. Při velkých hodnotách je v systému větší míra nejistoty. To v praxi znamená, že pro vytvoření dobrého odhadu je třeba udělat více pozorování, odměnou pak je schopnost vyrovnat se i se změnami zrychlení kamery.

2.7.3 Významné body

Významné body, nebo-li významné body jsou velice důležitou součástí celého problému SLAM. Bez nalezení těchto bodů by nebylo možné cokoliv odhadovat. Naštěstí je okolí plné míst, která jsou lehko zapamatovatelná. Tento fakt platí i pro strojové hledání významných bodů. Za významný bod může být považováno takové místo, které je dostatečně odlišné od svého okolí a je rozpoznatelné z více směrů a vzdáleností.

Významné body je možné dělit podle více kritérií. Jedním z nich je dělení podle způsobu vzniku místa s významným bodem :

- **Přírodní významný bod** - Oblast, která se ve scéně přirozeně vyskytuje. To znamená, že se jedná o útvar nebo předmět, který nebyl do scény záměrně dodán.
- **Syntetický významný bod** - Předmět, který se ve scéně vyskytuje díky zásahu člověka. Může se jednat například o výrazně zbarvený geometrický útvar, který pomáhá při orientaci mobilního robota.

Druhým možným dělením je dělení podle detekční metody, která je na body použita:

- Roh
- Přímkový úsek
- Oblast s výraznou texturou
- Barveně odlišená oblast

Některé metody detekce významných bodů budou dále rozebrány podrobněji.

2.7.4 Detekce významných bodů

Cílem metod pro detekci významných bodů je detekce dostatečně výrazných částí ve scéně tak, aby tyto body (oblasti) bylo možné najít na více snímcích. Pokud by scéna byla statická, pak by se tato vlastnost mohla nazývat opakovatelností. Právě opakovatelnost je jedna z nejdůležitějších vlastností těchto metod. Druhou důležitou vlastností je náchylnost na šum a na změny v obraze. V dalších kapitolách budou probrány jednotlivé metody, jejich vlastnosti a v kapitole Implementace a výsledky se následně shrnou praktické implementace a výsledky těchto metod.

Harrisův operátor

Metoda pro kombinovanou detekci hran a rohů byla poprvé představena v roce 1988 v článku [11]. Princip metody spočívá v nalezení míst v obraze, které jsou nejvíce odlišné od svého okolí. Respektive v překonání určitého prahu určeného pro rozhodnutí o označení konkrétního bodu. Jinak je možné také říci, že hledáme místo s největším gradientem vůči svému okolí. Vzhledem k tomu, že se tato metoda může potýkat s problémem šumu v obraze, je na obraz aplikováno Gaussovské rozostření G s rozptylem σ_1^2. Algoritmus metody je možné shrnout následovně

1. Načtení obrazu I a aplikace Gaussovského rozostření pro odstranění šumu.
2. Vypočtení gradientu pomocí konvoluce se speciálními vektory[10]

$$X = I \otimes (-1, 0, 1) = \frac{\partial I}{\partial x}$$

$$Y = I \otimes (-1, 0, 1)^T = \frac{\partial I}{\partial y}.$$

3. Sestavení matice $\mathbf{M}$ ve tvaru

$$\mathbf{M} = G(\sigma_2^2) \otimes \begin{bmatrix} x^2 & xy \\ xy & y^2 \end{bmatrix},$$

   kde $G(\sigma_2^2)$ je opět Gaussovské rozostření.

4. Dalším krokem je určení hodnoty funkce R, která vypovídá o typu a hodnotě významného bodu.

$$R = \alpha\beta - k(\alpha + \beta)^2,$$

   kde k se většinou volí $0,04$. Ve vztahu vystupují vlastní čísla. Funkci však lze vypočítat i bez nich zavedením následujících vztahů

$$\mathrm{Tr}(\mathbf{M}) = \alpha + \beta = x^2 + y^2$$

[10] $\otimes$ - znak pro konvoluci

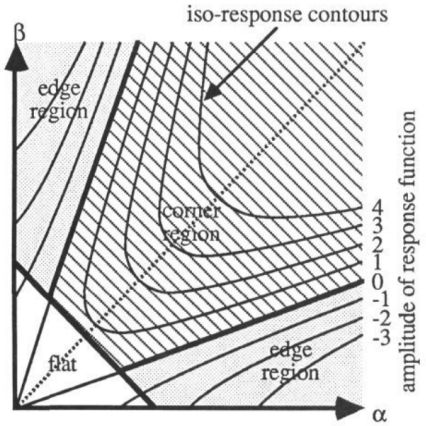
$$\mathrm{Det}(\mathbf{M}) = \alpha\beta = x^2y^2 - xy$$

Funkce R má pak tvar

$$R = \mathrm{Det}(\mathbf{M}) - k\mathrm{Tr}(\mathbf{M})^2$$

5. Funkce R je vypočtena pro každou dvojici x,y bodů v obrazu. Vzniká tak matice $\mathbf{M_R}$, kde je možné na pozicích x,y nalézt lokální maxima, které jsou při překročení nastaveného prahu označeny za významné body.

V algoritmu je uveden vztah mezi hodnotou funkce R a vlastními čísly α, β. Pomocí vlastních čísel je také možné určit typ významného bodu. Vše ilustruje obrázek 2.8.



Obrázek 2.8: Graf závislosti typu významného bodu na hodnotách vlastních čísel matice M. Zdroj : [11]

Shi-Tomasi

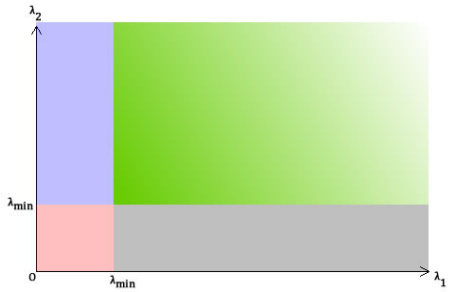
Metodou se zabývá článek [12] vydaný v roce 1994. Metoda z velké většiny vychází z Harrisova operátoru a ve výsledku pouze upravuje hodnotící funkci R na tvar

$$R = \min(\alpha, \beta).$$

Ačkoliv se zdá, že se jedná o jednodušší vzorec, je zde na rozdíl od Harrisova operátoru nutné znát vlastní čísla matice M. Tím se zvyšuje výpočetní náročnost.

Přesto je tento algoritmus používanější a dle dostupných informací dosahuje lepších výsledků než první zmíněný. Rozdělení prostoru na typy významných bodů podle vlastních čísel v případě Shi-Tomasi algoritmu je uvedeno na obrázku 2.9[11] Umístění oblastí je přibližně stejné jako u Harrisova operátoru.

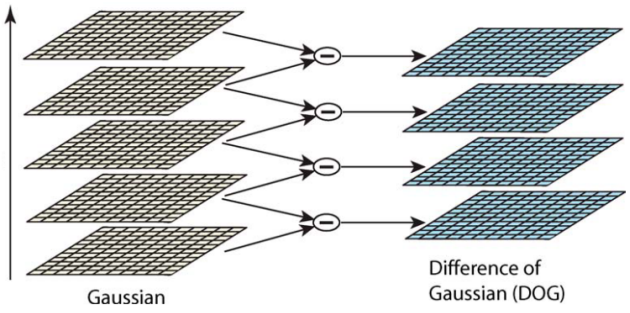
[11] Místo α a β je použito značení λ_1 a λ_2.



Obrázek 2.9: Graf závislosti typu významného bodu na hodnotách vlastních čísel matice M pro metodu Shi-Tomasi. Zdroj : *http://www.aishack.in/*

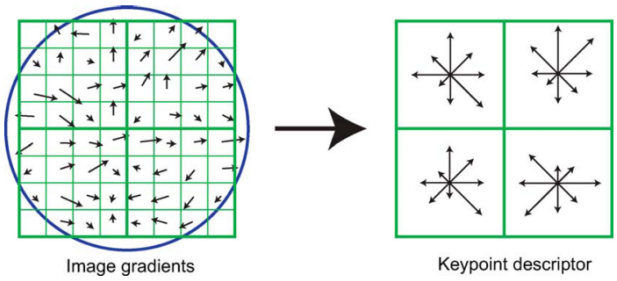
SIFT

V tomto případě se již jedná o více propracovanou metodu pro detekci významných bodů. Celý název metody je Scale-Invariant Feature Transform, z čehož je možné usuzovat, že se jedná o hledání významných bodů nezávisle na měřítku vstupních dat.
Prvním úkolem metody je získat takzvaný DoG (Diference of Gaussian - Diferenci Gaussovské funkce). K tomu je potřeba nejdříve provést konvoluci obrazových dat s Gaussovskou funkcí s různým nastavením rozptylu. Z toho vyplývá, že je získáno několik rozostřených obrazů. Odečtením jednotlivých po sobě jdoucích rozostřených obrazů je získán takzvaný scale-space složený z DoG obrazů.



Obrázek 2.10: Ukázka takzvaného scale-space. Zdroj : [18]

Detekce významných bodů pak probíhá na těchto snímcích hledáním oblastí s velkým kontrastem v jednom směru a malým ve směru kolmém. Celá oblast významného bodu se dále rozdělí na menší části, ve kterých se spočítají jednotlivé lokální orientace kontrastu. Z nich je následně vypočtena hlavní orientace, kterou je významný bod popsán.



Obrázek 2.11: Princip rozdělení oblasti významného bodu a následné určení orientace. Zdroj : [18]

SURF

Metoda SURF je do značné míry založená na předchozí metodě SIFT. Celý název této metody je Speeded-Up Robust Features. Jak už název napovídá vznikla kvůli urychlení celého procesu detekce, který může být u metody SIFT delší než je potřebné pro realtime aplikace. Mnoho informací k metodě lze najít v článku [14].

Narozdíl od předchozích metod se při použití SURF algoritmu musí provést preprocessing dat. Tato příprava spočívá v převedení obrazových dat do tvaru takzvaného integrálního obrazu. Tato reprezentace je získána tak, že do každé buňky na souřadnicích (x, y) je uložen součet všech pixelů v obdelníkovém útvaru začínajícím na souřadnicích $(0, 0)$ a končícím na pozici (x, y). Výhodou je, že pomocí těchto hodnot je možné následně získat jakýkoliv obecný obdelníkový útvar uvnitř integrálního obrazu. Matematicky lze proces tvorby zapsat následovně

$$I_\Sigma(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j),$$

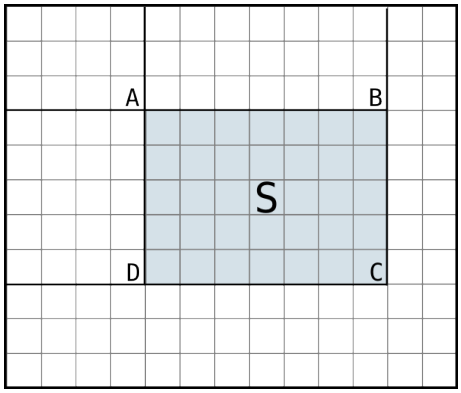
kde I jsou obrazová data a I_Σ je integrální obraz. Informace o obdelníku pak lze získat následujícím způsobem :

$$I_\Sigma(A, C) = C + A - B - D.$$

Význam jednotlivých písmen je zřejmý z obrázku 2.12
K detekci významných bodů se využívá integrálního obrazu, na který se v daném bodě aplikuje LoG[12] konvolucí s druhou derivací Gaussovy funkce. Vzniká tak Hessova matice ve tvaru

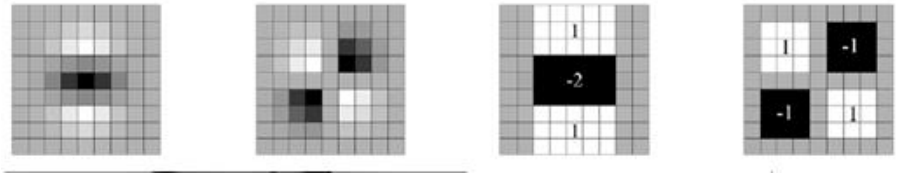
$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}$$

[12]Laplacian of Gaussian



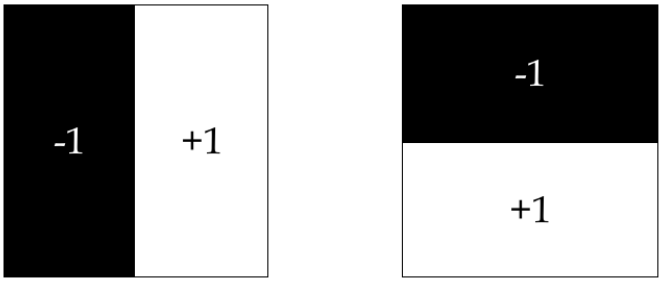
Obrázek 2.12: Integrální obraz. Zdroj : Obrázky Google

kde L je LoG a σ parametr Gaussovy funkce. V případě metody SURF je samotný LoG nahrazen aproximací tohoto operátoru. Na následujícím obrázku jsou vidět zleva originální LoG operátory a dále jejich používané aproximace.



Obrázek 2.13: Porovnání Log (obr 1 a 2 zleva) a aproximací používaných v metodě SURF (obr 3 a 4). Zdroj : [14]

Aplikací více různě velkých aproximací je získán scale-space jako u metody SIFT a významné body jsou hledány stejným způsobem. Tyto body lze následně popsat pomocí jejich orientace. tím je získána nezávislost na natočení bodu. Základem nalezení orientace je aplikování Haar wavelets operátoru viz obrázek 2.14.



Obrázek 2.14: Haar wavelets používané pro popis orientace bodu. Zdroj : [14]

Ty jsou aplikovány na okolí významného bodu neboli takzvanou oblast zájmu. Ta je rozdělena na 16 částí, pro každou z nich je následně spočteno 25 odezev na Haarův operátor. Následně jsou z výsledků vypočteny následující vlastnosti

$$\sum d_x \sum \mid d_x \mid, \sum d_y \sum \mid d_y \mid .$$

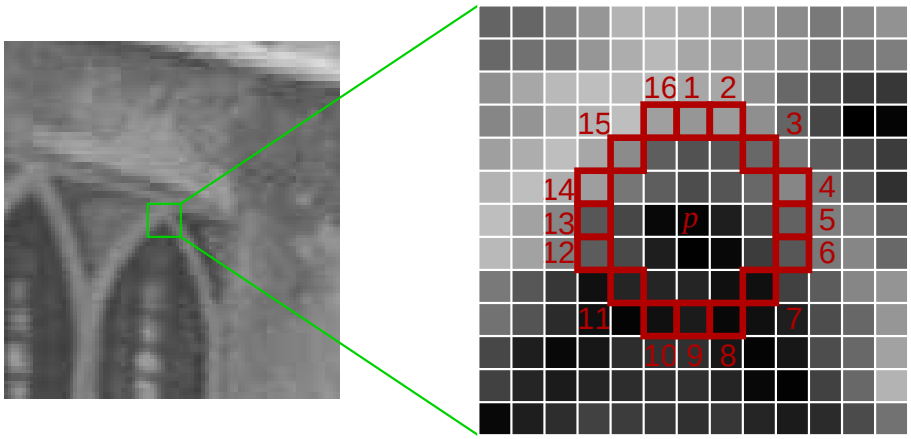
Jedná se o odezvy na operátor v jednotlivých směrech bez a s absolutní hodnotou. To v praxi znamená, že je získán vektor popisující každou vlastnost o délce 16 buněk. Po sjednocení všech vektorů je bodu přiřazen vektor o délce 64 popisující orientaci vektoru a sloužící k možné asociaci dat.

FAST

Metoda pracující na trochu jiném principu, než výše uvedené. Namísto aplikace určitého operátoru je zkoumáno okolí pixelu p o souřadnicích x, y přímo. Je vybráno 16 bodů na kruhovém okolí vybraného pixelu. Mohou nastat celkově 3 možnosti :

- Minimálně 12 bodů má vyšší hodnotu jasu o více než předem určený práh.
- Minimálně 12 bodů má nižší hodnotu jasu o více než předem určený práh.
- Ani jedno z předchozích neplatí.

V prvních dvou případech je daný pixel určený jako významný bod vzhledem ke svému okolí. Princip je dále ukázán na obrázku 2.15.



Obrázek 2.15: Princip metody Fast. Zdroj : [15]

Hodnoty pixelů s odečteným prahem se následně dají využít pro popis významného bodu. Budou tak existovat pozitivní a negativní významné body, přičemž není nutné porovnávat pozitivní a negativní body mezi sebou.

Ostatní metody

Mezi další metody patří například Moravcův operátor, ze kterého vychází výše uvedený Harrisův operátor a jemu podobné metody. Samostatnou skupinou jsou potom metody analyzující obraz na základě textury. Jednou z nich je například metoda Local Binary Patterns (LBP), která popisuje texturu obrazu na základě lokálních změn jasu v jednotlivých směrech. Tuto metodu by také bylo možné použít také jako doplňující metodu pro popis významných bodů, které však byly nalezeny jinou metodou. Ke každému bodu by byl přiřazen histogram četností obsahující veškerou informaci o textuře okolí významného bodu. Tento popis by mohl ulehčit následnou asociaci dat respektive ověření shodnosti dvou bodů, o kterém bude řeč v další kapitole.

2.7.5 Ověření shodnosti bodů

Shodnost bodů je velice důležitá pro simultánní lokalizaci a mapování. Tato asociace dat je složitá avšak nezbytná část celého algoritmu, neboť špatné přiřazení nových dat k těm, které již jsou v systému uložené by dozajista vedlo k znehodnocení výsledků a naprostému selhání SLAMu. V této práci budou uvedeny dvě metody. Přičemž u obou se předpokládá vysoký poměr frekvence snímkování ku rychlosti pohybu. Pokud bude tento poměr správný, pak na dvou po sobě jdoucích snímcích budou pouze malé posuny významných bodů. Metoda nejbližšího souseda by při nesplnění poměru selhala.

Metoda nejbližšího souseda

Nejjednodušší metoda pro asociaci nových dat s daty již používanými v EKF algoritmu. Principem algoritmu je k objevenému bodu najít co nejbližší již určený významný bod. Porovnání se provádí pomocí Euklidovské vzdálenosti, přičemž je snaha tuto vzdálenost minimalizovat. Obecný tvar výpočtu pro dimenzi n je

$$\min\left\{d(\mathbf{a},\mathbf{x}_i)\right\} = \min\left\{\sqrt{\sum_{j=1}^{n}(x_{ij}-a_j)^2}\right\},$$

kde a je nově nalezený bod a x_i je významný bod nacházející se již v systému. V úloze vizuálního slamu se pak většinou využívá dvou dimenzí. Tedy polohy bodů v obraze.

Problémem tohoto algoritmu je zejména náchylnost na chybovost při více detekovaných bodech na velmi malé oblasti. To samé platí pro rychlý pohyb kamery, neboť čím větší pohyb kamera vykoná, tím je větší pravděpodobnost, že se nový bod přiřadí k jinému bodu v systému. Řešením může být doplnění bodů o nějaký popis. Například texturní popis, nebo popis pomocí deskriptorů SIFT respektive SURF metody.

Ransac

Metoda Ransac (Random Sample Consensus) vhodně doplňuje deskriptory detekčních metod. Pomocí tohoto algoritmu se dají zpřesnit odhady o korespondenci dat ve dvou

snímcích. Detektory jsou i bez použití Ransacu kvalitní, ale mohou se mezi nimi vyskytovat falešné korespondence.

Nejjednodušší případ použití metody Ransac je rozhodnutí, jestli skupina bodů leží na dané přímce. Ze skupiny bodů je náhodně vybrána podmnožina a tou je proložena přímka. Dále se porovnává vzdálenost bodů od této přímky. Pokud vybrané body leží v menší vzdálenosti než je předem určený práh, pak jsou body označeny jako náležící přímce. Je vytvořen model a pokud vyhovuje modelovému prahu T, pak je potvrzena hypotéza, že body leží v přímce. Náhodný výběr podmnožiny se dělá opakovaně až do zvoleného počtu N opakování. Pokud není nalezena ani jedna vyhovující přímka, není hypotéza přijata. Obecný algoritmus je možné shrnout do následujících bodů:

1. Z množiny dat je vybrán určitý počet bodů potřebných pro určení dané hypotézy. Pro přímku je možné využít minimálně dva body.
2. Pro každý bod z dané množiny se vypočítá odchylka od vytvořené hypotézy.
3. Body s nižší odchylkou než práh se označí jako správné, ostatní jako špatné. Model je následně ohodnocen těmito hodnotami.
4. Pokud je nalezen model, který vyhovuje prahu T pro přijetí modelu, je model přijat a vytvořen pouze ze správných bodů. V opačném případě je celý algoritmus opakován až do určeného počtu N.

Z výše uvedeného je patrné, že metoda odstraňuje špatně přiřazené body. Obecně řečeno špatně zadaná data. O správnosti hypotézy rozhoduje pomocí výběru minimálního množství dat. V případě přímky se jedná o pouhé dva body. Pro metodu monokulárního SLAMu je dokonce popsána metoda nazvaná 1-point ransac uvedená v článku [17], která využívá pouze jednoho bodu

2.7.6 Inicializace a měření významných bodů

Inicializací významného bodu se v tomto případě myslí zařazení bodu do algoritmu EKF SLAMu. Toto zařazení může probíhat buď vícekrokově, nebo se jedná o inicializaci jednokrokovou, která významný bod do hlavní smyčky zařadí ihned po objevení bodu. Jednotlivé metody budou popsány v následujících odstavcích.

Vícekroková inicializace

Jedná se o nejzákladnější přístup k problému inicializace 3D polohy významného bodu. Vychází jednoduše z faktu, že z jednoho snímku nelze určit informaci o vzdálenosti od kamery. Proto tento bod není hned zařazen do hlavní smyčky, ale je namísto toho zpracováván zvlášť. Po pořízení dalších snímků a odhadů polohy významného bodu je pomocí triangulace odhadnuta vzdálenost a bod je konečně zařazen do algoritmu EKF.

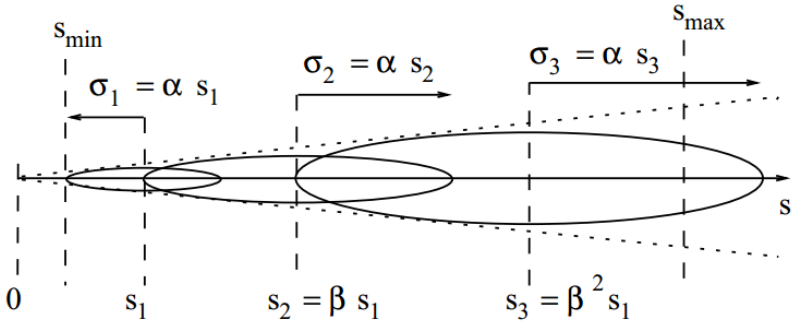
Metody, které slouží k odhadu vzdálenosti bodu od kamery využívající pohybu jsou popsány v článku [6]. Uvedené metody se liší ve způsobu, v jakém reprezentují data používané k odhadu třetí souřadnice.

- **2D do 2D** - Sady bodů detekované ze dvou různých snímků a uchovávané pouze jako 2D data[13]
- **3D do 3D** - Dvě sady bodů vytvořené triangulací z několika snímků.
- **3D do 2D** - Sada bodů z minulých dat je uchovávána ve 3D podobě určené pomocí triangulace. Nová data jsou uchovávána jen ve formě 2D dat.

Problémem tohoto přístupu může být rychlý pohyb kamery okolo významného bodu. Může nastat situace, kdy se významný bod ztratí dříve, než je nasnímáno dostatečné množství dat. Z toho vyplývá také fakt, že čím delší dobu kameře trvá načíst jeden snímek, tím pomalejší by měl být pohyb kamery, aby nemohlo dojít k této nepříjemné situaci. Řešením mohou být metody jednokrokové, které se snaží třetí souřadnici odhadnout okamžitě.

Jednokroková inicializace určení pomocí hypotézy

Jednou z metod, která se snaží o inicializaci bez zpoždění je metoda uvedená v článku [8]. Základem metody je určení polopřímky vedoucí od ohniska kamery směrem k bodu a dále za něj do nekonečna. Na tuto polopřímku je dále aplikována hypotéza v podobě řady geometrických Gaussových funkcí.



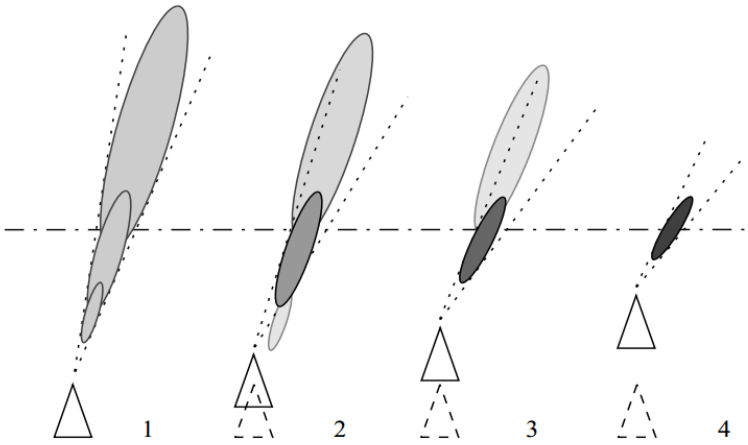
Obrázek 2.16: Hypotézy Gaussových funkcí zobrazených ve formě elips. Zdroj : [8]

Každá z těchto elips[14] představuje odhad vzdálenosti s určitou nejistotou. Z toho vyplývají dvě asi největší nevýhody této metody. Zaprvé není možné obsáhnout těmito elipsami všechny vzdálenosti od ohniska až po nekonečno. Z toho důvodu je nutné předem určit v jakém rozsahu bude hypotéza aplikovaná a v jakém rozsahu bude tím pádem možné určit hloubku bodu ve scéně. Druhý problém je vidět z výše uvedeného obrázku. Na daný

[13] Bez třetí souřadnice.

[14] Respektive elipsoidů, ovšem pro představu postačí se na celou scénu dívat shora a zanedbat jednu ze známých souřadnic.

rozsah je určena hypotéza obsahující omezený počet elips. To prakticky znamená, že pomocí této metody není možné přesně určit hloubku. Ta je pouze skokově přiřazena na základě toho, která z elips se do systému přiřadí. Tento výběr také nemusí být správný, což může způsobit chybu v celém algoritmu. Průběh odhadu polohy je ukázán na následujícím obrázku.



Obrázek 2.17: Zpřesňování polohy v čase. Zdroj : [8]

Jednokroková inicializace určení pomocí metody inverzní hloubky

Za nejvhodnější metodu pro jednokrokovou inicializaci je možné považovat metodu Inverzní hloubky představenou v článku [9]. Tato metoda k inicializaci využívá jiné reprezentace významného bodu, než je klasická Euklidovská XYZ reprezentace

$$\mathbf{X_i} = [x_i, y_i, z_i]^T.$$

Naproti tomu takzvaná Inverse Depth[15] (Inverzní hloubka) reprezentace se skládá z vektoru čítajícího celkem šest údajů

$$\mathbf{y}_i = (x_i, y_i, z_i, \Theta_i, \Phi_i, \rho_i)^T,$$

kde x_i, y_i, z_i je optický střed kamery, Θ, Φ představují azimut a elevaci natočení kamery v prostoru a ρ je člen inverzní hloubky, matematicky $\rho_i = \frac{1}{d_i}$, kde d_i je hloubka bodu. Vektor $\mathbf{y}_i$ obsahuje informace o bodu vzhledem k poloze, ze které byl tento bod poprvé pozorován.

Mezi Euklidovskou a ID reprezentací existuje převodní vztah, který je pro potřeby algoritmu SLAM nutné uvést

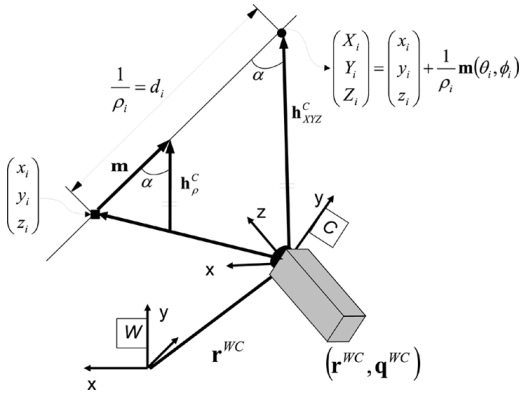
$$\mathbf{X_i} = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i}\mathbf{m}(\Theta_i, \Phi_i),$$

[15] ID reprezentace

kde vektor $\mathbf{m}$ se vypočítá z dostupných úhlů následovně

$$\mathbf{m} = (cos\Phi_i sin\Theta_i - sin\Phi_i, cos\Phi_i cos\Theta_i).$$

Inicializační hodnota inverzní hloubky ρ_0 se volí empiricky. Například v případě experimentů v článku [9] je zvolena hodnota $0,1$. Kompletní myšlenka je ilustrována na následujícím obrázku



Obrázek 2.18: Princip inverzní parametrizace. Zdroj : [9]

Ve zmíněném článku jsou k dispozici i podrobnější informace k celé problematice ID reprezentace. Navíc lze též některé informace najít ve starším článku [10].

2.7.7 Správa mapy

Nejdůležitější součástí systému je kvalitní správa mapy. Jedná se zejména o práci se sledovanými významné body. Zejména se musí kontrolovat následující vlastnosti

- Viditelnost
- Kvalita
- Výraznost
- Opakovatelnost - souvisí s použitou metodou pro detekci bodu

V případě nevyhovění některé vlastnosti nastaveným požadavkům je nutné významný bod vyřadit, neboť je pro celý algoritmus nepotřebný. Navíc by svou přítomností zatěžoval systém jak výpočetně tak i paměťově.

Další otázkou je uchovávání informací o jednotlivých významných bodech. Tento problém bude podrobněji rozebrán v následující kapitole společně s analýzou všech částí systému pro řešení SLAM úlohy.

2.8 Zajímavé druhy SLAMu

V této podkapitole budou uvedeny zajímavé přístupy k úloze simultánní lokalizace a mapování. Zejména pak algoritmy používající nezvyklé snímače, případně nezvyklý druh významných bodů.

2.8.1 WifiSLAM

Někdy také označovaný jako wiSLAM. K orientaci slouží RSS (received signal strength) signál. Poloha se určuje porovnáváním síly přijímaného signálu od jednotlivých wifi vysílačů v okolí. WiSlam může být také použit jako doplněk pro jiné druhy SLAMu. Například dále popsaný FootSLAM.

2.8.2 FootSLAM

Velice exotická podoba SLAM úlohy. Jako snímač je použit určitý druh krokoměru, který se snaží odhadovat pohyb člověka, který má tento snímač přichycený k noze. Více o tomto druhu SLAMu lze nalézt na [22].

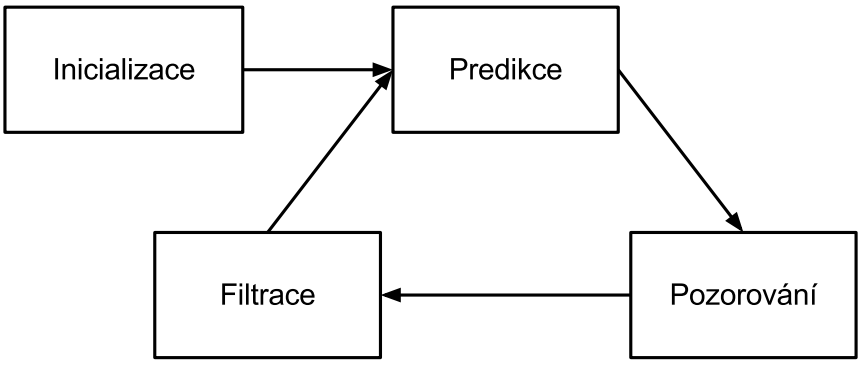
Kapitola 3

Analýza SLAM algoritmu

V následujících odstavcích se práce zaměří na analýzu SLAM respektive monokulárního problému SLAM. Cílem této analýzy je přehledné popsání úlohy jako celku i jako skupiny několika samostatných systémů za účelem snadnější implementace, která bude popsána v další kapitole. Z důvodu větší názornosti budou v dalších odstavcích uvedeny blokové diagramy SLAM úlohy. Dále pak budou jednotlivé části schémat popsány tak, aby bylo zřejmé o co se daný systém stará. V případě monokulárního SLAMu bude místo mobilního robota uvažována kamera pohybující se v prostoru zcela náhodně. To v praxi znamená, že žádný systém nebude mít informaci o pohybu robota z odometrického[1] měření.

3.1 Základní analýza SLAM úlohy

Základ celého algoritmu tvoří čtyři části, které na sebe bezprostředně navazují. Jejich návaznost je zobrazena na následujícím blokovém schématu.



Obrázek 3.1: Základní schéma SLAM algoritmu.

[1] Měření počtu otočení kol.

Každá část má v systému specifickou a nezbytnou funkci. Jako první se spouští inicializace systému, která má za úkol připravit prostředí systému na řešení SLAM úlohy a případně připravit i samotného mobilního robota (zapnutí, odbržední, ...). Po inicializaci se algoritmus přesouvá do uzavřeného cyklu skládajícího se z následující trojice :

Predikce zajišťuje zejména aplikaci pohybového modelu a aplikaci časového kroku EKF algoritmu.

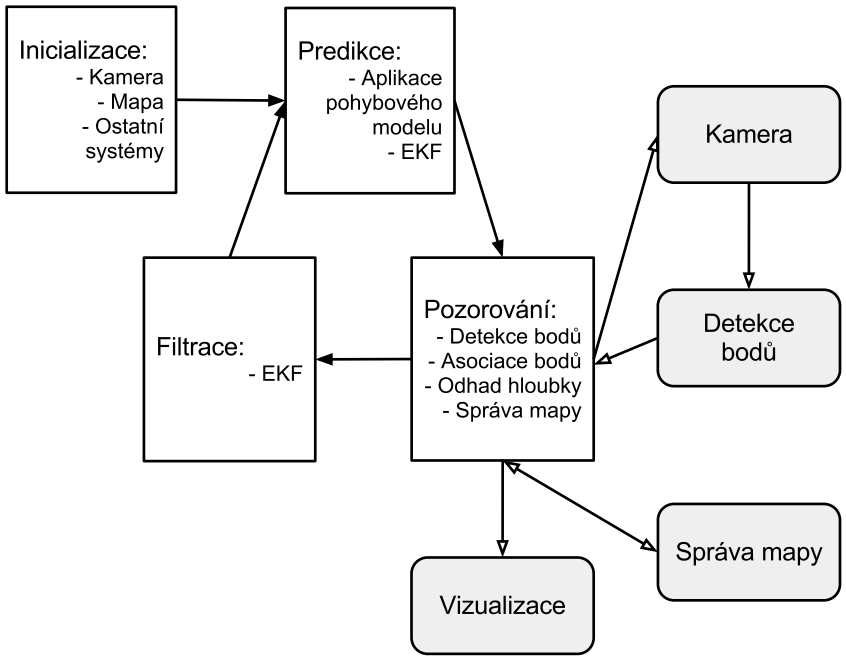
Pozorování obsahuje všechny úlohy pro správu pozorování a mapy.

Filtrace obsahuje filtrovací krok EKF algoritmu.

V dalších částech budou nejprve uvedené části popsány obecně a později bude text zaměřen podrobně také na drobnější systémové celky monokulárního SLAMu jako je systém pro detekci významných bodů nebo pro správu mapy.

3.2 Analýza monokulárního SLAMu

Na obrázku níže je vidět blokové schéma monokulárního SLAMu. Toto schéma vychází ze základního uvedeného v předchozím odstavci. Jsou zde navíc vyobrazené některé důležité systémy, které je potřebné později hlouběji rozebrat.



Obrázek 3.2: Schéma monokulárního SLAM algoritmu.

Zejména je třeba zmínit systém kamery pro získávání vstupních dat. V pozdějším rozboru bude tento systém navržen tak, aby byl co nejobecnější. Na systém kamery zajisté navazuje systém pro detekci a případný popis významných bodů. Díky vzájemné propojenosti je také možné tyto systémy sloučit do jednoho.

Dalším velice důležitým úkolem je správa mapy. Tento modul zajišťuje správné zacházení s významnými body, jejich párování s novými daty a případné mazání.

Posledním důležitým a velice variabilním modulem je modul vizualizace. Ta může být zajištěna jak ve 2D tak i ve 3D a podle toho je dále vybrána potřebná technologie. Další možností je vizualizace připomínající takzvanou rozšířenou realitu. Tedy situace, kdy jsou odhadované významné body promítány přímo do vstupních dat.

3.3 Podrobný rozbor monokulárního SLAMu

Jednotlivé části systému budou v následujících odstavcích rozebrány tak, aby šly následně implementovat v libovolném programovacím jazyku podporujícím objektově orientované programování.

3.3.1 Systém zajišťující inicializaci celé úlohy

Do této části můžeme shrnout všechny metody systému, které jsou volány před samotným započetím řešení problému SLAM. Konkrétně se jedná o rutiny zajišťující následující kroky

- Vytvoření programového modelu zajišťující spojení s kamerou, případně jiným zdrojem vstupních dat.
- Vytvoření programového modelu pro EKF a inicializace jednotlivých proměnných jako stav a kovariance a datová struktura pro uchovávání dat o významných bodech.
- Vytvoření potřebných částí systému pro vizualizaci. Například okno vstupních dat, okno 3D vizualizace a podobně.
- Předání všech inicializovaných součástí do hlavního cyklu aplikace.

3.3.2 Rozšířený Kalmanův filtr

Tento systém je na výše uvedeném schématu zakreslen celkem ve dvou blocích. To je ale v pořádku, jelikož se samotný Kalmanův filtr skládá ze dvou částí. Jedná se o predikci a filtraci.

Z uvedeného vyplývá, že nejlepším přístupem k implementaci EKF algoritmu je sjednocení těchto částí do společného celku. V terminologii OOP se mluví o třídě. Tato třída pak bude obsahovat dvě metody zajišťující aplikaci časového respektive filtračního kroku algoritmu. Kromě těchto metod by měl celek obsahovat také metody pro načtení a vyzvednutí jednotlivých proměnných (takzvané get a set metody).

3.3.3 Systém pro práci se vstupními daty

Tento systém by mohl být též nazván systémem pro práci s obrazovými daty. Prací s obrazovými daty se konkrétně myslí jejich načtení do vhodné reprezentace z různých zdrojů.

Vhodnou reprezentací je pro obrazová data nejlépe určitý druh matice. Případně objektu obsahujícího matici jednotlivých pixelů a další údaje doplňující načtená data. V dnešní době jsou systémy pro načtení obrazových dat přímo obsaženy ve většině programovacích jazyků, případně v doplňujících knihovnách. Horší situace je s načtením obrazových dat z jiných zdrojů, než přímo z obrázku uloženého v počítači. Přehled zdrojů je uveden v následujícím seznamu:

- Obrázek v počítači v různém formátu (jpg, png, bmp).
- Obrazová data uložená ve strojově zpracované podobě (například matice v matlabu).
- Webová kamera.
- Obrázek načítaný z webové kamery na jiném počítači.
- Data načítaná přes síť z jiného počítač/serveru.

Je patrné, že možných zdrojů dat je více a ke každému je nutné přistupovat jiným způsobem. Zvláště poslední dva zmíněné potřebují ke své činnosti přítomnost mezivrstvy komunikující přes síť s jiným počítačem, nebo například chytrým telefonem. To je důvod, proč je nutné tento systém navrhnout s ohledem na množství různých dat.

Je možné využít dva přístupy k tvorbě tohoto systému. Prvním je vytvoření rozhraní, které budou jednotlivé konkrétní vstupní systémy dědit a budou ho implementovat. To by mělo zaručit jednotnou práci se všemi zdroji vstupních dat. Druhou a jednodušší možností je vytvoření třídy umožňující pracovat s více zdroji. Toho je docíleno pomocí speciálního parametru, předávaného při vytváření objektu vstupního zařízení, určujícího typ zdroje.

3.3.4 Systém pro detekci významných bodů

S výše uvedeným souvisí i další systém, který má za úkol v načtených datech hledat významné body. V kapitole 2 byly některé vhodné metody uvedeny. Není samozřejmě problém do této části programu napsat více algoritmů a následně pomocí parametru volat konkrétní metodu.

Systém pro detekci významných bodů může být buď samostatný, nebo je možné ho implementovat v rámci systému pro práci se vstupními daty. V obou případech by však jeho metody měly splňovat určitá pravidla:

- Vstup metody by měl být v co největší míře jednotný. To znamená práci se stejnou reprezentací dat. Větším problémem je vyřešení ostatních parametrů úlohy. Ideálním řešením je vytvoření speciální obecné třídy pro uchování těchto parametrů. Díky dědičnosti je pak opět možné se v rámci aplikace k parametrům chovat jednotně, respektive nezávisle na typu metody, která bude použita.

- Výstup metody by měl být také určitým způsobem standardizován, aby odpadla nutnost sepisovat pro každý algoritmus speciální metodu pro práci s významnými body. Toho je dosaženo převodníkem uvnitř každé metody, které převede reprezentaci bodů na předem určený standard.

Splnění výše uvedených podmínek zaručí snížení objemu kódu a zvýšení efektivity a přehlednosti celého kódu. Je nutné však podotknout, že bez objektově orientovaného programování by se programátor zmíněným problémům jen těžko vyhýbal.

3.3.5 Systém pro práci s mapou

Nejrozsáhlejším systémem dle počtu metod a úkolů je systém pro správu mapy. Tento systém zajišťuje správné předávání dat mezi většinou ostatních systémů. Zodpovídá zejména za mazání nepotřebných významných bodů, za volání metody pro asociaci dat a za předání dat do systému pro vizualizaci celé úlohy.

Tohoto systému se také týká problém odhadu třetí souřadnice. V případě inverzní parametrizace je zde potřeba vytvořit systém, který bude pracovat jak s reprezentací euklidovskou obsahující souřadnice XYZ, tak i s reprezentací ID, která byla vysvětlena v kapitole 2. Součástí je samozřejmě i převod z ID reprezentace, ve které se body převádějí do souřadnic XYZ.

3.3.6 Systém pro asociaci dat

Význam asociace dat je velice důležitý, proto je tento systém popsán zvlášť a ne v rámci předchozího systému pro správu mapy. Tento systém by měl podobně jako systém pro detekci významných bodů pracovat velmi obecně, aby bylo možné do celého systému kdykoliv přidat novou metodu a zapojit jí do celku bez nutnosti velkých úprav. Opět zde ideálně zafunguje objektově orientované programování, díky němuž je možné vytvořit celý systém velmi obecně a definovat určitý standard volání jeho metod.

3.3.7 Systém pro vizualizaci dat

Tento systém není snadné popsat, jelikož může do značné míry záviset na typu úlohy. Celkově lze totiž vizualizaci pojmout různými způsoby:

- 3D vizualizace okolí se zobrazenými významnými body .
- 3D vizualizace s texturovaným okolím deformovaným pomocí významných bodů.
- 2D vizualizace.
- Vizualizace do vstupních dat.

Zajisté by bylo možné vymyslet další způsoby vizualizace. Jedinou podmínkou je pouze definice vstupních dat potřebných k vizualizaci a jejich správná interpretace. Není však od věci vytvořit mezi systémem vizualizace a zbytkem celé úlohy mezivrstvu obsahující rozhraní, díky které se z vizualizace vytvoří vyměnitelný modul.

3.4 Návrh testů

V této podkapitole budou popsány testy, které budou v rámci práce provedeny. Jedná se o dvě základní skupiny experimentů. Prvním bude ověření opakovatelnosti detekčních metod, což je jedna z nejdůležitějších vlastností těchto algoritmů pro úlohu monokulárního respektive vizuálního SLAMu. Druhou skupinou testů bude porovnání rychlostí jednotlivých metod.

3.4.1 Ověření opakovatelnosti detekčních metod

Opakovatelnost jako vlastnost se běžně testuje u různých druhů snímačů. Čím lepší snímač, tím lepší by měla být jeho opakovatelnost, tedy schopnost naměřit při stejných podmínkách vždy stejnou hodnotu.

U vizuálních snímačů se jedná nejen o vlastnost snímače, nýbrž i o vlastnost použité metody. Proto budou postupně vyzkoušeny jednotlivé metody s použitím stejné kamery.

Statická scéna a statická kamera

V prvním testu bude kamera obsahovat statickou scénu. Při teoretickém použití ideálního snímače by měla metoda ukazovat stále stejný počet bodů. Každá kamera však obsahuje šum, který může výsledky znehodnotit. Tento problém narůstá s nižší cenou výrobku. Při testech bude použita webová kamera Canyon, která patří do skupiny nejlevnějších webových kamer. Její maximální rozlišení je `640x480` pixelů s frekvencí 30 snímků za vteřinu.

Statická scéna s chybou a statická kamera

Při druhém testu bude do statického obrazu v polovině testu zanesena chyba ve formě změny světelných podmínek. Kromě samotné opakovatelnosti zde bude zkoumána také schopnost přizpůsobení se změně podmínek. Ideální metoda by byla invariantní vůči světlu. Takovou metodu si lze však jen těžko představit.

3.4.2 Test rychlosti jednotlivých metod

Kromě opakovatelnosti detekčních metod je též důležitá jejich rychlost. Ideálním případem by byla metoda s vynikající opakovatelností provedená s co nejmenší časovou náročností. Čím déle metodě celý proces detekce trvá, tím déle trvá jeden celý krok SLAM

algoritmu. To by mohlo vést ke zpomalení aplikace nebo dokonce ke kompletnímu znehodnocení výsledků. Cílem tohoto testu je nalezení metody, která bude vykazovat dobré výsledky v co nejkratším čase.

Kapitola 4

Návrh implementace SLAM algoritmu

Jedním z cílů této práce je, jak je patrné již z jejího názvu, návrh metod pro řešení úlohy vizuální simultánní lokalizace a mapování. V následující kapitole bude představen návrh implementace aplikace, která má za úkol daný problém řešit. Celý navržený systém bude podrobně rozebrán po menších celcích podobným způsoben, jako v kapitole 3. Rozdíl je zejména ve způsobu popisu jednotlivých částí. Zatímco v analýze byl text věnován obecnému popisu úkolu jednotlivých částí systému, následující kapitoly budou zaměřeny zejména na popis funkčnosti metod.

Při návrhu aplikace bude využito možností objektově orientovaného programování a celý systém bude představen tak, aby jej bylo možné implementovat v libovolném vhodném programovacím jazyce.

4.1 Programové vybavení

Před samotným popisem systému pro řešení problému slam je vhodné představit si programové vybavení, ve kterém je možné tento problém řešit a také to, ve kterém je psaný kód všech aplikací vytvořených v rámci této diplomové práce. Do programového vybavení patří výběr programovacího jazyka, knihoven rozšiřujících základní možnosti tohoto jazyka a také výběr jednotlivých technologií jako je například prostředí systému, pro který jsou tyto aplikace psány.

4.2 Programovací jazyk

Ve světě existuje velké množství programovacích jazyků. Faktem však zůstává, že každý z nich je vytvářen pro řešení určité kategorie problémů. Pro problém SLAM úlohy se nejvíce hodí následující jazyky:

- **Matlab** pro prototypování a zkoušení jednotlivých částí systému. Pro reálné nasazení se nehodí kvůli nízké rychlosti a vysokým nárokům na systém.

- **.NET** jazyky od společnosti Microsoft. Jedná se o rodinu jazyků s velmi kvalitním vývojovým prostředím. Nevýhodou zůstává oficiální podpora pouze pro operační systém Windows.
- **Java** je jazyk spravovaný firmou Oracle. Jedná se o jazyk interpretovaný a proto může být v aplikacích náročných na výpočty pomalejší.
- **C/C++** jsou jazyky vyznačující se dvěma výhodami. První z nich je vysoká rychlost výsledné aplikace. Zejména oproti interpretovaným jazykům jako .Net , jazyk Java apod. Druhou výhodou je možnost použít takové technologie a knihovny, které jsou dostupné na všech rozšířených operačních systémech, což vede k zajištění přenositelnosti kódu.

Z výše uvedených důvodů byl jako jazyk pro vytvoření návrhu aplikace pro SLAM vybrán jazyk C++. V další části bude uveden seznam použitých knihoven a s tím souvisejících technologií.

4.3 Knihovny

Kromě standardních knihoven jazyka C++ byly v aplikacích použity následující rozšiřující knihovny zajišťující zejména práci s obrazovými daty a zlepšení práce s matematickými výpočty.

OpenCV je knihovna obsahující metody pro práci s obrazovými daty a také pro práci s webovou kamerou. Jedná se o tematicky velmi rozsáhlý soubor algoritmů obsahující například metody pro transformace obrazu, práci se soubory obsahujícími obrazová data a také, pro SLAM důležité, metody pro detekci významných bodů v obraze.

Eigen do programovacího jazyka C++ přidává funkčnost týkající se matematických výpočtů. Díky této knihovně je možné velice jednoduše provádět maticové výpočty téměř tak pohodlně jako v prostředí Matlab, které je jednodušší snad jen v tom, že se všechny matice nemusí dopředu přesně alokovat v paměti počítače.

4.4 Technologie

Kromě knihoven bylo pro psaní a testování aplikace nutné použít další programové vybavení. Mezi tyto aplikace, které jsou obecně nazvány technologiemi určitě patří operační systém. V případě této práce byl používán operační systém Windows 7. Vzhledem k použitým technologiím by měl jít vytvořený software bez problému přeložit i v linuxových distribucích. Kromě operačního systému je však potřeba použít dvě další důležité technologie.

Cmake slouží k vytvoření projektu ze zdrojových kódů aplikace pro různá vývojová prostředí tak, aby se k aplikaci správně připojily všechny nezbytné hlavičkové soubory a k nim příslušející statické knihovny.

Visual studio bylo použito jako vývojové prostředí pro jazyk C++ v operačním systému Windows. Jeho předností je zejména doplňování psaného kódu a to včetně metod, které jsou napsané v jiném souboru aktuálního projektu.

4.5 Základní pohled na návrh aplikace

K vytvoření základního povědomí o struktuře aplikace nejlépe poslouží diagram vyobrazený na obrázku 4.1. Z diagramu je velice dobře vidět modulárnost celého systému. Tedy fakt, že je možné každou část celé úlohy modelovat jako samostatnou a díky tomu zajistit, aby bylo možné k těmto částem přistupovat jako k vyměnitelným modulům. Konkrétní sestavením těchto modulů následně vzniká konkrétní aplikace .

V diagramu je vidět několik základních tříd (modulů), které budou podrobněji popsány v dalších odstavcích. Mezi tyto třídy patří :

`MonocularSlam` je třída implementující základ algoritmu. Zejména metodu, která provede jeden časový krok algoritmu SLAM.

`EKF` je samostatnou třídou obsahující atributy a metody typické pro rozšířený Kalmanův filtr.

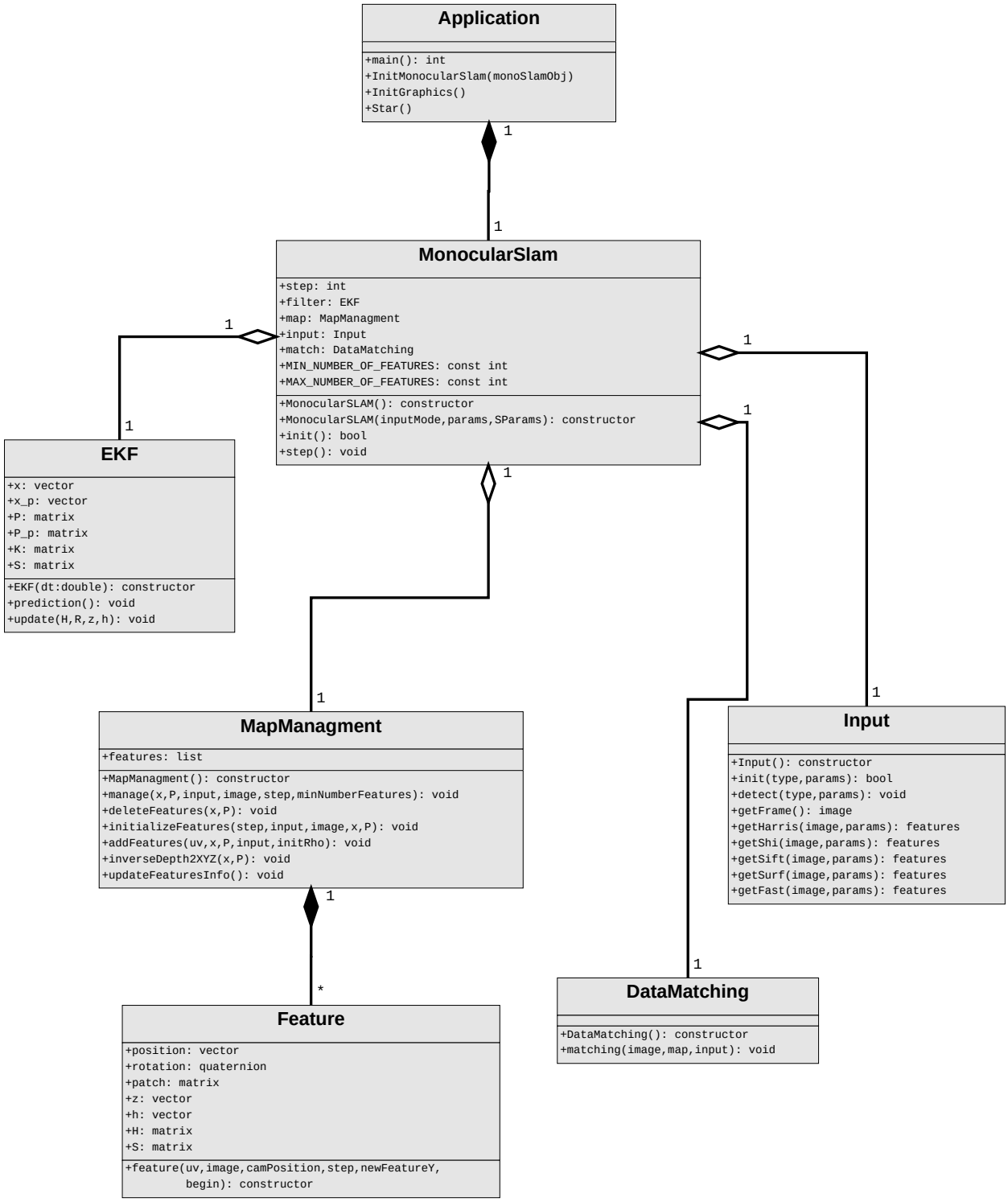
`MapManagment` se stará zejména o práci s mapou. Přidání, odebrání a aktualizace jednotlivých významných bodů. Ke své činnosti využívá třídu `feature`, která slouží k udržování informací o významném bodu.

`Input` je třída obsahující metody pro načítání nových dat a práci se zdrojovými zařízeními.

`DataMatching` obsahuje metody starající se o asociaci dat mezi novým snímkem a body uloženými ve vytvářené mapě.

Ve výše uvedeném výčtu záměrně není uvedená část v diagramu označená jako **Application**. V tomto případě se nejedná o samostatnou třídu, nýbrž o hlavní soubor celé aplikace pro řešení úlohy monokulárního SLAMu. Součástí hlavního souboru je také část pracující s grafikou. Tento systém není v návrhu konkrétně zaznamenán, jelikož se jedná až o finální výstup aplikace a do samotné úlohy nijak nezasahuje. Bude však dále ještě zmíněn.

Popis každého z modulů včetně hlavního souboru bude popsán v následujících podkapitolách. Součástí popisu bude také několik sekvenčních grafů ilustrujících postup činnosti důležitých metod.



Obrázek 4.1: Class diagram návrhu aplikace

4.6 Spuštění aplikace

Program má po načtení dva hlavní úkoly. Prvním je inicializace parametrů pro vstupní zařízení, vizualizační rozhraní a SLAM úlohu jako celek. Druhým úkolem je spuštění vizuálního rozhraní, které se bude starat o zobrazování výsledků a zároveň o volání potřebných metod SLAM algoritmu. O tyto činnosti se starají metody `InitMonocularSlam` vytvářející objekt třídy `MonocularSlam`, `InitGraphics` inicializující vizualizační část aplikace a `Start`, která celou úlohu zahájí.

Před spuštěním úlohy je potřeba nastavit několik parametrů, které budou dále použity. Parametry vstupního zařízení korespondují s kalibračními parametry kamery. Všechny jsou uvedeny v následujícím výpisu

Typ vstupního zařízení určuje typ snímače dat. Může se jednat o kameru, uložené soubory, síťové zařízení.

Číslo kamery je parametr použitelný pouze pokud je vstupním zařízením opravdu kamera. K počítači může být připojeno více kamer a číslem se provádí jejich výběr. Základní hodnota je 0.

Šířka a výška obrazových dat (snímků).

Střed snímku.

Ohnisková vzdálenost pro osu x a y.

Koeficient zkreslení obrazu.

Směrodatná odchylka představující míru šumu v obraze.

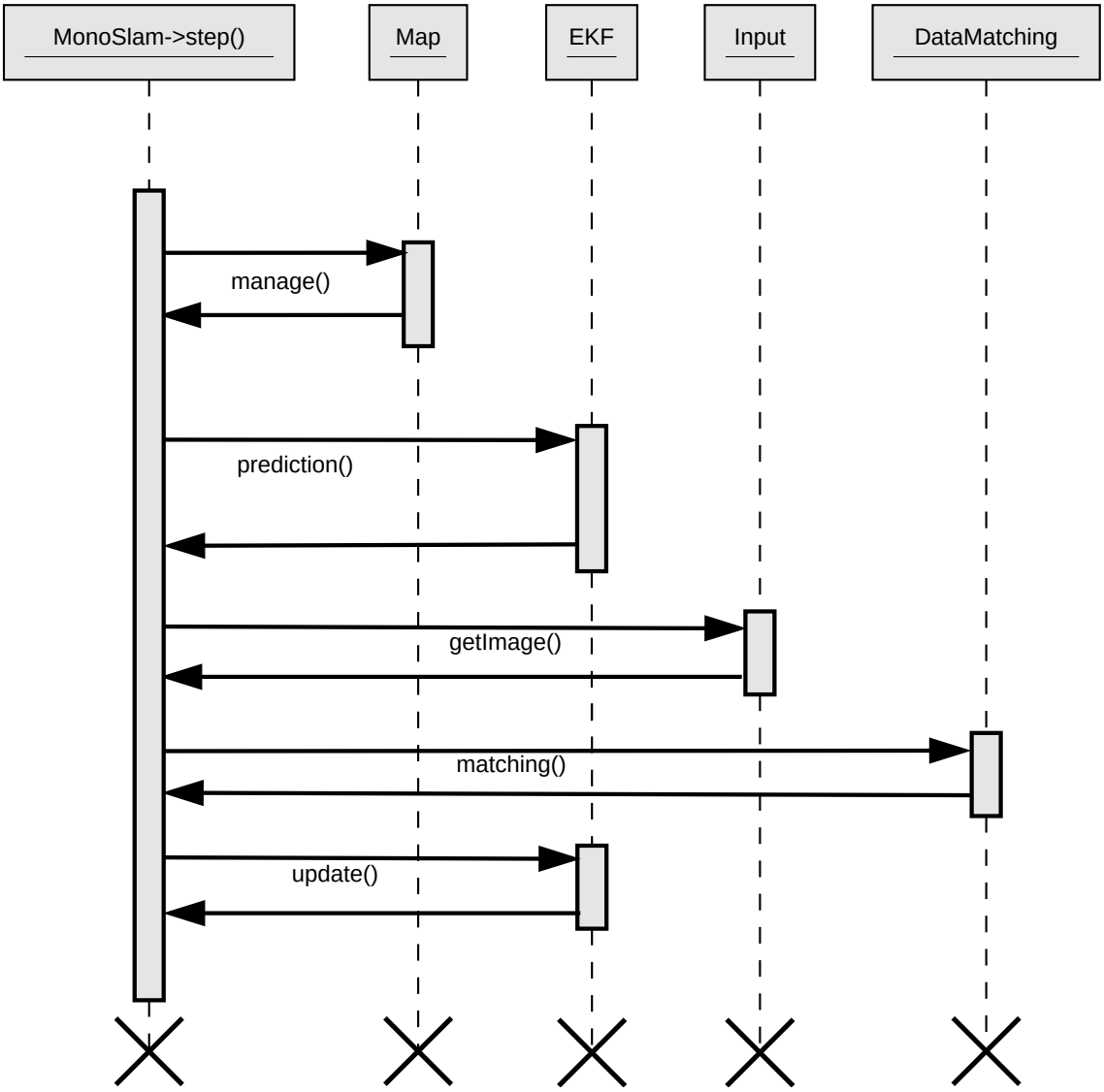
Mezi parametry SLAM úlohy patří časový krok **dt**, je ovšem vhodné vytvořit pro tyto parametry strukturu pro pozdější rozšiřování možností celé úlohy.

Hlavní objekt třídy `MonocularSLAM` bude popsán v další části. Jedná se o objekt, který v sobě uchovává všechny ostatní části úlohy a řídí pořadí jejich volání a vzájemnou interakci. Samotný objekt je však po inicializaci grafického rozhraní předáván metodě START, Která volá potřebné metody tohoto objektu. Zejména pravidelně spouští metodu `step()`.

4.7 Třída MonocularSLAM

Třída `MonocularSLAM` obsahuje metodu pro inicializaci a správu celé SLAM úlohy. Jako své členy obsahuje objekty všech ostatních tříd uvedených v diagramu na níže uvedeném obrázku 4.1.
Konstruktor má za úkol inicializovat vstupní zařízení dle předaného nastavení. Zavoláním metody `init()` se dále nastaví základní atributy jako stavový vektor **x** a kovarianční matice **P**. Metoda `step()` pak provádí jeden krok SLAM algoritmu. To znamená, že

postupně volá všechny systémy popsané v kapitole analýza. Fungování této metody je patrné ze sekvenčního diagramu na obrázku 4.2.



Obrázek 4.2: Sekvenční diagram metody step

Metoda step by mohla v konkrétním případě obsahovat mnohem více metod. Ve velké většině tyto nadbytečné metody budou pouze pomocné a v návrhu by mohly být zahrnuty do jedné z metod uvedených v diagramu.

4.8 Třída EKF

Rozšířený kalmanův filtr, neboli v řeči programovacího jazyka třída EKF obsahuje dvě základní metody korespondující s jednotlivými kroky kalmanova filtru. Konkrétně se jedná o metody:

prediction - metoda pro aplikaci časového kroku rozšířeného Kalmanova filtru.

update - metoda pro aplikaci filtračního kroku rozšířeného Kalmanova filtru.

Kromě těchto metod může třída EKF obsahovat ještě další soukromé metody sloužící k provedení jednotlivých částí predikce respektive filtrace. Například v případě monokulárního SLAMu je logické implementovat metodu pro odhad pohybu kamery dle vybraného pohybového modelu. V přiloženém kódu se jedná o pohybovém modelu pro volně pohyblivou kameru v 3D prostoru. Jedná se ovšem o neveřejné metody volané ze dvou výše uvedených metod.

4.9 Třída MapManagment

Velice rozsáhlým systémem je třída pro správu mapy. Tato třída se stará o všechny úkony spojené s mapou. Zejména pak o zařazování a vyřazování jednotlivých významných bodů, aktualizace informací o významných bodech a také o asociaci nových a již zařazených dat.

Zejména monokulární SLAM by bez precizní a velice složité správy mapy nemohl fungovat. To je způsobeno v první řadě absencí informace o třetí souřadnici významného bodu. Díky tomu je nutné rozlišovat body, u kterých je třetí souřadnice již známá a body, u kterých se teprve zjišťuje. To samozřejmě zahrnuje i metodu pro převod mezi reprezentacemi XYZ a inverzní hloubky významného bodu. Před uvedením struktury třídy pro správu mapy je nutné uvést třídu pro uchovávání významného bodu v mapě. Ta obsahuje následující vlastnosti významného bodu:

position - Pozice významného bodu.

rotation - Rotace významného bodu.

patch_wi - Okolí významného bodu ve vstupních datech při přidání bodu.

patch_wm - Okolí významného bodu ve vstupních datech při změření bodu.

type - typ reprezentace XYZ, ID

half_patch_size_wi - Velikost poloměru patch_wi.

half_patch_size_wm - Velikost poloměru patch_wm.

times_predicted - Počet predikcí.

times_measured - Počet pozorování.

y - Měření

Systém pro správu mapy pracuje s objekty třídy feature v podobě seznamu. Konkrétně v programovacím jazyku C++ je možné využít možností struktury List. Třída pak obsahuje následující metody:

`manage()` je metoda, která postupně provede kompletní správu mapy v daném časovém kroku.

`deleteFeatures()` vyřadí ze zpracování ty významné body, kteří nesplní podmínku na poměr predikovaných a skutečně pozorovaných bodů. Počet pozorování musí být větší než polovina počtu predikcí, jinak je bod vyřazen.

`initializeFeatures()` provádí inicializaci nových významných bodů ve chvíli, kdy je počet bodů v obraze menší než nastavený práh.

`addFeatures()` přidá inicializované významné body do vektoru stavu a kovarianční matice úlohy SLAM.

`inverseDepth2XYZ()` je metoda, která převádí významné body reprezentované pomocí inverzní hloubky do euklidovské reprezentace XYZ.

`updateFeaturesInfo()` aktualizuje informace o významných bodech v databázi a připravuje je na další časový krok.

Postup metody `manage()` je takový, že se nejdříve vymažou nepotřebné respektive nedostatečně výrazné významné body. Následně se aktualizují informace o jednotlivých významných bodech a provede se případné převedení některého z bodů na XYZ reprezentaci. Nakonec se v případě nutnosti načtou nové body tak, aby jejich počet splňoval nutné minimum.

Důležitá je zejména práce při přidávání a mazání významných bodů. Proto je vhodné tyto rozsáhlé metody rozložit do více specifičtějších. Konkrétně vytvořit speciální metodu pro přidání bodu v ID reprezentaci a speciální pro XYZ reprezentaci.

4.10 Třída Input - Systém pro práci se vstupními daty

Na diagramu uvedeném na obrázku 4.1 je uvedena třída Input. Tato třída může být chápána buď jako rozhraní, které se bude formou dědění konkretizovat pro určité zařízení a nebo je možné chápat ho jako třídu, která obsahuje metody pro práci s větším množstvím zdrojů dat.

V případě druhé možnosti je důležité při vytváření objektu třídy input specifikovat typ zařízení, se kterým bude třída pracovat. Toto nastavení je uloženo uvnitř třídy v podobě číselné proměnné. Díky tomu lze vytvořit jednotný přístup ke zdrojům dat z různých zdrojů. Vhodné dále je sjednotit také podobu vrácených dat. V programovacím jazyce C++ se dá s výhodou využít sada knihoven OpenCV, která umí přistupovat k usb kameře i k obrázkům uloženým v počítači, přičemž načtená data mají stejnou reprezentaci ve formě objektu třídy Mat.

4.11 Třída Input - Systém pro detekci významných bodů

S předchozím systémem je pevně spjat systém pro detekci významných bodů. Proto existují dvě možnosti jak tento systém implementovat. První způsob je implementace systému jako samostatné třídy obsahující jednotlivé metody pro detekci, případně speciální metodu pro výběr detekčního algoritmu.

Druhou možností je vřazení do systému pro práci se vstupními daty konkrétně do třídy Input. To znamená implementovat tyto metody přímo v rozhraní, respektive ve zděděných třídách. Seznam metod se shoduje se seznamem detekčních algoritmů uvedeným v kapitole 2. Názvy metod je možné volit například takto:

```
getHarris()

getShiTomasi()

getSift()

getSurf()

getFast()
```

Kromě zmíněných metod je vhodné vytvořit metodu, která bude výše uvedené algoritmy volat na základě parametrů. Tyto parametry je vhodné implementovat ve formě struktury, přičemž je možné ke každému algoritmu vložit jiné názvy nastavujících parametrů ve struktuře. Ty, které algoritmus nepotřebuje se pro daný typ ponechají nulové a ostatní se nastaví dle potřeby. Tento postup sice mírně zvýší paměťovou náročnost, jelikož je ovšem potřeba jen jednoho objektu této struktury, je zvýšení zanedbatelné. Výhodou pak je získání univerzálního přístupu k jednotlivým algoritmům. Toho je docíleno zejména parametrem určujícím typ metody, který bude fungovat podobně jako typ vstupního zařízení uvedený výše.

Použití detekčních metod je v C++ pomocí knihovny OpenCV velice jednoduché. Každá metoda má vytvořen svůj vlastní objekt detektoru, přes který je metoda přístupná. Ve všech případech stačí vytvořit příslušný objekt a zavolat jeho metodu `detect()` s konkrétními parametry pro danou úlohu. To významně ulehčuje tvorbu jednotného rozhraní pro systém detekce významných bodů.

4.12 Systém pro asociaci dat

Systém pro asociaci dat představovaný třídou DataMatching je velice specifická část celého problému SLAM. V celém algoritmu je vždy použita jen jedna metoda pro asociaci. A na základě výběru tohoto algoritmu se implementuje systém pro asociaci dat. V kapitole 2.4.5 již bylo uvedeno, že se může jednat například o **metodu nejbližšího souseda**, nebo metodu **RANSAC**. Použití každé metody je speciální, neboť vyžaduje silnou vazbu

na systém pro správu mapy. Nejvhodnějším přístupem je předat systému kompletní objekt mapy. Algoritmus pro asociaci dat má pak k dispozici všechny informace potřebné k rozhodování o validnosti nově načtených dat.

4.13 Systém pro vizualizaci dat

Jedná se o další systém, který nelze výrazně zobecnit. Záleží zde zejména na výběru formy vizualizace, jejichž seznam byl uveden v kapitole analýza. Při použití programovacího jazyka C++ je vhodné použít některou z dostupných přenositelných knihoven pro různé druhy vizualizací. Vhodnou volbou je například OpenGL pro práci s 2D i 3D grafikou. Pokud se programátor rozhodne použít pouze 2D grafiku, pak je možné využít jednodušší knihovnu SDL[1].

4.14 Ukázka CMake souboru

Na začátku této kapitoly byla uvedena aplikace CMake, která je vhodná pro vytvoření projektových souborů v jazyce C++. Aplikace je schopna vytvořit projekt pro většinu používaných vývojářských prostředí. Ve Windows je nejlepší volbou některá z verzí nástrojů Visual studio. Podmínkou však je, že pro danou verzi musí být zkompilovány všechny knihovny použité ve vyvíjené aplikaci. Příklad soubor `CMakeLists.txt`, do kterého se zapisuje kód, ze kterého jsou následně vytvořeny projektové soubory, je uveden v následující ukázce:

Algoritmus 1: Ukázka souboru `CMakeLists.txt`

```
PROJECT( monocular_slam )
  cmake_minimum_required(VERSION 2.6)
  SET(CMAKE_MODULE_PATH${CMAKE_MODULE_PATH}
      "${ PROJECT_SOURCE_DIR}/CMakeModules/")
  ADD_EXECUTABLE( slam input.h library.h library.cpp)
TARGET_LINK_LIBRARIES( slam ${LIBS})
```

4.15 Ukázková aplikace založená na popsaném návrhu

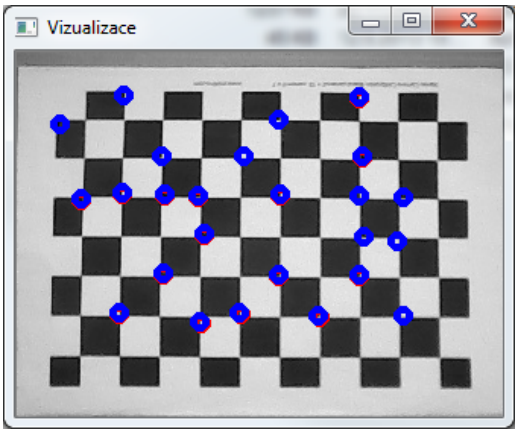
V rámci diplomové práce byla vytvořena aplikace pro monokulární SLAM splňující výše uvedené požadavky na návrh a implementaci. Program využívá jako zdroj dat sérii souborů s obrazovými daty. Na tyto obrazová data aplikuje metodu pro detekci významných bodů, které dále zařazuje do EKF-SLAMu. Výstupem aplikace je promítnutí predikcí a

[1] Simple Directmedia Layer : http://www.libsdl.org/

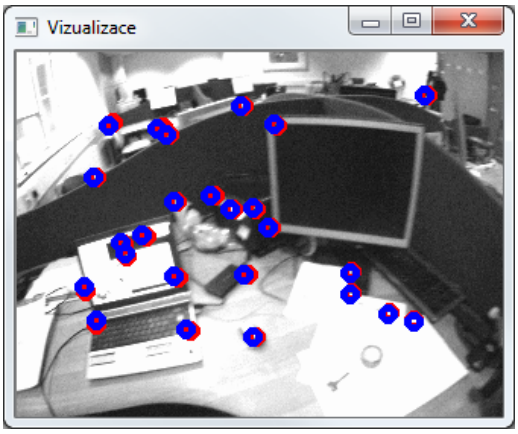
měření zpět do zdrojových dat viz obráky 4.3 až 4.6. Aplikace se spouští s jedním parametrem udávajícím cestu k adresáři, který obsahuje sérii zdrojových souborů.

Např.: `slam.exe ./images`

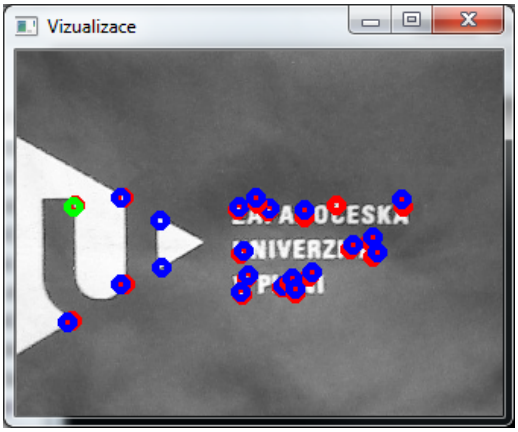
Na výstup jsou vykreslovány body tří barev. Červené jsou body změřené v daném kroku, modré představují predikci náležící k měření a zelené představující dodatečnou predikci, která je provedena ve chvíli, kdy je klasická predikce zamítnuta a bod přesto splňuje určitá kritéria. Podrobnosti lze najít v přiložených zdrojových kódech návrhu algoritmu SLAM.



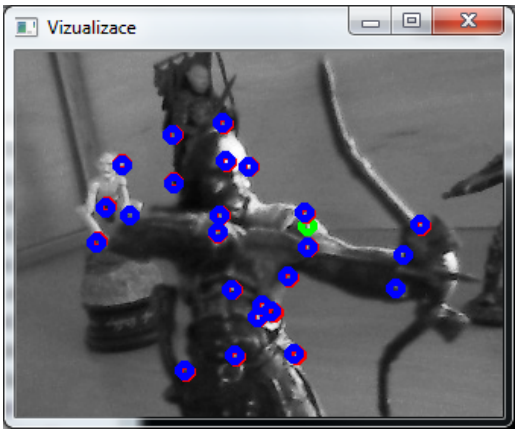
Obrázek 4.3: Výstup série s šachovnicí.



Obrázek 4.4: Výstup série z článku [17].



Obrázek 4.5: Výstup série se znakem ZČU.

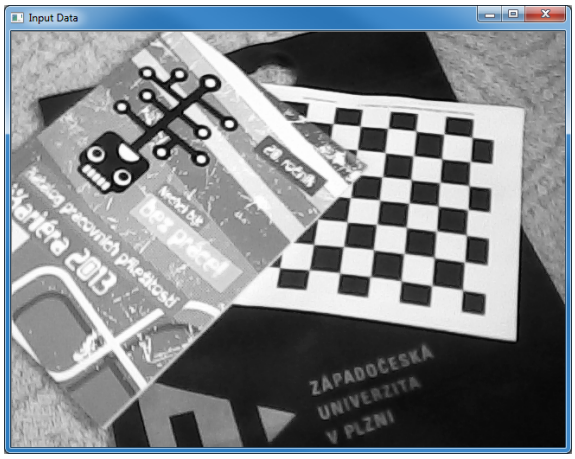


Obrázek 4.6: Výstup série s figurkou.

Kapitola 5

Testování detekčních metod

V této kapitole budou uvedeny výsledky testování detekčních metod. Nejdříve budou představeny a okomentovány výsledky dosažené při testu opakovatelnosti detekčních metod. Následně bude text zaměřen na rychlost jednotlivých metod. V závěru kapitoly budou výsledky sloučeny a bude vybrána nejvhodnější metoda pro nasazení v aplikaci pro monokulární slam. Při všech testech byla použita scéna uvedená na obrázku 5.1

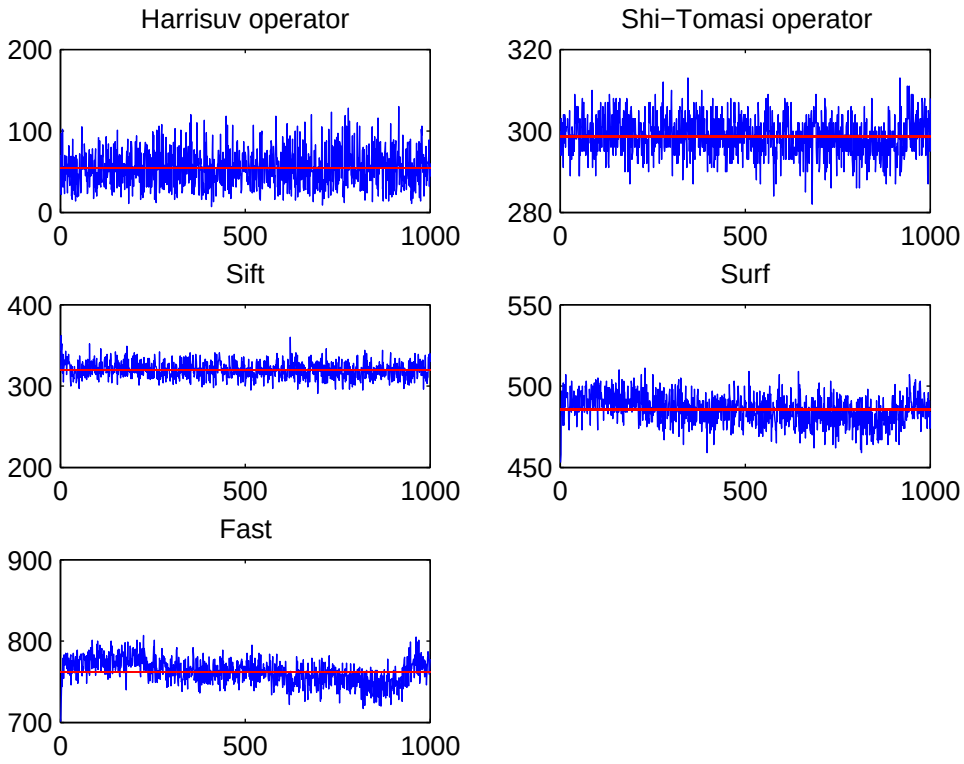


Obrázek 5.1: Scéna použitá při testování

5.1 Testování opakovatelnosti - statická scéna

Statickou scénou je v tomto případě myšlena neměnná scéna snímaná kamerou. Dosažené výsledky tak do značné míry záleží na kvalitě kamery. Čím lepší kamera, tím lepší výsledky budou jednotlivé metody mít. V tomto případě je použita levná usb kamera Kanyon, která byla uvedena na obrázku 2.7.
Při měření bylo pevně postavenou kamerou získáno 1000 snímků statické scény. Na každý snímek bylo aplikováno pět metod pro detekci významných bodů s využitím základního

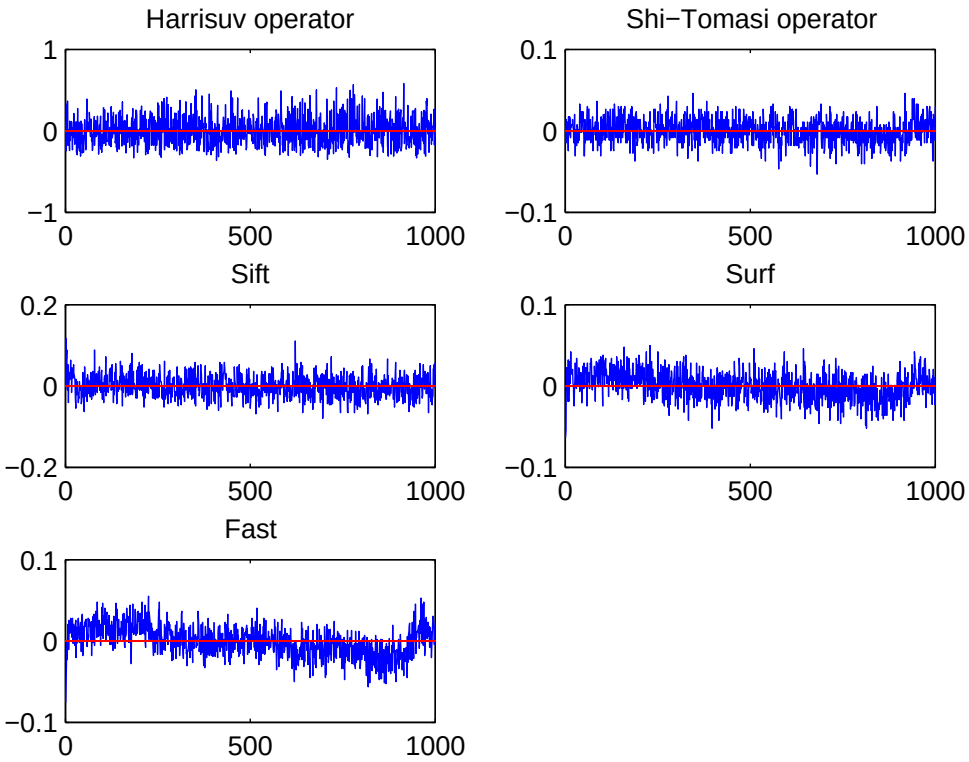
nastavení používaném v knihovnách `OpenCV`. Ze získaných informací byl pro další zpracování důležitý jen údaj o počtu zachycených bodů v daném časovém kroku. Na obrázku 5.2 jsou zobrazena data získaná při testování.



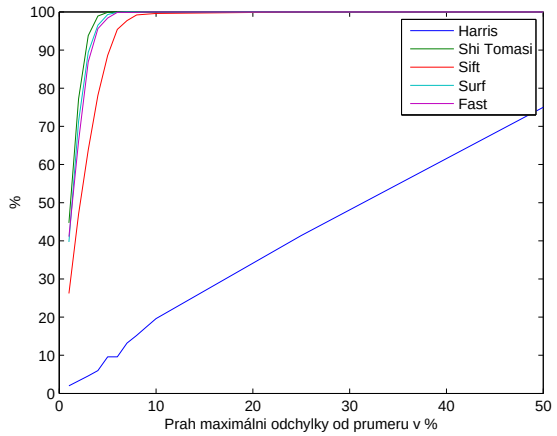
Obrázek 5.2: Data získaná při testování opakovatelnosti

Z dat je patrné, že každá metoda nalézá jiný počet bodů. Proto byla data pro lepší porovnání metod znormována viz obrázek 5.3. Grafy vypadají velice podobně, pouze se změnilo jejich měřítko. Navíc je v grafech namísto informace o počtu bodů zobrazena odchylka od střední hodnoty.

V grafu na obrázku 5.4 je dále uvedena informace o procentu měření vychýlených od průměru o méně než 1 až 50. Čím rychleji křívka roste, tím dává metoda konzistentnější výsledky. Favority v tomto případě byly metody Shi-Tomasi, Surf a Fast. Metoda Sift má mírně slabší výsledky a metoda Harrisova operátoru významně zaostává.



Obrázek 5.3: Normovaná data získaná při testování opakovatelnosti

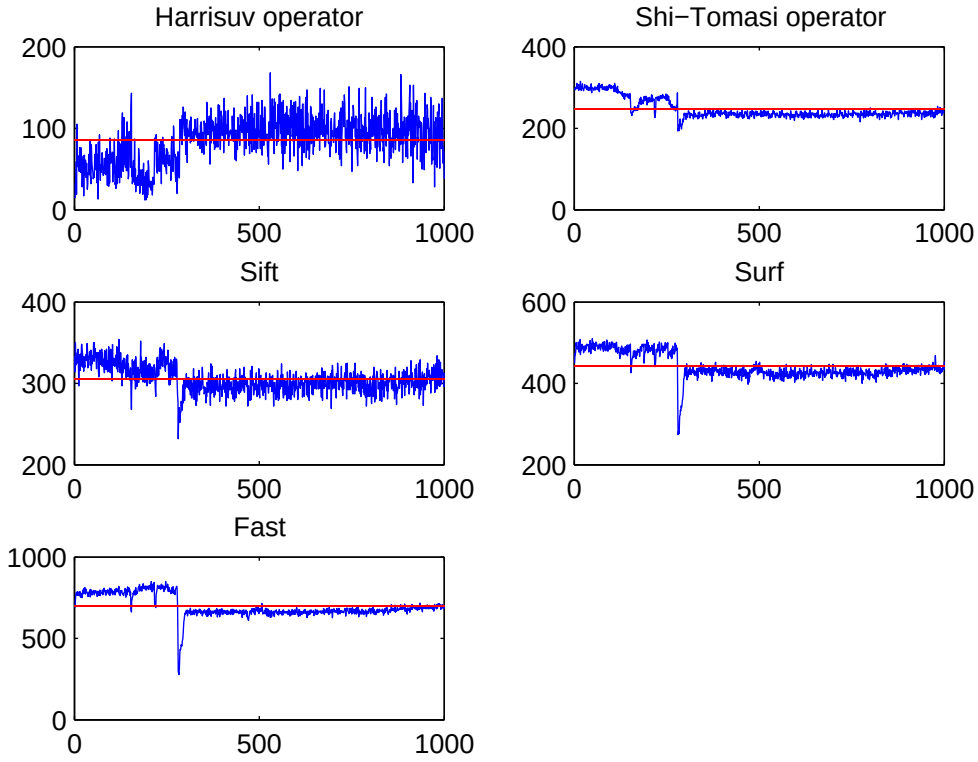


Obrázek 5.4: Závislost odchylky od průměru o méně než určitý práh v procentech

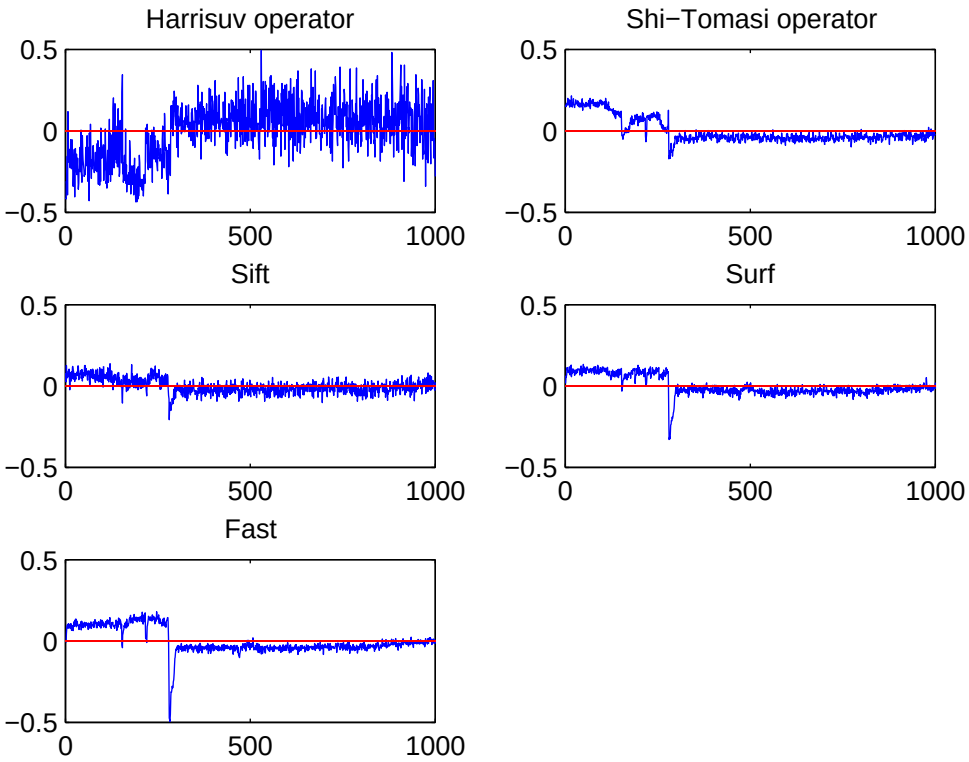
5.2 Testování opakovatelnosti - statická scéna s chybou

Dále byla testována scéna, ve které došlo v určitém momentu k jednorázové změně. V tomto případě se jednalo o změnu osvětlení scény. Získaná data jsou opět uvedena v základní a normované podobě na obrázcích 5.5 a 5.6. Kolem snímku s číslem 300 je v datech vidět výrazný výkyv počtu detekovaných bodů. Ten je způsoben přizpůsobováním optiky kamery momentálnímu osvětlení scény. Po vyrovnání tohoto výkyvu se počty detekovaných bodů opět ustalují, ovšem je patrné, že se hodnoty v změnily. Kromě metody Harrisova operátoru došlo ke snížení počtu detekovaných bodů.

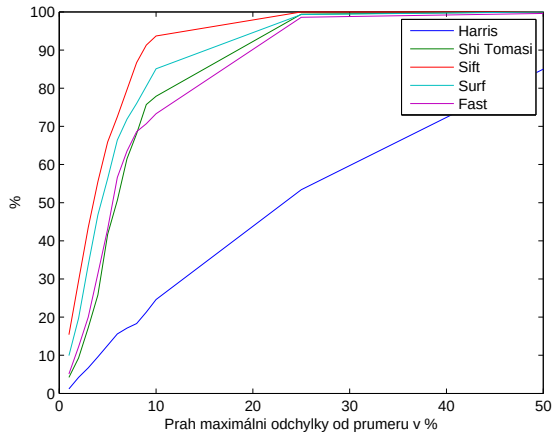
Na obrázku 5.7 je opět zobrazen procentuální graf snímků s počtem bodů odchýlených o méně než určitý práh lomený všemi snímky. Pořadí metod je podobné jako v případě statické scény. Pouze metoda SIFT se zde dostala na nejlepší pozici, jelikož nejrychleji roste ke 100%. Projevila se tedy jako nejodolnější vůči změně ve statické scéně.



Obrázek 5.5: Data získaná při testování opakovatelnosti



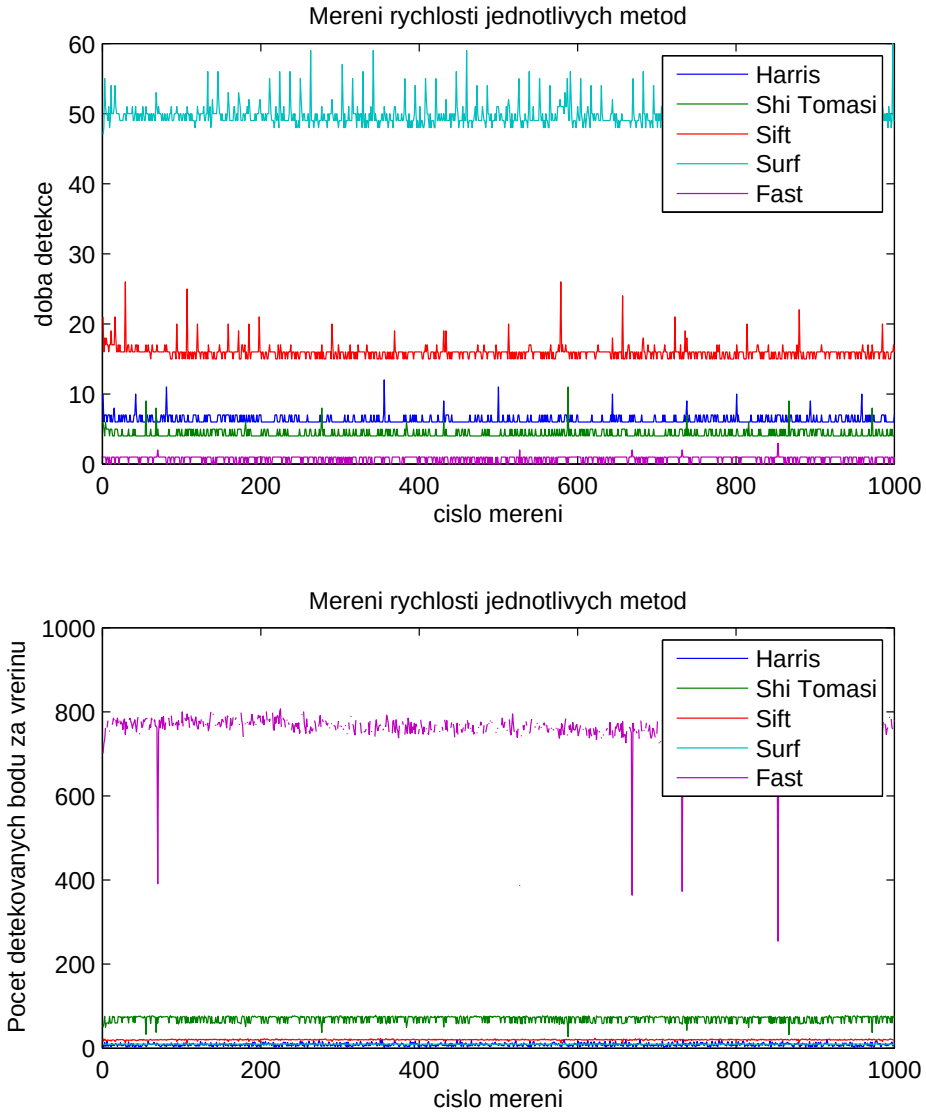
Obrázek 5.6: Normovaná data získaná při testování opakovatelnosti



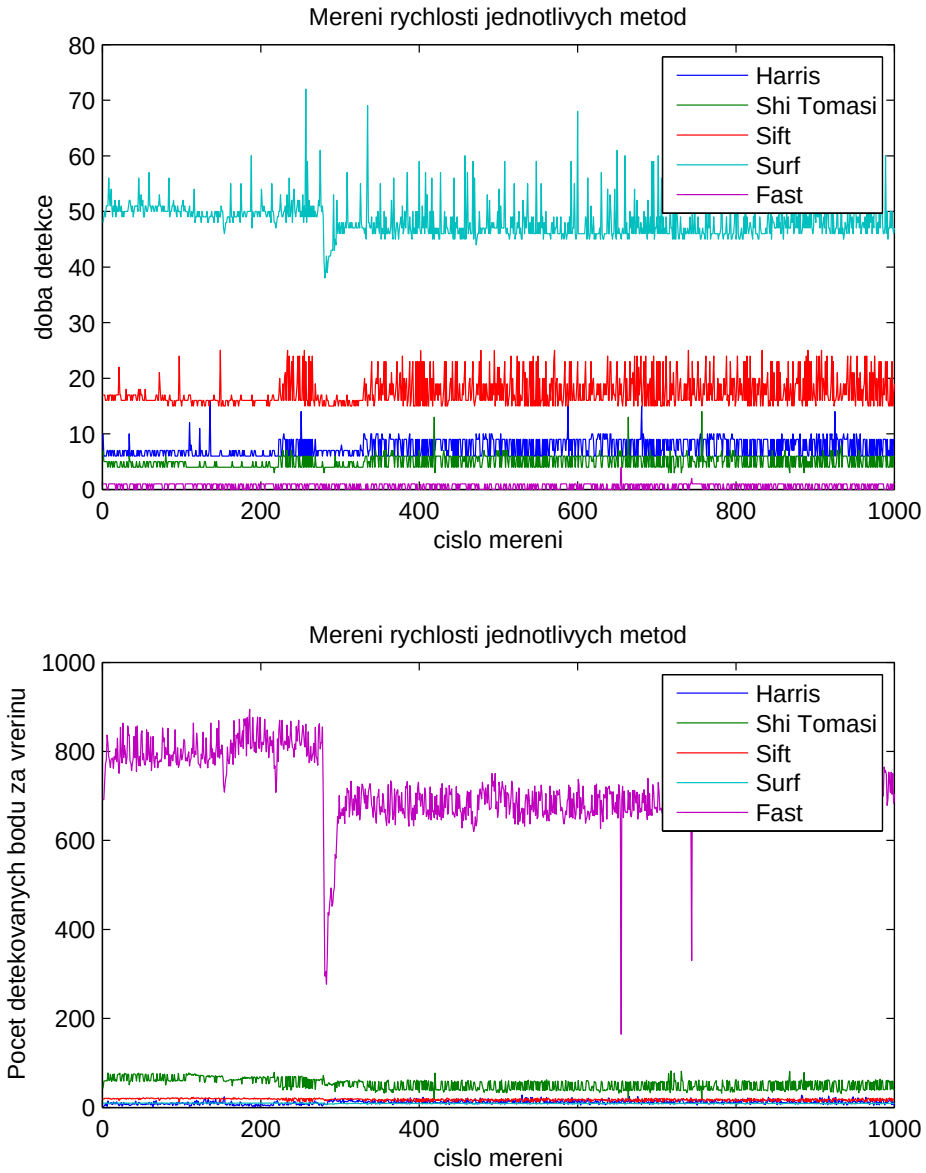
Obrázek 5.7: Závislost odchylky od průměru o méně než určitý práh v procentech

5.3 Testování rychlosti detekčních metod

Při testování byla zaznamenávána i doba běhu jednotlivých metod. Výsledky byly pro oba testy zaznamenány vždy do dvou grafů. První z nich obsahuje informaci o době běhu v milisekundách a druhý o počtu bodů, které by za dané rychlosti funkce zpracovala za jednu vteřinu. Na obrázku 5.8 jsou uvedeny grafy pro statickou scénu a na obrázku 5.9 pro scénu s chybou.



Obrázek 5.8: Grafy rychlosti metod pro statickou scénu



Obrázek 5.9: Grafy rychlosti metod pro scénu s chybou

Zajímavým údajem je také informace o počtu milisekund potřebných na detekci jednoho významného bodu.

	Harrisův operátor	Shi-Tomasi	Sift	Surf	Fast
Statická scéna	0,1162 ms	0,0145 ms	0,0498 ms	0,1083 ms	0,0010 ms
Scéna s chybou	0,0888 ms	0,0201 ms	0,0576 ms	0,1096	0,0009 ms

Tato informace lze také převést na počet bodů, které je metoda schopna zpracovat za jednu sekundu.

	Harrisův operátor	Shi-Tomasi	Sift	Surf	Fast
Statická scéna	8	69	20	9	999
Scéna s chybou	11	49	17	9	1119

5.4 Shrnutí výsledků

Z uvedených informací vyplývá, že nejvhodnější metodou k detekci bodů v monokulárním SLAMu je metoda FAST. Dle získaných výsledků se jedná o metodu schopnou v krátkém čase zpracovat nesrovnatelně větší množství bodů, než ostatní metody. Navíc za ostatními nezaostává ani z pohledu opakovatelnosti. Naopak nejhorší metodou je v tomto případě Harrisův operátor, který nevykazoval dobré výsledky v žádném testu.

Metody SIFT a SURF měly konzistentní výstupy, ovšem měly vyšší časovou náročnost. V dnešní době se tato náročnost dá odstranit použitím paralelních výpočtů na grafických kartách. V takovém případě by pak tyto metody nejspíše vykazovaly lepší výsledky než metoda FAST. V praktické části ovšem není tato možnost využita.

Kapitola 6

Závěr

Návrh metod pro vizuální simultánní lokalizaci a mapování vyžaduje znalosti z mnoha oblastí jakou jsou teorie odhadu, počítačová grafika či zpracování digitalizovaného obrazu. Zejména pak poslední jmenovaná je důležitá, neboť vstupem celé úlohy vizuálního SLAMu jsou obrazová data.

Impulzem k zadání této práce byl zájem osvojit si některé metody a algoritmy z oblasti teorie obrazu a za současného využití znalostí z oblasti zpracování digitalizovaného obrazu. Cílem práce pak bylo zejména shrnutí dostupných materiálů v ucelený text a navržení metod, které by byly schopny řešit úlohu vizuálního SLAMu, které by se mohly stát základem pro další výzkum v této oblasti.

V práci byla konkrétně vybrána verze monokulárního SLAMu využívající k získávání dat jedné kamery. Bylo též uvedeno, že s tímto výběrem souvisí řada problémů spočívajících v absenci informace o vzdálenosti jednotlivých objektů ve sledované scéně. Tato informace lze však zpětně získat analýzou pohybu kamery kolem sledované scény.

Práce se také zabývala detekcí významných bodů, které byly následně zařazeny do výpočetního algoritmu. Metody byly testovány z pohledu rychlosti a opakovatelnosti. Je zřejmé, že první zmíněná vlastnost souvisí nejen s metodou, ale také výkonem počítače, na kterém je metoda spuštěna, ovšem je třeba podotknout, že poměr rychlostí by měl být na všech strojích stejný, nebo velice podobný. Stejně tak je druhá vlastnost metod je spjata s kvalitou kamery, která data snímá. Čím kvalitnější senzor má kamera k dispozici, tím méně šumu by v obrazu mělo být a tím lepší výsledky by měly být získány. Porovnání jednotlivých metod co do pořadí výsledků bude opět stejné, nebo velice podobné i pro jiné kamery, než pro kameru použitou v příslušné části této práce. Jako nejlepší metoda pro využití v úloze vizuálního SLAMu se jeví metoda FAST, která vykázala velice kvalitní výsledky v testech opakovatelnosti a bezkonkurenční výsledek v rychlosti, kde o mnoho předčila ostatní metody.

V rámci praktické části této práce byl navržen algoritmus pro řešení monokulárního SLAMu, který vycházel z prací [5] a [17]. K práci je přiloženo několik sekvencí snímků, na kterých lze aplikaci vyzkoušet. Kromě přeložené aplikace jsou k dispozici také zdrojové kódy, které mohou být použity jako kostra pro jinou aplikaci s využitím jiných metod pro detekci významných bodů či asociaci nově načtených dat.

Jediným nedostatkem návrhu je prozatím jeho výpočetní náročnost, která zabírá mnoho času a proto zatím není možné nasadit aplikaci do prostředí, kde je potřeba skutečně běh v reálném čase. Tím se ovšem otvírají možnosti pro další rozšíření navržených metod a algoritmů. Například využití grafických karet pro urychlení výpočtů, využití dalších zdrojů obrazových dat, z nichž některé byly v práci zmíněny. Další zajímavou cestou výzkumu simultánní lokalizace a mapování je hledání nových možností v návrhu pohybových a pozorovacích modelů a případné připojení mobilního robota, který by do aplikace načítal užitečné odometrické informace, které v úloze s volnou kamerou chybí. Nejen těmito směry je potřeba metody simultánní lokalizace a mapování rozvíjet, neboť cesta k pomyslnému svátému grál robotiky, tedy plně autonomnímu mobilnímu robotu, je ještě dlouhá a klikatá.

Literatura

[1] DURRANT-WHYTE H, BAILEY T.: *Simultaneous Localization and Mapping: Part I The Essential Algorithms.* , Robotics and Automation Magazine, June 2006.

[2] DURRANT-WHYTE H, BAILEY T.: *Simultaneous Localization and Mapping: Part I State of art.* , Robotics and Automation Magazine, June 2006.

[3] RIISGAARD S, BLAS R, M.: *SLAM for Dummies : A Tutorial Approach to Simultaneous Localization and Mapping* , 2003.

[4] CHEESEMAN R, SMITH C R .: *On the Representation and Estimation of Spatial Uncertainty* , The international Journal of Robotics Research, MIT, 1986.

[5] DAVISON A. J, REID I. D, MOLTON N. D, STASSE O .: *MonoSLAM : Real-Time Single Camera SLAM* , IEEE, London, Imperial College, 2007.

[6] SCARAMUZZA D, FRAUNDORFER F.: *Visual Odometry Part I : The First 30 Years and Fundamentals* , IEEE, 2011.

[7] SCARAMUZZA D, FRAUNDORFER F.: *Visual Odometry Part II : Matching, Robustness, Optimization, and Applications* , IEEE, 2012.

[8] SOLÀ J, MONIN A, DEVY M, LEMAIRE T .: *Undelayed Initialization in Bearing Only SLAM* , LAAS-CNRS, Tolouse, France.

[9] CIVERA J, DAVISON A. J., MONTIEL J. M.: *Inverse Depth Parametrization for Monocular SLAM* , IEEE, 2008.

[10] CIVERA J, DAVISON A. J., MONTIEL J. M.: *Unified Inverse Depth Parametrization for Monocular SLAM* , IEEE, 2007.

[11] HARRIS Ch, STEPHENS M.: *A Combined Corner and Edge Detector*, UK,1988.

[12] SHI J, TOMASI C.: *Good Features to Track*, IEEE,1994.

[13] LOWE D. G.: *Distinctive image features from scale-invariant keypoints*, International journal of computer vision,2004.

[14] BAY H, ESS A, TUYTELAARS T, GOOL L. V.: *Speeded-Up Robust Features (SURF)*, Elsevier,2007.

[15] ROSTEN E, DRUMMOND T.: *Fusing Points and Lines for High Performance Tracking*, Department of Engineering, University of Cambridge, 2005.

[16] PSUTKA J.: *Materiály k předmětu Učící se systémy a klasifikátory, KKY/USK*.

[17] CIVERA J, GRASA O. G, DAVISON A. J., MONTIEL J. M.: *1-Point RANSAC for EKF-Based Structure from Motion*, 2010.

[18] MATTMANN P.: *Visual SLAM in pieces*, Curych,2009.

[19] ŠONKA M, HLAVAC V, BOYLE R.: *Image Processing, Analysis, and Machine Vision, 3rdedition* , Toronto, Thomson Learning, April 2007, 821 p, ISBN 049508252X (2nd edition Brooks/Cole, Pacific Grove, CA, 1999, 1st edition Chapman & Hall, London 1993).

[20] DOUCET A.: *On sequential simulation-based methods for Bayesian filtering*, Cambridge Univ., Tech. Rep.,1998.

[21] *Wikipedie, otevřená encyklopedie* [online] *http://cs.wikipedia.org/*

[22] *FootSLAM and PlaceSLAM* [online] *http://www.kn-s.dlr.de/indoornav/footslam_video.html*

Dodatky

Dostupný software řešící úlohu SLAM

V této části bude uveden krátký seznam ukázkových aplikací, které je možné nalézt na internetu. Mezi dostupným softwarem jsou aplikace simulující základní algoritmy, které lze využít pro řešení úlohy simultánní lokalizace a mapování. Zároveň lze nalézt i aplikace řešící kompletní úlohu visuzálního SLAMu. V následující tabulce je uveden seznam některých aplikací :

Název aplikace	**Autor**	**Popis**
Simulační skripty v Matlabu	Tim Bailey	Skripty simulující několik druhů algoritmu SLAM. Přesněji EKF, Fast a UKF SLAM. Použito prostředí Matlab
Zdroj : http://www-personal.acfr.usyd.edu.au/tbailey/software/		
Výukové skripty	Joaquim Salvi	Krátké ukázkové skripty pro 1D, 2D a 3D SLAM vytvořené v Matlabu
Zdroj : http://eia.udg.es/ qsalvi/recerca.html		
SceneLib 1	Andrew Davison	Aplikace v C++ řešící úlohu monokulárního SLAMu dle článku [5].
Zdroj : http://www.doc.ic.ac.uk/∼ajd/software.html		
SceneLib 2	Hanme Kim	Přepis předchozí aplikace za použití novějších knihoven. Využití USB namísto FireWire kamery.
Zdroj : https://github.com/hanmekim/SceneLib2		
1 Point Ransac MonoSlam	Javier Civera	Aplikace monokulárního SLAMu vytvořená v prostředí Matlab. Využívá techniky Inverzní hloubky a metodu ransac.
Zdroj : http://webdiis.unizar.es/ jcivera/code/1p-ransac-ekf-monoslam.html		

Obsah přiloženého CD

Na přiloženém CD jsou k dipozici všechny materiály spojené s prací. Od zdrojových dat, přes aplikace a zdrojové kódy až po elektronickou verzi dilpomové práce práce ve formátu PDF. Pro přehlednost zde následuje stromová struktura obsahu CD :

- Elektronická verze DP
  - diplomova_prace.pdf
- Zdrojová data
  - Jednotlivé nasnímané série obrázků
  - Část série z článku [17].
- Experimenty
  - Aplikace pro testování detekčních metod
  - Soubory s nasnímanými daty
- Zdrojové kódy aplikací
  - Návrh SLAM algoritmu
  - Ukázkový soubor ukazující možné propojení s knihovnou OpenGL pro vizualizaci dat.
- Zkompilované aplikace