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 KVADROKOPTÉRY V 3D PROSTORU POMOCÍ JEDNÉ KAMERY

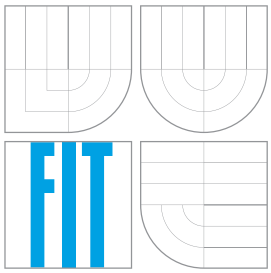
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR KUBICA

BRNO 2015

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 KVADROKOPTÉRY V 3D PROSTORU POMOCÍ JEDNÉ KAMERY

QUADCOPTER LOCALIZATION IN 3D SPACE BASED ON MONO-CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR KUBICA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2015

Abstrakt

V této bakalářské práci jsou prozkoumány možnosti existujících metod pro určení polohy UAV AscTec Pelican za pomoci monokulární kamery. Jsou testovány metody SVO, PTAM a LSD-SLAM ve vnitřních i venkovních prostorách za použití různých kamer, objektivů a různě výkonného hardware. Na základě výstupů testů jsou doporučeny postupy pro získání co nejlepších možných výsledků.

Abstract

This bachelor thesis focuses on the possibilities of existing methods for locating the UAV AscTec Pelican using a monocular camera. The methods SVO, PTAM and LSD-SLAM were tested in both inner and outer environments using various cameras, lenses and hardware with different performance. Based on the experience gained during the testing, guidelines on how to achieve the best results are made.

Klíčová slova

SVO, PTAM, LSD-SLAM, SLAM, AscTec Pelican, Testování metod, ROS, uEye, kalibrace kamery

Keywords

SVO, PTAM, LSD-SLAM, SLAM, AscTec Pelican, Testing methods, ROS, uEye, camera calibration

Citace

Petr Kubica: Lokalizace kvadrokoptéry v 3D prostoru pomocí jedné kamery, bakalářská práce, Brno, FIT VUT v Brně, 2015

Lokalizace kvadrokoptéry v 3D prostoru pomocí jedné kamery

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana Ph.D.

.....
Petr Kubica
20. května 2015

Poděkování

Na tomto místě bych rád velmi poděkoval vedoucímu mé bakalářské práce, panu Ing. Vítězslavovi Beranovi Ph.D., za jeho cenné rady, doporučení a směřování v průběhu jejího vypracování.

Obsah

Seznam obrázků

Úvod

V dnešní době je k dispozici velké množství způsobů, jak lokalizovat UAV[1] v prostoru. Mezi ně patří například lokalizace pomocí GPS signálu, získávání poznatků o okolí pomocí sonarů nebo pomocí obrazu pořízeného kamerou. Vzhledem k dnešnímu vysokému výkonu miniaturních zařízení se právě lokalizace v prostoru pomocí obrazu těší vysoké oblibě. Na základě posuvu kamery vůči prostoru lze detekovat pohyb zařízení pomocí zpětné projekce objektů zaznamenaných sousedními snímky. Takto získané informace o pohybu se pak mohou projevit při ovládání robota, například v situacích, kdy je třeba daného robota navádět po určité trajektorii nebo jej lokalizovat v místech, kde není dostupný GPS signál. Postup, kdy se pořízená obrazová data z kamery po zpracování metodami v reálném čase projevují na řízení robota, se nazývá vizuální odometrie.

Metody se člení podle typu robota, na kterém mají běžet. Trochu odlišným způsobem fungují metody pro UAV a pro pozemní roboty. U UAV se lokalizace na základě obrazu provádí hůře než je tomu například u jiných robotů. Důvodem je velká vzdálenost objektů od kamery, která většinou také snímá předmět zájmu uživatele.

Tato bakalářská práce si klade za cíl prozkoumat možnosti existujících metod pro lokalizaci UAV v různých podmínkách jako je jiný výpočetní výkon různých zařízení, vliv prostředí na kvalitu funkčnosti metod nebo také nastavení daných metod a kamery.

Pro výzkum a testování jsem měl k dispozici kvadrokoptéru AscTec Pelican od firmy Ascending Technologies GmbH, technicky označenou jménem Tyra. Jedná se o kvadrokoptéru, která patří mezi profesionální modely a je velmi vhodná právě pro výzkumné účely.

[1] Unmanned aerial vehicle http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

Kapitola 1

Polohování kamery v prostoru

Tato kapitola popisuje nezbytné informace k pochopení dané problematiky. Jsou zde představeny základní principy pro vizuální odometrii, moderní metody pro polohování kamery v prostoru, popřípadě mapovací metody a architektura kvadrokoptéry, na níž se daný výzkum bude provádět. Základní znalost představuje pochopení principům pro lokalizaci kamery v prostoru, možnosti kamer a platformy ROS včetně dostupných balíků, na kterém tento systém principielně pracuje.

1.1 Odometrie

Vlastní slovo odometrie se skládá z řeckých slov hodos (cesta) a metron (měřit) [22].

Odometrie je proces, kdy se data z různých senzorů vyhodnocují a poté se projeví například v pohybu robota. Stav, který je výsledkem těchto údajů, lze následně pomocí zpětnovazební smyčky považovat za další vstupní data. V případě UAV zařízení se kupříkladu může jednat o pozici a směru pohybu. Mezi senzory snímající polohu a stav zařízení může patřit například elektromagnetický kompas, ultrazvukové senzory pracující na bázi odrazu od přepážky nebo také za pomoci infračervených čidel.

Jedna z využívaných metod pracuje pomoci ultrazvukových sonarů, kdy jsou vysílány ultrazvukové pulzy do okolí a následně jsou snímány jejich odrazy. Tato metoda se hodí převážně pro pohyb v uzavřeném prostoru, kde se překážky vyskytují relativně blízko. Přesnost snímání je v řádech centimetrů. Další výhodou této metody je například také identifikace překážek s tím, že se robot drží v bezpečné vzdálenosti. Tento postup se hojně využívá u běžných pozemních robotů.

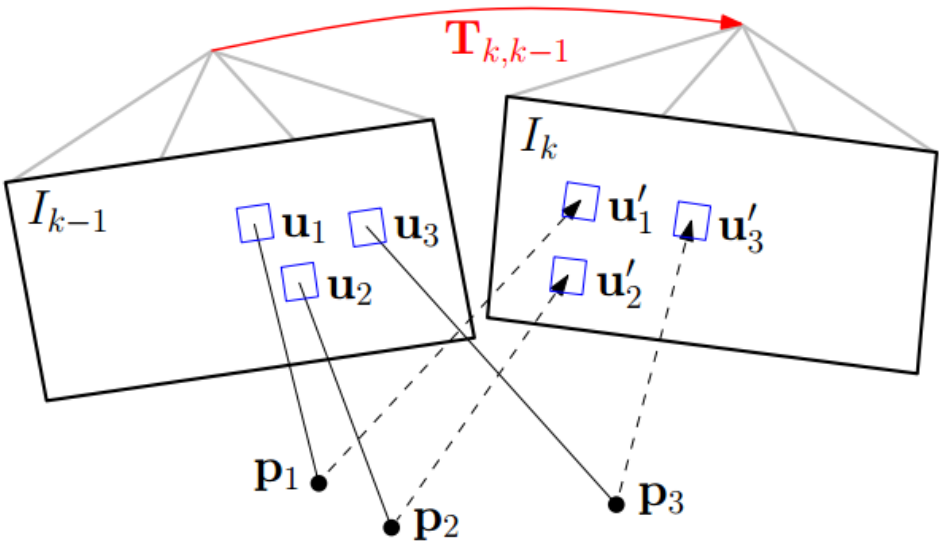
V případě UAV zařízení se tato metoda nepoužívá pro lokalizaci, neboť ve venkovním prostoru se signál ze senzorů nemá od čeho odrazit. Také při pohybu ve větších vzdálenostech od překážek, kterými mohou být například budovy, může docházet k nepřesnostem. V dnešním světě, kde spousta zařízení již mezi sebou komunikují pomocí elektromagnetických vln, může docházet k rušení.

Jednou z posledních možností, která se vyskytuje u moderních robotů, je využití obrazových dat ke svému pohybu.

Vizuální odometrie

Za pomocí kamerou pořízených obrazových dat lze po aplikaci lokačních metod získat údaje o snímaném prostoru. Základním principem je zde rozdíl mezi dvěma snímky [19] (obr. 1.1),

díky kterým robot získá stereoskopickou představu týkající se snímaného okolí a umožní tak detekovat případný pohyb kamery. Na tomto principu pak dokáže také vyhodnotit vzdálenosti od překážek ve snímaném prostoru. Většina algoritmů k detekci používá hrany daných předmětů, kde se mění barvy, případně kontrast.



Obrázek 1.1: Porovnání 3D bodů (u1, u2 a u3) dvou snímků (ik-1 a ik) pro určení posunu snímacího zařízení. Zdroj: [2]

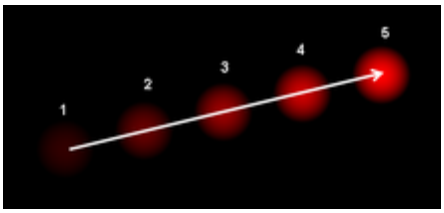
Pseudo-algoritmus vizuální odometrie

Většina stávajících přístupů k algoritmu vizuální odometrie je založena na následujících etapách [19]:

1. Získání vstupních dat pomocí monokulární, stereo nebo všesměrové kamery.
2. Korekce pomocí technik pro zpracování obrazu, odstranění zkreslení objektivu.
3. Detekce důležitých 3D bodů a jejich porovnání mezi snímky, tvorba optického toku:
   (a) Za pomocí korelace se detekují souvislosti mezi dvěma snímky. (Tyto souvislosti nejsou sledovány přes všechny snímky.)
   (b) Extrakce těchto souvislostí.
   (c) Tvorba optického toku za pomocí různých metod, z nichž nejznámější: Lucas-Kanade metoda.
4. Kontrola optického toku pro případné chyby, odstranění odlehlých hodnot.
5. Odhad pohybu kamery na základě optickému toku:
   (a) Za pomocí Kalmanova filtru.
   (b) Za pomocí nalezení geometrických a 3D vlastností bodů, které minimalizují chybu v re-projekci mezi dvěma snímky.

Optický tok

Optický tok [20] (anglicky optical flow) je vzor zdánlivého pohybu předmětů, povrchů a hran ve vizuální scéně způsobeného relativním pohybem mezi pozorovatelem (kamerou) a scénou. V robotice se používá v mnoha odvětvích jako je například detekce objektů, detekce pohybu včetně navigace robota. Informace obsažené v optickém toku slouží k určení posuvu kamery vůči scéně. Princip spočívá ve zpracování více snímků a odhadu pohybu objektů ve scéně. Tento jev je znázorněn na obrázku 1.2, který ukazuje záznam pěti pořízených snímků pohybující se koule vložených do jednoho obrázku. Snímače optického toku mohou být různé - od kamer až po různé senzory, jako například senzor ke snímání pohybu u optické myši.



Obrázek 1.2: Vektor optického toku pohybujícího se objektu. Zdroj: [19]

Lucas-Kanade metoda

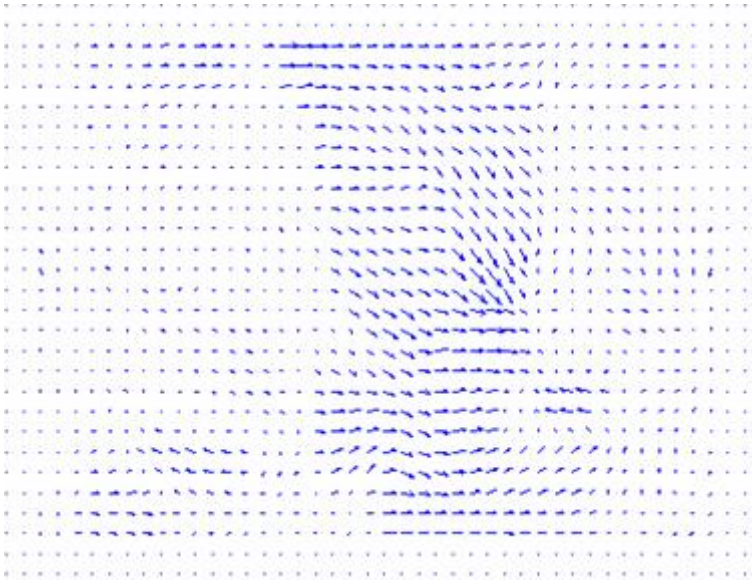
Jedna z diferenciálních metod pro výpočet optického toku pochází od Bruce D. Lucase a Takeo Kanade. Předpokládá se, že světelné složky po sobě jdoucích snímků (obr. 1.3 jsou konstantní. Rovnice 1.1 jednoho pixelu v sobě zahrnuje dvě neznámé vektoru u a v.

$$O = I_t(p_i) + Ip_i[uv] \tag{1.1}$$

Obrázek 1.3: Dva po sobě jdoucí snímky. Při bližším zkoumání je vidět pohyb kamery směrem doprava. Zdroj: [14]

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_n) \end{bmatrix} \tag{1.2}$$

Metoda vychází z principu, že okolní pixely budou mít ten samý vektor posuvu. Z toho vyplývá rovnice 1.2 pro výpočet vektoru n x n pixelů, která se pak minimalizuje metodou nejmenších čtverců. Výsledkem je pak matice vektorů posuvu bodů v obraze (obr. 1.4) [3].

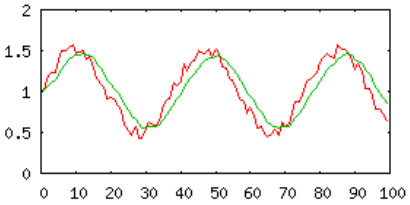


Obrázek 1.4: Vektory posunu vypočtené metodou Lucas-Kanade ze snímků 1.3. Zdroj: [14]

Kalmanův filtr

Kalmanův filtr se používá k modelování dynamického diskrétního systému. Ve velké míře se používá hlavně pro trasování objektů. Umožňuje predikovat polohu na základě předchozích stavů a souběžně ji upřesňovat pomocí měření. Získanou informací z Kalmanova filtru je trajektorie.

Kalmanův filtr se také obecně používá pro takzvaný plovoucí průměr [21] (obr. 1.5) kdy jedna naměřená hodnota, která se vyznačuje velkou nepřesností, neuvede systém do jiného stavu. Úplně však tuto naměřenou hodnotu ze systému nevyloučí. Tento filtr pracuje na bázi pravděpodobnosti stavů, ve kterých se dané hodnoty mohou nacházet.



Obrázek 1.5: Měření hodnoty a její korekce Kalmanovým filtrem. Zdroj: [21]

1.2 Typy polohovacích metod

Metody se dají členit podle způsobu přístupu k výpočtu pohybu kamery vůči scéně. Z tohoto pohledu můžeme rozlišovat metody přímé a metody na bázi extrakci bodů či hran.

Metody založené na extrakci bodů

Standardní přístupje extrakce 3D bodů z každého snímku a následné vzájemné přiřazení k bodům z následujícího snímku pomocí invariantního deskriptoru. Finálně se pak určení po-

hybu kamery zlepšuje skrze minimalizaci reprojekční chyby. Většina těchto metod používá výše zmíněný přístup nezávisle na použitých optimalizačních prostředcích. Důvodem úspěchu tohoto přístupu je rychlost detektoru pro extrakci bodů a invariantního deskriptoru, které umožňují mapování bodů i při větším rozdílu mezi dvěma po sobě jdoucími snímky.

Nevýhodou této techniky je spoléhání se na detekci potřebných údajů ze snímku. Vzhledem k tomu, že spousta detektorů je optimalizována spíše pro rychlost než pro přesnost je třeba provádět neustálá měření [2] [5].

Přímé metody

Přímé metodyurčují pohyb pomocí hodnot intenzity ve snímku. Na základě tohoto principu tyto metody dokážou využít všechny informace, které snímek nabízí a tím dokáží ušetřit čas pro detekování bodů. Bylo dokázáno, že předčí metody založené na extrakci bodů z hlediska použití s nízkým rozlišením textury nebo při rozostření či rychlejším posunu kamery. Výpočet fotometrické chyby je zde důležitější než je chyba v reprojekci, neboť jsou zde zahrnuty deformace a integrují se velké obrazové oblasti [2] [5].

Optimimalizace pozičního grafu

Jedná se o známou SLAM techniku pro výstavbu konzistentní globální mapy. Svět je reprezentován množinou klíčových snímků spojených pozičním omezením do struktury grafu, který může být optimalizován pomocí obecného nástroje pro optimalizaci grafu, jako je například nástroj g2o [5].

1.3 PTAM - Parallel Tracking and Mapping

PTAM [11] byl poprvé představen na Oxfordské univerzitě panem Georgem Kleinem a Davidem Murrayem v roce 2007. Metoda byla původně určená pro zjišťování polohy kamery v neznámém prostředí a možnost zobrazování vizualizovaných 3D objektů. Příkladem může být počítačově vytvořený 3D automobil jezdící po desce stolu.

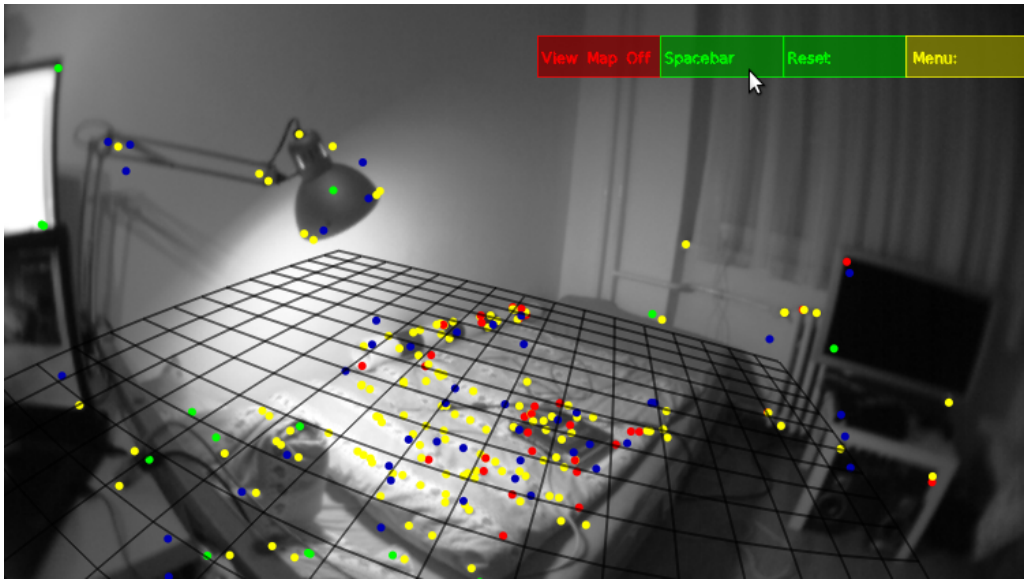
Tato metoda byla následně přejata pro systém ROS v podobě balíku `ethzasl_ptam`. S přídavným balíčkem `ethzasl_sensor_fusion` je možné propojení `asctec_mav_frameworku`, který slouží pro ovládání HL procesoru autopilota AscTec Pelican.

PTAM (obr. 1.6) je založen na detekci bodů a pro svůj výpočet současně mapuje v obraze pohyb několika tisíců 3D bodů. Metoda je primárně určena pro kancelářské použití. Pokud bychom uvažovali o využití této metody ve venkovních podmínkách. je třeba provést vícenásobnou modifikaci parametrů metody.

Hlavní podstata metody PTAM:

- Sledování 3D bodů a mapování je zde rozděleno do dvou paralelně běžících vláken.
- Mapování je založeno na klíčových snímcích.
- Mapa je hloubkově inicializována ze stereo páru (5-Point Algorithm[1]).
- Nové 3D body jsou inicializovány díky epipolárnímu[2] vyhledávání.
- V obraze je mapovaná spousta 3D bodů (několik tisíc).

[1] http://users.cecs.anu.edu.au/~hongdong/new5pt_cameraREady_ver_1.pdf
[2] http://en.wikipedia.org/wiki/Epipolar_geometry



Obrázek 1.6: Lokalizace pomocí metody PTAM v prostoru.

Projekt je napsán v jazyce C++ s použitím libCVD[3] a TooN[4] knihoven. Projekt nebyl vysoce optimalizován ale i přesto jsou použity některé z optimalizací, jako je například řádková vyhledávací tabulka použitá pro urychlení přístupu k poli v pyramidovém schématu.

Pro modifikaci ve venkovním prostředí je potřeba změnit některé parametry metody a tím docílit lehce odlišného chování ve velkých prostorách.

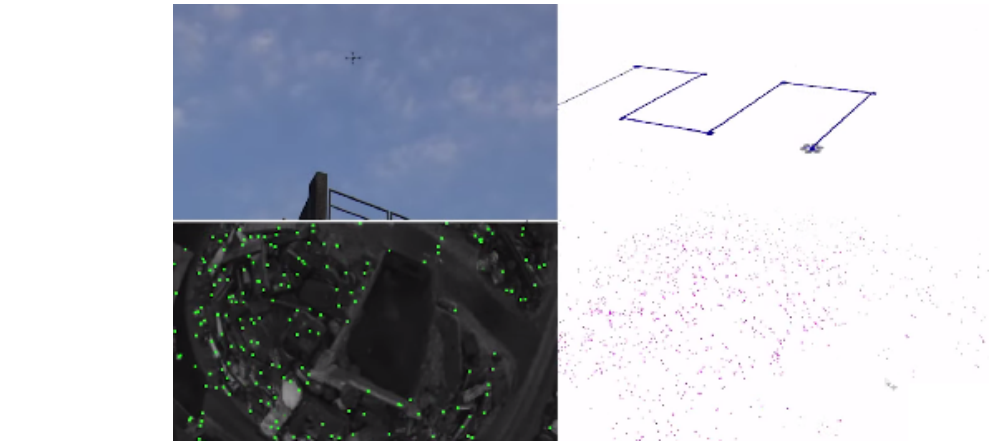
1.4 SVO - Semi-direct Monocular Visual Odometry

Na projektu SVO [2] (obr. 1.7), který zastřešovala univerzita v Curychu ve Švýcarsku, se podíleli Christian Forster, Matia Pizzoli a Davide Scaramuzza, kteří tento software uveřejnili jako open-source. Metoda pro svou správnou funkčnost doporučuje nasměrovat monokulární kameru kolmo dolů. Metoda je primárně určena pro navádění UAV zařízení v prostorách bez přístupu k informacím z GPS .

Tato metoda byla speciálně vyvinuta pro efektivnější zpracovávání snímků než umožňovaly současné metody. Polopřímý přístup eliminuje potřebu náročnou extrakci bodů a robustní mapování shodujících se bodů pro výpočet pohybu. Metoda pracuje přímo na intenzitě jednotlivých pixelů což má za následek vysokou přesnost při vysokém snímkovacím kmitočtu. Jedná se o pravděpodobnostní mapovací metodu. V případě přímých metod je shoda 3D bodů mezi snímky implicitním výsledkem. Extrakce bodů u této metody je potřebná pouze při inicializaci nových 3D bodů, tedy ne každý snímek podstoupí extrakci bodů a informace ze snímku pak putují k aktualizaci hloubkového filtru.

Metoda je implementována pomocí dvou paralelních vláken, přičemž jedno slouží pro výpočet pohybu kamery a druhé pro mapování prostředí tak, jak je kamerou prozkoumáváno. Toto rozdělení umožňuje rychlé sledování 3D bodů v jednom z vláken, zatímco v druhém se rozšiřuje mapa. Tímto odpadá problém zpracování v reálném čase.

[3] http://www.edwardrosten.com/cvd/
[4] http://www.edwardrosten.com/cvd/toon.html



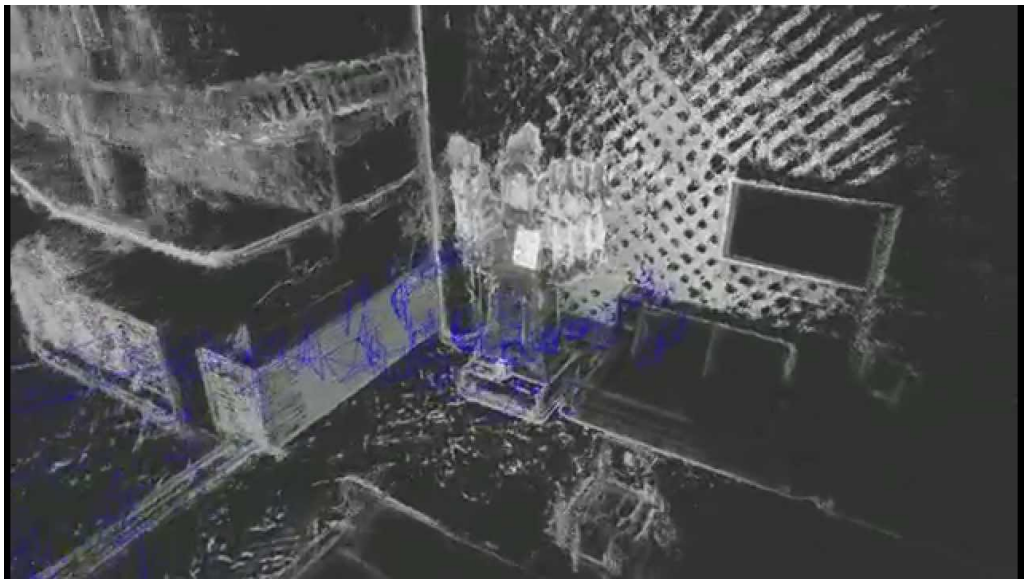
Obrázek 1.7: Ukázka hlavních částí metody SVO. Zdroj: [2]

1.5 LSD-SLAM - Large-Scale Direct Monocular SLAM

LSD-SLAM [5] představuje nové pojetí přímé monokulární mapovací techniky z podzimu 2014. Na projektu se podíleli Jakob Engel, Thomas Schöps a Prof. Dr. Daniel Cremers z Technické univerzity v Mnichově.

Místo používání klíčových bodů na snímku se přímo operuje s intenzitami pro určování polohy kamery a bodů. Algoritmus metody pracuje na bázi polohustých hloubkových map, získaných po zpracování rozdílu mezi více snímky. Klíčovými prvky jsou zde snímky, které se propojují vzájemně na sebe i s možností kruhové uzávěrky, tedy navazovat snímky do kruhu.

Tento systém pak umožňuje mapování objektů ve větším měřítku, jako je například místnost nebo venkovní ulice, a do vysokých detailů.



Obrázek 1.8: Rekonstrukce místnosti pomocí LSD-SLAM. Zdroj: [5]

Metoda nejenže dokáže sledovat pohyb kamery, ale dovoluje také mapování okolního

prostředí (obr. 1.8). Je implementována tak, aby ji bylo možno použít jak ve vnitřních tak i venkovních podmínkách. Doporučuje se použít rozlišení 640 x 480 .

1.6 Další dostupné metody pro polohování ve 3D

viso2_ros a libviso2

Balík obsahuje podporu pro monokulární a stereo kamery. U obou typů kamery se využívá odhadu pohybu kamery na základě příchozích snímků z kalibrovaných kamer. U mono kamery, která využívá pouze 2D informaci je potřeba získat hloubkovou informaci pro správnou funkčnost. U stereo kamer toto omezení neplatí. Metoda byla primárně vyvíjena pro pojízdné roboty, kde se používají převážně širokoúhlé kamery [23].

DEMO - Depth Enhanced Monocular Odometry

DEMO je založen na hloubkových mapách. Při přístupu k IMU metoda zpracovává i rotaci kolem osy. Autor udává, že daný program byl testován na laptopu se čtyřjádrovým 2.5 GHz procesorem s 6 GB pamětí. Metoda také podporuje RGB-D kamery. Zpracování je možné zrychlit pomocí GPU přes OpenCL[5] [10].

1.7 Kamery

Kamery se dají členit podle různých parametrů. Pro vizuální odometrii je podstatným parametrem, jakým způsobem je snímáno okolí. Podle tohoto kritéria se dají kamery členit do tří velkých skupin:

- monokulární kamery,
- stereo kamery,
- všesměrové kamery.

Monokulární kamera

Obraz je snímán pomocí jednoho objektivu. Jedná se o standardní klasický obraz. Při vizuální odometrii je třeba pro stereoskopické (3D) snímání okolí posunu kamery vůči prostoru a pořízení dalšího snímku. Z jednoho snímku není možné rekonstruovat vzdálenost předmětů ve snímaném prostoru.

Stereokamera

Stereokamera ke snímání prostoru využívá dvou objektivů, které jsou vůči sobě umístěny ve vzdálenosti 6.35 cm, což přibližně odpovídá vzdálenosti očí. Pořízený snímek nám pak do určité vzdálenosti poskytuje stereoskopickou představu o snímaném prostoru. Cena stereokamery je pak vyšší než cena za monokulární kamery. Stereokamery se v současnosti těší velké oblibě, a to zejména díky tomu, že jsou jimi tvořeny 3D filmy pro projekci v kinech.

[5] http://en.wikipedia.org/wiki/OpenCL

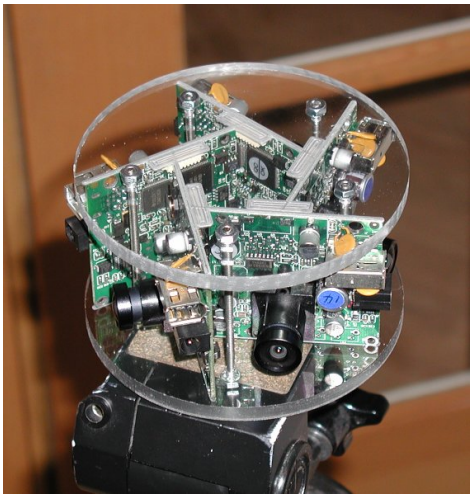
Všesměrová kamera

Všesměrové kamery nejsou tolik známé jako předchozí dva typy kamer. Jedná se o kameru, která má pozorovací úhel od 180° až po 360°. Díky této vlastnosti je hojně využívána pro pořizování panoramatických fotek (obr. 1.9), Ke snímání se využívá sada kamer nebo jedna kamera se speciálně umístěnou sadou zrcadel (obr. 1.10). Další využití se naskýtá například i v robotice při metodách SLAM, kdy se pomocí kamery mapuje okolí. Tato technologie se využívá například u Street-View od firmy Google Inc [15].

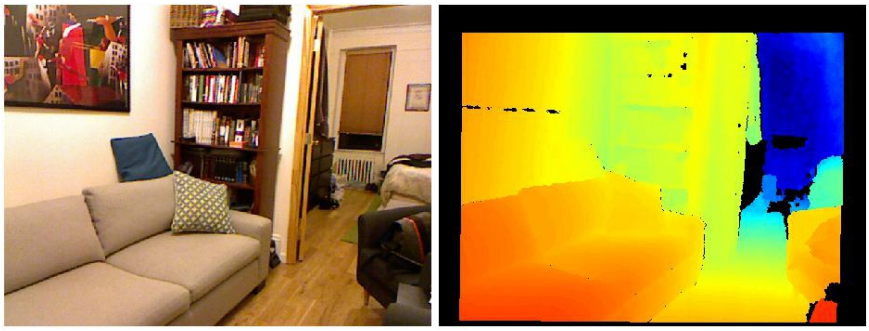
Obrázek 1.9: Obraz pořízený z všesměrové kamery. Zdroj [6]

Kamera s hloubkovým senzorem (RGB-D kamera)

Kamery s hloubkovým senzorem mají tu výhodu, že dokážou získávat údaje o hloubce obrazu, které se pak poskytují společně s obrazovými daty (obr. 1.11). Tyto údaje se dají využít ve spoustě případů, a to od mapování (techniky SLAM) až po vizuální odometrii. Existují ale už i různé algoritmy, které dokáží tuto informaci vygenerovat pro snímky ze stereo kamer či monokulárních kamer.



Obrázek 1.10: Příklady všesměrové kamery. Zdroj: [4]



Obrázek 1.11: Snímek s informací o hloubce pořízený RGB-D kamerou. Zdroj [16]

Specifikace kamery

Na vlastnosti kamery se můžeme dívat ze dvou úhlů. Můžeme sledovat parametry, které jsou dané montáží objektivu a kamery a pak také měnitelné nastavení kamery. Do specifikace objektivu a kamery patří například distorze čočky, ohnisková vzdálenost nebo rozlišení, ve kterém se snímání provádí. Dále lze ještě mluvit o vnějším rozhraní kamery, které může být buď USB nebo ethernet.

Ohnisková vzdálenost (Focal length)[6]

Ohnisková vzdálenost určuje vzdálenost mezi optickým středem čočky objektivu a rovinou, na kterou dokáže objektiv zaostřit snímaný objekt, neboli kde se dané paprsky ze skutečného obrazu protnou. Ohnisko objektivu je dáno jeho konstrukcí. U kamer IDS uEye s výměnným objektivem je pak třeba správně nastavit tuto vzdálenost vhodným počtem závitů při nasazování objektivu, jinak je obraz rozmazán.

[6] http://en.wikipedia.org/wiki/Focal_length

Rozlišení (Resolution)[7]

Rozlišení kamery se určuje v pixelech, případně v megapixelech (2^{20} pixelů). Tento parametr určuje, kolik obrazových bodů bude mít jeden snímek, tedy kolik má světlocitlivých buněk obrazový čip kamery. Kamery s vyšším rozlišením pak nemají problém s případným digitálním přiblížením (přiblížení bez mechanického posunu čoček). Naopak kamery s nižším rozlišením pak mají vyšší snímací rychlost.

Nastavení kamery

Množství nastavitelných parametrů se u kamer může lišit. Nebývá pevně dáno, co je přímo konstrukčním parametrem kamery a co naopak je možné softwarově nastavit. U kamer IDS uEye je těchto parametrů poměrně dost. Některé měnitelné parametry jsou nastavovány automaticky během natáčení kamery, jako například délka expozice kamery nebo korekce gammy.

Dále jsou zde parametry, které je třeba nastavit manuálně. Mezi takové parametry může patřit například barevné spektrum nebo rozlišení, které se pohybuje v hodnotách do maximálního rozlišení daného konstrukčním možnostem obrazového čipu.

Barevné schéma

Jedná se o důležitý nastavitelný parametr kamery. Pomocí něj je možné změnit barevné schéma výstupních snímků z kamery. Mezi dva důležité parametry pak patří `rgb8` a `mono8`, což reprezentuje barevné snímání a snímání ve stupních šedi.

Doba expozice (Exposure time)[8]

Kamera jako taková pracuje na principu fotoaparátu, kdy jsou pořizovány jednotlivé snímky. Stejně jako fotoaparát pracuje kamera s rychlostí závěrky. Jedná se o dobu, po kterou je závěrka otevřená a tudíž na obrazový čip kamery může dopadat světlo. Expoziční doba hraje významnou roli ve světlosti pořízeného snímku. Platí zde pravidlo, čím delší je expozice, tím déle dopadá světlo na obrazový čip a tím je výsledný snímek světlejší.

Počet snímků za sekundu (Framerate)[9]

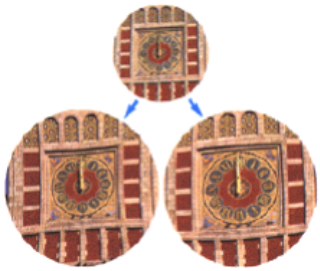
Tento důležitý parametr značí počet nasnímaných snímků za sekundu. Jedná se spíše o omezení vůči konstrukčním možnostem kamery. Pro plynulý obraz je minimálně vyžadováno 25 snímků za sekundu, některé kamery pak dokážou snímat i rychlost mávání křídel letícího hmyzu. Pro vizuální odometrii je vhodnější mít kameru s vyšší snímací rychlostí v závislosti na rychlosti posunu robota.

[7] urlhttp://cs.wikipedia.org/wiki/Pixel
[8] http://en.wikipedia.org/wiki/Shutter_speed
[9] http://en.wikipedia.org/wiki/Frame_rate

Specifikace objektivu

Distorze čočky (Distortion)

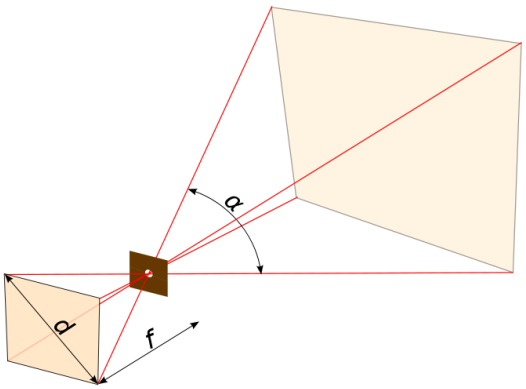
Jedná se o další důležitý aspekt ovlivňující obraz pořízený kamerou. Distorze čočky způsobuje takovou vadu obrazu, že nasnímaný obraz se liší od skutečnosti geometrickým zdeformováním, obraz se však jeví jako ostrý. Existují dva druhy distorze, pozitivní a negativní (obr. 1.12). Pozitivní distorze zvětšuje objekty na okrajích snímku zatímco negativní distorze přibližuje střed obrazu [13].



Obrázek 1.12: Příklad distorze čočky (pozitivní, původní obrázek, negativní). Zdroj: [13]

Zorné pole (Field of View)

Zorné pole je část prostoru (obr. 1.13), ze kterého přichází do objektivu světelné paprsky dopadající na obrazový čip kamery. Jedná se o maximální možný zachytitelný prostor. Velikost tohoto prostoru je hodně ovlivňována vlastnostmi čočky v objektivu [18].



Obrázek 1.13: Zorné pole s pozorovacím úhlem α. Zdroj: [18]

1.8 Princip kalibrace kamery

Pro zpracovávání obrazových dat nějakým pozičním či mapovacím algoritmem je třeba kameru správně kalibrovat. Bez známých parametrů jako je například ohnisková vzdálenost, hlavního bodu snímku nebo distorze čočky není možné bod následně rekonstruovat v prostoru. Příchozí světelný paprsek do objektivu je určitým způsobem lámán, než dojde k jeho

zaznamenání pomocí obrazového čipu. Snímek pak jako takový obsahuje jen množinu pixelů na konkrétních místech, které reprezentují zaznamenané paprsky přicházející ze scény v čase pořízení snímku. Aby se s těmito daty dalo dále pracovat je třeba určit, kde se dané paprsky nacházely ve scéně. K tomu právě slouží následná rekonstrukce těchto paprsků do scény. Aby tohle bylo možné, je třeba znát distorzi čočky a ohniskovou vzdálenost. Protože ale většina těchto hodnot záleží na kombinaci kamery a objektivu, není možné předem tyto hodnoty znát.

K výpočtu daných koeficientů se používají různé nástroje pracující s nějakým reálným obrazcem. Ve většině případů to bývá obrazec s motivem šachovnice u kterého jsou známy rozměry jednoho pole. Poté lze za pomocí složitého výpočtu spočítat rozdíl mezi snímkem a původní skutečnou podobou a tím získat potřebné koeficienty.

1.9 Kvadrokoptéra AscTec Pelican / Tyra

Bezpilotní letoun neboli také UAV zařízení je dnes známé zvláště díky svému hojnému využití. Jedná se většinou o malého bezpilotního robota, který je řízen ze země buď speciálně uzpůsobeným zařízením (RC vysílačem) nebo pouhým mobilním zařízením. Speciálním druhem UAV zařízení jsou kvadrokoptéry. K jejich hlavní přednostem patří snadná ovladatelnost a nízká pořizovací cena, která v dnešní době začíná na hranici jednoho tisíce korun a to od těch nejmenších modelů. Cena profesionálnějších modelů, které disponují již pokročilými funkcemi řízení, pak začíná přibližně od deseti tisíc korun. Využití nalézá v reportážních záběrech, pořizování fotografií ale také i v armádě a špionáži. Začíná také přibývat počet nadšenců, kteří si pomocí kvadrokoptéry dokumentují svou dovolenou nebo to mají jako svůj vlastní koníček. Na trhu existuje spousta druhů těchto bezpilotních robotů se širokou škálou již vestavěných funkcí a s možnostmi dalšího rozšíření. Hojně se v těchto zařízeních rozvinula technologie autopilota, která umožňuje uživateli zadat plán trasy předem a pak si už jen v pohodlí vychutnat pořízené video bez nutnosti dalšího zásahu do ovládání.

AscTec Pelican

AscTec Pelican[10] je model kvadrokoptéry od německé firmy Ascending Technologies GmbH. Jedná se o velmi škálovatelnou kvadrokoptéru, kterou lze dále rozvíjet. Hlavní jádro tvoří základní deska, která může nést výkonný vícejádrový procesor s architekturou x86-64 s ohledem na spotřebu a s nainstalovaným plnohodnotným operačním systémem a ROSem. ROS ale není realtime [17] systém, proto se nehodí k ovládání hardware, které musí pracovat v reálném čase, jako jsou rotory kvadrokoptéry. Ovládání jednotlivých rotorů se tedy neprovádí přímo z hlavní základní desky ale pomocí autopilota. Základní deska s autopilotem komunikuje, dochází tudíž mezi nimi k přenosu informací a příkazů. Důležité parametry kvadrokoptéry AscTec Pelican [7]:

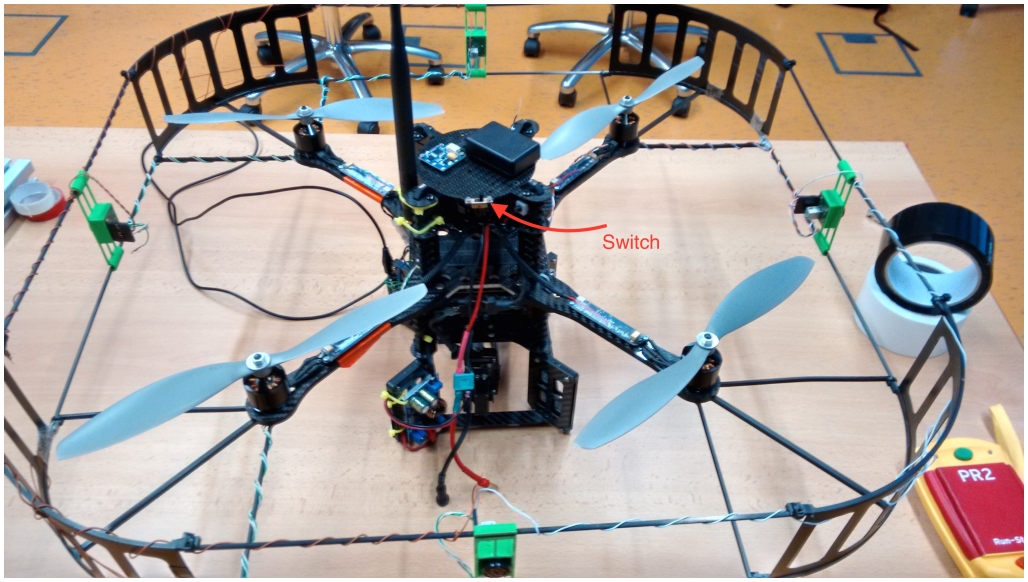
- velikost: 651 x 651 x 188 mm,
- maximální vzletová hmotnost: 1,65 kg,
- maximální doba letu: 16 minut,

[10] http://www.asctec.de/en/uav-uas-drone-products/asctec-pelican/

- bezdrátová komunikace: 2,4 GHz XBee[11] link, 10-63 mW, WiFi,
- autopilot: AscTec AutoPilot[12] s obnovovací frekvencí 1 kHz,
- letové módy: GPS mód, výškový mód, manuální mód.

Tyra

Tyra[13] (obr. 1.14) [9] je technické označení kvadrokoptéry, která je ve vlastnictví fakulty VUT FIT[14]. Má již spoustu modifikací, které neodpovídají modelu AscTec Pelican. Konkrétně jde o základní desku, na které běží operační systém Xubuntu 12.04 s ROS Hydro. Původní deska od Ascending Technologies byla cenově velmi drahá, proto byla nahrazena deskou `MIO-2261`[15] s procesorem Atom `N2800` pracující na frekvenci 1.86GHz. Tyra se dá ovládat pomocí R/C ovladače nebo pomocí mobilních zařízení přes WiFi. Pro práci a testování lze mít spuštěnou pouze základní desku aniž by bylo třeba zapínat celou kvadrokoptéru. K základní desce se pak připojí USB klávesnice, myš a monitor s rozhraním VGA, připojí se k internetu a lze na ní normálně pracovat.



Obrázek 1.14: Kvadrokoptéra Tyra na UPGM FIT VUTBR v Brně. Zdroj: [9]

Doplňující specifikace charakteristická pro Tyru:

- základní deska `MIO-2261` s Intel Atom `N2800`,
- 4GB DDR3 RAM paměti,
- 32 GB SSD disk.

[11] http://en.wikipedia.org/wiki/XBee
[12] http://wiki.asctec.de/display/AR/AscTec+AutoPilot
[13] http://merlin.fit.vutbr.cz/wiki/index.php/RoboLab_Tyra
[14] Vysoké učení technické, Fakulta informačních technologií v Brně http://www.fit.vutbr.cz/
[15] http://downloadt.advantech.com/ProductFile/PIS/MIO-2261/Product%20-%20Datasheet/MIO-2261_DS%2801.15.14%2920140128141904.pdf

Komunikační rozhraní:

- 2x externí USB + 2x interní USB,
- VGA konektor,
- ethernet RJ45,
- wireless LAN,
- RS232,
- kamera `UI122xLE-C` od firmy IDS uEye[16].

1.10 Robot Operating System (ROS)

Jedná se o open-source platformu využívanou především v robotice pro vývoj a sdílení již existujících řešení mezi vývojáři a uživateli. I přes svůj název se spíše jedná o framework, který umožňuje komunikaci mezi jeho komponentami a tím umožnit řešení typických úloh jako je ovládání pohybu, zpracování kamery nebo komunikaci s vnějším prostředím. Pro průmyslové účely je zde modifikace systému ROS v podobě ROS Industrial. Tento software pak využívají pro své robory velké firmy jako ABB, Kuka.

ROS [12] se především skládá z vývojových a simulačních nástrojů, knihoven a ovladačů k velké spoustě hardware.

První práce na systému ROS byla započata na Standforské univerzitě roku 2007, kde Morgan Quigley v rámci projektu STAIR (STandford AI Robot) napsal jádro Switchyard, které dodnes tvoří základ ROS. Projekt se ukázal natolik úspěšný, že další vývoj zastřešoval kalifornský podnik Willow Garage. Ten také vyvíjí projekty OpenCV[17], PLC Library a simulátor Gazebo. V dnešní době již přešel vývoj ROS pod značku OSRF (Open Source Robotic Foundation).

Mezi podporované systémy officiálně patří Ubuntu. Mezi experimentální pak OS X, Arch Linux a UbuntuARM. Podporované komunitou jsou ale i další operační systémy jako Fedora a Microsoft Windows [17].

Architektura ROS

Architektura ROS je založená na objektovém principu a má strukturu grafu. Hlavní podstata je ve zpracování dat v peer-to-peer síti, která je složena z uzlů (nodes). Mezi základní části systému ROS patří uzly, Master (roscore), zprávy (messages), témata (topics), služby (services). Uzly si mezi sebou posílají zprávy a mohou mít na starosti ovládání a správu různých částí robota. Většinou jde o určité elementární části nebo o abstrakci nad určitou částí hardware (např. snímání dat z kamery).

Master se stará o registraci názvu uzlů, aby se jednotlivé uzly dokázaly navzájem dohledat z důvodu výměny zpráv. ROS umožňuje mezi uzly různé způsoby komunikace. Přenášené zprávy jsou jednoduché datové struktury tvořené základními datovými typy (např. `integer`, `float`, `bool`...). Tyto zprávy jsou vkládané na různé komunikační kanály v závislosti na využití těchto informací.

[16] https://en.ids-imaging.com/
[17] http://opencv.org/

Prvním způsobem přenosu zpráv je pomocí témat, která pracují na principu publikace a odebírání zpráv. Výhodou této komunikace je asynchronní komunikace mezi uzly ale jedná se pouze o jednosměrnou komunikaci. Uzel, který vkládá danou zprávu na téma a druhý uzel naopak danou zprávu odebírá. Uzly mohou být připojeny k libovolnému počtu témat.

Druhým způsobem je komunikace pomocí služeb. Zde se již jedná o synchronní komunikaci, která pracuje na principu dotaz - odpověď. Uzel, který je v roli dotazovacího, pošle zprávu typu dotaz na kanál a čeká na odpověď [12].

ROS balíky

V současné době je k dispozici na webových stránkách ros.org veřejně ke stažení přes 3000 balíků k různým robotům. Tyto balíky se dají jednoduše stáhnout a nainstalovat pomocí `apt-get` ve tvaru: `apt-get install ros-<distro>-<balík>`

Pokud ale neexistuje příslušný deb balíček, lze jej například stáhnout pomocí `rosws` a na závěr přeložit. Některé balíčky se stahují přímo z repozitářů, ve kterých se vyvíjí, a to pak například nástrojem `git` a s následným přeložením pomocí `catkin_make` nebo `rosmake`.

Balík uEye

Balík [8] slouží pro interakci mezi ovladačem pro kameru od firmy IDS uEye a systémem ROS. Nezajišťuje ale samotnou funkcionalitu kamery, tedy je nutné pro kameru nejprve nainstalovat příslušný ovladač ze stránek výrobce. Projekt z repozitáře se pak dá přeložit pomocí nástroje `catkin_make`.

Uzel se spouští přes `roslaunch` a v souboru `nodelet.launch` se dají přednastavit příslušné parametry, které je možné vkládat i druhým způsobem a to při spouštění `ueye` uzlu.

Uzel publikuje obrazová data na `/image_raw` v podobě matice, případně pak informace o kameře na `/camera_info`. Tyto informace jsou čteny ze souboru a je nutné je případně předem nastavit.

Důležité spouštěcí parametry:

- `hardware_gamma (bool)`
  - povolí automatickou korekci gamma,
- `auto_exposure (bool)`
  - povolí případnou automatickou korekci délky expozice,
- `exposure_time (double)`
  - nastaví manuální dobu uzávěrky, pokud je automatická korekce zakázána,
- `frame_rate (double)`
  - určí maximální počet snímků za sekundu,
  - tento parametr neumožňuje získat větší počet snímků, než to umožňuje kamera,

Autor tohoto balíku doporučuje kalibrovat kameru pomocí balíku `camera_calibration`[18]. Balík uEye metodu podporuje a případná kalibrační data by měla být nastavena automaticky.

[18] http://wiki.ros.org/camera_calibration

Balík `image_view`[19]

Balík slouží k zobrazování obrazových dat do uživatelsky přívětivé podoby. Pro zobrazení je třeba mít k dispozici grafický režim.

Důležitými parametry jsou zde parametry image a pak zvolený přenos režimu. Pokud není zadán, použije se typ RAW[20]. Lze tedy pomocí tohoto balíku i zobrazovat komprimovaná data přicházející ze vzdáleného uzlu (například ze sítě).

Balík `image_transport`[21]

Obrazová data přenášená ze vzdálených uzlů je třeba komprimovat. Pro svůj objem v RAW podobě by rychlost většiny komunikačních linek nedostačovala. Pro převod mezi komprimovanou a nekomprimovanou podobou slouží právě tento balík.

ROSBAG

Nástroj rosbag je důležitou součástí systému ROS. Umožňuje zaznamenávat vybraná témata do jednoho bag souboru. Tyto soubory je pak možné následně přehrát takovým způsobem, aby ostatní uzly nepoznají, že data nepochází ze skutečného zařízení [12].

[19] http://wiki.ros.org/image_view
[20] http://en.wikipedia.org/wiki/Raw_image_format
[21] http://wiki.ros.org/image_transport

Kapitola 2

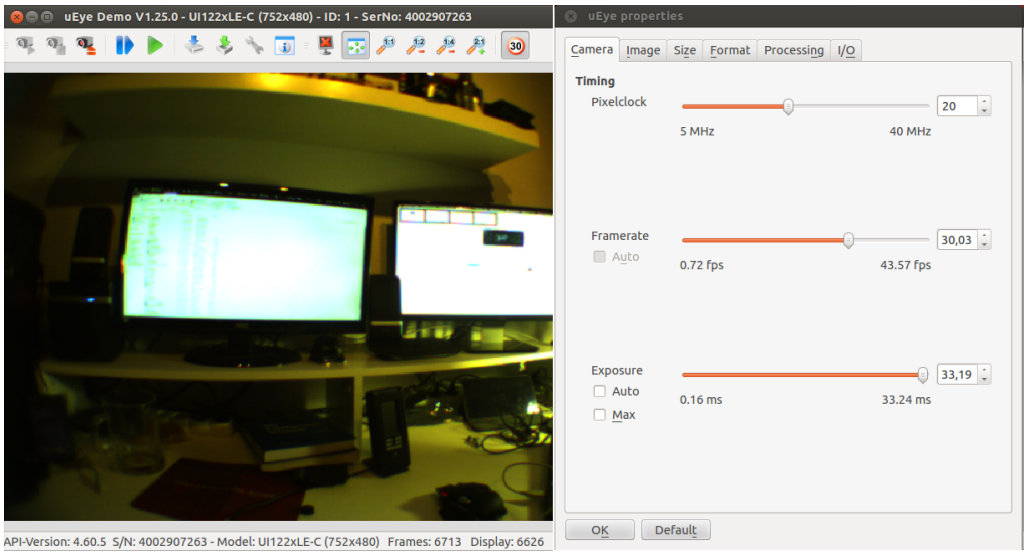
Instalace a oživování algoritmů

Následující kapitola popisuje postup pro instalaci a oživování jednotlivých metod. Základem je mít nainstalovaný systém ROS na systému Ubuntu. Vzhledem ke skutečnosti, že na Tyře se již nachází Xubuntu 12.04 s ROS Hydro, byly tyto postupy uzpůsobeny této konfiguraci. S jinou distribucí linuxu, kterou systém ROS podporuje by byl postup pravděpodobně v některých bodech jiný.

V době, kdy byla tato práce vypracovávána došlo k nekompatibilitě některých balíků pro metodu SVO. Z tohoto důvodu bude pro jistotu sestava balíků k funkčnosti všech testovaných metod dostupná na přiloženém DVD.

2.1 Oživování kamery z Tyry

K uvedení uEye kamery do provozu je třeba si stáhnout příslušný balík uEye od ROSu v aktuální verzi. Před překladem balíčku pomocí `catkin_make` je vhodné stáhnout a nainstalovat ovladače pro množinu kamer IDS uEye. Po instalaci zmíněných ovladačů se do počítače nainstaluje služba `ueyeusbrc`, kterou je třeba spustit. Test kamery se pak provádí pomocí příkazu `ueyedemo`.



Obrázek 2.1: Nástroj `ueyedemo` pro nastavování kamery a pro její testování v praxi.

V nastavení kamery (obr. 2.1) lze různě otestovat vliv jednotlivých parametrů nastavení na výsledný obraz. Změny se projevují okamžitě a není třeba znovu zapínat čtení z kamery, jako je tomu například u ROS uzlu `ueye`. Taktéž pomocí tohoto nástroje lze poznat reálné možnosti kamery. Tato aplikace se skvěle hodí pro testování v různých světelných podmínkách, kdy je třeba kameru správně nastavit, aby dokázala získat co nejvíce podstatných informací. Aplikace však výsledné parametry neukládá pro systém ROS.

Následně, po instalaci ovladačů již lze přeložit balíček v ROSu. K přeložení je třeba některých knihoven, které se právě doinstalují pomocí ovladače ke kameře.

2.2 Oživování mobilní kamery

Původní plán byl na mobilní telefon nainstalovat systém Ubuntu alespoň ve verzi 12.04. K dispozici jsem měl mobilní telefon smartphone HTC 3D Evo[1] s vestavěnou dvojitou kamerou pro 3D vidění. Po několika neúspěšných pokusech telefon rootnout jsem od tohoto plánu musel upustit a smířit se s aplikací na Google Play, která dokáže zaregistrovat kameru jako uzel do vzdáleně běžícího ROSu. Aplikaci jsem našel pod názvem ROS Sensors[2] od autora Tal Regev. Aplikace je z března 2015 stojí v přepočtu 30 Kč, nicméně ještě není úplně vyladěná a místy padá.

Aplikace jako taková umožňuje zvolit název kořenového uzlu a zvolit senzory, které se budou posílat. Doporučuji zde zvolit pouze kameru a pokusit se o připojení k této kameře. V ROSu na počítači by se daný uzel měl zobrazit. Pokud se nezobrazí, nedošlo k připojení telefonu k ROSu v počítači.

Obraz se standardně přenáší v komprimované podobě. Měl jsem k dispozici pouze wifi spojení s nominální rychlostí 54 Mbps. Po přepnutí do nekomprimované podoby v menu aplikace nebyl přenos obrazových dat dostatečně rychlý, což mělo za následek zpoždění a trhanost. Pro dekomprimaci komprimovaného toku jsem pak použil balík `image_transport` což mohlo mít za následek snížení kvality obrazu.

2.3 SVO

SVO [2] může být nainstalováno jako ROS balík nebo jako systémová aplikace. Primárně jsem se zaměřil na použití SVO v prostředí ROS, ale dodatečně zde přidávám i postup pro využití mimo systém ROS. SVO je primárně napsané pro verzi BOOST[3] 1.53+. S menšími modifikacemi lze toto obejít a překládat i na verzi BOOST 1.49, která je obsažena v Ubuntu 12.04.

Prerekvizity pro překlad:

Sophus[4] - Lie groups

Algoritmus pro výpočet transformací tuhého tělesa je napsán v jazyce C++ za pomocí knihovny Eigen, která slouží pro práci s maticemi, konkrétně s Lieovými grupami. Projekt stačí naklonovat, pak použít `cmake` a `make` pro přeložení nezávisle na systému ROS.

[1] http://www.htc.com/cz/support/htc-evo-3d/howto/82839.html
[2] https://play.google.com/store/apps/details?id=org.ros.android.android_all_sensors_driver
[3] http://www.boost.org/
[4] https://github.com/strasdat/Sophus.git

Fast[5] - Corner Detector

Pro detekci je použit algoritmus od Edwarda Rostena pomocí knihovny libCVD. Využitá je však jen modifikovaná podmnožina této knihovny. Použitá verze pro SVO je k dispozici opět na githubu. Projekt se pak přeloží opět za pomocí `cmake` a `make`, není třeba mít k dispozici ROS.

g2o[6] - General Graph Optimization

Jedná se o optimální balík, který není přímo potřebný pro metodu SVO. Nabízí se využití pro techniky SLAM a BA (Bundle Adjustment). V případě použití tohoto balíku je potřeba doinstalovat tyto systémové knihovny: `libeigen3-dev`, `libsuitesparse-dev`, `libqt4-dev`, `qt4-qmake`, `libqglviewer-qt4-dev` které jsou dostupné v systémových repozitářích. Projekt je možný klonovat opět z githubu a následně přeložit a nainstalovat (`cmake` a `sudo make install`) nezávisle na systému ROS.

vikit[7]

Vikit obsahuje některé z matematických a interpolačních funkcí použité ve SVO. Jedná se o catkin projekt. Nyní ve většině případů je potřeba nainstalovat balíček pro ROS ohledně `cmake` (`ros-hydro-cmake-modules`), jinak by projekt nemohl být přeložen. Pokud by se jednalo o verzi bez použití ROS, je třeba editovat soubor `vikit_common/CMakeLists.txt` a změnit hodnotu `USE_ROS` na `FALSE`.

SVO[8]

V konečné fázi se klonuje samotný algoritmus SVO. Pokud byl přidán dodatečný balík `g2o`, je třeba ho ještě povolit před překladem ve `svo/CMakeList.txt`. SVO může být přeloženo jako systémová knihovna a není zde zapotřebí prostředí ROS. Tato volba se dělá v témž souboru `CMakeList.txt`. V případě ROS verze se projekt přeloží pomocí `catkin_make`.

Troubleshooting

1. Během překladu se vyskytne chyba v návratu z funkce s `nullptr`. Projekt je uzpůsoben novější verzi ROSu kompilovaného pomocí vyšší verze BOOSTu než je nainstalována ze systémových repozitářů operačního systému. Je třeba celou třídu přepsat z `boost` na `std` ve všech souvisejících souborech (lze dohledávat opakovaným překladem).

2. Chybějící balík `ros-hydro-cmake-modules`, je třeba mít nainstalován při překladu catkin projektů.

3. Chybová hláška "Undefined symbol…". S velkou pravděpodobností se zde jedná o nestejné verze balíčků ROS a systémových knihoven. Je nutné provést update systému! Ideální je pak smazání složek `build` a `devel` v `catkin` složce a pokusit se znovu o kompletní překlad všech projektů. Je nutné znovu přeložit i předchozí balíky.

[5] https://github.com/uzh-rpg/fast.git
[6] https://github.com/RainerKuemmerle/g2o.git
[7] https://github.com/uzh-rpg/rpg_vikit.git
[8] https://github.com/uzh-rpg/rpg_svo.git

4. Problém s rviz[9] (padá, nespustí se). Většinou se jednalo o problém s chybějícím grafickým ovladačem nebo o pokus rozběhnout rviz ve VirtualBoxu. Na stránkách vývojáře rviz lze dohledat spoustu druhů chyb a lze se s nimi různými způsoby vypořádat. Mým doporučením je mít nainstalován grafický ovladač a pouštět rviz na fyzickém počítači.

Pro překlad pod ARM architekturou je třeba před překladem catkin projektů nastavit proměnnou `export ARM_ARCHITECTURE=True`

Oživování SVO

Pro otestování funkčnosti metody je vhodné si stáhnout předpřipravený `rosbag` soubor. Pokud algoritmus signalizuje chyby, je nutno, ve většině případů, hledat chyby přímo v programu. Zdrojový kód se stále ještě dolaďuje a na různých verzí knihoven pracuje odlišným způsobem.

Před ostrým spuštěním je třeba mít připravena kalibrační data pro daný typ kamery (dírkový model popřípadě model atan). Tyto údaje se pak vloží do souboru ve zdrojové složce `src/rpg_svo/svo_ros/param`. Pro spuštění je třeba také editovat příslušný spouštěcí soubor ve složce `launch` hned vedle složky `param`, kde je třeba změnit odebírané téma s RAW obrazovými daty, otočení kamery směrem dolů a případné změnění cesty ke kalibračním datům. Otočení kamery se provede vložením konfiguračních řádků do spouštěcího souboru:

```
<param name="init_rx"value="3.14"/>
<param name="init_ry"value="0.00"/>
<param name="init_rz"value="0.00"/>
```

Pro vizuální otestování algoritmu lze použít rviz. Konfigurační data pro rviz se nachází ve složce `rpg_svo/svo_ros/rviz_config.rviz`. Tímto souborem je třeba přepsat soubor `.rviz/default.rviz` v domovské složce. Po spuštění rviz je již vše nakonfigurované. Předchozí soubor doporučuji zálohovat! K dispozici je také nástroj `rqt_svo` pro sledování počtu trakčních bodů a dalších zajímavých informací.

SVO dokáže pracovat ve dvou módech. Mód `accurate` a mód `fast`. Výběr módu se provádí taktéž ve spouštěcím souboru. Rozdíl mezi těmito módy je pak v náročnosti na výpočetní výkon v závislosti na kvalitě obrazových dat a rychlosti kamery.

Inicializace SVO se provádí translačním pohybem přes kamerou zabíranou scénu.

2.4 PTAM

PTAM[10] [11] narozdíl od SVO nebylo těžké nainstalovat. Tento balík totiž není závislý na nainstalovaných balících kromě catkin, který je součástí jádra ROSu. Projekt se pouze naklonuje a přeloží standardní cestou pomocí `catkin_make`. Během překladu jsem se nesetkal s jediným problémem.

Oživování PTAM

Poměrně složité je však oživování PTAM. Projekt sám o sobě je poměrně robustní. Disponuje svým vlastním kalibračním mechanismem, který je poté třeba správně nastavit v konfiguračním souboru. V první řadě je však nutné nastavit barevné schéma kamery z `rgb8`

[9] http://wiki.ros.org/rviz
[10] https://github.com/ethz-asl/ethzasl_ptam

na `mono8` a nastavení rozlišení kamery do souboru `PtamFixParams.yaml`. Tyto úpravy jsou nutné pro spuštění kalibrační části PTAM.

Kalibrační obrazec se nachází ve složce `ptam` pod názvem `calib_pattern.pdf`. Pdf dokument je třeba vytisknout s velikostí 1:1, jinak by se provedla chybná kalibrace kamery. Pro kalibraci je nutné mít kolem obrazce co nejméně rušivých elementů.

Více informací ke správné kalibraci kamery lze nalézt v kapitole "Kalibrace kamery". Po provedení kalibrace kamery je třeba kalibrační data zapsat do souboru `PtamFixParams.yaml`. Dále lze ještě nastavit spouštění grafického režimu, které je doporučeno na vestavěných systémech zcela vypnout z důvodu zvýšení výkonnosti metody. Zakázání GUI se provede pomocí parametru `gui` v tomtéž souboru. Při snímání podobné textury je vhodné nastavit parametr `NoLevelZeroMapPoints`. Pro venkovní užití je taktéž vhodné nastavit parametr `EpiDepthSearchMaxRange`.

Další nastavení lze provést v souboru `PtamParams.cfg`, například povolení autoinicializace.

PTAM podporuje dynamickou rekonfiguraci parametrů. Parametry se dají modifikovat za běhu. Je k tomu třeba si spustit příslušný uzel `rqt_reconfigure`.

Inicializace PTAM se provádí stisknutím `Spacebar` v pravém horním rohu a translačním pohybem kamery přes scénu. Po úspěšné inicializaci je v okně ve scéně zabírané kamerou vidět spousta různobarevných 3D bodů. Pokud se tak nenastalo, jedná se nejspíše o špatnou kalibraci kamery.

2.5 LSD-SLAM

Instalace LSD-SLAM[11] [5] je taktéž poměrně jednoduchá. Pro instalaci na cílovém systému je třeba mít tyto balíky: `ros-hydro-libg2o` `liblapack-dev`, `libblas-dev`, `freeglut3-dev`, `libqglviewer-qt4-dev`, `libsuitesparse-dev`, `libx11-dev`. Po nainstalování je třeba naklonovat celý projekt do pracovního prostředí a pak pomocí `rosmake lsd_slam` přeložit. Překlad byl prováděn pro Ubuntu 12.04.

Oživování LSD-SLAM

Pro oživení metody je nutné nahrát potřebná kalibrační data do `camera.cfg`. Kalibrační data pro tuto metodu je vhodné získat pomocí kalibračního nástroje v metodě PTAM.

Existuje více možných formátů, ve kterých se daný soubor může nacházet v závislosti na kalibraci a typu kamery. Je vhodné použít rozlišení 640x480, neboť algoritmus je právě na toto rozlišení optimalizován.

Případné změny v chování metody se dají provést v souboru `LSDParams.cfg`. Je zde ale i možnost dynamické rekonfigurace parametrů za běhu pomocí uzlu `dynamic_reconfigure`.

Algoritmus jako takový produkuje spoustu bodů a aktualizuje tyto body pomocí aktualizací klíčových snímků. Z tohoto důvodu není možné se na výsledný obraz podívat pomocí nástroje rviz. Pro prohlédnutí nasnímané mapy lze využít uzel `lsd_slam_viewer`

Kvůli vysokému stupni paralelismu zavedeném v algoritmu je zde nedeterministické, jak bude výsledný obraz vypadat. I ze stejných obrazových dat nahraných pomocí `rosbag` se výsledek může lišit. Za toto je zodpovědné přepínání kontextu systému.

Inicializace algoritmu se provádí translačním pohybem okolo snímaného prostoru. Metoda hůře snáší rotační pohyby kamery.

[11] https://github.com/tum-vision/lsd_slam

Kapitola 3

Testování vybraných metod

V první řadě je třeba si vymezit způsoby testování a kritéria hodnocení. Pro testování jsem neměl k dispozici žádné externí speciální zařízení, pomocí kterého by se dala měřit odchylka algoritmů ve správnosti jejich výpočtu. Co naopak lze sledovat, je uživatelský pohled na dané algoritmy, jejich plynulost a spolehlivost. Z tohoto hlediska budou spíše výsledky reprezentovány slovním způsobem než statistickými daty.

Metody jsem se rozhodl testovat v různých situacích - jakým způsobem se chovají ve venkovním a vnitřním prostředí, jaký je vliv osvětlení v různou denní dobu na požadovaný výsledek. Taktéž jsem se rozhodl testovat metody při změně objektivu - jestli dojde ke zkvalitnění výsledku nebo zdali by bylo lepší pořídit novější kameru s lepšími parametry.

Pro testování jsem měl k dispozici dva objektivy - jeden s malým zorným polem a druhý s efektem rybí oko. Tyto dva objektivy jsou kompatibilní na kameru, která je již na Tyře.

K testování různých kamer jsem nakonec použil kameru z mobilního telefonu HTC 3D EVO. Pomocí programu z Google Play lze propojit telefon s běžícím ROSem na jiném počítači. Bohužel, ačkoliv telefon disponuje dvěma kamerami pro 3D snímání, aplikace neumožňuje přeposílání dat z obou kamer.

Výkon jsem se rozhodl otestovat na různých platformách. Pro testování se mi podařil uzpůsobit tablet Google Nexus 7 se čtyřmi jádry o frekvenci 1.6GHz s architekturou ARM s operačním systémem Ubuntu 13.04. Dále jsem se rozhodl otestovat výkon reálné kvadrokoptéry, konkrétně Tyry která disponuje dvoujádrovým Intel Atomem 1.8 GHz a v poslední řadě též na notebooku ASUS ROG G73-JH s Intel i7 se čtyřmi jádry 1.6 GHz.

Výběr vhodných kandidátů k testování

Důležitým kritériem pro výběr metody je jeho způsobilost pro daný typ úlohy. V mém kritériu hraje také roli možnost zprovoznění a smysluplného užití na Tyře. Taktéž metoda musí být kompatibilní s kamerou, která je k dispozici.

Pro testování jsem nakonec zvolil algoritmy PTAM, SVO a LSD-SLAM. Důvodem pro PTAM a SVO bylo již předchozí použití právě v UAV zařízení, kde měly dobré ohlasy. LSD-SLAM pak jako horkou novinku, která vyšla velmi nedávno ačkoliv se od účelu polohování zařízení odchyluje. Cílem je prozkoumat možnosti, které Tyra má. Její krajní meze s metodami, které jsou nyní dostupné. Pro testování by jistě bylo vhodné zahrnout více metod ale pro časovou náročnost to není zcela možné.

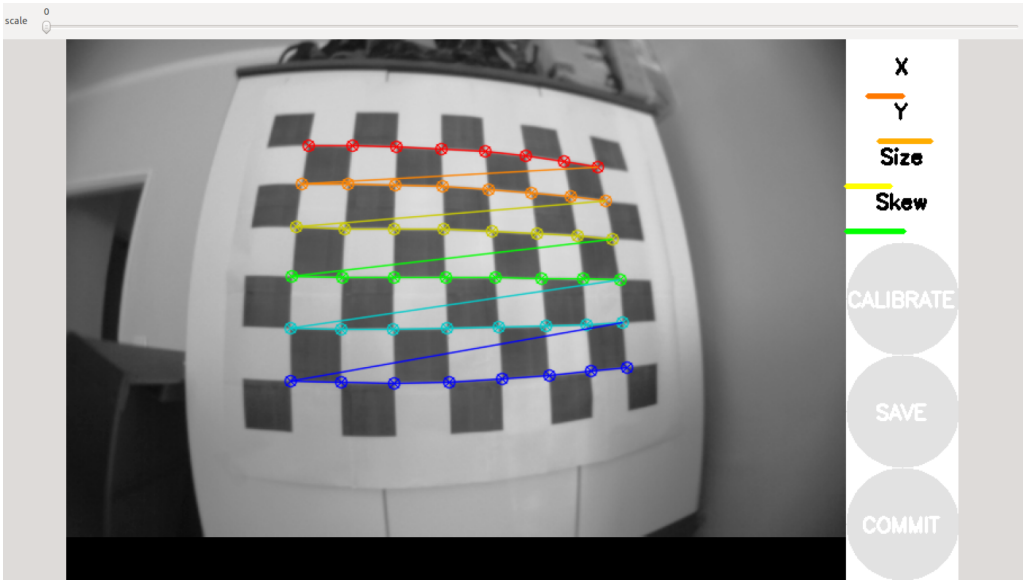
3.1 Kalibrace kamery

Při hledání způsobu kalibrace kamery, která byla k dispozici na Tyře, jsem narazil zprvu na balíček `camera_calibration` již vestavěný v systému ROS. Na tento balíček přímo odkazuje i balíček uEye, který zajišťuje zpracovávání dat z kamery.

Později se ale ukázalo jako vhodnější využít kalibračního nástroje u metody PTAM. Získaná kalibrační data pak byla využita pro nastavení všech použitých metod.

Kalibrace kamery pomocí balíku `camera_calibration`

Kalibrace zde probíhá pomocí velkého plátna s motivem šachovnice o rozměru cca 1 m x 2 m. Obrazec jsem sestrojil pomocí 25 papírů A4, které jsem sestavil přesně do výsledného kalibračního obrazce. Kalibrování probíhá docela jednoduše. Je důležité nasnímat tento obrazec z různých úhlů a vzdálenosti. Pro správné kalibrování jsou v programu uvedeny čtyři proužky (obr. 3.1) které svou délkou a barvou určují kvalitu dat pro kalibrování. Po nasnímání dostatečného počtu snímků se jen klikne na "`calibrate`"a započne se výpočet kalibračních parametrů. Výsledná kalibrační data se uloží do archivu a údaje se pro jistotu vytisknou i na obrazovku terminálu. V případě mnou prováděné kalibrace balíček napoprvé nenaběhl korektně a stalo se mi, že spadl při ukládání dat. Vždy se ale minimálně vytiskly kalibrační data do terminálu. Při opakovaném kalibrování se rozdíly vypočtených hodnot pohybovaly v řádu několika desítek, maximálně stovek. Taktéž po kalibrování následující algoritmy nepracovaly dobře. Například u SVO hned po inicializaci vypadávaly 3D body.



Obrázek 3.1: Nástroj camera_calibration určený pro kalibraci kamery.

Kalibrace kamery pomocí vestavěného nástroje v PTAM

PTAM v sobě nabízí přímo kalibrační nástroj. Pro kalibrování kamery je zde potřeba pouze jednoho papíru formátu A4 s motivem šachovnice. Kalibrace zde probíhá odlišným způsobem a vyžaduje větší úsilí než u předchozího případu. Je třeba kamerou zabrat co největší část šachovnice. Pro odebrání snímků je třeba zmáčknout v horním pravém panelu příslušné

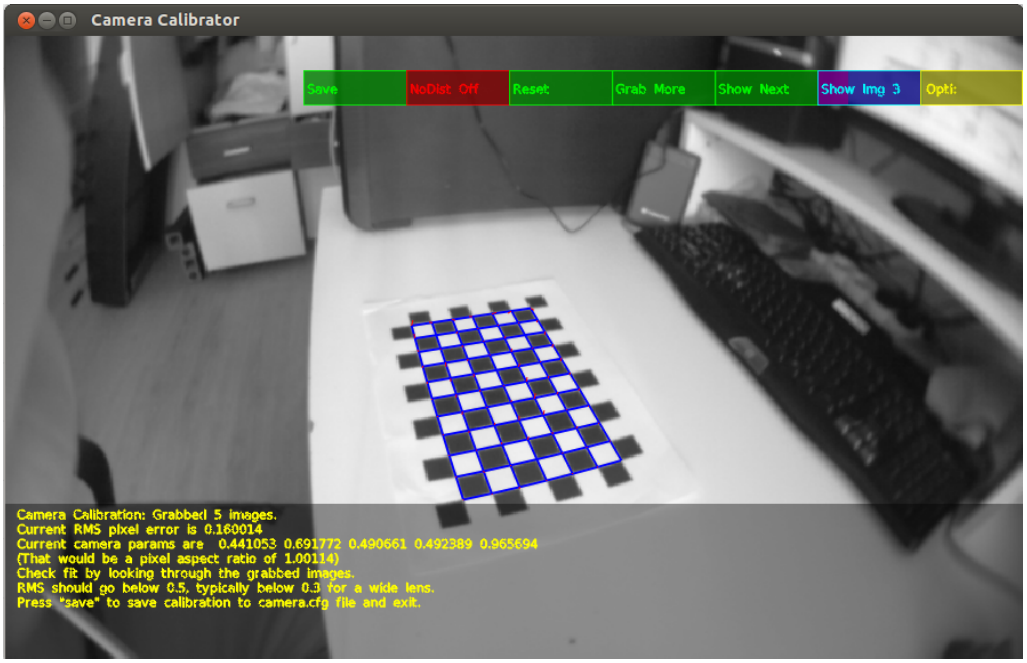
tlačítko. Po šachovnici se vykreslují obvody modrých čtverců (obr. 3.2). Čím více jich je, tím je obraz kvalitnější pro následné využití k výpočtu kalibračních dat. Nicméně i snímky s nižším počtem čtverců nejsou neužitečné a jsou zahrnuty k výpočtu.

Sejmutí jednoho snímku trvá téměř 20 sekund a osvědčilo se mi držet kameru během snímání ve stejné poloze. Většinou tento nástroj působí, že se přestal pracovat. Je zde třeba mít velkou trpělivost. Snímání pomocí kamery s malým zorným úhlem může být docela náročné.

Po pořízení přibližně 5 - 8 snímků z různých úhlů je vhodné zkusit vypočítat kalibrační data. Je zde vidět, jak se data během výpočtu zpřesňují. Důležitou hodnotou je zde pro kalibraci hraje `RMS pixel error`[1], kterou je ideální držet co nejníže. Kameru jsem zde kalibroval s přesností 0.16 což by měla být přijatelná chyba.

Může se stát, že při sbírání více snímků bude `RMS pixel error` narůstat. Za těchto okolností je vhodné začít od začátku. Při mnou prováděné kalibraci se mi ani po nasbírání 25 snímků nepovedlo získat lepší odchylku. Následující problém nastal při zacyklení výpočtu kalibračních dat. Dá se to rozeznat podle neustále měnících se kalibračních hodnot v levém dolním rohu okna. Pomohlo až začít s procesem kalibrace opět od začátku..

Pokud výše zmíněné problémy přetrvávají, je nutné se spokojit s případnou nejnižší naměřenou chybou při alespoň čtyřech až pěti snímcích.



Obrázek 3.2: Kalibrace kamery pomocí nástroje přiloženého u balíku ethzasl_ptam.

Vliv chybného nastavení parametrů kamery a kalibrace kamery

Špatná kalibrační data se dají poznat velmi rychle. V prvotních chvílích se metody zinicializují jakoby bez vážnějšího problému, následně ale začnou vypadávat a některé ještě stačí udat nesmyslná data. U všech tří metod se v terminálu vypisují varovná hlášení o výpadku

[1] http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Franczyk_RMSE_Accuracy.pdf

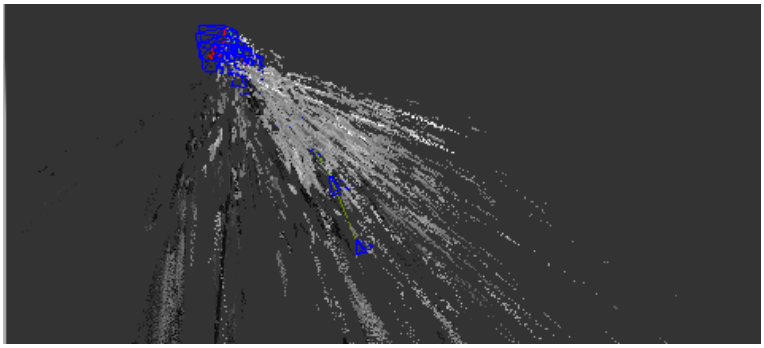
metody. LSD-SLAM navíc ještě přitom generuje nevalidní data, která získává špatnou rekonstrukcí bodů ve snímcích.

U LSD-SLAM se nevalidní kalibrace pozná hlavně generovanou 3D mapou, jak je patrné na obrázku 3.3. Obraz, který se pak v tomto případě rekonstruuje, je deformovaný a údaje z něj matoucí.

U metody PTAM se test kalibrace dá provést při zobrazení mapy v grafickém režimu a sledováním mapování 3D bodů při jednoduchých translačních pohybech nad rovným povrchem. Pokud nebudou body ležet převážně v rovině, je určitě na vině špatná kalibrace.

SVO narozdíl od předchozích dvou metod, při pokusu o pohyb kamerou, ihned ztratí zaměření v prostoru a nedokáže se znovu uchytit.

Všechny metody při špatné kalibraci vykazují neschopnost sledování 3D bodů v obraze.



Obrázek 3.3: Metoda LSD-SLAM s neplatnými kalibračními informacemi.

3.2 Testování různých objektivů

objektiv 6.0mm IR Mega

Při testování jsem měl možnost vyzkoušet na kameře dva objektivy. Původní objektiv, který sloužil pro uživatelské účely, konkrétně nahrávání obrazových dat a přenos dat například do mobilního telefonu, se ukázal jako omezující faktor pro kvalitu výsledných hodnot. Měl úzké zorné pole a navíc ještě přibližoval zabírané objekty (obr. 3.4). S tímto objektivem se u metod projevovaly velmi časté výpadky. SVO v tomto případě bylo nepoužitelné. U metody se vyskytovalo omezení ve sledování pohybu trakčních bodů ve snímcích. V prvních okamžicích se metoda inicializovala se spoustou bodů ale po zahájení se body velmi rychle vytratily. Pokud byla inicializační fáze úspěšná, bylo možno sledovat pouze body, které se zaměřily během inicializace.

PTAM se s tímto objektivem vyrovnal o poznání lépe než SVO. Projevovaly se také výpadky při rychlejším přesunu záběru kamery ale při malých a pomalých posuvech se na nových oblastech dokázal uchytit. Odhalování problému objektivu bylo komplikované. Bylo třeba nesčetněkrát kalibrovat kameru, zaostřovat objektiv a brát v úvahu všechny možné kombinace nastavení kamery a SVO. Po zralé úvaze a pozorování různých oficiálních rosbag záznamů jsem usoudil, že by problém nejspíše vyřešil objektiv s lepším pozorovacím úhlem.

Více se pomocí tohoto objektivu získat nedalo. S tímto objektivem bylo neúčelné pokračovat v dalších pokusech, neboť trpěl výše zmíněnými nedostatky.

Obrázek 3.4: Obraz pořízen objektivem 6.0mm IR Mega.

Objektiv typu rybí oko

Druhý objektiv, který jsem si vypůjčil pro testování, měl efekt rybí oko. Nedokázal jsem dohledat přesný typ objektivu, neboť na něm nebylo označení. Oproti předchozímu měl obrovský pozorovací úhel (obr. 3.5). Kalibrace tohoto objektivu probíhala podstatně lépe než tomu tak bylo u předchozího objektivu. Objektiv odstranil z velké části problém s trakčními body u metody SVO a podstatně vylepšil vlastnosti u PTAM. Kamerou pak bylo možno pohybovat podstatně rychleji a nové 3D body se detekovaly lépe. S tímto objektivem pak bylo možno zprovoznit i mapování pomocí metody LSD-SLAM jak v budově, tak i ve venkovních podmínkách.

Pro správnou činnost těchto metod je nezbytné používat objektiv s výrazně velkým zorným úhlem.

3.3 Použití různých kamer

Brzy po odstranění problémů s objektivy a nastavením algoritmů jsem se dostal k dalšímu limitu, kdy docházelo k občasným výpadkům. Tím bylo podezření na nedostatečnou kvalitu kamery. Zprvu jsem měl za to, že by případné rozlišení obrazových dat mohlo pomoci ke zkvalitnění funkčnosti testovaných algoritmů. K testování jsem použil dvě kamery - původní kameru, kterou už Tyra používala k uživatelským záznamům a pak kameru s 5Mpx rozlišením v telefonu HTC EVO 3D.

původní kamera IDS ueye `UI-1221LE-C-HQ`

Z kamery jsem dokázal získat proměnlivě od 20 až po maximálně 30 snímků za sekundu v závislosti na snímaném okolí. Při vyšším osvětlení se rychlost snímání zmenšovala až se v externích podmínkách zcela zastavila. Problém jsem pak nalezl hlavně v délce uzávěrky, která byla nastavená na maximální hodnotu. Kameru bylo nutné při každém prostředí

Obrázek 3.5: Obraz pořízen s objektivem typu rybí oko.

speciálně nastavit. Někdy se vyplatilo využívat automatických hodnot, v tmavších místech bylo lepší si parametry kamery přesně určit aby se dosáhlo plného potenciálu kamery.

testování kamery v telefonu HTC EVO 3D (pouze mono)

Data z kamery byla přenášená přes WiFi do počítače. Pro přenos jsem použil balíček image_transport, který data přenášel v komprimované podobě. Na mobilním telefonu se mi nepodařilo nainstalovat Ubuntu, proto jsem byl nucen využít aplikaci pro přenos senzorových dat pro ROS.

Zjistil jsem, že používání mobilní kamery v algoritmech je posun k horšímu. Toto zhoršení připisuji hlavně čočce obsažené v mobilním zařízení a nedostatečnou plynulostí obrazu, způsobený bezdrátovým přenosem dat.

Na výše zmíněný problém jsem narazil až při kalibraci kamery v mobilním telefonu. Nedokázal jsem se dostat u hodnoty `RMS pixel error` pod 0.4. Použitá kalibrační data pak nejsou zcela dostačující k provozování měřících algoritmů pro mapování a vizuální odometrii.

3.4 Test náročnosti metod na různá zařízení

Metody jsem měl možnost vyzkoušet na třech různých zařízeních - notebooku ASUS ROG JH-71G s procesorem `i7-720 QM`[2], na atomboardu obsaženého v Tyře a na architektuře ARM díky tabletu ASUS Google Nexus 7 (starší verze) se čtyřjádrovým procesorem nVidia Tegra 3[3]. Snažil jsem se nasadit algoritmy na těchto zařízeních do ostrého provozu v reálném čase a zjistit případné projevy chování metod na daném zařízení.

[2] http://ark.intel.com/products/43122/Intel-Core-i7-720QM-Processor-6M-Cache-1_60-GHz
[3] http://www.nvidia.com/object/tegra-3-processor.html

Mezi kritéria hodnocení patřila hlavně spolehlivost a plynulost zpracovávání obrazových dat a kvalita výsledné trajektorie, popřípadě rekonstruovaného 3D modelu.

ASUS ROG JH-71G

Notebook jsem používal jako hlavní výzkumný stroj. Vycházel jsem z předpokladu. že co se nebude dát realizovat na tomto notebooku, nepůjde uskutečnit ani jinde. Výkon zde dostačoval naprosto pro cokoliv, a to i pro souběžný chod všech tří metod současně s plným vytížením počítače. Ve výstupech jednotlivých metod nebylo znát, že by se nějakým výrazným způsobem zhoršily. I s tímto dostupným výkonem byla kalibrace nástrojem PTAM poměrně náročná. Získat alespoň pět snímků pro výpočet kalibračních parametrů se ukázalo být v řádech několika minut, přestože bylo třeba pouze nastavit kameru do nějakého úhlu a zmáčknout na tlačítko "Grab Frame".

Nejnižší zatížení procesoru jsem pozoroval u metody SVO, která dokázala zaměstnat kompletně jedno jádro procesoru. Metoda toto zatížení držela stabilně i při výpadku. Metoda PTAM ke svému běhu spotřebovala takřka dvojnásobný výkon, než tomu bylo třeba pro algoritmus SVO. PTAM stabilně běžel na dvou jádrech, které dokázal plně zatížit. Při výpadku zprvu nárok na výkon klesl o pouhou osminu, kdy se čekalo o pokus znovulokalizaci 3D bodů. Postupem času požadavek na výkon klesl na obsazenost jednoho jádra, nicméně po obnově polohy se nárok metody na výkon počítače opět rozrostl na dvě jádra.

Nejnáročnější metodou se zde ukázala metoda LSD-SLAM. Tato metoda se v závislosti na rychlosti posunu kamery a množství snímků již pořízených v dané oblasti, pohybovala obecně mezi zatížením dvou až tří jader. Větší vliv měl na zatížení pohyb než rychlost přidávání nového prostoru.

Nezanedbatelný vliv má taky běh uzlu uEye pro obsluhu kamery a dodávání obrazových dat algoritmům, nicméně tento uzel spotřeboval pouze 25% výkonu jednoho jádra.

ASUS Google Nexus 7

Na tabletu byl nainstalován systém Ubuntu ARM desktop ve verzi 13.04. Oživování kamery nakonec probíhalo trochu odlišným způsobem než u standardní x64 architektury, ale tato změna by neměla mít žádný vliv na výsledky algoritmů.

Nejdříve jsem testoval kalibraci pomocí obou nástrojů. Výkon tabletu by určitě nestačil ani trpělivému uživateli pro kalibraci kamery. Tablet několikrát přestal pracovat na delší dobu a bylo velmi obtížné dokončit kalibraci. Podstatně lépe na tom byla kalibrace `camera_calibration` než za pomoci nástroje v metodě PTAM. Nicméně přes veškerou snahu se mi ani jedním nástrojem nepodařilo získat data, která byla naměřená při kalibraci na počítači.

Ačkoliv se mi metody dokázaly přeložit na tabletu, s jejich chodem vyvstaly další problémy. Ukázalo se, že tablet má s chodem metod problémy. Pomocí SVO se podařilo vygenerovat správnou trajektorii pohybu kamery, protože docházelo k častým výpadkům plynulých dat z kamery. K rozběhnutí dalších metod by bylo potřeba výrazným způsobem změnit jejich nastavení, která by však měla za následek výrazné snížení kvality výsledných hodnot z tohoto zařízení, přičemž minimálně metody SVO a PTAM by mohly být použity v omezené míře a za ideálních okolních podmínek.

Tyra

Nejdůležitější částí bylo testování metod na Tyře, na kterou by dané metody měly být nakonec instalovány. Pro testování se s kvadrokoptérou nelétalo, jelikož se jedná o test z pohledu výkonu kvadrokoptéry pro provoz těchto metod. Zde se nejlépe uplatňovala metoda SVO, která nejméně vytěžovala procesor atomboardu kvadrokoptéry. Společně se však s kamerou jednalo o celkové využití dostupného výkonu. Metoda se ukazovala jako stabilní a případný nedostatek výkonu neovlivnil funkčnost metody Metoda PTAM však musela být nepatrně přenastavená, aby se došlo k optimálnímu výsledku, bylo nutné upravit parametry [1]:

- `InitLevel` na 1,
- `MaxStereoInitLoops` na 4,
- `MaxPatchesPerFrame` na 300,
- `MaxKF` na 15,
- `UseKFPixelDist` na `true`
- `NoLevelZeroMapPoints` na `true`.

Úpravou těchto parametrů určitě došlo k ovlivnění výsledků metody. Poté metoda pracovala dobře, ačkoliv zde opět docházelo k plnému využití procesoru.

Pro metodu LSD-SLAM se však ukázalo, že nemá smysl využívat mapování při běhu kvadrokoptéry. Ukázalo se, že je naopak prozíravější nahrát si pomocí `rosbag` nasnímaná obrazová data, která je možno až pak na počítači zpětně rekonstruovat.

Teoreticky by šlo posílat v reálném čase nasnímaná obrazová data do nějakého výpočetního zařízení, například notebooku, který by je zpracoval a výsledná rekonstrukční data by pak poslal pro zobrazení uživateli na tablet. Tento model se mi částečně podařilo realizovat. Setkal jsem však s problémem v přenosu nekomprimovaných dat přes síť WiFi, kdy rychlost nebyla dostačující. K dispozici jsem měl jen router s rychlostí 54 Mbps.

3.5 Testování metod v různých prostředích

Metody jsem otestoval v různých prostředích. Velmi záleželo na světelných podmínkách a zdali se jednalo o měření uvnitř místnosti, nebo měření ve venkovním prostředí. V tomto bodě důležitou úlohu zaujímá kvalita nastavení kamery pro různá prostředí. I přes veškerá úsilí se zcela nepovedlo kompenzovat tyto odchylky prostředí pomocí změn parametrů kamery. Odchylky se daly pouze zmírnit do přijatelných podob pro algoritmy. I tak je ale třeba vyvarovat se při používání metod různým situacím.

Chování metod ve vnitřních prostorách

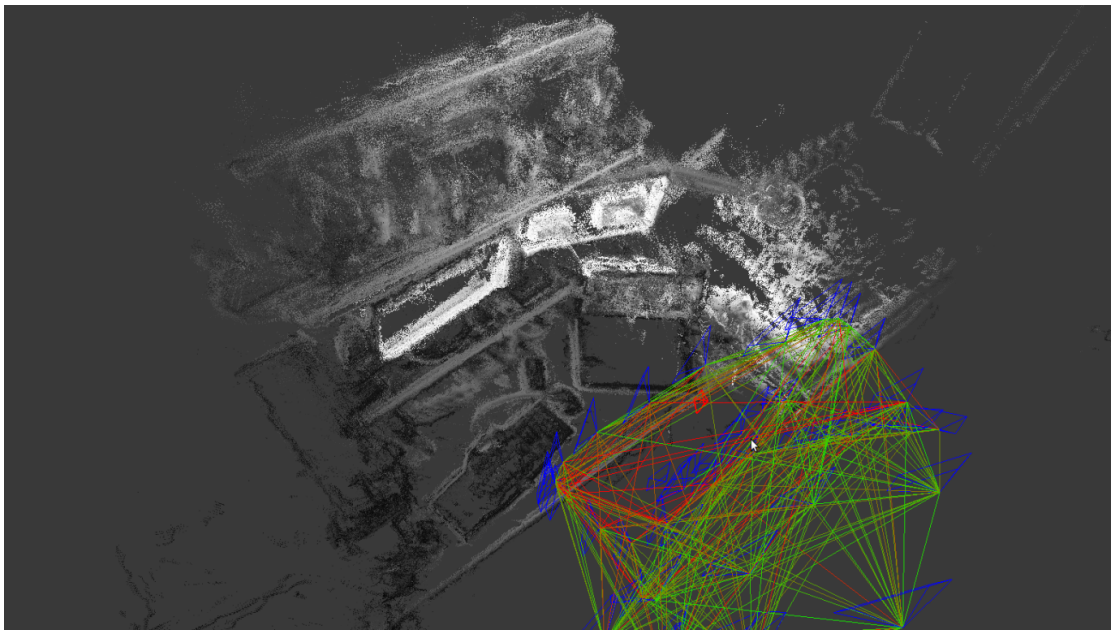
Při testování metod ve vnitřních prostorách jsem neměl kameru směřovanou svisle dolů. Většinou jsem cíleně zabíral prostor, kde se vyskytovaly nejrůznější předměty. Podlaha měla dost jednolitý vzor, který kamera nedokázala rozpoznat. Metody se na snímcích podlahy nedokázaly vůbec udržet.

Při normálním osvětlení

Za normální osvětlení jsem považoval denní světlo, kdy do místnosti nesvítilo příliš slunce a nedocházelo k oslňování kamery. Ukázalo se, že pro používání metod v uzavřených prostorách se jedná o ideální podmínky a nedochází k žádným větším výpadkům. Problém by mohl nastat v případě jednolitějšího prostředí snímaného kamerou, jako jsou například holé zdi nebo podlaha. Je tedy vhodné kamerou sledovat kontrastně členité prostředí. Rychlost závěrky na kameře nebyla nastavována automaticky, ukázalo se, že nejlepší variantou je, pokud je závěrka nastavena na maximum. Toto nastavení ale může způsobovat problémy s rušivými elementy, jako jsou například okna, a to i za oblačného dne.

Nízké osvětlení

Testování metod jsem také prováděl při večerních podmínkách, kdy bylo třeba mít k dispozici jako zdroj světla umělé osvětlení. Podstatou tohoto testu bylo, jak se metody vyrovnají s omezeným počtem údajů ve snímku pořízené v přirozeném prostředí, ve kterém by bylo možno metody provozovat.



Obrázek 3.6: Pracovní stůl zachycen mapovací metodou LSD-SLAM společně s vyobrazeným pozičním grafem propojující hlavní snímky za nižších světelných podmínek a s rušením od monitorů.

Ukázalo se, že metody PTAM a SVO se nedokážou uplatnit na částech, které nebyly nasvíceny umělým světlem. Navádění bylo tedy možno pouze na místech, která byla osvětlena nějakým výraznějším světlem. SVO prokazovalo větší výpadky než PTAM, který se dokázal na osvětlených částech lépe uchytit a prokázal se jako stabilnější metoda pro použití v těchto podmínkách.

Největším překvapením však byla metoda LSD-SLAM, která perfektně dokázala zmapovat daný prostor bez jediného zřetelného výpadku nebo vytvoření nečekaného šumu. Na obrázku 3.6 je vidět mírný šum bodů od pravého monitoru (viditelný zářící obdélník ve středu obrázku), kdy byla lehce oslněná kamera, ale jinak bylo vše zmapováno velmi dobře.

S předchozí kalibrací, která byla o 0.03 desetiny na `RMS pixel error` horší byl výskyt tohoto šumu od monitorů výraznější.

Při výskytu rušivých elementů

Mezi důležité rušivé elementy je možno zařadit rozsvícená světla, monitory počítačů, lampičky nebo také okna. Tyto elementy bohužel ovlivňují činnost metod negativním způsobem. Prakticky jsem se snažil s tímto problémem vypořádat pomocí automatické korekce obrazových dat při zpracovávání scény kamerou. Problém lze částečně vyřešit automatickým nastavením času uzávěrky. Přechod však není okamžitý a kameře trvá několik sekund, než se přizpůsobí novým světelným podmínkám. Při výskytu těchto elementů je vhodné s nimi předem počítat a přizpůsobit jim pohyb kamery přes tyto ovlivněné části prostoru.

Obrázek 3.7: Metoda SVO při běhu v místnosti s rušivým elementem v podobě slunečního záření přicházejícího zprava skrze okno.

Jak můžete vidět na obrázku 3.7, na méně osvětlené části slunečním zářením pracuje metoda SVO velmi dobře. Problém nastává při posunu kamery k více osvětleným částem, kdy se kamera ještě nestačí zcela adaptovat. Rychlejší přesun pak znamená výpadek polohy. Tímto problémem trpí všechny metody v závislosti na použité kameře.

Chování metod ve venkovním prostředí

Pro testování metod jsem měl většinou kameru namířenou směrem dolů kromě metody LSD-SLAM. Při nízkých výškách se toto však ukázalo jako nevhodný faktor pro kvalitu běhu různých metod. Lepší by bylo ponechat kameru ve vodorovné pozici vůči zemi, pokud se nejedná o otevřený prostor, jako je například pole. Pravděpodobně by bylo nejlepší mít otočnou kameru, která se bude natáčet z kolmé polohy k zemi do vodorovné a naopak v závislosti na výšce.

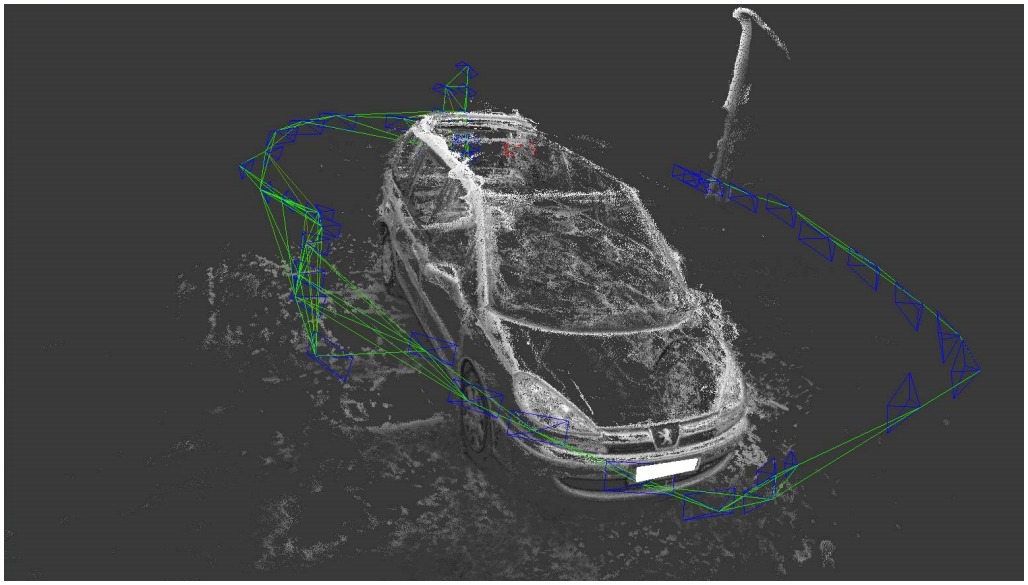
Při slunečném počasí

Slunečné počasí se ukázalo jako ne zrovna ideální faktor pro používání metod. Dochází zde k častému oslňování a kamera se chrání proti zničení. Během oslnění kamera neposkytuje žádná obrazová data a projevuje se jakoby přestala pracovat. Až se dostane z oslnění, po pár sekundách se znovu rozběhne. Stejný efekt vzniká, pokud je kamera výrazně oslněná například z odlesku oken nebo aut. Ačkoliv je uzávěrka nastavená na minimum, nestačí to k tomu, aby se těchto situací dalo vyvarovat.

Pro SVO a PTAM to v takovém případě znamená opětovnou inicializaci a všechna předchozí data o poloze jsou zbytečná. Už neexistuje žádná souvislost mezi starými a novými daty. LSD-SLAM se s tímto vyrovnává podobně avšak následek zde způsobuje větší problém. Pokud nastane výpadek, ze kterého se metoda nedokáže zotavit, začne také mapovat nanovo. Předchozí zmapované oblasti jsou již vyobrazeny a proto dochází ke skládání již zmapovaných objektů, například pod jiným úhlem a s jiným posunutím přes sebe. Validní nahraná data jsou pak pouze do nebo až po výpadku kamery.

Při zataženém počasí

Nejideálnější podmínky pro nejlepší funkčnost metod se ukázaly při oblačném až zataženém počasí, kdy nesvítilo slunce. Kameru tak neoslňovaly žádné odrazy či přímo slunce. To se hodně projevilo při mapování pomocí metody LSD-SLAM a ostatní poziční metody pak měly lepší odolnost vůči výpadkům.

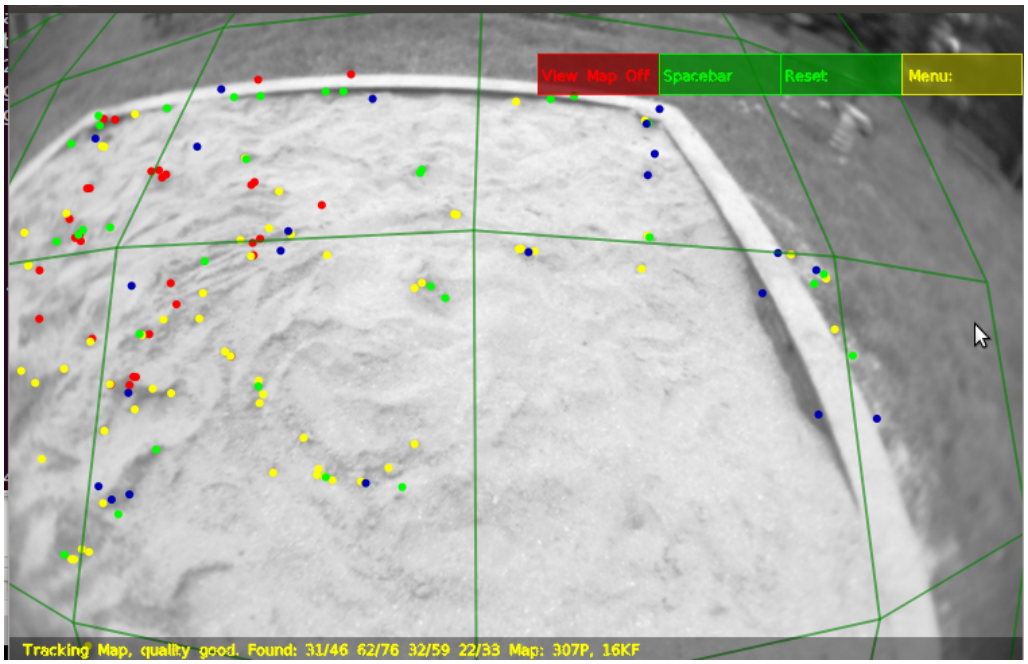


Obrázek 3.8: Použití LSD-SLAM při zataženém počasí pro zmapování auta Peugeot 807.

V obraze 3.8 i při stříbrné barvě původního auta nejsou vidět žádná rušení z odlesků, která by určitě byla patrná při slunečném počasí. Video bylo pořizováno pomocí tabletu, jelikož jsem neměl dostupný jiný hardware pro nahrávání obrazových bag souborů.

Chování metod při jednolitějším povrchu

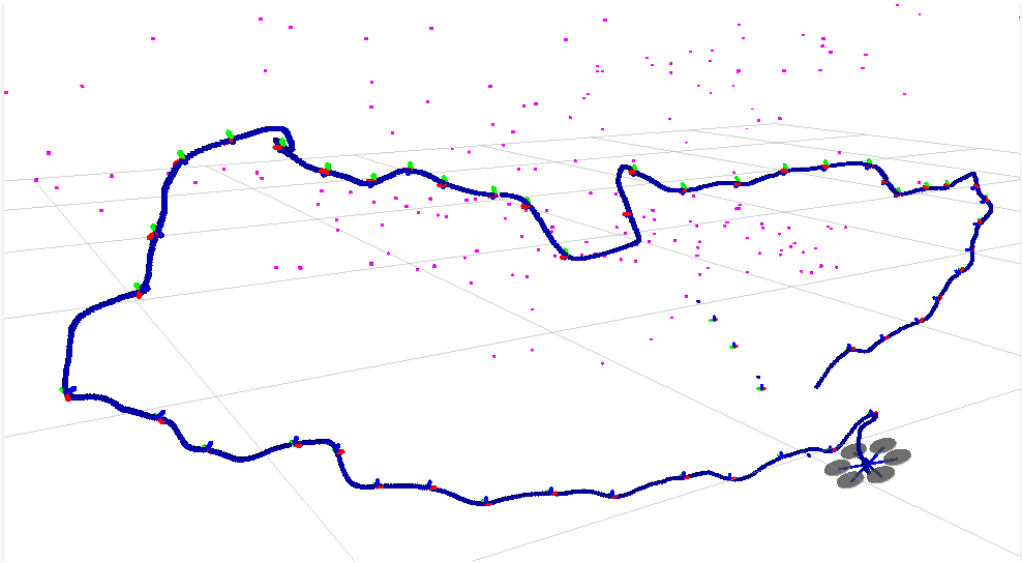
Zajímavým zjištěním bylo nalezení nedostatku u metody SVO ve výškách do jednoho metru od venkovního povrchu. V některých jednolitějších prostředích, jako je například tráva nebo pískoviště, má někdy problém s inicializací a velmi rychle dochází k výpadku. PTAM se oproti SVO ukázal jako mnohem stabilnější variantou pro tato prostředí. Dokázal se udržet s občasnými výpadky v nízkých výškách jak nad pískovištěm (obr. 3.9), tak i nad trávou. Ve vyšších výškách od jednoho a půl metru nad zemí však dokázala metoda SVO pomalu držet krok s metodou PTAM, kde také dokázala sledovat trajektorii pohybu nad jednolitějším povrchem (obr. 3.10).



Obrázek 3.9: Metoda PTAM a test zachycení trakčních bodů v pískovišti.

Nejstabilněji z těchto metod se však opět ukázala metoda LSD-SLAM, pomocí které bylo možno zmapovat i povrch pískoviště. Netrpěla žádným výpadkem a dokázala celou dobu mapovat prostor snímaný kamerou.

V nastavení kamery se velmi osvědčilo nastavení parametru `pixel_clock` u kamery při spouštění na vyšší hodnotu - přesněji na hodnotu 35, kdy se výsledný obraz z kamery zlepšil a byly v záběru rozeznat lepší detaily. Je třeba ale dávat pozor na přehřátí ovládacího čipu kamery.



Obrázek 3.10: Metoda SVO zachycující trajektorii pohybu nad travnatým povrchem ve výšce 1,5m bez jediného výpadku.

Kapitola 4

Závěr

Testování hodně zkomplikoval fakt, že létání s Tyrou je možné pouze v omezené míře, a to ve výšce pouhých pár metrů nad zemí (1-2m), kdy je kvadrokoptéra uvázána na zajišťovací tyč, aby v případě problému nehavarovala. Toto bezpečnostní opatření bylo navrženo v předchozích letech poté, kdy Tyra havarovala a došlo k jejímu dlouhodobému poškození. Testování zde pak mohlo probíhat pouze v omezené výšce. Ukázalo se, že při použití těchto metod v takové výšce je lepší mít kameru natočenou vodorovně v závislosti na rozmanitosti okolního prostředí.

Používání těchto metod venku s kamerou, která je k dispozici na Tyře je ideální pouze při oblačných podmínkách, kde nedochází k oslnění kamery. Jedině tak lze bezpečně zajistit ideální výsledky a takřka bezproblémový chod metod. Také metodám nedělá problém pracovat správně těsně před tím, než se začne stmívat. Předem je však nutné správně nastavit kameru pro dané prostředí. K tomuto postačí i subjektivní pocit uživatele.

Pro získání lepších výsledků nehraje přílišnou roli samotné nastavení metod nebo používání různých módů metod. Nastavení by hrálo větší roli při použití lepší kamery a také objektivu na kterém stojí správná a stabilní funkčnost metod. Vhodnější kamera s objektivem by měla mít pro tento účel ještě větší zorný úhel a taktéž je třeba větší snímkovací rychlosti, alespoň 50 fps a výše.

Na Tyře lze bez větších problémů použít metody PTAM a SVO, kde PTAM se ukázala při současné kameře jako stabilnější metoda. Trpěla méně častými výpadky než SVO. Metoda SVO by se však mohla při použití lepší kamery ukázat jako lepší volba. Výstupy z obou metod se pak dají využit k polohování Tyry v prostoru a hlavně k registrování posunu zařízení.

K běhu LSD-SLAM doporučuji použít externí výpočetní zařízení, které dané výsledky bude buď zpětně nebo v reálném čase promítat uživateli a rekonstruovaná data bude zasílat například do mobilního zařízení.

Literatura

[1] Achtelik, M.: ptam - ROS Wiki. [Online; citováno 23-02-2015].
URL http://wiki.ros.org/ptam

[2] Christian Forster, D. S., Matia Pizzoli: SVO: Fast Semi-Direct Monocular Visual Odometry. Technická zpráva, Robotics and Perception Group, University of Zurich, Switzerland, 2014, [Online; Citováno 28-03-2015]. Dostupné z http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf, web:http://www.ros.org/news/2014/06/new-package-svo-semi-direct-monocular-visual-odometry.html.

[3] Collins, R.: Lecture 30: Video Tracking: Lucas-Kanade. CSE486, Penn State, department of Computer Science and Engineering, The Pennsylvania State University.
URL http://www.cse.psu.edu/~rtc12/CSE486/lecture30.pdf

[4] Daniilidis, K.: The Page of Omnidirectional Vision. [Online; citováno 17-01-2015].
URL http://www.cis.upenn.edu/~kostas/omni.html

[5] Engel, J.; Schöps, T.; Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. Technická zpráva, Computer Vision Group, Technische Universität München, Deutschland, September 2014, [Online; Citováno 16-04-2015]. Dostupné z http://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf, web:http://vision.in.tum.de/research/lsdslam.

[6] Fermoso, J.: Omni-directional Camera Sensor Makes Pictures Look Like HAL-9000 Clones. [Online; citováno 19-01-2015].
URL http://www.wired.com/2008/07/olympus-omni-di/

[7] GmbH, A. T.: AscTec Pelican /// R&D Drohne für Computer Vision ; SLAM : Ascending Technologies GmbH. [Online; citováno 30-01-2015].
URL http://www.asctec.de/uav-uas-drohnen-produkte/asctec-pelican/

[8] Hallenbeck, K.: ueye - ROS Wiki. [Online; citováno 15-01-2015].
URL http://wiki.ros.org/ueye

[9] Ing. Vítězslav Beran, P.: RoboLab Tyra. [Online; citováno 30-03-2015].
URL http://merlin.fit.vutbr.cz/wiki/index.php/RoboLab_Tyra

[10] Ji Zhang, M. K.; Singh, S.: Real-time Depth Enhanced Monocular Odometry. Technická zpráva, Robotics Institute at Carnegie Mellon University, Pittsburgh, 2014, [Online; Citováno 16-04-2015]. Dostupné z http://people.csail.mit.edu/kaess/pub/Zhang14iros.pdf.

[11] Klein, G.; Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. Technická zpráva, Active Vision Laboratory, Department of Engineering Science, University of Oxford, Nara, Japan, November 2007, [Online; Citováno 15-03-2015]. Dostupné z http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf.

[12] O'Kane, J. M.: *A Gentle Introduction to ROS*. 2014, ISBN 978-1492143239. URL http://www.cse.sc.edu/~jokane/agitr/

[13] Prof. Oldrich Zmeskal, P.: 03 Vady optických zobrazovacích prvků. [Online; citováno 25-03-2015]. URL http://www.fch.vutbr.cz/~zmeskal/obring/presentace_2003/03_vady_optickych_zobrazovacich_prvku.pdf

[14] Rodriguez, M.: Implementing Lukas and Kanade?s Optical Flow. [Online; citováno 03-01-2015]. URL http://www.cs.ucf.edu/~mikel/Research/Optical_Flow.htm

[15] Scaramuzza, D.: Omnidirectional Camera. Technická zpráva, GRASP Lab, University of Pennsylvania, [Online; Citováno 20-04-2015]. Dostupné z http://rpg.ifi.uzh.ch/docs/omnidirectional_camera.pdf.

[16] Silberman, N.: Nathan Silberman. [Online; citováno 17-01-2015]. URL http://cs.nyu.edu/~silberman/

[17] Wikipedia: Robot Operating System - Wikipedia, the free encyclopedia. [Online; citováno 29-12-2014]. URL http://en.wikipedia.org/wiki/Robot_Operating_System

[18] Wikipedia: Zorné pole - Wikipedia, the free encyclopedia. [Online; citováno 28-03-2015]. URL http://cs.wikipedia.org/wiki/Zorn%C3%A9_pole

[19] Wikipedia: Visual odometry - Wikipedia, the free encyclopedia. 2014, [Online; citováno 08-01-2015]. URL http://en.wikipedia.org/wiki/Visual_odometry

[20] Wikipedia: Optical flow - Wikipedia, the free encyclopedia. 2015, [Online; citováno 08-03-2015]. URL http://en.wikipedia.org/wiki/Optical_flow

[21] Winkler, Z.: Měření rychlosti - robotika.cz. 2005, [Online; citováno 07-01-2015]. URL http://robotika.cz/guide/filtering/cs

[22] Winkler, Z.: Odometrie - robotika.cz. 2005, [Online; citováno 07-01-2015]. URL http://robotika.cz/guide/odometry/cs

[23] Wirth, S.: viso2_ros - ROS Wiki. [Online; citováno 01-01-2015]. URL http://wiki.ros.org/viso2_ros

Příloha A

Obsah CD

- Zdrojové kódy metod v src/ roztříděné podle ROS workspace (catkin a rosmake)
- Kalibrační a konfigurační soubory metod v config/ roztříděné do složek
- Dodatečné obrázky získané pomocí metod ve složce images/
- Spouštěcí skripty pro jednotlivá prostředí ve složce run/ vysvětlujícím souborem README
- Demonstrační rosbag soubory ve složce rosbag/
- Zdrojové kódy této bakalářské práce ve složce bp/src
- pdf dokument této bakalářské práce ve složce bp/
- Demonstrační video
- Plakát