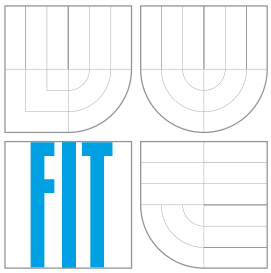
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LOKALIZACE ROBOTA POMOCÍ KAMERY

ROBOT LOCALIZATION USING CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR HEŘMAN
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Cílem této práce je návrh jednoduché lokalizační metody a její implementace pro robotický operační systém ROS. Tato metoda využívá monokulární kameru jako jediný senzor a odhaduje pozici v předem známé mapě. V rámci experimentů s prototypem jsou vyzkoušeny klíčové body typu SURF, SIFT a ORB.

Abstract

The objective of this work is to design a simple localization method and its implementation in robot operating system ROS. This method uses a monocular camera as the only sensor and estimates the position in a known map. In experiments with prototypes are tested key points of type SURF, SIFT and ORB.

Klíčová slova

lokalizace ve známém prostředí, odhad pozice, monokulární kamera, srovnání kamer, robotický operační systém, ROS, klíčové body, SIFT, SURF, ORB, FLANN

Keywords

localization in known environment, pose estimation, monocular camera, cameras comparison, robot operating system, ROS, key points, SIFT, SURF, ORB, FLANN

Citace

Petr Heřman: Lokalizace robota pomocí kamery, bakalářská práce, Brno, FIT VUT v Brně, 2013

Lokalizace robota pomocí kamery

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Petr Heřman
13. května 2013

Poděkování

Děkuji mému vedoucímu práce za neskutečnou trpělivost a motivaci, kterou mi poskytl při každé konzultaci.
Děkuji mé rodině, že mě podporovala během celého mého studia.

Obsah

Kapitola 1

Úvod

Mobilní robotika je moderní vědní odvětví, kombinující několik oborů, jako jsou počítačové vidění, řízení, plánování, modelování, umělá inteligence a další. Rychlost vývoje se zvyšuje spolu s růstem výkonu počítačů, miniaturizací součástek a jejich zlevňováním. Není tedy divu, že se roboti[1] stále častěji stávají součástí našich životů.

Jednou ze základních úloh robotiky je lokalizace mobilních robotů. Otázka „Kde jsem?“ je stejně důležitá pro roboty jako pro lidi a bez odpovědi na ni často nelze řešit další úlohy. Lidé pro určení své pozice nejčastěji využívají zrak, zatímco dnešní roboti častěji využívají různých typů dálkoměrů a odometrů. Zrak robotů, jež představují kamery, používají spíše pro zpřesnění lokalizace. Přitom kamery jsou proti dálkoměrům několikanásobně levnější a lze je použít i při dalších činnostech robota.

Tato práce se snaží dokázat, že mobilní robot se může lokalizovat i bez matematicky a fyzikálně přesných senzorů jako jsou dálkoměry. Vychází z předpokladu, že ne vždy je třeba znát svou pozici s geometrickou přesností, ale že stačí vědět, v které části mapy se nachází.

Pro usnadnění vývoje robotických aplikací existuje několik knihoven, frameworků a meta-operačních systémů. Ty si berou za cíl urychlit implementaci aplikací a usnadnit jejich začlenění do existujících robotických systému. Jedním z nich je i meta-operační systém ROS, který bude použit pro implementaci lokalizační metody. Využitelnost této práce díky němu nezůstane u jednoho konrétního typu mobilního robota.

1.1 Struktura práce

První dvě kapitoly této práce jsou zaměřeny na teorii. Kapitola 2 se zabývá tématy robotiky, jako lokalizace (sekce 2.1), simultální lokalizace a mapování (sekce 2.2), senzory (sekce 2.3) a ROS (sekce 2.4). V sekci o senzorech jsou detailně rozebrány kamery.

V kapitole věnované zpracování obrazu (viz 3) je vysvětlena problematika obrazových příznaků (sekce 3.1 a 3.4) a jejich porovnávání (sekce 3.5).

O specifikaci požadavků řešení a jeho návrhu pojednává kapitola 4. Hned v první sekci této kapitoly (tj. 4.1) je uveden přehled již existujících řešení, které posloužili jako inspirace pro návrh a implementaci.

Implementace prototypu a experimenty s ním jsou shrnuty v předposlední kapitole 5.

[1] Slovo robot, dnes přejaté do většiny jazyků na světě, použil poprvé Karel Čapek ve hře R.U.R.V češtině se dnes používá v životné i neživotné variantě, zpravidla podle míry autonomie nebo humanoidní podoby robota.

Kapitola 2

Teorie - robotika

2.1 Lokalizace

Lokalizace robota je proces zjišťování pozice v prostoru/mapě z dostupných senzorických dat a je důležitým předpokladem pro správné určení dalších kroků spojených nejen s řízením robota.

Jedná se tedy o problém, který je velice obecný, a je možné ho rozdělit do několika kategorií. Dle Thruna [23] lze problém kategorizovat na základě znalostí, které získáváme při startu lokalizace a během jejího běhu:

- **Sledování pozice (ang. position tracking)** - předpokladem této lokalizační techniky je znalost počáteční pozice robota. Následující pozice jsou odvozovány od pohybu, který robot provádí. Pro použití této metody je důležité, aby byla chyba senzorů, které měří pohyb, co nejmenší. Chyba se totiž s časem zvětšuje a přesnost lokalizace klesá.

  Neřešitelným problémem této metody je výpadek systému (i krátký), při kterém robot ztratí pojem o pohybu v čase výpadku a nedokáže již navázat lokalizační proces.

- **Globální lokalizace** - při této lokalizaci vycházíme z toho, že počáteční pozice v mapě je neznámá a při určování pozice je vhodné počítat z více možnými pozicemi [17]. Globální lokalizace se obvykle dokáže lépe vypořádat z chybami senzorických měření.

- **Problém uneseného robota (ang. kidnapped robot problem)** - řeší případ, kdy je robot během provozu náhodně přenesen na novou pozici (unesen) a tuto změnu nemohl registrovat. K tomuto problému může dojít při rušení signálu, nárazu s následkem náhlé změny pozice, příčném posunu diferenčně řízeného[1] robota, jehož pozice je zjišťována z inkrementálních enkodérů.

  Problém uneseného robota bývá často řešen současně s problémem globální lokalizace, respektive globální lokalizace je speciálním případem tohoto problému, kdy robotu dáme vědět, že byl unesen.

Jak uvádí Skalka [22], lze také lokalizaci kategorizovat dle možnosti lokalizačního systému cíleně ovlivňovat řízení robota:

- **Pasivní lokalizace** - pozice robota je odhadována na základě proudu senzorických dat, a robot nemůže nikterak zasahovat do ovládání (např. určovat směr jízdy, otočení).

[1] Diferenciální podvozek se sestává z dvou nezávisle řízených kol. Změna směru jízdy je dosažena různou rychlostí otačení kol. Takovýto podvozek se nedokáže pohybovat do boku.

- **Aktivní lokalizace** - lokalizační systém má možnost ovlivnit řízení robota z důvodu zpřesnění odhadu pozice. Může tedy například otočit kamerou, popojet, změnit nastavení senzorů (např. clony, filtry) a jiné. Aktivní lokalizace je častěji využívána u plně autonomních robotů, jelikož zde je přesnost pozice kruciální.

2.1.1 Statická a dynamická prostředí

Podstatný vliv na složitost lokalizace má prostředí, ve kterém se robot lokalizuje. Statická (neměnná) prostředí jsou při každé návštěvě robota stejná a odhad pozice v nich je snazší.

Dynamická prostředí se mohou měnit v čase. Změny mohou probíhat přímo během lokalizace nebo v čase, kdy se robot nenachází v prostředí. Při takovéto lokalizaci jsou změny považovány za šum senzorů. Jsou dva principy řešení: Buď si robot vytváří vektor prostředí v čase a je schopen pracovat s více stavy jednoho místa a nebo se snaží filtrovat změny a vycházet ze statických částí prostředí.

2.1.2 Zaměření práce

Tato práce se zabývá pasivní globální lokalizací v dynamickém prostředí.

2.2 SLAM

Simultální lokalizace a mapování (SLAM) je technika používaná roboty, kteří neznají své prostředí a jejichž cílem je navigace v tomto prostředí. Během navigace a pohybu v tomto prostředí si vytvářejí mapu a lokalizují se v ní [21].

Mapování a lokalizace v mapě jsou dva problémy, které je nutné řešit současně. Před tím, než robot může odpovědět na otázku „Kde jsem?“, musí nejprve udělat několik pozorování okolí, přičemž u každého takového pozorování musí znát pozici, ze které bylo provedeno. Jedná se o oblíbenou filozofickou hříčku „Byla dřív slepice nebo vejce?“ [24]

SLAM je tedy spíše princip/technika, než konkrétní algoritmus (ty se často liší v přístupech k výše zmíněným problémům).

2.2.1 Mapy

Mapování je proces tvorby modelu světa - mapy. Mapa je hlavním orientačním prvkem robota a je důležitá pro jeho pohyb a plánování. Tvorba mapy je závislá na použitých senzorech a úrovni abstrakce, která je vyžadována.

Nejrozšířenější typy map [3]:

- **Geometrické** - k popisu prostředí jsou použita geometrická primitiva (nejčastěji úsečky a křivky druhého řádu), která jsou propojována do objektů a ty je možné dále spojovat. Je složité tvorbu geometrické mapy ze senzorických dat automatizovat, jelikož jsou data zatížena chybou.

- **Topologické** - Bývají nadstavbou map geometrických. Abstrahují reprezentaci prostředí a často nesou další informace spojené s místy a cestami. Topologické mapy jsou často ve formě grafu což je vhodné pro aplikaci algoritmů. Při lokalizaci v těchto mapách není cílem metrická přesnost, ale povědomí robota o místě a jeho možnostech a vlastnostech.

- **Senzorické** - obsahují data přímo ze senzorů nebo jen částečně upravená. Typickým zástupcem takovéto mapy je mřížka obsazenosti, která dělí prostor do buněk. Každá buňka si drží informaci o své obsazenosti (jako pravděpodobnostní hodnotu) a má svou pozici v soustavě souřadné.

  Tento typ mapy se nehodí pro globální lokalizaci, jelikož mapa větších prostor je vysoce paměťově náročná.

- **Symbolické** - jsou vytvářeny pro komunikaci operátora s robotem. Obsahují informace důležité pro operátora, které však nemůže robot zjistit pomocí svého senzorického systému (což jsou nejčastěji názvy míst).

Senzorické a geometrické mapy řadíme mezi metrické, což znamená, že udržují informace o polohách, vzdálenostech a velikostech. Takovéto mapy mají jednoznačné měřítko.

2.2.2 Zaměření práce

Tato práce si nebere za cíl návrh SLAM metody, ale vychází z poznatků o lokalizaci jako součásti komplexnějších robotických úloh. SLAM metody poskytují bohatý zdroj informací o lokalizaci v nejrůznějších případech, mapách a prostředích.

Lokalizace bude probíhat v pseudotopologické mapě. Tento typ jsem nazval *mapa zkušeností*.

2.3 Senzory

Senzory jsou zařízení převádějící fyzikální hodnoty reálného světa na digitální signály a poskytují robotům zdroj informací. Při tomto převodu vždy vzniká chyba, která je vlastností senzoru. Mezi typické senzory v robotice patří tlačítka, akcelerometry, gyroskopy, kompasy, dálkoměry, mikrofony, kamery a další.

V rámci lokalizace jsou nejčastěji používány:

- **Inkrementální enkodér** - typické využití je u lokalizace position tracking. Tento senzor je umístěn u kol a sleduje jeho otáčky. Matematicky pak lze spočítat rychlost a směr pohybu, což je pro tuto lokalizaci dostačující. Nevýhodou je ztráta přesnosti například při prokluzu kol, ale lze ji částečně eliminovat použitím dalších senzorů (např. akcelerometrů, gyroskopů, aj.).

- **Kompas** - znalost otočení robota je pro lokalizace výhodou. Pro vyšší přesnost je používán v kombinaci s gyroskopem a akcelerometrem. Kombinací těchto tří senzorů lze získávat informaci o poloze v 3D prostoru.

- **Laserový dálkoměr** - jsou dva typy: 2D a 3D. 2D laserové dálkoměry měří vzdálenost bodů pouze v jedné rovině, což může mít za následek přehlédnutí překážky pod úrovní senzorů. 3D dálkoměry tímto problémem netrpí, ale prozatím se jedná o velice drahá zařízení (i 2D dálkoměry jsou proti výše uvedeným senzorům drahé).

  Výslednou informací, kterou dálkoměry poskytují, jsou mračna bodů (ang. point cloud). Dálkoměry neposkytují spojité veličiny.

- **GPS přijímač** - nejjednoduší a dnes i nejrozšířenější způsob globální lokalizace. Lokalizace probíhá v mapě světa a přesnost je určena počtem satelitů, z kterých přijímá signál. Problémem u tohoto senzoru zůstává lokalizace v místech, kde není dostatečný GPS signál (např. v budovách, pod zemí, aj.).

2.3.1 Kamery

V dnešní době na důležitosti získávají senzory, které zachycují obraz. Mezi jeden z důvodů úspěchu těchto senzorů patří fakt, že pro člověka je zrak nejdůležitější smysl. Cena běžné kamery je velice nízká a její možnosti jsou vysoké. Tyto možnosti jsou ale často vykoupeny vysokým výkonem, který je třeba ke zpracování obrazových dat.

Důležitými parametry kamer jsou: rozlišení, šířka záběru, frekvence snímků.

I nejlevnější kamery dnes dosahují rozlišení přesahující 3 megapixely, ale z důvodu výpočetní a datové náročnosti jsou v robotech používány kamery s rozlišením 640px x 480px. Reálné rozlišení kamer tedy vysoce přesahuje využitelné hodnoty.

Velice podobně na tom je frekvence snímků (ang. frames per second - FPS). Za běžnou hodnotu lze pokládat 25 snímků za sekundu, ovšem při složitějším zpracování klesá počet využitelných snímků na polovinu a méně. Pro vizuální lokalizaci je lepší méně kvalitních snímků, než více takových, které budou rozmazané. Existují i kamery, jejichž FPS je 100 a více, avšak z pohledu lokalizace jsou zbytečné.

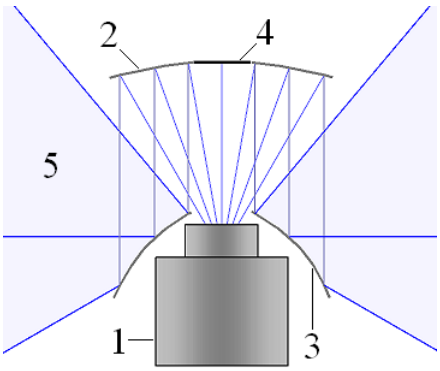
Posledním a pro tuto práci důležitým kritériem je horizontální úhel záběru.

15° až 120° - běžné kamery - nejběžnější senzor. Tento úhel je použit u většiny kamer na desce (ang. onboard) bez přidaných optických objektivů. Takovéto kamery jsou asi nejčastěji využívaným senzorem pro zpracování obrazu v robotice.

120° až 180° - širokoúhlé kamery - tohoto zorného úhlu je dosahováno pomocí optických objektivů. Speciálním případem jsou objektivy typu rybí oko, jejichž horizontální úhel záběru může být až 220°, ale trpí sférickým zkreslením.

Pro lokalizaci pomocí kamer je výhodné mít zorný úhel co největší, jelikož se zvyšuje pravděpodobnost, že se v záběru bude nacházet část prostředí důležitá pro orientaci. Druhým aspektem, který hovoří pro velký zorný úhel, je rychlost změn, které se při pohybu robota projevují v krajních mezích snímků a pravděpodobnostní lokalizace tak může proběhnout z nižšího počtu snímků (tedy v kratším čase).

360° - omnidirekcionální kamera se zrcadly - soustavou zrcadel (viz obrázek 2.1) je možné snímat obraz celého okolí robota. Jelikož zrcadla musí být velice přesně vyrobena a sestavena, není cena tohoto senzoru nízká, a to brání většímu rozšíření těchto kamer i přes fakt, že se jedná o velice výhodný robotický senzor. Právě tento typ kamer bývá úspěšně využíván pro vizuální odometrii, lokalizaci a v různých SLAMech [19] [16].



Obrázek 2.1: Pohled na konstrukci omnidirekcionální kamery. (1) kamera, (2) horní zrcadlo, (3) dolní zrcadlo, (4) mrtvý úhel kamery, (5 - světle modrá zóna) pohled kamery. Převzato z [7].

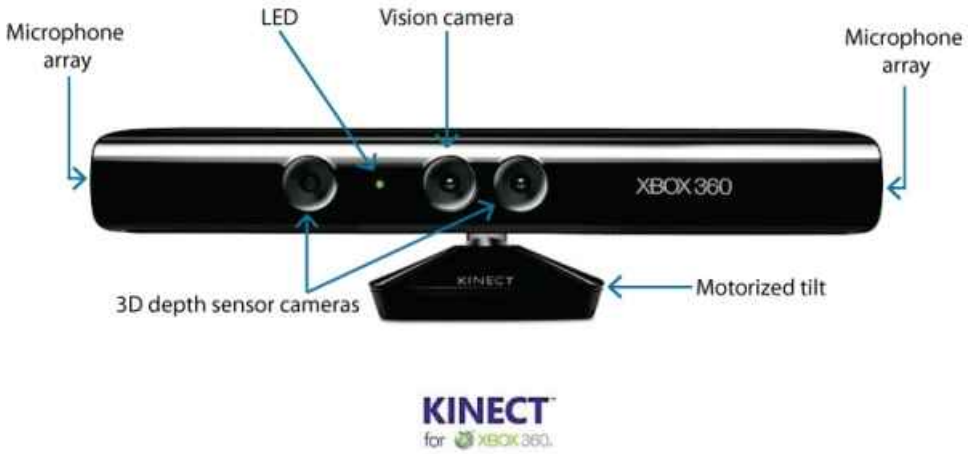
360° - složení z více kamer - levnější variantou, jak dosáhnout vícestupňového zorného pole, je použití více kamer a jejich spojení obrazů v jeden. Programů zajišťujících spojení (tvorbu panoramat) je sice mnoho, ale spotřebovávají část výpočetního výkonu, který mají roboti omezený.

Výhody a nevýhody monokulární kamery:

+ cena
+ široké možnosti využití (jeden senzor může sloužit pro více úkolů robotiky)
+ mnoho typů kamer a jejich vysoká variabilita
– výpočetně náročné zpracování a vysoké paměťová nároky
– obraz nenese žádné informace o hloubce zachyceného prostředí

Stereokamery řeší jeden z podstatných problémů monokulárních kamer. Opět se jedná o věc inspirovanou biologickými principy (lidské oči). Pomocí dvou kamer lze matematicky vypočítat hloubku obrazu a odhadovat vzdálenosti jednotlivých částí snímku. Nevýhodou stereokamery je nutnost kalibrace

Poměrně novou kategorii tvoří RGB-D senzory. Jedná se o kamery, které dokáží u každého pixelu určit jeho hloubku (ang. depth). RGB značí barevnou škálu pixelu a D jeho hloubku v prostoru. Hloubka je určována díky promítané neviditelné síti. Tato síť je snímána speciální kamerou a z deformací sítě je vypočtena hloubka jednotlivých obrazových bodů. Tyto senzory lze však použít pouze pro snímání v rozmezí několika metrů a při vhodném osvětlení (nelze použít například na přímém slunci). To je pro lokalizaci v jiném než vnitřním (ang. indoor) prostředí nepřijatelné. Ukázka tohoto senzoru je na obrázku 2.2.



Obrázek 2.2: RGB-D senzor KINECT™ od společnosti Microsoft. Převzato z [4].

2.4 ROS - Robot Operating System

Robotický meta-operační systém poskytuje knihovny a nástroje sloužící k usnadnění práce při tvorbě software robotů a je distribuován pod otevřenou licencí BSD[2]. Systém se soustředí na abstrakci hardwarové vrstvy, podporu ovladačů různých zařízení (motorů, senzorů, ...), vizualizaci rozličných dat, komunikaci mezi podsystémy, implementaci často využívaných funkcionalit a další. Mezi důležité vlastnosti patří ještě vlastní prostředí pro sestavování a běh programů, které je multiplatformní (cíleno především na UNIXové systémy) [8].

2.4.1 Struktura ROS

Důležitou vlastností ROSu je jeho modularita. Jednotlivé moduly se nazývají **uzly (ang. node)** a jsou to libovolné programy, které vzájemně komunikují pomocí zpráv a služeb, které definuje ROS. Tyto programy tedy nejsou vázány programovacím jazykem. V době psaní této práce jsou však plně podporovány jen jazyky C++ a Python (další jsou ve vývoji).

2.4.2 Balíčkovací systém ROS

Součástí ROS je balíčkovací systém, který má svou hierarchii.

Nejmenší jednotka tohoto systému je **balíček (ang. package)**. Balíček obsahuje jeden čí více uzlů, knihovny, ovladače a další související komponenty. Jeho nedílnou součástí je seznam závislostí na dalších balíčcích, ale pro dobrý návrh platí balíčku, že by jich nemělo být mnoho, aby bylo možné balíček použít v rozličných situacích.

Pro vytváření komplexnějších řešení, složených z mnoha funkcionalit, ROS zavádí **štos (ang. stack)**. Typickým využitím je stack složený z balíčků zajišťujících oživení celého jednoho robota (ovládání pohybu, zprostředkování dat ze senzorů, mapování, lokalizace, ...). Na rozdíl od balíčků nejde u stacků o co nejmenší počet závislostí, ale o zjednodušení instalace větších celků.

2.4.3 Komunikace v ROS

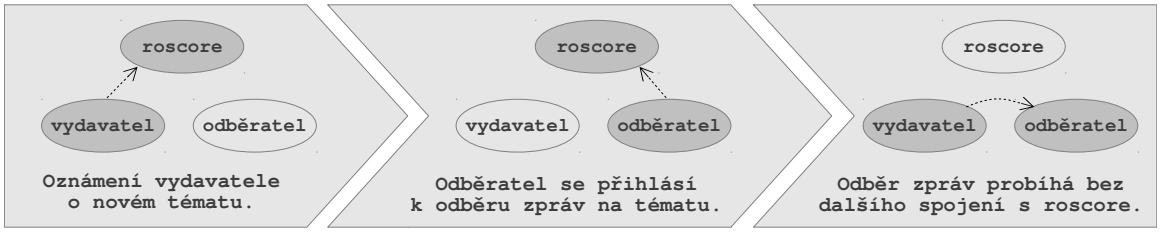
O zprostředkování komunikace se stará vždy `roscore`. Je to centrální uzel, který musí být spuštěn před ostatními uzly. Po spuštění udržuje přehled o ostatních uzlech a umožňuje jim nalezení protějšího uzlu ke komunikaci. Další komunikace pak probíhá bez nutnosti komunikace s `roscore`.

ROS implementuje dva způsoby komunikace:

- **Asynchronní, pomocí zpráv** - sdíleným kanálem pro tuto komunikaci jsou **témata (ang. topics)**, přes která jsou distribuovány **zprávy (ang. messages)**. Uzel, který odesílá zprávy na určité téma, je nazýván **vydavatel (ang. publisher)** a jiný uzel, který na něm zprávy přijímá, je **odběratel (ang. subscriber)**. Pro jedno téma platí, že na něj může posílat zprávy více vydavatelů a ty pak z něj může přijímat zaráz více odběratelů.

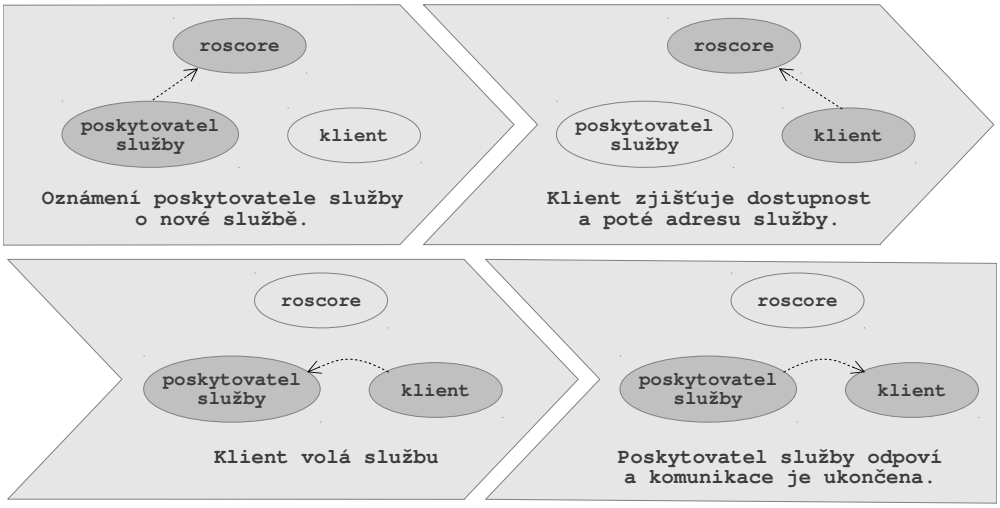
  Komunikace mezi uzly a `roscore` a poté mezi vydavatelem a odběratelem je realizována pomocí protokolu vycházejícího z XML-RPC.

[2] `http://www.linfo.org/bsdlicense.html`



Obrázek 2.3: Schéma asynchronní komunikace pomocí zpráv. Převzato z [8].

- **Synchronní, služby a klienti - služba (ang. service)** je implementována v uzlu a ten oznámí `roscore`, že je k dispozici. **Klient (ang. client)** pak může od `roscore` zjistit adresu služby a volat ji přímo (s požadovanými parametry). Uzel poskytující službu na toto volání odpovídá přímo klientskému uzlu a spojení je ukončeno.



Obrázek 2.4: Schéma synchronní komunikace mezi poskytovatelem služby a klientem. Převzato z [8].

Pro definici zpráv a služeb zavádí ROS jednoduchý textový formát, který zajišťuje nezávislost na programovacím jazyce (soubory v tomto textovém formátu se při sestavení pomocí `rosmake` rozgenerují pro všechny jazyky).

Všechny zprávy i služby jsou silně typované. Lze využít několik předdefinovaných typů nebo definovat vlastní, které mohou být následně použity v definici komplexnějších zpráv. Všechny strany, které spolu chtějí komunikovat, musí mít stejné definice zpráv a služeb, jinak je komunikace neúspěšná. Shodnost definic je ověřena z MD5 otisků definičních souborů zpráv/služeb.

Kapitola 3

Teorie - zpracování obrazu

3.1 Význačné obrazové příznaky

Přesná definice význačného obrazového příznaku (ang. image feature) neexistuje, jelikož jím může být cokoliv. Záleží vždy na požadavcích, které specifikuje každá úloha. Obecně lze za význačný obrazový příznak považovat něco, co je pro zvolenou úlohu důležité. Může se jednat o část nebo části obrazu, klíčové body, úsečky, křivky, barvy a další.

Pro detekci příznaků se používají detektory. Kvalita příznaků je dána kvalitou detektoru. Dobrý příznak by měl být znovudetekovatelný a invariantní vůči běžným transformacím obrazu - otočení, změna úhlu, zvětšení/zmenšení. Výhodou je invariantnost vůči dalším změnám, nejen obrazu (jako osvětlení, změna barevnosti aj).

Detekce význačných příznaků bývá základním krokem zpracování obrazu, a to je důvodem vzniku mnoha různých typů příznaků. Každý má své charakteristické vlastnosti a využití [6].

Vlastnosti, které jsou vyžadovány u význačných obrazových příznaků [14]:

- **Robustnost** - co nejmenší ovlivnění šumem, rozmazáním, kompresí, diskretizací
- **Lokálnost** - předchozí segmentace obrazu není vyžadována
- **Rozlišitelnost** - příznaky jsou porovnatelné v databázi příznaků
- **Invariantnost k fotometrickým deformacím a geometrickým transformacím**
- **Dostatečný počet** - v každém obraze lze nalézt dostatek příznaků
- **Výkonnost** - detekce musí být dostatečně rychlá

Příznaky můžeme rozdělit do tří hlavních skupin - hrany, rohy/význačné body a klíčové body s deskriptory.

3.2 Hrany

Hranou je myšlena křivka nebo soustava bodů mezi dvěma regiony obrazu, kde se prudce mění jas. Existují dvě metody detekce: **metoda gradientu** a **Laplaceova metoda**. Použití hran jako obrazováho příznaku je možné například pro detekci objektů v obraze [10], ale pro lokalizaci se většinou hranové detektory nepoužívají, protože prostředí lze jen obtížně popsat pomocí hran tak, aby byly pozice jednoznačné a dostatečně invariantní.

3.3 Rohy a význačné body

Původní detektory rohů pracovali na základě hranového detektoru, kdy v hranách byla určena místa ostrých zlomů a ta byla označena jako příznaky.

Pokročilé detektory rohů již pracují se sofistikovanějšími matematickými přístupy a mezi nejznámější a často využívané patří **Harrisův detektor** a algoritmus **FAST**.

Pro lokalizaci jsou tyto příznaky využitelné, jelikož jejich detekce v obraze je velice rychlá, ale z důvodu nízké invariantnosti se při určování pozice nepoužívají tolik jako klíčové body typu SIFT, SURF a další.

3.4 Klíčové body s deskriptory

Nalezený klíčový bod, proti výše uvedeným skupinám příznaků, je popsán deskriptorem. Deskriptorem je nazýván vektor příznaků. Dva deskriptory jsou při rozlišování dvou bodů vzájemně porovnávány. Tyto body bývají náročnější na výpočet, ale jsou invariantní k některým geometrickým transformacím.

V rámci lokalizace pomocí obrazu se jedná o nejpoužívanější typ příznaků. Mezi běžně využívané patří SIFT, SURF. Mezi poměrně nové patří ORB.

3.4.1 SIFT

Je to zkratka anglického *Scale-invariant feature transform*. Jedná se o klíčové body nezávislé na měřítku, rotaci, šumu a osvětlení, jež jsou popsány 128 dimensionálním vektorem (každý). Relativní pozice těchto bodů se na snímcích nemění.

Při extrakci těchto bodů je využito rozdílu dvojího Gaussova rozostření, které je aproximací druhé derivace snímku. Tento rozdílový snímek (ang. difference of Gaussians - DoG) obsahuje maxima a minima, a ta jsou brána jako možné klíčové body.

Výpočet rozdílu je opakován pro postupně zmenšovaný obraz tak dlouho, dokud není zmenšený obraz příliš malý pro detekci.

Lokální extrémy obrazu jsou poté vybrány z rozdílových (DoG) snímků a porovnává se hodnota pixelu s okolím (osm pixelů) a s devíti pixely v rámci rozdílových snímků s jiným měřítkem. Pokud je hodnota pixelu maximum nebo minimum v porovnání se všemi pixely, pak je tomuto bodu určen 128 dimensionální vektor popisující tento extrém. Následně se ještě dopočítává orientace klíčového bodu [15].

Pro lokalizaci je SIFT výbornou volbou, jelikož poskytuje stabilní klíčové body, které jsou detekovány v různých snímcích z různých pozic. Jedná se však stále o detekci bodů a ne objektů. 2D objekt tedy může být popsán za pomocí klíčových bodů, jejich relativních pozic a deskriptorů. Pokud budeme pracovat v 3D prostoru, bude muset být objekt popsán za pomoci více obrazů (a jejich bodů).

3.4.2 SURF

Je to zkratka anglického sousloví *Speeded Up Robust Features*. Jedná se o podstatně mladší detektor klíčových bodů (SIFT - 1999, SURF - 2006) a byl částečně inspirován právě SIFT detektorem. Standardní verze SURF detektoru je rychlejší a robustnější než SIFT. Stejně jako SIFT popisuje klíčové body deskriptory. Je invariantní vůči rotaci a změně měřítka [2].

Extrakci lze rozdělit do dvou fází: vyhledání klíčových bodů (rohy, skvrny, T-spoje) a výpočet deskriptoru.

Rozdílem mezi SURF a SIFT je, že SIFT porovnává stejný obraz ve více měřítkách, zatímco SURF používá integrální obraz. Integrální obraz je obraz, jehož každý pixel je součtem hodnot předchozích pixelů vlevo a nahoře. Poslední bod obrazu (pravý spodní) tedy obsahuje součet všech pixelů obrázku. Díky tomu je možné získat informace o intenzitě oblasti obrázku pouze z krajních bodů oblasti.

Detekce klíčových bodů pak probíhá za pomoci detektoru založeného na výpočtu tzv. Hessiánu (Hessovy matice) a extrakce deskriptorů za pomoci výpočtu odezvy Haarových vlnek v okolí významných bodů.

Kompletní vzorce a přesný matematický postup pro zisk SURF klíčových bodů lze nalézt v článku Speeded-Up Robust Features [2].

Jelikož byl SURF detektor patentován, vznikla jeho obdoba OpenSURF, která funguje na podobném principu.

SURF je pro svou rychlost a kvalitu bodů velice často používán pro lokalizaci a v různých SLAM metodách [20] [1]. Pro dostatečně rychlou detekci je důležité správně nastavit práh, který určuje kvalitu a tím i počet detekovaných bodů.

Porovnání SURF a SIFT popisuje článek A Comparison of SIFT, PCT-SIFT and SURF [13].

3.4.3 ORB

Alternativou k SURF a SIFT jsou klíčové body ORB. ORB je anglická zkratka pro *Oriented FAST and Rotated BRIEF*. Z tohoto názvu plyne, že detekce bodů je založena na algoritmu FAST a výpočet deskriptorů obstarává BRIEF extraktor. [18]

ORB je implementován v knihovně OpenCV a proto je zde zmíněn. Jelikož se jedná o nový algoritmus, není mnoho prací, které ho použili v lokalizačních úlohách. V rámci této práce bude vyzkoušen. Výsledky lze najít v sekci 5.4.4.

3.5 Porovnávání klíčových bodů

Získané klíčové body je třeba porovnávat. Porovnání může být mezi dvěma obrazy nebo celou databází. V případě dvou obrazů není nutné hledět na časovou náročnost, avšak při vyhledávání v databází obrázů již je čas porovnání zásadní.

Korespondence SURF nebo SIFT klíčových bodů mezi dvěma obrazy funguje na základě srovnání **Euklidovských vzdáleností** deskriptorů. Čím je vzdálenost nižší, tím jsou si body podobnější. Porovnávání vzdáleností deskriptorů a hledání minima vždy vede k nalezení korespondence, i když mezi prohledávanými deskriptory korespondence neexistuje. Tento jev se odbourává výpočtem druhé nejmenší vzdálenosti a platnost korespondence je určena na základě porovnání těchto dvou hodnot [5].

Při porovnání je možné použít metody hrubé síly (ang. brute force), které srovnávají obrázek po obrázku, nebo je možné využít sofistikovanějších algoritmů, které umí efektivně ukládat a následně prohledávat databázi snímků. Tyto algoritmy velice často pracují s pokročilými stromovými strukturami. OpenCV implementuje několik typů porovnávačů (ang. matcher) využívajících hrubou sílu: **NORM_L1**, **NORM_L2**, **NORM_HAMMING**, **NORM_HAMMING2**, **L1** a **L2**. Algoritmus **FLANN** je v OpenCV jediný, který je určen pro rychlé prohledávání velkých databází obrazů.

Kapitola 4

Návrh řešení

Základní motivací této práce bylo vytvoření systému pro lokalizaci mobilního robota ve zmapovaném prostoru, jehož hlavními vlastnostmi bude jednoduchost, co nejmenší výpočetní náročnost a nízká cena využitých senzorů. Jedná se o požadavky, které nebývají v robotice běžné, spíše naopak. Většina robotů vytvořených ve vědeckých laboratořích nese množství senzorů, jejichž ceny se pohybují v řádech statisíců korun a tím je jejich použití pro triviální úkoly běžného života nemožné. Výpočetní výkon těchto robotů vysoce přesahuje možnosti běžného přenosného počítače a ani tak není dostačující pro plnou operabilitu v reálném čase.

Návrh mého řešení se snaží vycházet z principů blízkých člověku. Nejen proto je jako hlavní a jediný senzor vybrána kamera. Člověk vybavený zrakem nemá problém se lokalizovat v jakémkoliv známém prostředí a nemusí znát přesné vzdálenosti jednotlivých objektů v zorném poli, jako například roboti, kteří se lokalizují pouze za pomoci laserových dálkoměrů.

Na lokalizaci lze nahlížet z několika směrů. Buď jako na problém, kdy robot musí znát svou pozici s přesně určenou přesností, nebo jako na úkol najít svou přibližnou pozici, která je postačující pro letmou orientaci, a v poslední řadě také jako na kombinaci těchto dvou přístupů. Kombinace obou principů je opět blízká lidskému chování. Pro globální lokalizaci stačí znát přibližnou pozici, kde se nacházíme (např. ve kterém části parku se nacházíme), a přesnou lokalizaci využíváme například proto, abychom nesešli z cesty, po které aktuálně jdeme. V této práci se budu zabývat globální lokalizací, jejíž prioritou nebude metrická přesnost, ale správnost.

Jak bylo tedy nastíněno, tato práce se dívá na lokalizaci v jiném světle, než je v robotice běžně vnímána, a proto musí být navržena s jistou variabilitou, aby bylo možné snadno měnit nastavení a experimentovat s výsledky.

V robotice bylo dříve běžné pracovat s každým robotem znovu od začátku. Naprogramovat ovladače pro jeho hardware, vytvořit prostředí pro robotovu kontrolu a teprvé poté začít programovat aplikace. Dnes již tento přístup bývá ojedinělý, jelikož je snahou roboty stavět ze sériově vyráběných součástí, pro které jsou programovány generické ovladače a to umožňuje celý hardware robota abstrahovat. Právě z důvodu co největší abstrakce, sjednocení přístupů a zaměření se přímo na tvorbu aplikací robotů, jsou vytvářeny různé knihovny, frameworky a meta-operační systémy určené pro robotiku, jejichž velkou výhodou je snadná přenositelnost naprogramované aplikace na jiné roboty.

Tato práce tedy bude navržena jako soubor aplikací pro robotický meta-operační systém ROS a bude co nejvíce využívat jeho možností.

4.1 Existující řešení

Využitím ROSu, jehož součástí je již mnoho hotových aplikací (balíčků), vyvstává otázka, zda-li již tento problém někdo neřešil. Proto jsem prozkoumal a vyzkoušel několik balíčků, které řeší lokalizaci.

4.1.1 ethzasl_ptam

Nejedná se o jeden balíček, ale o celý ROS stack, poskytující SLAM techniku založenou na monokulární kameře a inerciální jednotce (IMU). Je určen pro létající robotické platformy a je význačný svou vysokou přesností a relativně nízkou výpočetní náročností.

4.1.2 hector_localization

Jde o ROS stack, který zajišťuje plný 6DoF (*six Degree of Freedom* - šest os volnosti) odhad pozice. Lokalizace probíhá primárně na základě inerciálního senzoru, ale podporuje další senzory jako GPS a kompas. Je součástí ROS stacku **hector_slam**, který přidává k lokalizaci schopnost mapování prostředí

4.1.3 amcl

Tento balíček zahrnuje implementaci pravděpodobnostní lokalizace (Monte Carlo) robota pohybujícího se v 2D prostředí. Odebírá zprávy z laserového dálkoměru a promíta je do mřížky obsazenosti.

4.1.4 ar_kinect

Zlepšuje lokalizaci pomocí značek (ang. markers) za využití dat z RGB-D senzoru. Data ze senzoru jsou přijímána jako mračno RGB bodů (ang. point cloud). Výstupem je transformovaná pozice mezi kamerou a rozpoznanou značkou.

4.1.5 humanoid_localization

Poskytuje 6DoF lokalizaci ve speciálním typu 3D map - OctoMap. Jedná se o mapu obsazenosti ovšem oproti klasické mřížce, která je jen dvourozměrná, mapuje 3D prostor. Důležitou vlastností OctoMap je její snížená paměťová náročnost. Jako vstupní údaje pro lokalizaci tento ROS balíček přijímá data z laserového dálkoměru nebo mračna bodů. Pro přesnější určení pozice zpracovává data z inerciální jednotky a používá odometrická data.

4.1.6 gmapping

Jedná se o implementaci SLAM metody z webových stránek [24] pro ROS. Vstupními daty jsou opět data z laserového dálkoměru a výstupem je pozice v mapě. Jelikož se jedná o SLAM, tak si zároveň vytváří mapu ve formě mřížky obsazenosti.

Z výše uvedených informací je vidět, že požadované vlastnosti nastíněné na začátku této kapitoly (viz 4) nesplňuje ani jeden z již hotových ROS balíčků, ale přesto poskytují množství poznatků, které jsou přínosné pro návrh mé vlastní lokalizační metody. Například z **ethzasl_ptam** je vidět, že pro lokalizaci lze efektivně využívat obyčejnou monokulární

kameru. V **ar_kinect** jsou při lokalizaci používány značky, vůči kterým robot určuje svou pozici, což je rozhodně originální způsob.

Většina balíčků však pracuje s laserovým dálkoměrem, inerciální jednotkou nebo s jiným senzory, které byly hned v úvodu této kapitoly zamítnuty jako nevhodné pro tuto práci. Proto byl nutný průzkum mezi řešeními, které nejsou součástí ROS. V rámci tohoto průzkumu jsem nalezl mnoho prací, které s mou souvisí. Mezi nejdůležitější patří:

4.1.7 RatSLAM

Jedná se o SLAM inspirovaný biologickým principem krysího mozku, který udržuje povědomí o své pozici ve světě na základě vizuální podobnosti míst. Mapa, která je touto metodou vytvářena, je hybridem mezi prostorovou a topologickou mapou, kde jsou pro každé místo ukládány zkušenosti ve formě obrazů. Pozice míst jsou určovány pomocí odometrie a vizuální zkušenosti jsou používány pro uzavírání smyček. RatSLAM pro lokalizaci nepopužívá žádné geometrické výpočty ze snímků.

Informace byly získány z článku Outdoor simultaneous localization and mapping using RatSLAM [16].

Variantou této SLAM techniky je **OpenRatSLAM**, který je implementován i pro ROS.

4.1.8 Lokalizace využívající SIFT a aktivní monokulární kameru

Diplomová práce Holtkampa a Jonga [15] vychází z jednoduchého principu lokalizace. Zaměřuje se na přibližný odhad pozice za využití aktivní kamery a klíčových bodů typu SIFT. Zajímavou částí tohoto projektu je optimalizace databáze, která zde slouží jako mapa.

4.1.9 Monokulární vize pro lokalizaci robota

Článek Royera, Dhoma a Lavesta [9] se zabývá vytvořením mapy pomocí techniky **hloubka obrazu z pohybu (ang. structure from motion)**. Za pomocí této techniky lze získat informace o hloubce obrazu z několika po sobě jdoucích snímků vytvořených během pohybu robota. Práce robota je zde rozdělena do dvou úkolů - vytvoření mapy a navigace v ní. Uváděné výsledky této lokalizační metody jsou zatíženy překvapivě malou chybou (v jednotkách centimetrů).

4.1.10 Odhad pozice na základě vizuální informace

Lokalizace ve vnitřním prostředíje hlavní náplní článku Virtual reference view generation for CBIR-based visual pose estimation [12]. Poskytuje poznatky, které lze aplikovat i pro určování pozice ve venkovním prostředí. Popisuje získávání pozice v mapách, které jsou vytvořeny z dat různých senzorů (pro lokalizace jsou však využity pouze kamerové snímky).

Nevýhodou posledních tří prací je absence zdrojových kódů. Kdyby tyto programy byly přepsány jako balíčky pro ROS, jistě by byly hojně využívany, jelikož kvalitní lokalizační metody pro venkovní prostředí zatím v ROS nejsou.

Výše uvedené práce dokazují, že lokalizace za pomocí monokulární kamery je možná a to hned na několika úrovních přesnosti a složitosti (algoritmické i výpočetní).

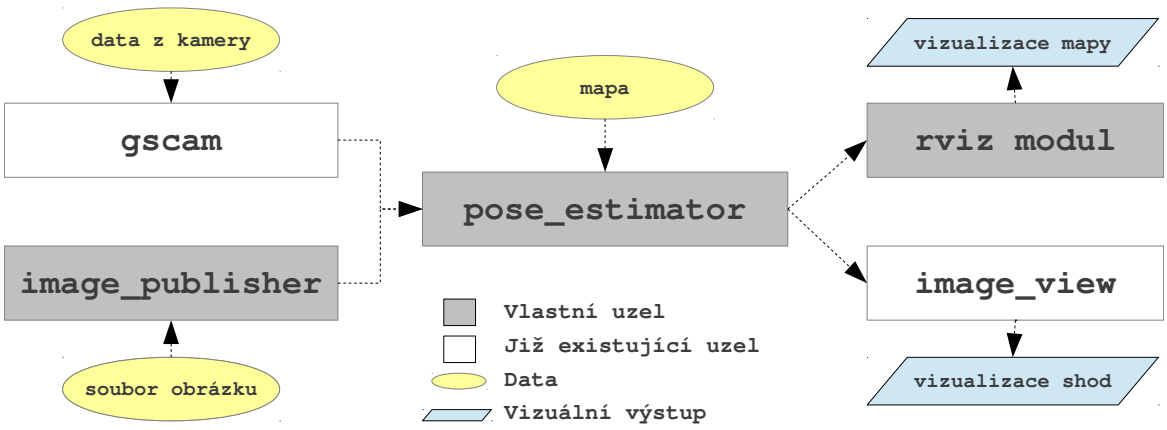
4.2 Specifikace požadavků

Z poznatků získaných během studia existujících řešení, které jsou v předchozí sekci 4.1 shrnuty, ze zadání této bakalářské práce a z mých představ uvedených na začátku této kapitoly (viz 4) je nutné přesně specifikovat problém, jehož návrh řešení bude dále uvažován.

- Řešení bude plně využívat možností meta-operačního systému ROS s důrazem na znovupoužitelnost jednotlivých balíčků.
- Cílem je vytvoření globální lokalizační metody pro odhad pozice ve známem dynamickém prostředí.
- Program bude primárně navržen pro práci ve venkovních prostředích, ale neměl by být na tomto prostředí závislý.
- Jako hlavní a jediný senzor pro lokalizaci bude využita monokulární kamera (webkamera) a snímky z ní.
- Metoda bude pracovat s více možnými hypotézami odhadu pozice.
- Jedna pozice bude určena jako identifikátor značky, která odpovídá naučené zkušenosti (odpovídající jednomu snímku v daném místě).
- Součástí řešení musí být jednoduchý mapový formát uchovávající zkušenosti a jejich geometrické pozice, který bude sloužit pro lokalizaci a její vizualizaci.
- Nastavení bude snadno měnitelné, jelikož by tato práce měla sloužit hlavně k experimentům.

4.3 Návrh řešení v ROS

V teorii bylo uvedeno (viz 2.4), jak vypadá struktura ROS a jak funguje. Celou práci jsem tedy rozdělil do několika částí, které budou tvořit jednotlivé uzly. Toto rozdělení lze vidět na následujícím diagramu:



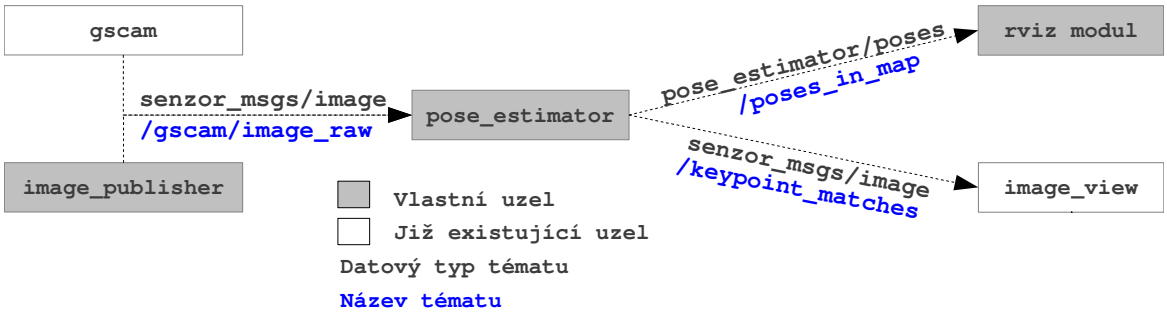
Obrázek 4.1: Rozvržení prototypu na uzly.

Hlavním uzlem je *pose_ estimator*, který se stará o odhad pozice v mapě. Tu si musí při svém startu vytvořit. Přijímá zprávy od uzlu *gscam*, který již je v ROS implementován nebo od uzlu *image_ publisher*, který je nutné vytvořit pro snadné testování a experimentování. Uzel *image_ publisher* proto musí umět publikovat jeden obrázek, který mu je zadán ve formátu obrazových zpráv (v takovém jako publikuje *gscam*).

Uzel *image_ view* je také balíčkem, který je volně dostupný a stará se o vizualizaci porovnání aktuálního lokalizovaného snímku s nejpodobnějším ze snímků, které jsou součástí mapy. Uzel *image_ view* toto porovnání neprovádí, to je zajištěno uzlem *pose_ estimator*, jeho náplní je pouze zobrazování obrazových zpráv. Druhým vizualizačním prvkem je *rviz_ module*, zobrazující pozici v satelitní mapě. *Rviz* je vizualizační nástroj ROSu a poskytuje možnost vlastních rozšíření ve formě modulů.

4.4 Komunikace uzlů

Protože je funkcionalita rozdělena mezi několik na sobě závislých uzlů, musí být ustanoveny komunikační kanály. Jako nejvýhodnější se jeví využití asynchroní komunikace ROS ve formě zpráv.



Obrázek 4.2: Rozvržení komunikace mezi uzly.

Filozofií ROS komunikace pomocí témat je standardizace různých typů zpráv, aby bylo možné kombinovat uzly z různých balíčků. V diagramu lze pozorovat, že i tato práce se snaží co nejvíce využít zavedených témat. Většina zpráv je typu *sensor_ msgs/image*. ROS poskytuje několik typů zpráv pro práci s odhadovanou pozicí, avšak žádná není pro tuto práci vhodná, protože všechny pracují s geometrickou pozicí.

Vlastní typ zprávy *pose_ estimator/poses*, nesoucí informace o výsledcích lokalizace, musí obsahovat identifikátory pozic a jejich pravděpodobnost.

4.5 Mapa

Mapa, v které bude lokalizace probíhat, bude složena ze zkušeností. Tvorba takovéto mapy je složitější, ale výhodou je, že při následné lokalizaci v takovéto mapě stačí jeden snímek bez informací o jeho hloubkové struktuře. Výpočet geometrické pozice je tedy proveden pouze ve fázi tvorby mapy.

V rámci využití mapy jak pro lokalizaci, tak pro vizualizaci je nutné, aby obsahovala:

- satelitní snímek mapy místa, kde byly zkušenosti pořízeny
- zkušenosti

Jedna každá zkušenost musí obsahovat:

- jednoznačný identifikátor zkušenosti
- snímek spojený s touto zkušeností (obraz)
- klíčové body získané z tohoto snímku
- extrahované deskriptory klíčových bodů
- geometrickou pozici v satelitním snímku celého prostředí, přepočtenou jako pozici v mřížce pixelů obrázku

Mapa bude uložena v jednoduchém textovém formátu. Přímou součástí tohoto souboru tedy nebudou obrázky, ale pouze relativní cesty k nim. Mapa proto bude tvořena tímto konfiguračním souborem a složkou, obsahující veškerá obrazová data, na které je v konfiguračním souboru odkazováno.

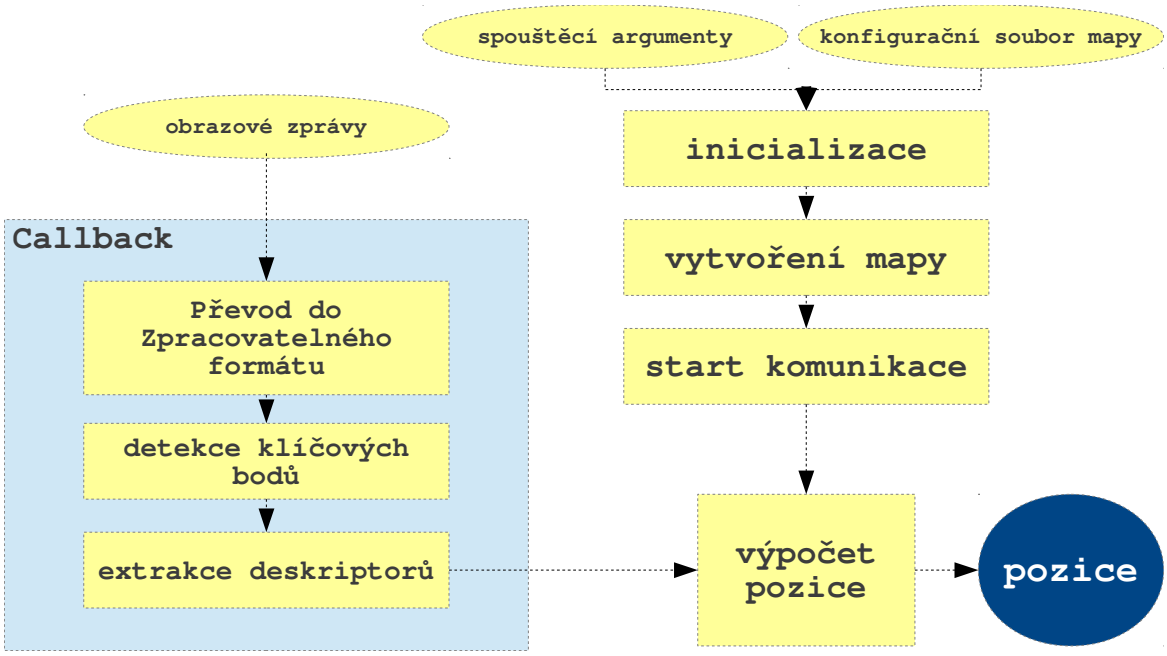
Klíčové body a deskriptory nesmí být přímou součástí uložené mapy, ale musí se detekovat a extrahovat až při způštění uzlu *pose_ estimator*. Kdyby byly ukládány jako přímá součást mapy, musely by být vygenerovány ve všech možných typech (SIFT, SURF, ORB) a parametrech (různé prahy detekce), což by jednak zvyšovalo paměťovou náročnost a také omezovalo rozšiřitelnost a nastavitelnost.

Pro usnadnění a částečnou automatizaci tvorby mapy musí být součástí řešení aplikace nebo skript, mající na starost generování mapy v definovaném textovém formátu. Přímo se nabízí využití dnes běžně dostupné technologie, umožňující automatické ukládání GPS souřadnic místa, kde byl obraz pořízen. Pokud vezme obrazy, které mají v EXIF datech uloženy tyto souřadnice, není problém je extrahovat a přepočíst je na pozici určenou v pixelech. Stačí znát GPS souřadnice rohů satelitní mapy.

Pro získání satelitního snímku mapy lze použít některou z mnoha existujících mapových aplikací. Tytéž aplikace dokáží poskytnout informace o GPS souřadnicích rohů vybrané části mapy.

4.6 Lokalizace

Nejdůležitějším uzlem celého řešení je *pose_ estimator*, jelikož se stará o lokalizaci, která je jádrem celé této práce. Pro odhad pozice bude využita pouze monokulární kamera. Získáme z ní snímek a budeme se snažit najít co nejlepší shodu v mapě tvořené zkušenostmi. Celý proces lokalizace zobrazuje následující diagram.

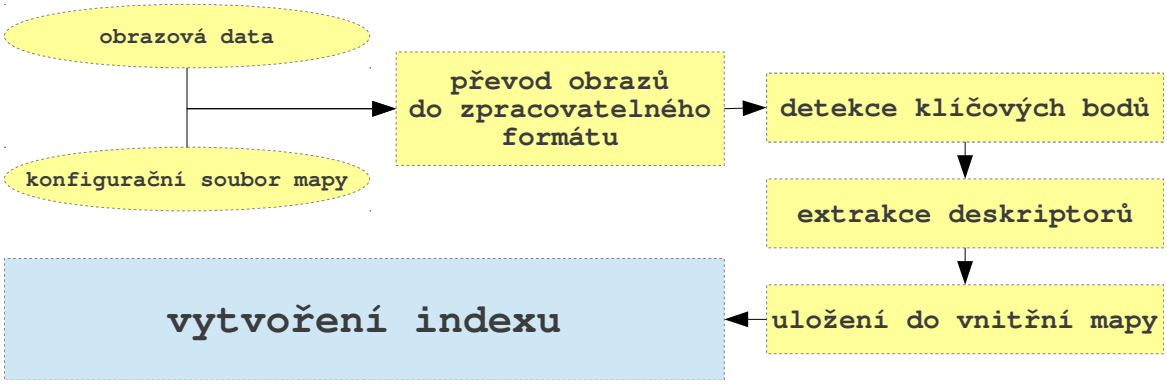


Obrázek 4.3: Diagram lokalizace pomocí uzlu *pose_ estimator*.

Uzel je spuštěn s parametry, které zajistí konfiguraci detektoru klíčových bodů, extraktoru deskriptorů a porovnávacího algoritmu. Při spuštění je uzlu také předán název konfiguračního souboru mapy, ve které bude lokalizace probíhat. Na začátku běhu je nezbytná kontrola správnosti zadaných konfigurací. Kontrola by měla odhalit chyby, které by způsobily pád programu nebo vysoce nepřesnou lokalizaci.

Prvním krokem po úspěšné konfigurace uzlu je vytvoření mapy. Tato mapa musí být k dispozici po celou dobu běhu programu. Pro každou zkušenost, která je v konfiguračním souboru mapy uvedena, se vytvoří záznam, který se skládá z položek: identifikátor zkušenosti, snímek v dále zpracovávatelném formátu, detekované klíčové body a extrahované deskriptory těchto bodů. Po průchodu všech záznamů je vytvořen index pro porovnávání a tím je mapa dokončena.

Činnosti při tvorbě mapy ilustruje diagram 4.4.

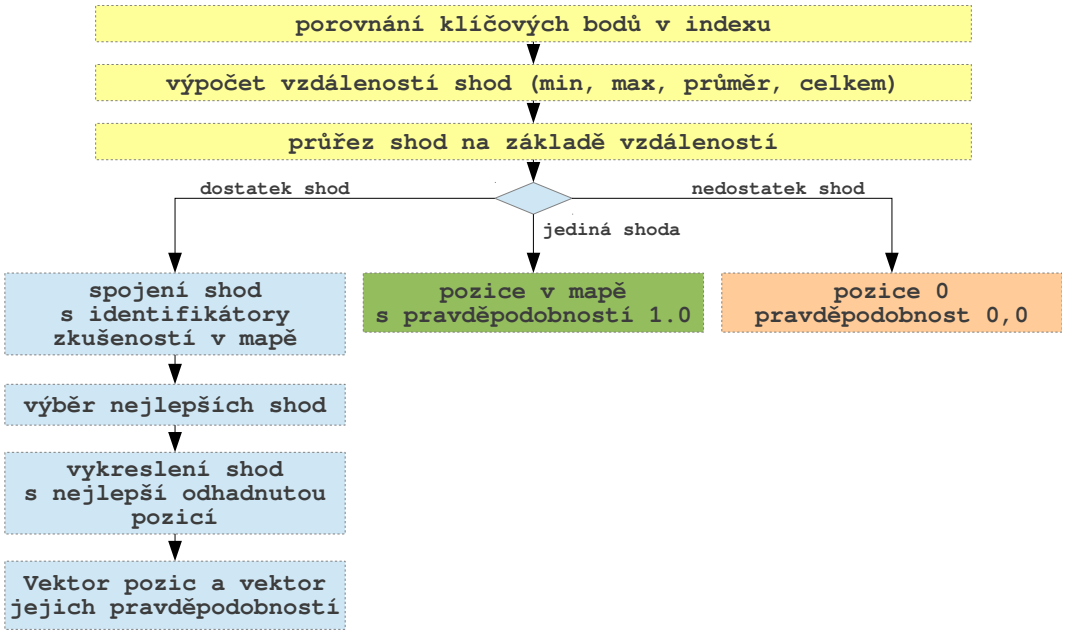


Obrázek 4.4: Postup tvorby mapy zkušeností.

Teprve po vytvoření mapy lze začít s lokalizací. V tuto chvíli tedy musí být navázána komunikace z dalšími ROS uzly. Zaregistruje se publikování zpráv pozic a proběhne přihlášení k odběru zpráv z kamery (nebo uzlu *image_publisher*).

Po příchodu zprávy na odebíraném tématu se bude volat lokalizační funkce, jež přímo vrací pozice ve formátu zprávy typu *pose_estimator/poses*.

Lokalizační funkce nejprve musí zpracovat obraz do formátu, s kterým je možno dále pracovat. Z tohoto obrazu jsou následně detekovány klíčové body a extrahovány deskriptory klíčových bodů pomocí stejného detektoru a extraktoru použitého při vytváření mapy. Z těchto údaju probíhá výpočet odhadu pozice.



Obrázek 4.5: Postup výpočtu pozice.

V diagramu 4.5 lze vypozorovat průběh výpočtu pozice. Základem celého výpočtu je porovnání klíčových bodů aktuálního snímku se všemi zkušenostmi v mapě. Z tohoto porovnání je zjištěno několik poznatků: minimální, maximální, průměrná a celková euklidovská vzdálenost mezi porovnávanými klíčovými body. Tyto poznatky slouží k průřezu (ang. pruning) shod klíčových bodů. Prořezány jsou ty shody, jejichž vzdálenost je vyšší než průměr.

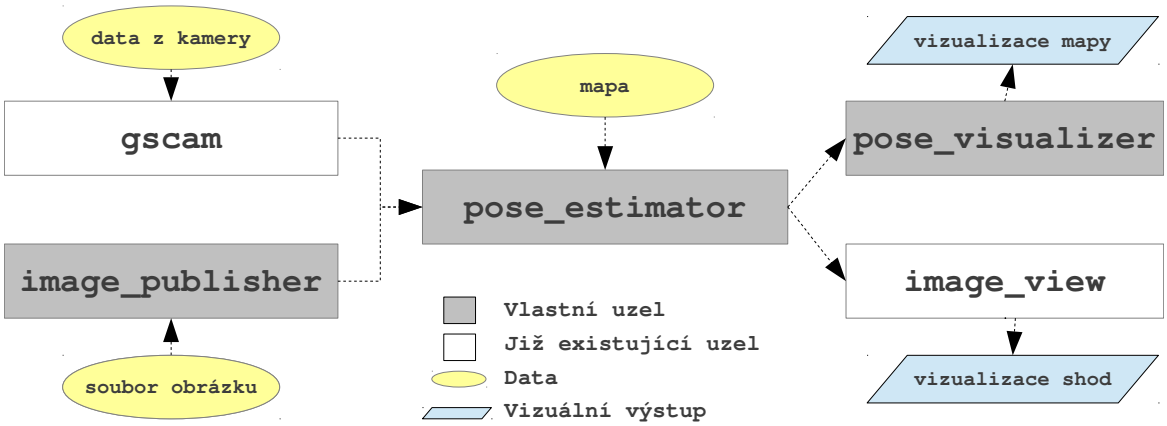
Po průřezu se musí rozhodnout, zda-li zbyl dostatek shod pro určení pozice, nebo zda-li byla lokalizace neúspěšná. Pokud je shod dostatečný počet, jsou spojeny se zkušenostmi, kterým odpovídají. Zkušenosti, které jsou spojeny s největším počtem shodných, jsou vráceny (jejich identifikátory) jako pravděpodobné pozice, kde se robot nachází.

4.7 Aktualizace návrhu během implementace

Během realizace návrhu jsem dospěl k několika poznatkům, které znamenaly nutnost změny návrhu.

První problém návrhu se objevil během realizace uzlu *rviz_module*, starajícího se o vizualizaci mých vlastních zpráv typu *pose_estimator/poses*. Složitost zobrazení barevného satelitního obrázku jako mapy na pozadí mě přimělo k návrhu vlastního jednoduchého vizualizačního uzlu *pose_visualizer*, který si uloží mapu v navrženém formátu.

Uzel musí zobrazit satelitní mapu a všechny zkušenosti v ní. Každou zkušenost je třeba zobrazit jako značku v barvě, která určí pravděpodobnost výskytu robota na dané pozici. Aktualizovaný diagram na obrázku 4.6.



Obrázek 4.6: Aktualizované rozvržení prototypu na uzly.

Druhá změna návrhu byla spojena s tvorbou mapy. Při pořizování snímků jsem zjistil, že přesnost GPS přijímače, použitého pro získání pozice, kde byl snímek pořízen, je silně nedostatečná a protože pořízení kvalitnějšího GPS přijímače by bylo příliš nákladné, rozhodl jsem se vytvořit jednoduchou aplikaci pro tvorbu map, ve které bude pozice určována manuálně. Tato aplikace nemusí být implementována jako součást ROS, jelikož se jedná pouze o podpůrnou funkcionalitu.

Kapitola 5

Realizace, experimenty a vyhodnocení

Volba hlavního implementačního jazyku byla omezena využitím meta-operačního systému ROS, který plně podporuje pouze C++ a Python. Rozhodl jsem se využít jazyk C++ z důvodu osobní preference a také kvůli jeho větší znalosti proti jazyku Python. Pro implementaci pomocné aplikace na tvorbu map jsem zvolil kombinaci jazyků PHP, Javascript, HTML a CSS.

5.1 Použité prostředky

Mezi nejdůležitější prostředky lze jednoznačně zařadit ROS verze Groovy, který je spojovacím prvkem celé implementace (a také celé bakalářské práce). Mezi jeho nejdůležitější a nejvíce využité části lze zařadit: systém pro kompilování a sestavování balíčků, asynchronní komunikaci pomocí zpráv, vizualizační nástroj *rxgraph*, konzolové nástroje *rostopic*, *roscreate-pkg* a *rosrun*.

Pro veškerou práci související se zpracováním obrazu byla použita knihovna OpenCV, která je úzce spjata s ROS. Díky této knihovně je dnes možné používat velice pokročilé algoritmy zpracování obrazu s minimální námahou. S využitím OpenCV souvisí i využití jeho podknihovny HighGui, která umožňuje rychle vytvářet jednoduchá multiplatformní uživatelská rozhraní.

Implementace prototypu je závislá na ROS balíčcích *roscpp*, *opencv2*, *cv_bridge*, *image_transport*, *sensor_msgs*, *sensor_msgs* a také na knihovně Boost. Pro práci s kamerou je využit balíček *gscam* a pro vizualizaci zpráv obrazového typu *image_view*.

Všechny využívané prostředky jsou distribuovány pod otevřenými licencemi a jsou multiplatformní, i když zatím spíše v teoretické rovině.

5.2 Implementace

Prototyp se skládá ze tří balíčků, obsahující vždy jeden uzel, a aplikace pro tvorbu map. Balíčky byly vždy vytvořeny pomocí programu roscreate-pkg, který je k tomu určen. Balíček v ROS je adresář, který musí obsahovat soubor *manifest.xml*, popisující specifikaci balíčku a *CMakeLists.txt*, což je soubor typu CMAKE, který popisuje kompilaci a linkování.

5.2.1 image_publisher

Tento balíček obsahuje uzel *image_ publisher*, který je naprogramován, aby přijímal jako parametr adresu obrázku, který se má publikovat na téma `/gscam/image_raw`. Typ zprávy tohoto tématu obsahuje více než jen data obrazu a bez vyplnění těchto dat není možné obrázek publikovat. Tento uzel tedy vyplní celou datovou strukturu zprávy typu *sensor_ msgs/Image* a tím umožní obraz publikovat.

5.2.2 pose_estimator

Implementace tohoto uzlu kopíruje návrh, který byl uveden v 4.6. Kód tohoto uzlu je rozdělen do tříd:

- *pose_ estimator* - zajišťující vytvoření mapy, zahájení komunikace a odhad pozice
- *config*, který zpracovává konfigurační soubor mapy
- *experience_ map*, poskytující metody pro práci s mapu vytvořenou při běhu uzlu
- *error*, což je třída výjimky.

Pro implementaci bylo nutné definovat nový typ zpráv *pose_ estimator::poses*, který má následující tvar:

```
int32[] poses
float32[] probabilities
```

Ukázka 5.1: Formát zprávy *pose_ estimator::poses*

Jedná se o dva vektory, které musí být vždy stejně dlouhé a jedna pozice je tvořena z páru hodnot pod stejným indexem. Vektor `poses` je určen pro identifikátory zkušeností a `probabilities` pro pravděpodobnosti jednotlivých pozic.

Při práci s detektorem, extraktorem a srovnávačem bylo využito univerzálních rozhraní pro tyto objekty, které OpenCV podporuje, čímž bylo dosaženo snadné nastavitelnosti typu klíčových bodů a srovnávače a také snadné rozšiřitelnosti aplikace o nové typy.

Uzel je spouštěn s jedním povinným parametrem, cestou ke konfiguračnímu souboru mapy, a třemi nepovinnými: typ detektoru (SIFT, SURF, ORB), typ extraktoru (SIFT, SURF, ORB) a typ srovnávače (viz 3.5). Po spuštění je vytvářena mapa, což je výpočetně a časově nejnáročnější část programu, avšak v tomto místě to není překážkou. Rychlost odhadu pozice v mapě závisí na velikosti mapy, ale probíhá v přijatelné rychlosti.

Komunikace uzlu probíhá přes tři ROS témata: `/gscam/image_raw` a `/keypoint_matches`, z nichž jsou zprávy přijímány, a `/poses_in_map`, na kterém jsou publikovány odhadované pozice. Názvy témat lze měnit jen v kódu ve volání funkce `startCommunication`, která přijímá názvy jako parametry.

Při přijmutí zprávy z tématu `/gscam/image_raw` je volána callback funkce, převádějící snímek do formátu matice OpenCV (`cv::Mat`) a po úspěšném převedení je volána funkce `estimatePose(cv::Mat& image)`, jejíž parametr je právě převedená matice. Tato funkce je jádrem lokalizace.

5.2.3 pose_visualizer

Kód uzlu *pose_ visualizer* využívá knihovny HightGui a využívá stejnou třídu pro zpracování konfiguračního souboru mapy jako předchozí balíček. Při spuštění, které vyžaduje jediný

povinný parametr, cestu k souboru mapy, přečte konfiguraci mapy a zobrazí z ní odkázaný satelitní snímek. Druhým parametrem lze nastavit jméno, s jakým bude tento uzel spouštěn, čímž lze dosáhnout vícenásobného spuštění tohoto programu.

Po první a každé další přijaté zprávě z tématu `/poses_in_map` vykreslí všechny zkušenosti z mapy jako tečky jedné barvy a ty, které jsou obsaženy ve zprávě, vykreslí jinou barvou, jejíž světlost je odvozena od pravděpodobnosti uložené ve zprávě.

5.2.4 Aplikace pro tvorbu map

Tato aplikace na rozdíl od ostatních není implementována v ROS. Jedná se totiž jen o podpůrnou aplikaci pro vytvoření konfiguračního souboru mapy.

První vytvořená verze je pouhý PHP skript (*create_map_from_GPS.php*) procházející složku, v níž jsou uloženy obrázky zkušeností. Z každého takového obrázku získá z jeho EXIF dat GPS souřadnice místa, z kterého byl pořízen. Takto získané souřadnice přepočte na pozici v pixelové mřížce satelitního obrázku, který také musí být součástí procházené složky. Získaná data nakonec uloží do konfiguračního souboru mapy, jehož formát je uveden v 5.2.

Druhá verze (*create_map_manually.php*) byla vytvořena, protože jsem při testování zjistil, že GPS pozice uložené v EXIF obrázku nemají dostatečnou přesnost. Tato verze slouží k manuálnímu určování pozice, kde byl snímek pořízen, a byla implementována včetně jednoduchého grafického uživatelského rozhraní. Podoba tohoto rozhraní byla vytvořena pomocí HTML a CSS a ovládací logika s využitím javascriptového frameworku jQuery. Uživateli je vždy ukázán obrázek, který je potřeba umístit do satelitní mapy pomocí kliknutí. Jakmile je všem obrázkům určena pozice, aplikace sama vytvoří obsah konfiguračního souboru mapy, který je třeba zkopírovat a uložit. Formát je stejný jako v první verzi této aplikace (viz 5.2).

Pomocnou aplikací pro získání satelitního snímku v libovolném přiblížení a bez perspektivní projekce je *google_maps_without_perspective.html*. Je v ní využito Google maps API.

```
relativni/cesta/satelitniho_obrazku.png;615;514
0;relativni/cesta/obrazku_zkusenosti1.jpg;20;20
1;relativni/cesta/obrazku_zkusenosti1.jpg;50;50
```

Ukázka 5.2: Formát konfiguračního souboru mapy

První řádek konfiguračního souboru mapy určuje satelitní obrázek míst, ke kterému se vztahují následně vypsané zkušenosti. Tento řádek obsahuje relativní cestu k satelitnímu obrázku, šířku tohoto obrázku v pixelech a výšku, taktéž v pixelech. Oddělovačem těchto údajů je středník. Další řádky jsou již ve stejném formátu a obsahují: číselný identifikátor zkušenosti, relativní cestu k obrázku, který je se zkušeností spjat, a pozici, která je určena dvěma hodnotami (počet pixelů od levého okraje a počet od horního okraje obrázku). Oddělovačem údajů je opět středník.

Při programování všech balíčku ROS byl dodržován jednotný styl psaní kódu definovaný pro ROS[1].

[1] `http://www.ros.org/wiki/CppStyleGuide`

5.3 Průběžné testování a ladění

Při programování každé aplikace je důležité průběžné testování a ladění. ROS pro toto testování poskytuje několik výborných nástrojů.

Jednoduché ladící a informační vypisy implementují makra `ROS_INFO()`, `ROS_WARN()`, `ROS_ERROR()` a další. Tato makra zapouzdřují standardní funkci `printf()`. K výpisu zprávy pomocí těchto maker je přidán údaj o čase, který jsem několikrát využil (například pro měření rychlosti algoritmů). `ROS_INFO()` je využit pro informování uživatele o průběhu lokalizace.

Další specialitou ROSu je například program *rostopic*, díky kterému je možné snadno testovat funkčnost komunikace pomocí zpráv. Pomocí tohoto programu není nutné psát všechny komunikující uzly zároveň, protože to, jestli daný uzel přijímá nebo odesílá zprávy, lze zjistit právě za pomocí *rostopic*. Tento program například umí vypsat všechna témata, která jsou k dispozici nebo dokáže vypisovat zprávy z konkrétního tématu.

Program *rxgraph* slouží ke grafickému znázornění uzlů a spojů mezi nimi, což mi také několikrát pomohlo odhalit chybu.

5.4 Experimentování s prototypem

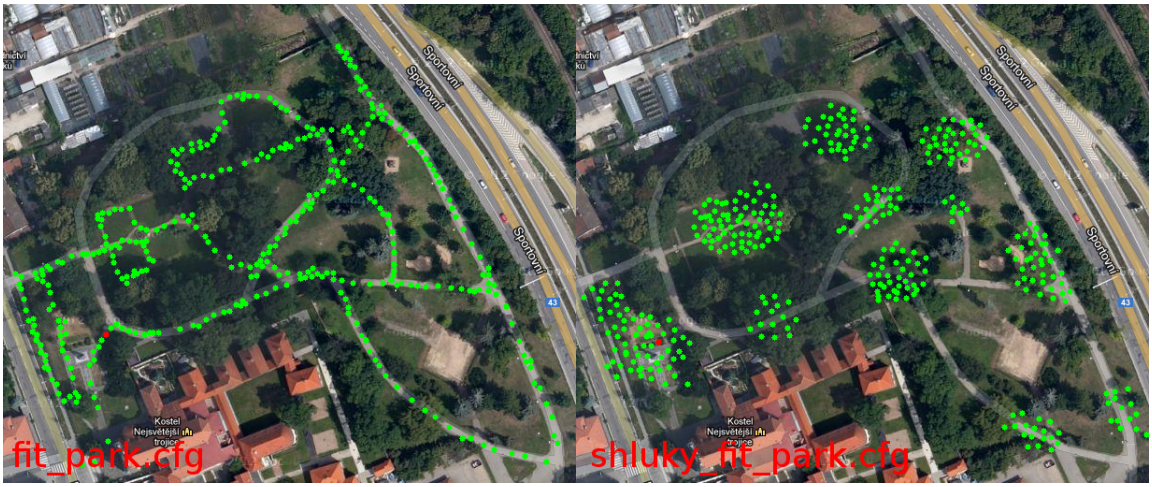
Experimenty probíhaly na přenosném počítači s operačním systémem GNU/Linux v distribuci Ubuntu 12.10 (64-bitové verzi). Procesor tohoto počítače je Intel® Core™ i5-3210M, který je taktován na frekvenci 2500MHz. Operační paměť má velikost 4GB a disk je typu SSD. Kamery byly použity dvě: webkamera (1,3 Mpx - rozlišení 1024×768px) a fotoaparát (8Mpx - rozlišení 3264×2448px).

Jako hlavní prostředí pro experimenty jsem zvolil park sousedící s budovou Fakulty informačních technologií Vysokého učení v Brně na Božetěchově ulici. Toto místo má rozlohu asi 50 000m^2, je protkáno několika cestami, které se mnohokrát kříží a poskytuje rozumnou variabilitu prostředí - obsahuje několik záchytných částí, ale i spousty takových, které jsou si velice podobné. Toto prostředí jsem zvolil záměrně i proto, že většina prací zabývajících se globální lokalizací používá městské prostředí, zatímco prostředí s převládající zelení je opomíjeno.

Park jsem několikrát prošel a během toho jsem vytvořil sadu 360 fotografií. Fotografie byly vždy vytvořeny z cesty a zachycují pohledy v nahodilých směrech. Foceny byly z ruky, takže výška od země, z které byly pořízeny, odpovídá přibližně 1,5m. Mapování jsem provedl v měsíci dubnu.

Pomocí aplikací popsaných v 5.2.4 jsem vytvořil satelitní mapu parku a umístil na ni získané fotografie dvěma způsoby. V prvním konfigurační souboru mapy *fit_park.cfg* byly umístěny na pozice, z nichž byly pořízeny. V druhém - *shluky_fit_park.cfg* - byly rozmístěny do několika shluků. Obě mapy vychází z totožné sady fotografií ve stejném pořadí. Tyto dvě mapy tedy zobrazují jednu zkušenost dvěma způsoby. Vizualizace výsledných map je zachycena na obrázku 5.1.

Mapa shluků lépe demonstruje cíl mé lokalizační metody - určení části mapy, kde se robot nachází, zatímco mapa s přesnými pozicemi ukazuje místa zvolená jako jeho nejpravděpodobnější pozice na cestě.



Obrázek 5.1: Satelitní snímky použité pro vizualizaci výsledků lokalizace. Převzato z [11].

5.4.1 Popis experimentu

V rámci experimentu je nutné spustit vždy čtyři uzly v uvedeném pořadí:

1. *pose_visualizer* - načte mapu shluků zkušeností, ale vykreslí ji až při přijetí první zpravy s odhadovanými pozicemi
2. *pose_visualizer* - načte mapu s přesnými pozicemi zkušeností, ale vykreslí ji až při přijetí první zpravy s odhadovanými pozicemi
3. *pose_estimator* - načte mapu a zkušenosti v ní a vyčkává na zprávy s obrazem
4. *image_publisher* - odesílá zprávy s obrazem k lokalizaci

```
rosrun pose_visualizer pose_visualizer mapa_shluku.cfg
rosrun pose_visualizer pose_visualizer mapa_s_pozicemi.cfg pose_visualizer2
rosrun pose_estimator pose_estimator mapa_shluku.cfg SURF SURF FlannBased
rosrun image_publisher image_publisher snimek_pro_lokalizaci.jpg
```

Ukázka 5.3: Ukázka spouštěcích příkazů pro jeden experiment.

Pro lepší sledování výsledků neprobíhá lokalizace z videosekvencí, ale z jednotlivých obrazů. Určil jsem proto testovací sadu fotek, ve které jsou zastoupeny různé typy snímků:

- Snímek pocházející přímo z mapy zkušeností.
- Snímek zmapovaného místa pořízený během podzimu.
- Snímek zmapovaného místa.

Očíslované náhledy snímků jsou zobrazeny na obrázku 5.2.

Každý snímek z testovací sady je publikován pomocí uzlu *image_publisher* a po každém obrázku jsou zaznamenány výsledky lokalizace. Výsledek lokalizace je popsán následujícím způsobem:

Obrázek 5.2: Testovací sada snímků použitá pro experimenty.

- **Počet detekovaných klíčových bodů** v testovaném snímku.
- **Počet shod klíčových bodů** - první číslo říká pro kolik klíčových bodů z testovaného snímků byl nalezen protějšek v mapě. Druhé odpovídá počtu bodů po prořezání na základě vzdáleností.
- **Vzdálenosti** shod klíčových bodů jsou uvedeny čtyři: minimální, maximální, průměrná a celková.
- **Doba lokalizace** - čas, který byl potřebný pro odhad pozice v sekundách.
- **Přesnost** - protože cílem práce není návrh metody s geometrickou přesností je určena přesnost slovně. Stupnice je určena hodnotami v následujícím pořadí: *výborná*, *dobrá*, *dostatečná*, *nedostatečná*. První ze dvou hodnot, které výsledek obsahuje, je přesnost vyhledané pozice, označené jako nejpravděpodobnější, a druhá je hodnocením zbylých pozic.

  Vychází z vizualizace pozic v obou mapách zobrazovaných pomocí uzlů *pose_visualizer* a také ze zobrazeného vykreslení shod pro nejpravděpodobnější pozici.
- **Geometrická přesnost** - jedná se o velice hrubý odhad, který je uveden spíše pro zajímavost (odchylka činní přibližně 3 metry). Je určen vzdáleností mezi správnou pozicí a odhadovanou pozicí.

  Je uvedena zvlášť pro nejpravděpodobnější pozici a zvlášť pro nejhůře odhadnutou pozici.

5.4.2 SURF

Tento experiment ověřuje lokalizaci pomocí SURF klíčových bodů. Detektor je tedy `SURF`, extraktor `SURF` a jako porovnávač byl zvolen FlannBased. U detektoru jsem nastavil práh 600. Typ porovnávače a práh detektoru jsem stanovil na základě zkušebních měření.

Mapa, kterou si při spuštění vytvořil uzel *pose_ estimator*, obsahuje 691219 klíčových bodů (jejich deskriptorů), které zabraly, spolu s vytvořeným indexem, 545MB operační paměti počítače. Vytvoření mapy zabralo 2 minuty a 40 sekund.

Výsledky tohoto experimentu jsou následující:

snímek	**klíčové body**	**shody**	**vzdálenosti**	**doba**	**přesnost**	**geom. přesnost** [$\pm 3m$]
1.	1895	1895 1895	min: 0.0000; max: 0.0000 prům: 0.0000; celk: 0.0000	$0,475s$	výborná -	$0m$ -
2.	1568	1568 817	min: 0.0794; max: 0.3850 prům: 0.2108; celk: 330.5491	$0,457s$	výborná dobrá	$3m$ $129m$
3.	1491	1491 758	min: 0.0517; max: 0.3994 prům: 0.2069; celk: 308.6125	$0,440s$	výborná nedostatečná	$5m$ $118m$
4.	1229	1229 559	min: 0.0165; max: 0.4019 prům: 0.1917; celk: 235.6231	$0,394s$	výborná dobrá	$3m$ $83m$
5.	1829	1829 927	min: 0.0695; max: 0.4218 prům: 0.2045; celk: 374.0735	$0,523s$	dobrá nedostatečná	$9m$ $92m$
6.	1197	1197 588	min: 0.0594; max: 0.5537 prům: 0.2031; celk: 243.1917	$0,406s$	nedostatečná dobrá	$189m$ $35m$
7.	731	731 361	min: 0.0598; max: 0.3844 prům: 0.2026; celk: 148.1037	$0,276s$	výborná nedostatečná	$22m$ $144m$
8.	1213	1213 571	min: 0.0463; max: 0.4284 prům: 0.2025; celk: 245.7129	$0,382s$	výborná dobrá	$3m$ $120m$
9.	2066	2066 1004	min: 0.0174; max: 0.4015 prům: 0.1986; celk: 410.5111	$0,600s$	výborná výborná	$8m$ $76m$
10.	1740	1740 871	min: 0.0437; max: 0.4235 prům: 0.2064; celk: 359.2211	$0,513s$	dostatečná dostatečná	$27m$ $67m$
11.	2316	2316 1164	min: 0.0584; max: 0.3510 prům: 0.2041; celk: 472.8997	$0,633s$	výborná dobrá	$14m$ $80m$
12.	2901	2901 1507	min: 0.0762; max: 0.4428 prům: 0.2000; celk: 580.2101	$0,758s$	výborná nedostatečná	$5m$ $78m$
13.	2425	2425 1262	min: 0.0587; max: 0.4347 prům: 0.2050; celk: 497.1610	$0,659s$	nedostatečná nedostatečná	$101m$ $167m$
14.	1498	1498 788	min: 0.0669; max: 0.4566 prům: 0.2034; celk: 304.7306	$0,478s$	dostatečná výborná	$7m$ $27m$
15.	1466	1466 706	min: 0.0294; max: 0.4220 prům: 0.2031; celk: 297.8281	$0,508s$	výborná výborná	$12m$ $15m$
16.	1767	1767 942	min: 0.0720; max: 0.4034 prům: 0.2124; celk: 375.4444	$0,519s$	nedostatečná dostatečná	$46m$ $52m$
17.	1843	1843 948	min: 0.0963; max: 0.3891 prům: 0.2104; celk: 387.8307	$0,529s$	nedostatečná nedostatečná	$53m$ $69m$
18.	1862	1862 949	min: 0.0621; max: 0.3602 prům: 0.2085; celk: 388.3531	$0,545s$	výborná nedostatečná	$5m$ $104m$
19.	1320	1320 657	min: 0.0740; max: 0.4067 prům: 0.1998; celk: 263.8462	$0,417s$	výborná dostatečná	$12m$ $81m$
20.	1972	1972 1028	min: 0.0675; max: 0.3701 prům: 0.2081; celk: 410.5324	$0,541s$	nedostatečná nedostatečná	$35m$ $124m$

První řádek tabulky ukazuje, že pokud je lokalizace prováděna na základě snímku z mapy zkušeností, není pozice odhadnuta, ale je určena se stoprocentní přesností. Vzdálenost shod klíčových bodů je v takovémto případě rovna nule ve všech případech (minimální, maximální, průměrná i celková) a množina bodů před a po prořezání je totožná. Zbylé řádky podobné hodnoty neobsahují, a proto se nikdy nejedná o snímky, které byly využity k vytvoření mapy.

Průměrná doba lokalizace je asi půl sekundy, což je skvělý výsledek na to, kolik dat se musí zpracovávat. Spojitost mezi dobou odhadu pozice a počtem klíčových bodů je logická - čím více klíčových bodů, tím náročnější je porovnávání a čas odhadu tedy vyšší. Ze získaných časových údajů lze říci, že lokalizace probíhá v reálném čase.

Přesnost navržené globální lokalizace je různá. U posledních pěti řádků je vidět značný propad přesnosti. Lze říci, že hodnoty *výborně* jsou zde minimální a pozice jsou odhadovány špatně. Těchto pět řádků odpovídá snímků, z podzimu a je na nich vidět, že se prostředí parku velice změnilo. Na stromech ještě není listí, což se projevuje větší „vzdušností“ krajiny a změnou siluet stromů. Přitom právě na siluetách stromů (a jiných částech obrazů z hranami) je detekováno a úspěšně porovnáváno nejvíce klíčových bodů. Lokalizační metoda tedy v takto dynamickém prostředí selhává.

Přesnost nejpravděpodobnějších pozic u snímků 2 až 15 hodnotím jako dostačující v rámci stanovených cílů této práce (viz 4). Tyto dobré výsledky ale ztrácí svou hodnotu v porovnání s dalšími odhadovanými pozicemi, kde se vyskytuje již mnoho velmi nepřesně odhadnutých pozic. Geometrická přesnost, i přes to že se jedná o velice orientační položku, často koreluje se slovními pozicemi.

Z výsledků plyne, že má-li snímek minimální a maximální vzdálenost shod klíčových bodů nízkou, lze predikovat, že výsledek bude dobrý. Celková vzdálenost nemá u těchto výsledků žádnou spojitost s přesností.

5.4.3 SIFT

Druhý experiment využívá SIFT klíčových bodů. Detektor je tedy `SIFT`, extraktor `SIFT` a jako porovnávač byl zvolen FlannBased. Parametry detektoru a extraktoru byly ponechány tak, jak je definuje knihovna OpenCV.

Mapa, kterou si při spuštění vytvořil uzel *pose_estimator*, obsahuje 1365162 klíčových bodů (jejich deskriptorů), které zabraly, spolu s vytvořeným indexem, 1,7GB operační paměti počítače. Vytvoření mapy zabralo 3 minuty a 31 sekund.

Výsledky tohoto experimentu jsou následující:

snímek	klíčové body	shody	vzdálenosti	doba	přesnost	geom. přesnost [$\pm 3m$]
1.	3966	3966 3966	min: 0.0000; max: 0.0000 prům: 0.0000; celk: 0.0000	$0,676s$	výborná -	$0m$ -
2.	3212	3212 1338	min: 67.2681; max: 379.5668 prům: 254.6411; celk: 817907.4716	$0,733s$	výborná dobrá	$3m$ $184m$
3.	3000	3000 1213	min: 51.1859; max: 361.3640 prům: 258.9310; celk: 776793.0948	$0,710s$	výborná nedostatečná	$5m$ $72m$
4.	3161	3161 1198	min: 24.7991; max: 381.6032 prům: 247.1051; celk: 781099.3652	$0,725s$	výborná výborná	$3m$ $176m$
5.	3611	3611 1509	min: 62.3778; max: 363.4377 prům: 258.0708; celk: 931893.7466	$0,791s$	výborná nedostatečná	$9m$ $100m$
6.	2106	2106 944	min: 54.9090; max: 380.2459 prům: 230.5149; celk: 485464.4616	$0,593s$	výborná dostatečná	$8m$ $123m$
7.	1932	1932 752	min: 51.0392; max: 352.8512 prům: 255.1496; celk: 492949.0467	$0,573s$	výborná dostatečná	$22m$ $107m$
8.	2698	2698 1039	min: 33.9852; max: 365.8537 prům: 251.4226; celk: 678338.2355	$0,670s$	výborná výborná	$3m$ $61m$
9..	4171	4171 1651	min: 57.4978; max: 370.2593 prům: 251.9497; celk: 1050882.2761	$0,864s$	výborná výborná	$8m$ $10m$
10.	3473	3473 1459	min: 50.4579; max: 375.6820 prům: 253.8174; celk: 881507.9531	$0,776s$	dobrá dobrá	$3m$ $52m$
11.	4374	4374 1848	min: 62.0966; max: 371.3879 prům: 255.9034; celk: 1119321.6003	$0,887s$	výborná výborná	$14m$ $46m$
12.	5347	5347 2243	min: 48.0416; max: 381.2400 prům: 254.7589; celk: 1362196.1169	$0,994s$	výborná nedostatečná	$5m$ $73m$
13.	4814	4814 2003	min: 38.7814; max: 380.4812 prům: 265.0379; celk: 1275892.6841	$0,938s$	výborná dostatečná	$11m$ $83m$
14.	3231	3231 1307	min: 55.9374; max: 378.1335 prům: 253.4777; celk: 818986.5359	$0,733s$	výborná výborná	$23m$ $84m$
15.	2102	2102 862	min: 37.5765; max: 361.7830 prům: 230.8668; celk: 485282.0455	$0,604s$	výborná výborná	$12m$ $15m$
16.	3457	3457 1460	min: 69.1158; max: 365.4709 prům: 253.1118; celk: 875007.7783	$0,776s$	nedostatečná nedostatečná	$138m$ $167m$
17.	2929	2929 1269	min: 69.9571; max: 360.9709 prům: 250.7646; celk: 734489.5801	$0,705s$	nedostatečná nedostatečná	$52m$ $84m$
18.	3657	3657 1598	min: 43.8748; max: 381.9672 prům: 251.6761; celk: 920379.5000	$0,795s$	výborná nedostatečná	$5m$ $106m$
19.	2837	2837 1129	min: 25.9615; max: 369.9770 prům: 240.4378; celk: 682122.1341	$0,749s$	výborná výborná	$5m$ $24m$
20.	3495	3495 1611	min: 59.9332; max: 379.8394 prům: 249.6390; celk: 872488.3471	$0,771s$	nedostatečná nedostatečná	$69m$ $115m$

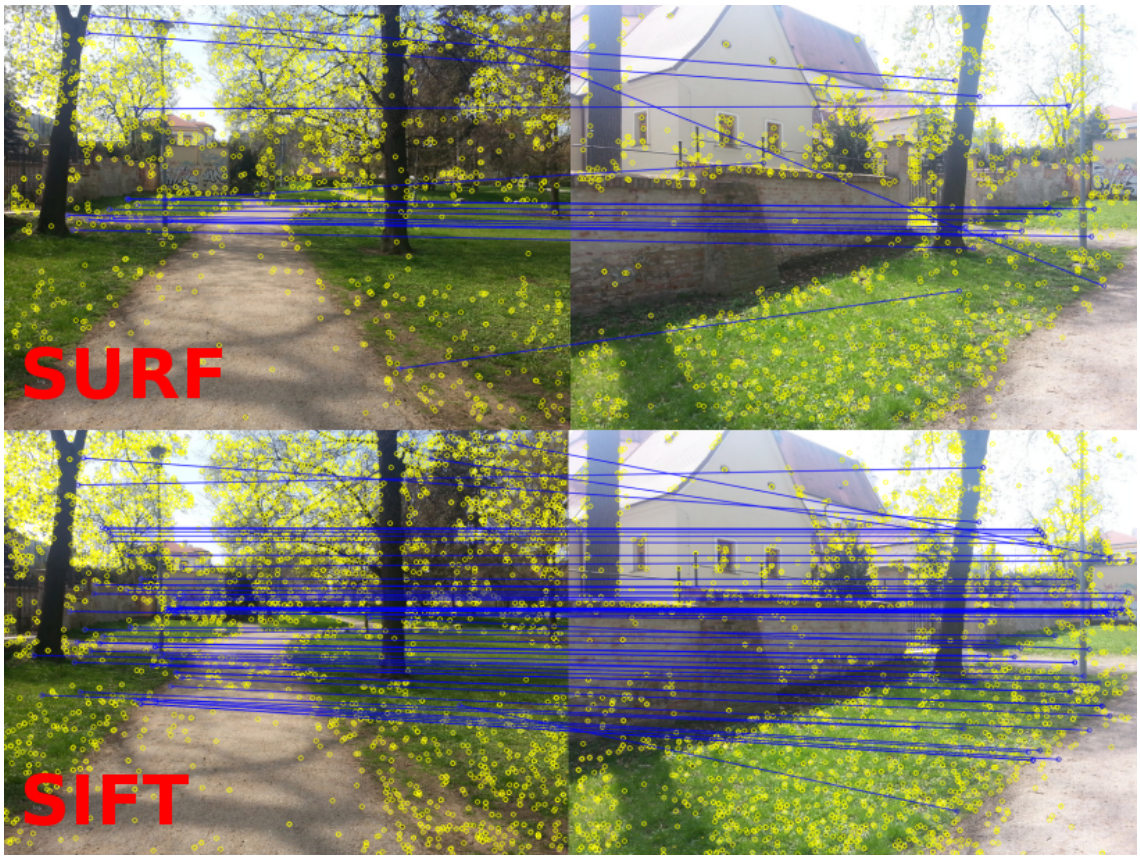
Již při inicializaci mapy si lze všimnout diametrálních rozdílů mezi SURF a SIFT klíčovými body. Detekovaných SIFT bodů je skoro dvojnásobný počet a zabírají třikrát více operační paměti. U SURF detektoru sice lze snížit práh a tím získat více bodů, nicméně takto získané body navíc nemají proti SIFT bodům dostatečnou kvalitu. S počtem detekovaných bodů souvisí i delší čas potřebný pro vytvoření mapy.

Doby trvání lokalizace jsou proti předchozímu experimentu delší, ale stále se pohybují pod jednou sekundou. Rozdíl je dán pouze vyšším počtem porovnávaných bodů, což lze zjistit při porovnání času u snímku 7 této tabulky a snímku 20 u tabulky pro SURF.

Přesnost lokalizace pomocí SIFT klíčových bodů je lepší než pomocí SURF bodů skoro ve všech případech, ať už se jedná o nejpravděpodobnější pozice, ostatní odhadované pozice nebo lokalizaci v dynamickém prostředí (snímky 16 až 19). Geometrická přesnost dokonce

předčila mé očekávání. I v takto dobrých výsledcích ale bylo mnoho chyb, obvzláště v dynamickém prostředí, které by v reálném provozu robota mohly způsobit komplikace. Navíc paměťová náročnost mapy je tak vysoká, že omezuje běh dalších aplikací na použitém počítači.

Z vykreslovaných shod mezi lokalizovaným snímkem a nejlépe odpovídajícím snímkem z mapy je možné dobře vidět rozdíl mezi místy, kde jsou detekovány body SURF a kde SIFT. I přesto, že SURF by měl být invariantní vůči rozměrům objektů v obrázku, nelze vidět takové výsledky jako u SIFT (viz obrázek 5.3). Pro zvolené prostředí je tedy tato invariantnost důležitější než rychlost detekce bodů.



Obrázek 5.3: Porovnání mezi shodami klíčových bodů při využití SIFT a SURF.

5.4.4 ORB

Poslední experiment ověřoval možnosti lokalizace při využití ORB klíčových bodů. Detektor je tedy `ORB`, extraktor `ORB` a jako porovnávač byl zvolen **BruteForce-Hamming(2)**. Použití FLANN porovnávače není pro ORB podporováno.

Mapa, kterou si při spuštění vytvořil uzel *pose_ estimator*, obsahuje 181000 klíčových bodů (jejich deskriptorů), které zabraly spolu s vytvořeným indexem 32MB operační paměti počítače. Vytvoření mapy zabralo 13 sekund.

Výsledky tohoto experimentu jsou následující:

snímek	klíčové body	shody	vzdálenosti	doba	přesnost	geom. přesnost [$\pm 3m$]
1.	500	500 500	min: 0.000; max: 0.000 prům: 0.000; celk: 0.000	2.364s	výborná -	0m -
2.	500	500 227	min: 20.000; max: 59.000 prům: 43.888; celk: 21944.000	2.426s	výborná nedostatečná	3m 230m
3.	500	500 232	min: 18.000; max: 57.000 prům: 41.584; celk: 20792.000	2.405s	výborna dostatečná	5m 107m
4.	500	500 210	min: 7.000; max: 62.000 prům: 39.954; celk: 19977.000	2.459s	výborná dostatečná	3m 276m
5.	500	500 231	min: 23.000; max: 63.000 prům: 44.634; celk: 22317.000	2.418s	dobrá nedostatečná	18m 121m
6.	500	500 210	min: 26.000; max: 59.000 prům: 41.898; celk: 20949.000	2.422s	nedostatečná dostatečná	81m 89m
7.	500	500 260	min: 22.000; max: 62.000 prům: 45.154; celk: 22577.000	2.486s	výborna nedostatečná	30m 166m
8.	500	500 216	min: 12.000; max: 60.000 prům: 41.602; celk: 20801.000	2.417s	výborná dobrá	3m 126m
9.	500	500 239	min: 12.000; max: 58.000 prům: 39.358; celk: 19679.000	2.453s	výborná výborna	12m 18m
10.	500	500 252	min: 24.000; max: 62.000 prům: 44.358; celk: 22179.000	2.445s	nedostatečná dostatečná	124m 138m
11.	500	500 240	min: 17.000; max: 60.000 prům: 45.170; celk: 22585.000	2.418s	výborná výborná	14m 38m
12.	500	500 234	min: 22.000; max: 59.000 prům: 43.206; celk: 21603.000	2.493s	nedostatečná nedostatečná	100m 110m
13.	500	500 251	min: 17.000; max: 60.000 prům: 45.348; celk: 22674.000	2.412s	nedostatečná nedostatečná	156m 172m
14.	500	500 246	min: 26.000; max: 61.000 prům: 43.710; celk: 21855.000	2.443s	nedostatečná nedostatečná	176m 192m
15.	500	500 244	min: 20.000; max: 60.000 prům: 41.600; celk: 20800.000	2.406s	výborná výborná	12m 15m
16.	500	500 240	min: 22.000; max: 61.000 prům: 44.528; celk: 22264.000	2.424s	nedostatečná nedostatečná	46m 129m
17.	500	500 243	min: 24.000; max: 60.000 prům: 44.048; celk: 22024.000	2.427s	nedostatečná nedostatečná	92m 92m
18.	500	500 234	min: 12.000; max: 58.000 prům: 43.558; celk: 21779.000	2.426s	výborná nedostatečná	6m 90m
19.	500	500 254	min: 18.000; max: 59.000 prům: 43.184; celk: 21592.000	2.445s	dobrá nedosatečná	18m 156m
20.	500	500 248	min: 21.000; max: 59.000 prům: 43.366; celk: 21683.000	2.464s	nedostatečná nedostatečná	69m 97m

Hned první nedostatek ORB klíčových bodů je absence rychlého porovnávače. Použitý BruteForce-Hamming(2) dosahoval sice nejlepších časů ze všech nabízených algoritmů (výčet v 3.5), ale doba lokalizace přes dvě sekundy je příliš dlouhá. Kvůli tomu musel být snížen počet detekovaných bodů, protože při osmistech klíčových bodů trval lokalizační proces přes deset sekund. Detekce ORB bodů a jejich deskriptorů je přitom rychlejší než u SURF a SIFT bodů.

Vzdálenosti shod klíčových bodů ukazují, že jsou spíše neodpovídající slovní přesnosti, kromě minima, které lze částečně brát jako měřítko kvality lokalizace.

Výsledky u lokalizace za využití ORB bodů jsou špatné. Přesnost je dobrá jen u snímků, které jsou z více jak 80% totožné s některým ze snímků z mapy zkušeností.

5.4.5 Vyhodnocení experimentů

Nejlepší výsledky se podařilo dosáhnout pomocí detekce a extrakce SIFT klíčových bodů a jejich porovnání pomocí FLANN algoritmu. Doba potřebná pro lokalizaci je vynikající ale paměťová náročnost nikoliv. Výhodou využití SIFT bodů je jejich přesnost nejen u nejpravděpodobnější shody, ale i u dále odhadovaných pozic. První (nejpravděpodobnější) pozice bývá velice přesná hlavně díky dobré shodnosti s některým ze zmapovaných snímků, což značí že počet a typ snímků mapy byl vhodně zvolen. Lokalizace pomocí SIFT lze tedy reálně nasadit v případech, kdy je mapa dostatečně aktuální. Lidé, auta a jiné drobné změny ve snímcích měli na kvalitu minimální vliv.

Lokalizace za pomocí SURF bodů nedosahuje takových výsledků jako za pomocí SIFT bodů, ale v rámci ušetřené operační paměti by bylo možné zařadit více snímků do mapy zkušeností, což by se pozitivně projevilo na přesnosti. Protože rychlost zpracování je výborná, je také možné pořídit více snímků (v různých směrech), statisticky zpracovat odhady ze všech těchto snímků a tak získat lepší odhad aktuální pozice.

Využití ORB bodů se pro tuto metodu neosvědčilo. Lokalizace selhávala jak u snímků, kde bylo prostředí zatíženo změnou, tak i u snímků z plně aktuálního prostředí.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat lokalizační metodu pracující pouze z daty získanými z monokulární kamery. V rámci práce na tomto zadání jsem nastudoval problematiku detekce příznaků v obrazu, rozdíly mezi kamerami s různou šířkou obrazu, strukturu meta-operačního systému ROS a několik již existujících řešení. Získané informace jsem poté využil při návrhu vlastní jednoduché lokalizační metody, jejíž cílem je globální lokalizace v předem známé mapě, konkrétně určením přibližné oblasti v ní, kde se robot nachází. Přesto, že mnou navržená metoda globální lokalizace je založena na opravdu jednoduchém principu porovnávání obrazů v mapě zkušeností, nenašel jsem žádnou publikaci, která by tento princip popisovala a experimentovala s ním.

Navrženou metodu jsem implementoval jako balíček ROS. Spolu s ní jsem implementoval i nástroje pro testování a vizualizaci výsledků mé práce, které lze využít mimo tuto práci, jelikož se jedná o samostatné balíčky pro ROS.

S implementovaným prototypem jsem provedl několik experimentů, které se zaměřují hlavně na porovnání různých typů klíčových bodů a jejich využitelnost pro navrženou metodu. Výsledky jsem vyhodnotil a zjistil, že lokalizace pouze pomocí levné kamery (bez nutnosti kalibrace) a za využití běžného přenosného počítače je možná, ale ne ve všech případech. Tyto případy jsem popsal. Tato metoda nemá sloužit jako výhradní způsob lokalizace, ale jako určení globální pozice robota, na kterou poté může navázat například lokalizace pomocí sledování pozice. Tyto metody dále můžou fungovat společně a s vyšší přesností.

Mou práci lze využít jako dobrý základ pro další bádání v cílové oblasti. Možnosti rozšíření jsou zde opravdu široké. Kupříkladu by bylo možné vyzkoušet reprezentaci zkušeností za pomoci bag-of-words modelu nebo použít kompas jako další senzor a omezit tím množství prohledávaných zkušenosti. Možné by také bylo zpřesnit geometrickou přesnost na základě rekonstrukce hloubky ze dvou a více obrazů odhadnutých pozic, což by však vyžadovalo podrobnější mapu s více snímky.

Literatura

[1] A. C. Murillo, J. J. G.; SAGUÉS, C.: SURF features for efficient robot localization with omnidirectional images. In *ICRA'07*, 2007, s. 3901–3907.

[2] BAT, H.; ESS, A.; TUYTELAARS, T.; aj.: Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, ročník 110, č. 3, 2008: s. 346–359, ISSN 1077-3142.

[3] BJELKA, J.: *Návrh a realizace metody mapování okolí pro mobilní roboty*. Diplomová práce, FSI VUT v Brně, 2007.

[4] BOUCHARD, S.: Robotiq [online]. 2011-08-11 [cit. 2013-05-06].
URL http://blog.robotiq.com/bid/40428/Using-The-Kinect-For-Robotic-Manipulation

[5] BÍLEK, P.: *Významné body v obraze: detekce, korespondence a lokalizace ve 3D*. Diplomová práce, Fakulta elektrotechnická ČVUT v Praze, 2007.

[6] contributors, W.: Feature detection (computer vision) [online]. 2013-04-06 [cit. 2013-05-06].
URL http://en.wikipedia.org/wiki/Feature_detection_(computer_vision)

[7] contributors, W.: Ominidirectional camera [online]. 2013-04-06 [cit. 2013-05-06].
URL http://en.wikipedia.org/wiki/Omnidirectional_camera

[8] Documentation: Documentation - ROS Wiki.
URL http://www.ros.org

[9] Eric ROYER, M. L.; DHOME, M.; LAVEST, J.-M.: Monocular vision for mobile robot localization and autonomous navigation. *JOURNAL OF COMPUTER VISION*, ročník 74, č. 3, 2007: s. 237–260.

[10] FERRARI, V.; JURIE, F.; SCHMID, C.: From Images to Shape Models for Object Detection. *International Journal of Computer Vision*, ročník 87, č. 3, 2010: s. 284–303, ISSN 0920-5691.

[11] Google mapy [online]. 2013-05-06 [cit. 2013-05-06].
URL https://maps.google.cz/

[12] HUITL, R.; SCHROTH, G.; HILSENBECK, S.; aj.: Virtual reference view generation for CBIR-based visual pose estimation. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, 2012, ISBN 978-1-4503-1089-5, s. 993–996.

[13] JUAN, L.; GWON, O.: A Comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing (IJIP)*, ročník 3, č. 4, 2009: s. 143–152.
URL http://www.cscjournals.org/csc/manuscript/Journals/IJIP/volume3/Issue4/IJIP-51.pdf

[14] MATUSZEK, M.: *Automatické třídění fotografií dle obsahu*. Diplomová práce, FIT VUT v Brně, 2010.

[15] Michiel HOLTKAMP, S. d. J.: *Robot Localisation Using SIFT and Active Monocular Vision*. Diplomová práce, Department of AI, University of Groningen, 2006.

[16] PRASSER, D.; MILFORD, M.; WYETH, G.: Outdoor Simultaneous Localisation and Mapping using RatSLAM. In *International Conference on Field and Service Robots*, 2005.

[17] ROZMAN, J.: *Navigace mobilních robotů*. Dizertační práce, FIT VUT v Brně, 2011.

[18] RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; aj.: ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, ISSN 1550-5499, s. 2564–2571.

[19] SCARAMUZZA, D.; SIEGWART, R.: Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. *Robotics, IEEE Transactions on*, ročník 24, č. 5, 2008: s. 1015–1026, ISSN 1552-3098.

[20] SHEN, Y.; LIU, J.: *Vision based SLAM for Robot Navigation with Single Camera*. Diplomová práce, Department of Information Science and Electronic Engineering, Zhejiang University.

[21] SICILIANO, B.; KHATIB, O.: *Springer Handbook of Robotics*. Gale virtual reference library, Springer, 2008, ISBN 9783540239574.

[22] SKALKA, M.: *Srovnání lokalizačních technik*. Diplomová práce, Matematicko-fyzikální fakulta UK v Praze, 2011.

[23] THRUN, S.; BURGARD, W.; FOX, D.: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, ISBN 0262201623.

[24] WWW stránky: OpenSLAM.
URL http://openslam.org