České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Cabicar**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Rekonstrukce 3D modelu prostředí helikoptérou**

Pokyny pro vypracování:

1. Prostudujte princip měření vzdálenosti kamerovým stereo-párem a vhodné metody současné lokalizace a mapování (SLAM) pro registraci jednotlivých měření a vytvoření 3D mapy.
2. Seznamte se s použitím kamery ZED.
3. Navrhněte a implementujte metodu vhodnou pro použití na malé bezpilotní helikoptéře.
4. Proveďte experimenty v reálném prostředí a vyhodnoťte použitelnost metody pro helikoptéru, její přesnost, robustnost a omezení.

Seznam odborné literatury:

[1] Weingarten, J. Feature-based 3D SLAM. PhD Thesis, Swiss Federal Institute of Technology Lausanne, EPFL, no 3601, Dir. Roland Siegwart, (2006).
[2] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions, Journal of the Optical Society of America, 4:629-642, 1987
[3] Mázl Roman - Lokalizace pro autonomni systemy, disertační práce – Praha, 2007

Vedoucí: Ing. Jan Chudoba

Platnost zadání: do konce letního semestru 2016/2017

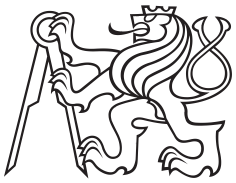
L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 2. 2016

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídící techniky

Rekonstrukce 3D modelu prostředí helikoptérou

Jan Cabicar

Vedoucí: Ing. Jan Chudoba
Studijní program: Kybernetika a robotika
Obor: Systémy a řízení
Květen 2016

Poděkování

Děkuji svému vedoucímu práce, panu Ing. Janu Chudobovi, za ochotu a pomoc při vypracování této práce. Dále děkuji svému kolegovi Davidu Česenkovi za spolupráci při konstrukci kvadrokoptéry.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 25. 5. 2016

.....

Abstrakt

Práce se zabývá algoritmem pro tvorbu 3D modelů prostředí. Hlavním zaměřením této práce je použití již existujících algoritmů pro mapování a jejich aplikaci pro stereokameru. Použitý algoritmus je založený na sledování význačných prvků v pořízených snímcích pomocí KLT trackeru. K výpočtu transformace mezi jednotlivými snímky využívá algoritmus RANSAC a Hornovu metodu. Pro implementaci algoritmu byla použita knihovna Mobile Robotic Programming Toolkit. Dále se práce věnuje i konstrukci mobilní platformy pro vykonávání experimentů.

Klíčová slova: SLAM, RANSAC, KLT tracker, Hornova metoda, MRPT, stereokamera ZED

Vedoucí: Ing. Jan Chudoba
BLOX:7.91,
Evropská 11,
16000 Praha 6

Abstract

Thesis is concerned with algorithm for creation of 3D models of environment. Main focus is application of existing algorithms for mapping with use of stereocamera. Used algorithm is based on tracking features in pictures using KLT tracker. For computing transformation RANSAC and Horn's method is used. Algorithm implementation was done using Mobile Robotic Programming Toolkit. It also describes construction of mobile platform for experiments executions.

Keywords: SLAM, RANSAC, KLT tracker, Horn's method, MRPT, stereocamera ZED

Title translation: 3D model reconstruction of environments using helicopter

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V posledních letech zažívají velký rozmach malé bezpilotní helikoptéry. Využití nalézají nejen v podobě hraček, ale také v komerčním sektoru. Nejznámějším případem je pravděpodobně internetový obchod Amazon, který nedávno oznámil, že tyto stroje hodlá využít k rozvozu zboží. Za další nejvíce rozšířenou aplikaci helikoptér se dá považovat průzkum a mapování neznámého prostředí.

Helikoptéry mohou být autonomní nebo jsou vzdáleně ovládané člověkem. Takové stroje patří mezi mobilní roboty. Základní úlohou, kterou se roboty zabývají, je orientace v prostředí. Úloha se dá rozložit na dva základní podproblémy: lokalizaci a mapování. Oba podproblémy jsou na sobě závislé - jedná se o tzv. problém „slepice a vejce“: pro lokalizaci potřebujeme dobrou mapu a stavba mapy vyžaduje informaci o přesné poloze robota. Výstupem je pak např. vizuální mapa neznámého prostředí. V angličtině se tento typ úlohy nazývá SLAM - Simultaneous Localization And Mapping.

Robotické přístroje v současnosti nalézají uplatnění i v lidských obydlích. Netýká se to jenom helikoptér, ale třeba i robotických vysavačů (Dyson, Neato Robotics) či sekaček. Levnější z nich se sice v prostoru neorientují (využívají v podstatě jen jednoduchý algoritmus pro vyhýbání se překážkám), ale dražší modely už mají v sobě zabudovaný SLAM a plánování cesty. Díky tomu jsou si schopny vytvářet model prostředí, zjišťovat svoji polohu v něm a plánovat cestu pro úplné pokrytí prostoru.

Klíčovými faktory problému jsou také různé dálkoměrné senzory, kterými daný robot „vidí“. Mezi nejlevnější patří ultrazvukové senzory, k nejdražším naopak laserové scannery. Cena je dána hlavně přesností - scannery jsou přesnější než sonary. Další vhodnou volbou mohou být 3D kamery, např. v poslední době rozšířený Kinect od společnosti Microsoft.

Náplní práce je vytvoření modelu uzavřených prostor s využitím výše zmíněných technik. K rekonstrukci scény byla použita 3D stereo kamera, kterou nesla námi vyrobená modelářská kvadrokoptéra. Použitý algoritmus pro SLAM využívá sledování význačných prvků ve snímcích a kombinace algoritmu RANSAC s Hornovou metodou k výpočtu pozic robotu.

Kapitola 2

Kamery a jejich využití při měření vzdálenosti

Měření vzdálenosti pomocí kamer je poměrně náročné vzhledem k faktu, že kamery snímají trojrozměrný prostor a převádějí ho do dvourozměrného zobrazení. Tyto problémy se řeší využitím více senzorů a kamer. Nejčastějšími příklady jsou stereo kamery nebo kamery doprovázené laserovým scannerem. Existuje několik možných způsobů měření vzdálenosti, které však nejsou aplikovatelné u problému jako je SLAM, který vyžaduje získávání měření v reálném čase.

2.1 Model kamery

Při záznamu obrazu kamerou se ztratí informace o vzdálenosti objektů. Dojde k tomu proto, že se 3D souřadnice snímaného bodu vzhledem k souřadnému systému kamery transformují do 2D souřadnic obrázku. Promítnutí bodu do obrazové roviny realizované ideální dírkovou kamerou je popsáno touto rovnicí:

$$l \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = KR \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.1}$$

kde l je měřítko, u a v jsou 2D souřadnice promítnutého bodu v obrazové rovině, K je matice vnitřních parametrů kamery, R je matice vnějších parametrů kamery a X, Y a Z jsou 3D souřadnice promítaného bodu.

Matice vnějších parametrů popisuje polohu kamery v souřadném systému a transformaci snímané scény vzhledem ke kameře. Matice vnitřních parametrů popisuje transformaci z 3D souřadnic promítaného bodu v souřadném systému kamery do 2D souřadnic bodu promítnutého do obrazové roviny. Matice K je dána touto rovnicí:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

Parametry této matice mají následující význam:

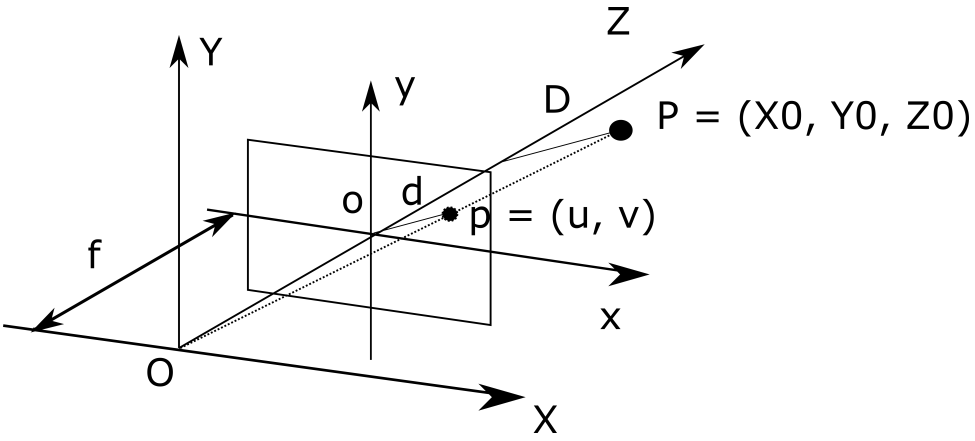
c_x, c_y ... souřadnice středu obrazové roviny v souřadnicích obrázku.

f_x, f_y ... ohnisková vzdálenost. U ideální dírkové kamery jsou oba parametry stejné.

s ... zkosení osy. U ideální dírkové kamery je většinou nulové.

Rozmístění parametrů v matici je dáno povahou matematických operací, které chceme s maticí provádět. Pokud máme souřadnice transformované maticí vnějších parametrů, musíme je ještě posunout a naškálovat.

Na obrázku 2.1 můžeme vidět promítnutí bodu P na obrazovou rovinu. Jak již bylo řečeno, tímto procesem ztratíme souřadnici Z0.



Obrázek 2.1: Model dírkové kamery

K rekonstrukci scény můžeme přistoupit po změření vzdálenosti Z0. Na obrázku 2.1 najdeme podobné trojúhelníky O, p, o a O, P, Z0. Z těchto trojúhelníků můžeme odvodit následující vzorec 2.3, kde d je vzdálenost zobrazeného bodu od optické osy, D je vzdálenost skutečného bodu od optické osy, f je ohnisková vzdálenost a $Z0$ je souřadnicí získanou z měření.

$$\frac{d}{f} = \frac{D}{Z0} \tag{2.3}$$

Vzorec upravíme tak, že místo vzdáleností bodu od optické osy dosadíme u a v, což jsou souřadnice zobrazeného bodu v souřadném systému obrázku, který má podle konvencí počátek v levém horním rohu a osa y směřuje dolů. Tyto souřadnice vztáhneme k počátku O. Výpočet souřadnic X a Y bodu je popsán rovnicemi 2.4 a 2.5.

$$X = \frac{u - c_x}{f_x} \cdot Z \tag{2.4}$$

$$Y = \frac{v - c_y}{f_y} \cdot Z \tag{2.5}$$

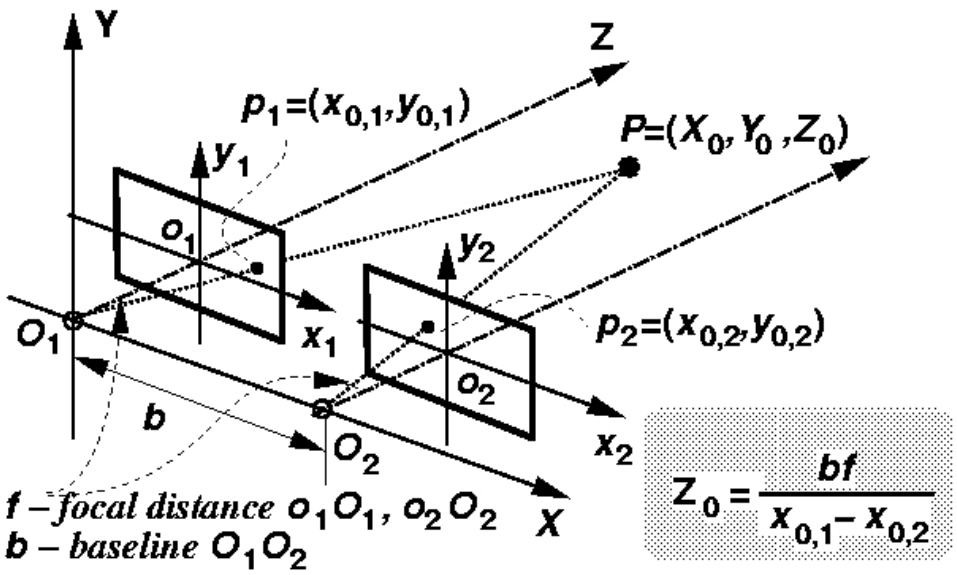
2.2 Metody pro měření vzdálenosti

2.2.1 Pasivní triangulace

Tato metoda se nazývá pasivní triangulace proto, že k měření vzdálenosti se nevyužívá přídavný zdroj světla. Pracuje na podobném principu jako lidské oči. K získání vzdálenosti jsou zapotřebí dvě kamery s rovnoběžnými optickými osami, jejich kalibrační parametry a vzájemná vzdálenost. Pro správné měření vzdálenosti bodu potřebujeme, aby byl viditelný v zorném poli obou kamer.

Rozteč kamer má vliv na přesnost měření - s větší vzdáleností roste přesnost, ale zmenšuje se prostor snímaný oběma kamerami.

Body ve snímaném prostoru se souřadnicemi x, y a z snímají dvě kamery a každá je vidí v jiném místě. K získání vzdálenosti potřebujeme najít korespondující body ve snímcích z obou kamer. Z nalezených korespondencí můžeme spočítat jejich vzdálenost. K odvození vzorce použijeme schéma stereo kamery na obrázku 2.2.



Obrázek 2.2: Stereo triangulace[10]

Z podobnosti trojúhelníků O1, p_1, o1 a O1, P, Z_0 vyplývá vztah 2.6.

$$\frac{x_{0,1}}{f} = \frac{X_0}{Z_0} \tag{2.6}$$

Z podobnosti trojúhelníků O2, p_2, o2 a O2, P, Z_0 vyplývá vztah 2.7.

$$\frac{x_{0,2}}{f} = \frac{X_0 - b}{Z_0} \tag{2.7}$$

Souřadnice zobrazeného bodu $x_{0,1}$ a $x_{0,2}$ jsou vyjádřené v souřadné soustavě obou kamer, X_0 a Z_0 jsou skutečné souřadnice bodu, b je rozteč optických

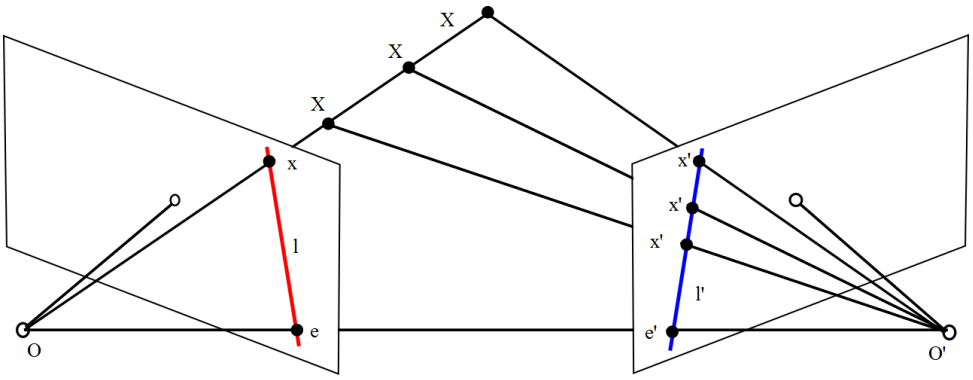
středů kamer, tzv. baseline a f je ohnisková vzdálenost. Odečtením rovnic 2.6 a 2.7 získáme vztahy 2.8 a 2.9 pro výpočet *Z0*.

$$\frac{x_{0,1} - x_{0,2}}{f} = \frac{b}{Z_0} \tag{2.8}$$

$$Z0 = \frac{bf}{x_{0,1} - x_{0,2}} \tag{2.9}$$

Rozdíl $x_{0,1} - x_{0,2}$ se nazývá horizontální paralaxa nebo také disparita.

Pro použití vzorce potřebujeme znát $x_{0,1}$ a $x_{0,2}$, kde $x_{0,1}$ je souřadnice bodu snímaného levou kamerou a $x_{0,2}$, což je souřadnice toho samého bodu snímaného pravou kamerou. K získání parametrů potřebujeme najít bod ve snímcích z obou kamer. Nejjednodušším řešením je porovnání intenzit a barev všech bodů. K zúžení výběru využijeme tzv. epipoláru. Epipolára je přímka, která je projekcí spojnice pozorovaného bodu se středem levé kamery na pravou kameru. Na obrázku 2.3 je tato přímka zvýrazněna červeně v pravé obrazové rovině. Hledaný korespondenční bod pak leží na této přímce.



Obrázek 2.3: Epipolární přímka[11]

Podstata metody přináší úskalí ve chvílích, kdy je snímaná scéna jednolitá, má stejnou strukturu a barvu - např. bílá zeď. Problém se dá částečně vyřešit hledáním korespondujících skupin bodů ,čímž se zlepší výpočetní náročnost problému.

2.2.2 Metoda Time-of-Flight

Metoda Time-of-Flight je založená na měření doby letu světla. Pro měření vzdálenosti se nejčastěji používá rozmítaný světelný paprsek, kdy měříme dobu mezi vysláním světelného paprsku a přijetím jeho odrazu od snímaného objektu. Využívá se faktu, že rychlost světla je konstantní, z čehož vyplývá, že doba letu t_l je dána vztahem:

$$t_l = \frac{2D}{c} \tag{2.10}$$

kde c je rychlost světla a D je vzdálenost daného objektu. Vyjádřením D získáme vztah 2.11 pro výpočet vzdálenosti.

$$D = 0.5 \cdot t_l \cdot c \tag{2.11}$$

Problém zvolené metody nastává ve chvílích, kdy chceme s velkou přesností měřit kratší vzdálenosti. Situaci lze v praxi řešit modulací vysílaného světelného paprsku sinusoidovým nebo čtvercovým signálem[12] a následným měřením fázového posunu mezi vyslaným a přijatým paprskem. Velkou nevýhodou oproti stereokamerám je právě rozmítaný paprsek, který se může na odrazivém povrchu odrazit jinam, než zpět do kamery.

2.2.3 Další metody

Mezi další metody měření vzdálenosti patří např. rekonstrukce tvaru ze stínování, což je soubor technik, které zjišťují normály povrchů za různého osvětlení. Dále lze zmínit metodu, při níž je pořízeno více snímků s odlišnou hloubkou ostrosti[1]. Vzdálenost zaostřených objektů pak lze spočítat za pomoci známých vlastností čoček.

Kapitola 3
SLAM

SLAM je spojením dvou podproblémů - lokalizace a mapování. Oba problémy samostatně jsou náročné, protože se jedná o zmíněný problém slepice a vejce. Pro spojení obou problémů potřebujeme znát počáteční polohu robota. Poté může robot stavět inkrementálně mapu z dat (scanu) vhodného senzoru. Jakmile se robot posune, vypočítá svoji novou polohu a pak upraví mapu podle nového scanu. [3]

Dnes již existuje mnoho přístupů ke SLAMu založených na různých metodách. První počátky výzkumu v této oblasti pocházejí z devadesátých let minulého století a byly založené na rozšířeném Kalmanově filtru[3]. Dnes existuje mnoho postupů pro lokalizaci založených na pravděpodobnostním rozložení (Markovská lokalizace, Monte Carlo...), na srovnávání scanů (Iterative Closest Point, RANSAC) nebo přístupy, kde se hledají v prostředí význačné body a podle nich se roboty orientují (Landmarková lokalizace). Mapy lze dělit do dvou kategorií: mapy vizuální, srozumitelné pro člověka, a mapy pro orientaci robota v prostředí. Mezi vizuální mapy patří například symbolické mapy, topologické mapy a případně mřížky obsazenosti. Pro roboty jsou srozumitelné mapy založené na geometrických prvcích[3] - používají se např. mřížky obsazenosti. Všechny postupy pak lze kombinovat podle požadavků na řešení úloh. [4]

Výše uvedená řešení se většinou provádějí pouze dvourozměrně kvůli jednoduchosti implementace. Výpočetní a paměťová náročnost 2D SLAMu je také menší. Velkou nevýhodu představuje skutečnost, že některé překážky robot nemusí zaznamenat - nemusí být v úrovni jeho senzoru a robot je „nevidí“. Řešení SLAMu v třírozměrném prostoru lze provést s využitím rotačního laserového scanneru nebo kamery, která spolu s obrazem snímá i vzdálenost objektů (viz kapitola 2).

3.1 Kalmanův filtr

Kalmanův filtr je algoritmus, který z měření zatížených šumem vytvoří odhad měřené veličiny. V případě SLAMu se jedná o odhad polohy. K reprezentaci neurčitostí a chyb v datech používá unimodální gaussovské rozložení, kde pozice robota je popsána střední hodnotou a rozptylem[4]. Rozšířený Kalmanův filtr se oproti standardnímu Kalmanovu filtru používá pro nelineární

systémy, kde linearizuje střední hodnotu a rozptyl.

SLAM založený na této metodě pracuje s výraznými body v prostoru. Body se nazývají landmarky a největší problém představuje jejich nalezení a asociace. Jedním z problémů algoritmu je klasifikace landmarku jako nového pozorování, i když se jedná o již zaregistrovaný landmark, a naopak, asociace ještě nezaregistrovaného landmarku s nějakým starším. Další problém představuje kvadratická složitost algoritmu - při rozpoznání každého landmarku je třeba aktualizovat stará pozorování a tím se omezuje maximální počet landmarků, které si je robot schopen zapamatovat.[4]

3.2 ICP SLAM

Iterative Closest Point je metoda, která spočívá v iterativním hledání nejlepší transformace mezi scany. Přístup vychází z metody nejmenších čtverců - minimalizuje se suma čtverců vzdáleností korespondujících bodů dvou scanů[4].

Algoritmus uvažuje první přijatý scan za počátek souřadnicového systému a všechny ostatní scany pak registruje vzhledem k prvnímu. Na začátku se inicializuje proměnná pro počet iterací k = 0. Pak následuje hledání transformace mezi novým pozorováním X a posledním registrovaným scanem P_k. Algoritmus probíhá následovně[5]:

Algorithm 1 ICP

1: Výpočet nejbližších bodů mezi X a P_k.
2: Výpočet transformace mezi těmito dvěma scany.
3: Do proměnné nového posledního registrovaného scanu P_{k+1} uložíme první scan posunutý podle nově vypočítané transformace.
4: Ukončení iterace pokud je rozdíl čtverců vzdáleností bodů menší než požadovaná přesnost algoritmu.

Výhody algoritmu spočívají v tom, že není nutné ve scanu hledat jakékoliv význačné prvky a ani nijak význačně upravovat naměřená data[5]. Další výhodou je pak nízká výpočetní náročnost. Na druhou stranu však tato metoda selhává, pokud se v měření nacházejí velké odchylky[5] a je velmi citlivá na chybu v počáteční orientaci robotu[4].

3.3 FastSLAM

Tento algoritmus kombinuje vlastnosti více metod, Kalmanova filtru a Monte Carlo lokalizace neboli particle filtrů. Particle nebo také částice jsou struktury, které obsahují Kalmanův filtr pro odhad pozic landmarků podmíněných trajektroií robotu. Dále metoda spočívá v odhadu posteriorních distribucí pozic robotu a landmarků. Využívá se Bayesovského přístupu a tyto distribuce se faktorizují. Technice se říká Rao-Blackwellized particle filtry.

Výhodou přístupu je menší výpočetní náročnost oproti algoritmům využívajícím Kalmanovy filtry. Zatímco tyto metody mají kvadratickou výpočetní náročnost, FastSLAM má výpočetní náročnost O(MK), kde M je počet particlů a K je počet landmarků. Díky tomu je tento algoritmus schopen uložit do paměti větší množství landmarků a může vytvářet větší mapy prostředí[13].

3.4 Použité metody

Technika pro SLAM použitá v této práci vychází z programu napsaného s Mobile Robotic Programming Toolkit (dále jen MRPT) pro senzor Kinect a upraveného pro účely této práce. Přístup staví na třech základních metodách: RANSAC[6], Kanade-Lucas-Tomasi feature tracker[7, 8] a Hornově quaternionu[9].

3.4.1 RANSAC

RANSAC je zkratkou pro RANdom SAmple Consensus (shoda náhodných vzorků). Jde o algoritmus pro odhad parametrů modelu popisujícího experimentální data. Jeho hlavní výhodou je imunita vůči velkým výchylkám v datech (v angličtině nazývané outliers). Tento přístup je nejvíce využívaný v odvětví zpracování obrázků a byl pro něj přímo navrhnut.

Nejznámější metodou pro odhad parametrů modelu je metoda nejmenších čtverců, která se ale snaží nalézt tyto parametry pro celý dataset s předpokladem, že velké výchylky lze zanedbat vzhledem k velikosti datasetu. Tento předpoklad není vždy splněn a je potřeba algoritmus, který není založen na průměrování a tím je právě RANSAC. Na rozdíl od ostatních metod začíná s nejmenší možnou podmnožinou datasetu, zatímco ostatní metody se snaží najít prvotní odhad nad celým datasetem, který pak iterativně zlepšují. RANSAC nakonec využívá metody nejmenších čtverců pro optimalizaci na výsledné podmnožině.

Algorithm 2 RANSAC

1: Z datasetu D se náhodně vybere nejmenší možná podmnožina P_1 pro odhad parametrů modelu.
2: Nalezený model se využije pro výběr podmnožiny S z datasetu D, která leží v intervalu daném maximální povolenou odchylkou.
3: Pokud je určená podmnožina S větší než předem určená hodnota t, která je odhadem počtu velkých odchylek v celém datasetu, použije se pro výpočet nových parametrů modelu metoda nejmenších čtverců.
4: Pokud je určená podmnožina S menší než t, náhodně se vybere nová nejmenší podmnožina P_2 a celý proces se opakuje.
5: Pokud se proces opakuje víckrát než předem určený maximální počet iterací, spočítají se parametry pro model nad největší nalezenou podmnožinou S nebo celý algoritmus skončí neúspěšně.

Tři důležité parametry algoritmu jsou toleranční interval pro maximální

možnou odchylku v datasetu, maximální možný počet iterací a odhad počtu velkých odchylek t.

Toleranční interval se podle teorie dá spočítat analyticky, nicméně je to pro aplikace v reálném čase nepoužitelné a proto bývá určen experimentálně.

Maximální možný počet iterací se určuje podle pravděpodobností. Nechť je tedy P pravděpodobnost, že jakýkoliv vybraný bod z datasetu je v tolerančním intervalu, Z je pravděpodobnost, že aspoň jedna z náhodně vybraných podmnožin neobsahuje bod s velkou výchylkou, N je počet iterací pro nalezení této podmnožiny a m je počet bodů v datasetu. Pak můžeme napsat vztah 3.1, kde $1 - Z$ je pravděpodobnost, že náhodně vybraná podmnožina obsahuje bod s velkou výchylkou a $1 - P^m$ je pravděpodobnost, že náhodně vybraný bod je bod s velkou výchylkou. Z tohoto vzorce už můžeme jednoduše odvodit následující vztah 3.2 pro výpočet potřebného počtu kroků.

$$(1 - P^m)^N = 1 - Z \tag{3.1}$$

$$N = \frac{log(1 - Z)}{log(1 - P^m)} \tag{3.2}$$

Odhad počtu velkých odchylek t se určí také podle pravděpodobností. Nechť je tedy Y pravděpodobnost, že daný bod je v tolerančním intervalu nesprávného modelu a n je počet bodů ve vybrané podmnožině. Pak je žádoucí, aby pravděpodobnost Y^{t-n} byla velmi malá. Ve skutečnosti neexistuje obecný způsob, jak určit Y. Můžeme ale říct, že je menší, než již dříve zmíněná pravděpodobnost P, která počítá se správností určeného modelu. Pokud budeme předpokládat, že $Y < 0.5$ a hodnota t-n $= 5$, máme víc jak 95% pravděpodobnost, že konečná podmnožina nebude určená z nesprávného modelu[6].

Tento odhad však může záviset také na tom, jakou metodu použijeme pro výpočet parametrů modelu nad vybranou podmnožinou. U různých přístupů se dá předem toto číslo určit za použití obecné metody.

3.4.2 Hornův quaternion

Hornova metoda představuje closed form solution pro hledání translace a rotace mezi dvěma kartézskými souřadnicovými systémy. Jako reprezentaci rotace využívá quaterniony, což je vektor o čtyřech složkách, který se dá reprezentovat jako komplexní číslo, které má tři různé imaginární části. Hlavní výhodou přístupu je skutečnost, že představuje řešení metody nejmenších čtverců, které nevyžaduje iterativní vylepšování parametrů.

Nechť máme dvě kartézské souřadnicové soustavy, levou a pravou. Abychom byli schopní najít translaci a rotaci mezi těmito soustavami, potřebujeme minimálně 3 body z levé soustavy a jejich korespondující body z pravé soustavy. Nechť máme n těchto bodů, přičemž n splňuje podmínku $n \geq 3$.

Algorithm 3 Hornova metoda

1: V obou soustavách nalezneme centroidy, které jsou definovány jako aritmetický průměr všech bodů v soustavě (součet všech bodů podělený jejich počtem).
2: Vypočítáme nové souřadnice bodů vzhledem k centroidu.
3: Pro každé dva páry nových souřadnic spočítáme všech devět možných součinů: $x_l x_r$, $x_l y_r$, $x_l z_r$, ..., $z_l z_r$.
4: Vypočítáme součet těchto součinů pro všechny body. Získáme:
$$S_{xx} = \sum_{i=1}^{n} x_{l,i} x_{r,i},\ S_{xy} = \sum_{i=1}^{n} x_{l,i} y_{r,i},\ ...,\ S_{zz} = \sum_{i=1}^{n} z_{l,i} z_{r,i}$$
5: Z těchto čísel vytvoříme následující 4x4 matici:
$$\mathrm{N} = \begin{pmatrix} (S_{xx}+S_{yy}+S_{zz}) & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{pmatrix}$$
6: Dále získáme z matice N vlastní vektory. Quaternion reprezentující rotaci je jednotkovým vektorem, který má stejný směr jako největší vlastní vektor matice N.
7: Spočítáme měřítko $s = \sqrt{\frac{\sum_{i=1}^{n} \left\| r_{r,i} \right\|^2}{\sum_{i=1}^{n} \left\| r_{l,i} \right\|^2}}$ kde r je vektor obsahující souřadnice bodu.
8: Translaci spočítáme jako rozdíl pravého centroidu a transformovaného levého centroidu.

3.4.3 Kanade-Lucas-Tomasi feature tracker

V některých algoritmech pro SLAM se využívá význačných bodů v prostředí, tzv. landmarků. Náš algoritmus je založený na podobném principu - hledá korespondující body (features), mezi dvěma snímky z kamery. Kanade-Lucas-Tomasi feature tracker (dále KLT tracker) představuje rychlý algoritmus k hledání těchto korespondencí.

Nechť F(x) a G(x) jsou funkce, které popisují hodnotu pixelu na pozici dané vektorem x ve dvou obrázcích. Hledáme takový vektor h, který minimalizuje rozdíl mezi F(x+h) a G(x) pro x v nějaké oblasti zájmu. Problém řešíme ve dvou rozměrech, protože význačné prvky hledáme ve snímku z kamery.

Nejjednodušším řešením je spočítat rozdíl mezi snímky pro všechny možné hodnoty h, což je ale velmi časově náročné. Na rozdíl od tohoto řešení, KLT tracker hledá lokální korespondence za pomoci váženého gradientu prostorové intenzity.

Hledáme rozdíl h mezi funkcemi F(x) a G(x) = F(x + h). Nejdříve provedeme následující linearizaci:

$$F(x+h) \approx F(x) + hF'(x) \tag{3.3}$$

Dále se snažíme minimalizovat následující míru rozdílu mezi funkcemi F(x+h) a G(x):

$$E = \sum_{x} [F(x+h) - G(x)]^2 \tag{3.4}$$

Chybu E minimalizujeme tak, že ji parciálně zderivujeme podle h a porovnáme s nulou:

$$0 = \frac{\partial E}{\partial h} \approx \sum_x [F(x) + hF'(x) - G(x)]^2 = \sum_x 2F'(x)[F(x) + hF'(x) - G(x)] \tag{3.5}$$

Odkud můžeme napsat vzorec pro h:

$$h \approx \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2} \tag{3.6}$$

Tento vzorec vylepšíme použitím vážící funkce, která je daná inverzí druhé derivace:

$$F''(x) \approx \frac{G'(x) - F'(x)}{h} \tag{3.7}$$

Protože chceme vážící funkci, můžeme se zbavit konstanty h a vytvořit následující vztah:

$$w(x) \approx \frac{1}{|G'(x) - F'(x)|} \tag{3.8}$$

Nyní už můžeme popsat iterační formu algoritmu:

Algorithm 4 KLT tracker

1: $h_0 = 0$
2: $h_{k+1} = h_k + \frac{\sum_x w(x)F'(x+h_k)[G(x)-F(x+h_k)]}{\sum_x w(x)F'(x+h_k)^2}$

Tento postup lze rozšířit tak, že vezmeme v úvahu rotace, translace a další transformace. Můžou se také pečlivěji vybírat prvky, které se sledují - sledují se pouze ty, u kterých vlastní čísla gradientní matice jsou větší než předem daná hodnota.

3.5 Použitý algoritmus pro 3D SLAM

Použitý algoritmus pro 3D SLAM byl převzat z knihovny MRPT a upraven pro stereokameru. Další úpravy zahrnovaly převedení algoritmu do offline režimu. Původní algoritmus v samostatném vlákně získával z kamery měření, ze kterých rovnou stavěl mapu. K nalezení korespondencí mezi jednotlivými měřeními je zapotřebí, aby se snímky co nejvíce překrývaly. Při delším běhu původního algoritmu se prodleva mezi aktualizací mapy a načtením nového měření zvětšovala, což mělo za následek zvětšování rozdílu mezi po sobě jdoucími snímky. Korespondence se pak nedaly nalézt a algoritmus skončil. Pro správnou funkci algoritmu nejdříve provedeme měření a pak ho necháme snímky zpracovat, což zaručí návaznost měřených snímků a nezávislost na době výpočtu.

Původni algoritmus si ukládal poslední viditelné význačné prvky, které použil jako referenci při hledání transformace. Tyto prvky si ukládal i když nenalezl novou transformaci, což mělo za následek znemožnění nalezení dalších

Algorithm 5 SLAM

```
    trackedFeats                          ▷ seznam sledovaných features
    tracker                                                ▷ KLT tracker
    globalPntsMap                                          ▷ hlavní mapa
    localPntsMap                                          ▷ pomocná mapa
    observation                                            ▷ nové měření
    previous_image                                       ▷ předchozí snímek
    lastVisibleFeats                  ▷ seznam posledních viditelných features
    curVisibleFeats              ▷ seznam features viditelných v novém snímku
    corrs          ▷ korespondence mezi curVisibleFeats a lastVisibleFeats
    relativePose                ▷ poloha robota vzhledem k předchozí pozici
    globalPose                                      ▷ současná poloha robota
    confidenceMap                                          ▷ mapa konfidencí
    step_num                                               ▷ mapa konfidencí
    numOfObs                                                 ▷ počet snímků
1:  nastavení parametrů pro tracker, inicializace mapy
2:  step_num ← 0
3:  numOfObs ← 100
4:  while step_num<pocetSnimku do
5:      observation ← newMeasurement
6:      theImg ← observation.newImage
7:      if step_num>1 then
8:          tracker.trackFeatures(previous_image, theImg, trackedFeats)
9:          převedení features do 3D a uložení do curVisibleFeats
10:         corrs ← correspondencies(curVisibleFeats, lastVisibleFeats)
11:         if corrs.size() > 3 then
12:             if relativePose ← findNewPose(corrs) then
13:                 globalPose ← globalPose + relativePose
14:                 lastVisibleFeats ← curVisibleFeats
15:                 confidenceMap ← Observation.newConfidence
16:                 localPntsMap ← Observation.newPointcloud
17:                 localPntsMap.applyConfidence(confidenceMap)
18:                 globalPntsMap.add(localPntsMap)
19:                 previous_image ← theImg
20:             end if
21:         end if
22:     end if
23:     if step_num == 1 then
24:         globalPose ← newPose(0, 0, 0, 0, 0, 0)
25:         confidenceMap ← Observation.newConfidence
26:         localPntsMap ← Observation.newPointcloud
27:         localPntsMap.applyConfidence(confidenceMap)
28:         globalPntsMap.add(localPntsMap)
29:         previous_image ← theImg
30:     end if
31:     step_num ← step_num + 1
32: end while
```

pozic robotu a nepříznivě to ovlivňovalo mapu. To jsme upravili tak, že si algoritmus pamatuje poslední viditelné prvky pouze pokud transformaci nalezne. Díky tomu mapa zůstává neovlivněná špatnými snímky. V malém procentu případů algoritmus může nalézt novou transformaci .

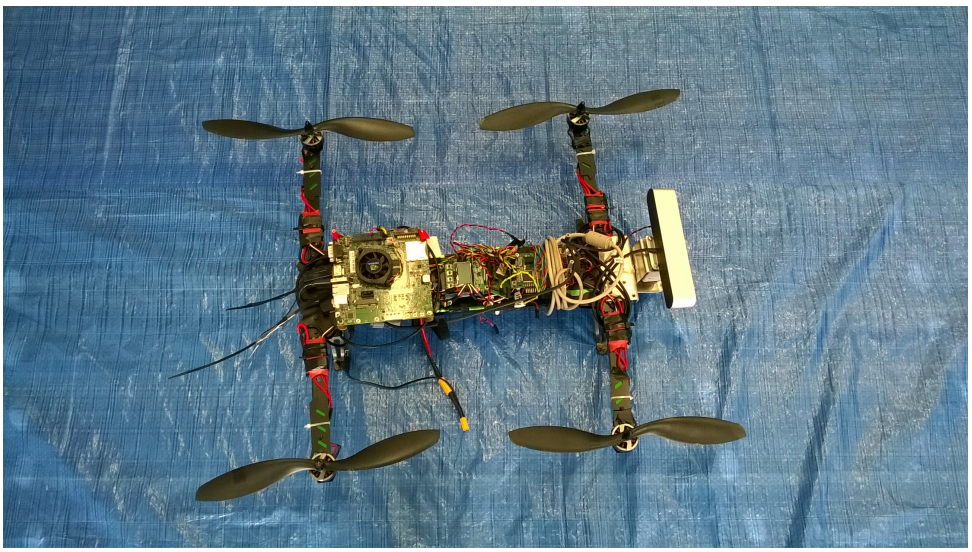
Použitá stereokamera poskytuje k měření i mapu konfidencí. To je matice, která popisuje jistotu měření vzdálenosti (pravděpodobnost, že je správné) pro každý pixel. Další úprava spočívala ve vymazání bodů z mapy podle jejich konfidence.

Algoritmus 5 iterativně prochází naměřená data. Tato data jsou zapouzdřená v proměnné *Observation*. Při první iteraci se nastaví jako první poloha počátek souřadné soustavy. Pak se vzhledem k této poloze vykreslí první pointcloud, upravený podle mapy konfidencí a do proměnné *previous_image* se uloží první obrázek. Ten slouží při další iteraci pro sledování prvků z něj do nového obrázku. Pokud se mezi prvky viditelnými ve starém snímku a těmi viditelnými v novém snímku najdou víc jak tři korespondence, spočítá se nová poloha kombinací RANSACu a Hornovy metody. Nová poloha je vypočítaná vzhledem k poslední známé poloze. Pokud najdeme novou polohu, uložíme si snímek a jeho prvky jako starý snímek. Novou polohu použijeme jako referenci pro vykreslování mapy, upravené podle konfidencí. Takto inkrementálně budujeme 3D mapu snímané scenerie.

Kapitola 4

Kvadrokoptéra

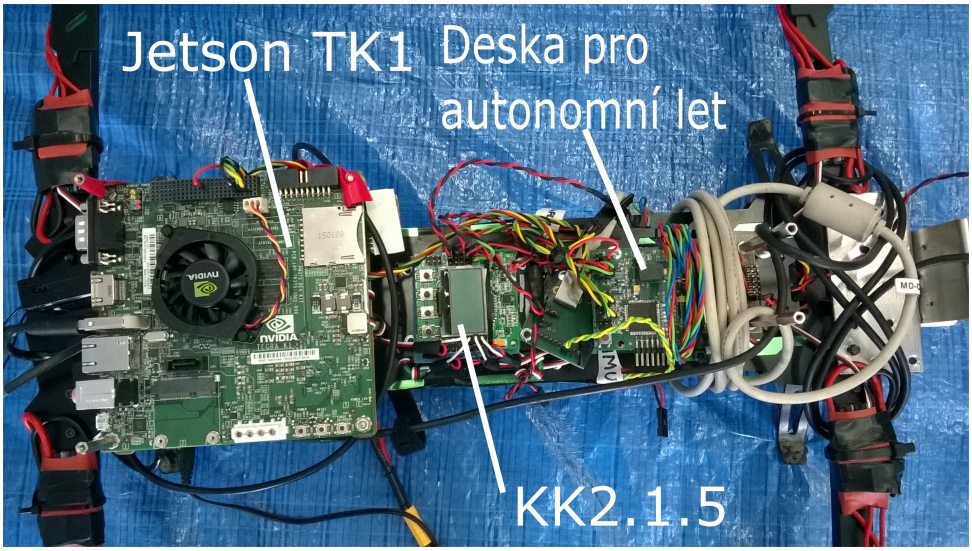
Jako mobilní platformu pro tento projekt jsme použili kvadrokoptéru vyrobenou z modelářských součástek z obchodu Hobbyking (obrázek 4.1). Rám je typu H o velikosti 470x116 mm. Kvadrokoptéra je vybavena čtyřmi motory pro vrtule a pro každý z nich má 5V regulátory. Pro komunikaci s vysílačkou a řízení motorů je využita řídící deska Hobbyking KK2.1.5 o velikosti 50.5x50.5x12 mm. Tato deska obsahuje osmibitový mikrokontrolér Atmel Mega644PA a LCD displej s ovládáním. Další vlastností této desky je stabilizace kvadrokoptéry během letu. Obsahuje proto gyroskop s accelerometrem MPU-6050, který předává informace mikrokontroléru, kde je implementovaný PI regulátor. Uživateli je umožněno nastavovat konstanty tohoto regulátoru.



Obrázek 4.1: Kvadrokoptéra

Dále kvadrokoptéra nese embedded systém Jetson TK1 od společnosti NVIDIA a přídavný USB 3.0 hub. Pro budoucí aplikace byla vyrobena deska plošných spojů, která by měla umožňovat bezpilotní let kvadrokoptéry. Kvadrokoptéra je také vybavená různými senzory. Zespodu kvadrokoptéry je připevněna kamera PX4FLOW, která měří optický tok. Tato kamera je schopná z pozemních textur a z měření vestavěného sonaru zjistit rychlost

kvadrokoptéry. Kromě této kamery je na kvadrokoptéře připevněn i gyroskop.



Obrázek 4.2: Popis kvadrokoptéry

Na straně ve směru letu kvadrokoptéry je připevněn držák s dvěma hlavními senzory, stereokamerou ZED od firmy Stereolabs a laserovým dálkoměrem Hokuyo (obrázek 4.3). Tento dálkoměr spolu s kamerou PX4FLOW a gyroskopem je využitý pro jiný projekt.

Obrázek 4.3: Stereokamera a laserový dálkoměr

Napájení všech těchto komponent zajišťuje lithium polymerová baterka o kapacitě 4000mAh. Při plně nabité baterce vydrží kvadrokoptéra létat přibližně 10 minut.

4.1 Počítač Jetson TK1

Použitá stereokamera potřebuje k zpracování dat počítač s grafickou kartou od společnosti NVIDIA. K získání hloubkové mapy využívá výpočty na grafické kartě. Kvadrokoptéra dále musí být takový počítač schopná unést. Všechny tyto požadavky splňuje počítač Jetson TK1 od společnosti NVIDIA. Celý počítač je umístěn na jediné desce plošných spojů, která má rozměry 127x127 mm a váží 120 g. Grafická karta umístěná na počítači je srovnatelná s těmi, které se nachází v normálních stolních počítačích. Dosahuje rychlosti až 326 miliard floating point operací za sekundu (326 GFLOP/s).

Počítač je vybaven operačním systémem Linux4Tegra založený na Ubuntu 14.04. Na desce počítače se nachází čtyřjádrový procesor ARM Cortex-A15 s frekvencí 2.32 GHz a grafická karta Tegra K1 Kepler GK20. Paměť počítače je 16 GB a je vybaven DRAM o velikosti 2 GB. Počítač je také vybaven velkým počtem portů. Mezi nejdůležitější patří USB 3.0, slot pro sběrnice miniPCIe, UART slot pro sběrnici RS232 a výstup pro audio a video.

4.2 Stereokamera ZED

Jako senzor pro snímání okolí byla použita stereokamera ZED od firmy Stereolabs. Rozměry kamery jsou 175x30x33 mm a váží 160 g. Její baseline je 120 mm a s počítačem je propojená přes USB 3.0. Video může natáčet v různém rozlišení podle tabulky 4.1. Frekvence snímků uvedené v tabulce platí v případě, že kamera natáčí pouze video.

Režim	Frames per second	Rozlišení
2.2K	15	2208×1242
1080p	30	1920×1080
720p	60	1280×720
VGA	120	640×480

Tabulka 4.1: Režimy natáčení videa

Kamera využívá k výpočtu vzdáleností objektů ve snímané scéně pasivní triangulaci (kapitola 2.2.1). Tato informace je pak ve stejném rozlišení, jako snímaný obrázek - ke každému pixelu obrázku máme informaci o jeho vzdálenosti od kamery. Tuto vzdálenost je kamera schopna zjistit v intervalu od 1 do 20 metrů.

Kamera je schopná se sama zkalibrovat a poskytuje nám svoji matici vnitřních parametrů. Aby kalibraci mohla provést, je třeba kameru namířit na dobře osvětlené místo a odstranit příliš blízké objekty ze snímaného prostoru. Kamera většinou špatně snímá první čtyři snímky, které jsou v odstínech zelené.

4.2.1 ZED SDK

Spolu s kamerou je dodáno i SDK, které nám umožňuje vytvořit program k využití kamery pro naše účely. K správné funkci SDK vyžaduje knihovny OpenCV (Open source Computer Vision), která se zabývá počítačovým viděním a machine learningem. Dále SDK vyžaduje knihovnu CUDA (Computer Unified Device Architecture), což je softwarová a hardwarová architektura společnosti NVIDIA, která umožňuje výpočty na grafické kartě.

Součástí SDK jsou funkce pro získání měření, která kamera poskytuje. Ty popisuje tabulka 4.2:

DEPTH	Depth map
DISPARITY	Disparity map
CONFIDENCE	Certainty/confidence of disparity map
XYZ	Všechny souřadnice bodů
XYZRGBA	Všechny souřadnice bodů s barvou

Tabulka 4.2: Druhy měření

SDK ukládá měření do třídy sl::zed::Mat. Náš algoritmus pro SLAM vyžaduje kromě snímků i měření vzdálenosti jednotlivých pixelů a jejich konfidencí. Ukládá se snímek z levé kamery, která se používá jako počátek souřadného systému.

Spojení počítače Jetson TK1 a kamery ZED nám umožňuje pořizovat snímky v rozlišení 720x1280 s frekvencí 15 snímků za sekundu. Toto rozlišení má i měření vzdálenosti a mapa konfidencí. Ukládání jednotlivých měření způsobuje zpomalení fps na 2 snímky za sekundu.

Zde je program, který nám umožňuje získat z kamery snímky, hloubkovou mapu a mapu konfidencí:

```
#include <iostream>;
#include <opencv2/core/core.hpp>;
#include <opencv2/highgui/highgui.hpp>;
#include <opencv2/imgproc/imgproc.hpp>;
#include <zed/Camera.hpp>;

sl::zed::Camera* zed = new sl::zed::Camera(sl::zed::HD720,15);
sl::zed::ERRCODE err;
err = zed->init(sl::zed::MODE::PERFORMANCE, 0, true);
if (err != sl::zed::SUCCESS) {
        std::cout << "Init failed:" << errcode2str(err) << "\n";
        delete zed;
        return 1;
}

while(1){
 if (!zed->grab(sl::zed::SENSING_MODE::FULL)){

        // Retrieve left color image
        sl::zed::Mat left;
        left = zed->retrieveImage(sl::zed::SIDE::LEFT);
```

```
        // Retrieve depth map
        sl::zed::Mat depth;
        depth = zed->retrieveMeasure(sl::zed::MEASURE::DEPTH);

        // Retrieve confidence map
        sl::zed::Mat conf;
        conf = zed->retrieveMeasure(sl::zed::MEASURE::CONFIDENCE);
 }
 //process data ...
}
```

SDK umožňuje také získání matice vnitřních i vnějších parametrů obou kamer a dalších vlastností kamery. Další součástí SDK je uživatelské rozhraní ZED Explorer pro ovládání kamery a pořizování videa.

Kapitola 5

Experimentální výsledky

Experimenty byly provedeny tři, z nichž dva se konaly na schodišti v interiéru budovy Blox. Při těchto experimentech byla kvadrokoptéra pouze nesena. Třetím experimentem byl venkovní let, který se konal v ulici Kafkova.

Ke stavbě kvalitní mapy je potřeba, aby se snímky z kamery co nejvíce překrývaly a bylo na nich co nejvíce význačných prvků. Tyto podmínky jsou důležité pro sledování těchto prvků mezi snímky a pro výpočet polohy robotu. Pokud tyto podmínky nejsou splněny, stavba mapy může selhat po dvou nebo třech krocích.

Problém může nastat při venkovních experimentech, kdy má na měření značný vliv okolní prostředí a také počasí. Také můžeme očekávat, že vzdálenost některých objektů je větší než interval, ve kterém měří stereokamera. Chování kamery je v těchto případech nepředvídatelné a měření je nepřesné.

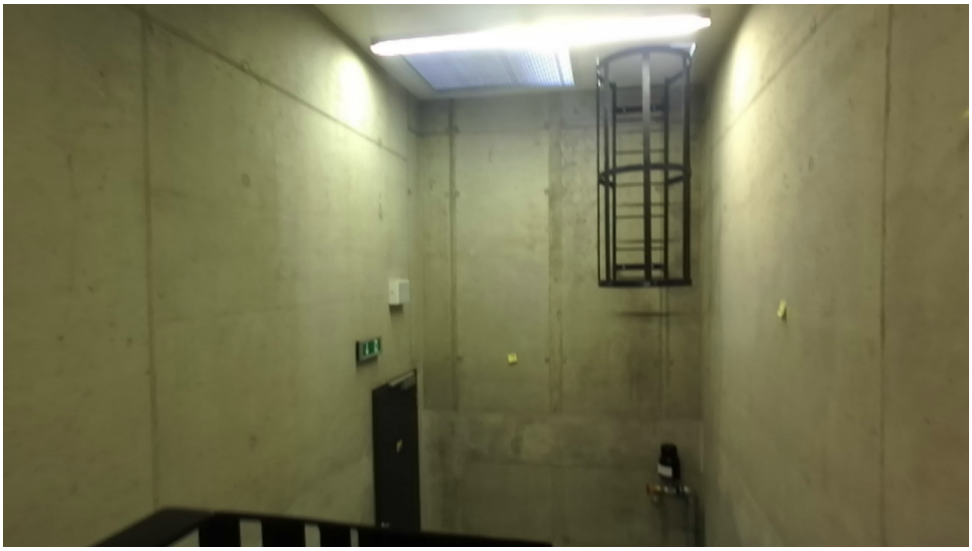
5.1 Rekonstrukce scény jednoho snímku

Nejdříve jsme řešili problém úpravy mapy podle konfidencí jednotlivých bodů. Do výsledné mapy jsme body pouze přidávali, včetně těch, které už v mapě byly. Tím narůstalo využití paměti a při stavbě větších map program z tohoto důvodu neočekávaně skončil.

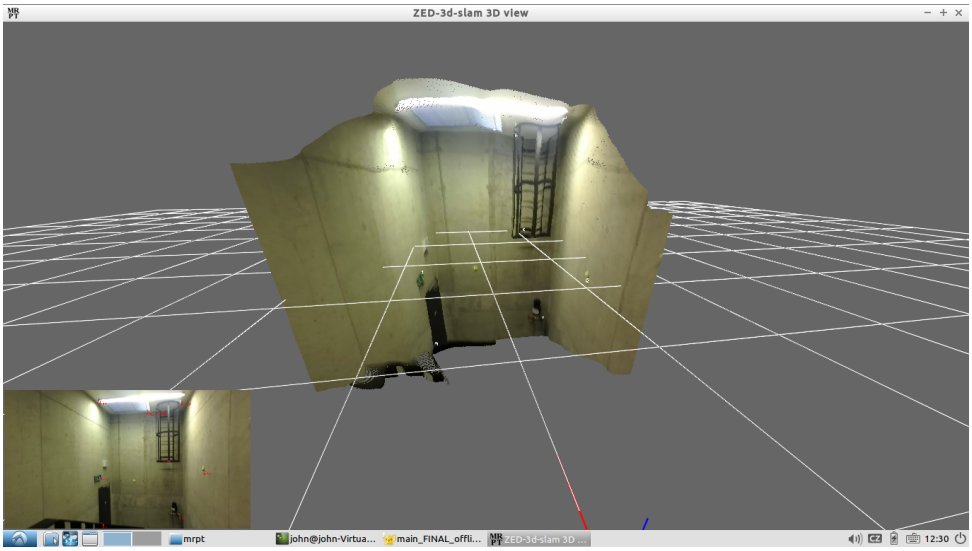
Na obrázcích 5.1, 5.2 a 5.3 je vidět snímaný prostor a vytvořená mapa. Na obrázku 5.3, kde je pohled na mapu ze strany, můžeme pozorovat několik chyb způsobených stereokamerou. Světlo na stropě způsobuje špatný výpočet vzdálenosti v jeho okolí. Další chybou je ohyb zdí, které jsou nejblíže kameře (znázorněné barevným počátkem souřadné soustavy na obrázku 5.3). Je tomu proto, že tyto části zdí jsou na pomezí zorného pole kamery.

Na obrázku 5.4 je vidět, jaké kamera přiřadila konfidence jednotlivým bodům. Náš algoritmus odstraní všechny body z obrázku, které mají konfidenci menší než 50%. Jak můžeme pozorovat, velmi vysokou konfidenci mají i místa, u kterých je vzdálenost špatně spočítaná (světla, zdi na pomezí zorného pole kamery).

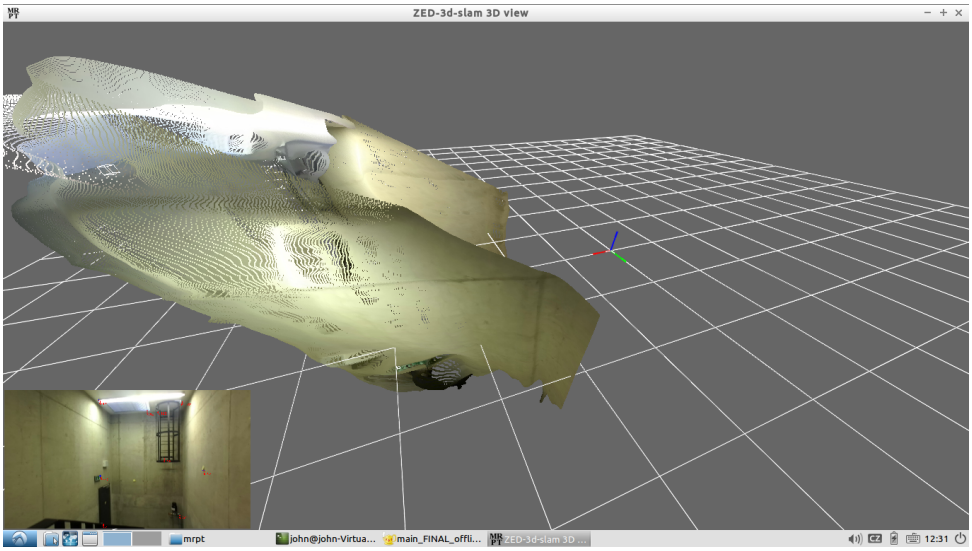
Na obrázcích 5.5 a 5.6 je vidět upravená mapa. I zde můžeme pozorovat několik chyb, které kamera nedokáže rozeznat a přidělit jim nízkou konfidenci. Zadní stěna místnosti je nakloněná a okolí stropního světla by mělo být rovné.



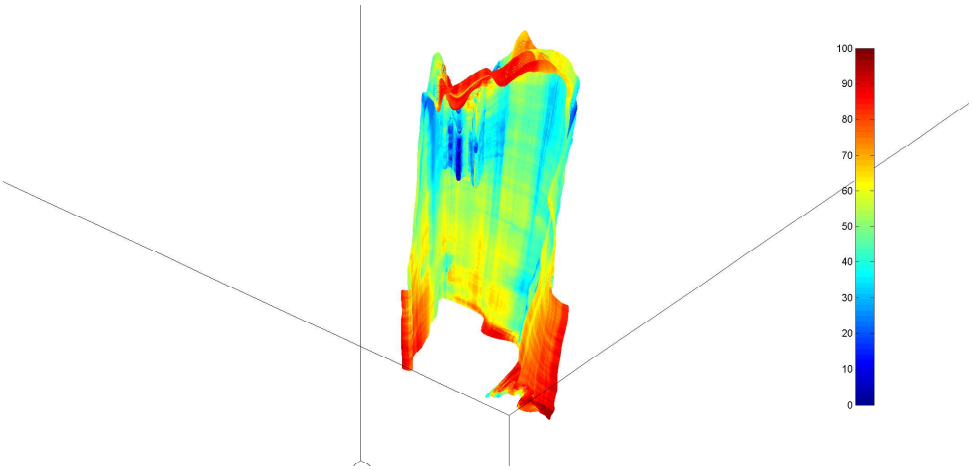
Obrázek 5.1: Snímaný prostor



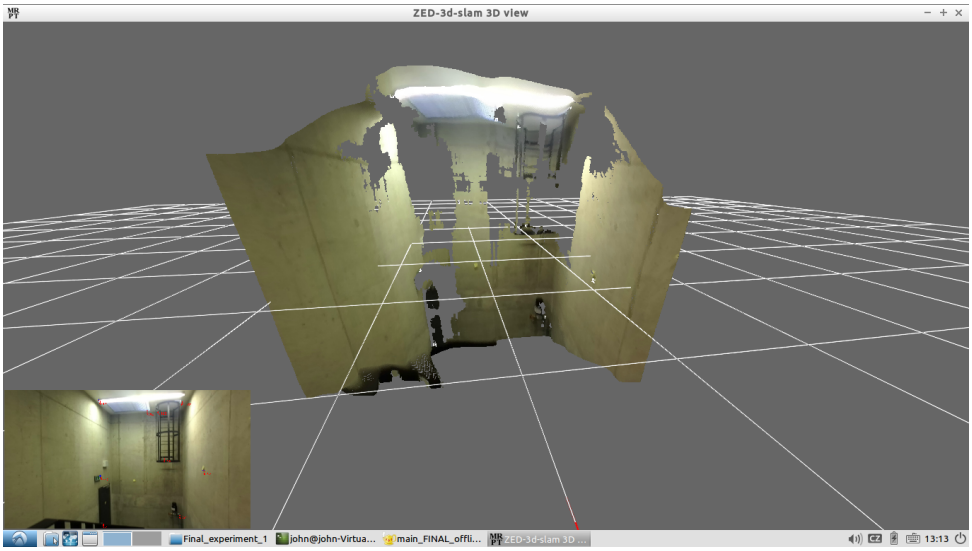
Obrázek 5.2: Vytvořená mapa bez úprav



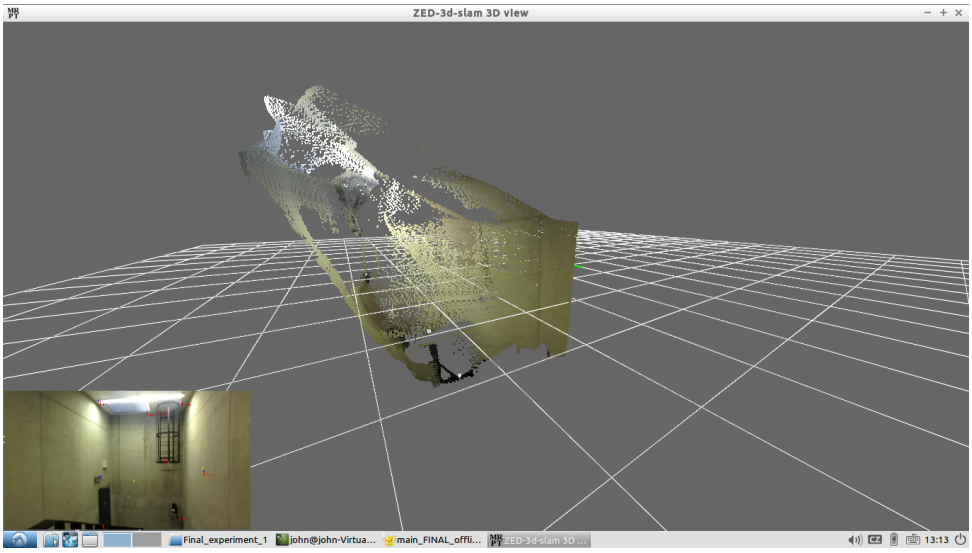
Obrázek 5.3: Pohled na mapu ze strany



Obrázek 5.4: Mapa konfidencí



Obrázek 5.5: Mapa upravená podle konfidencí



Obrázek 5.6: Pohled ze strany na upravenou mapu

5.2 Tvorba mapy z více snímků

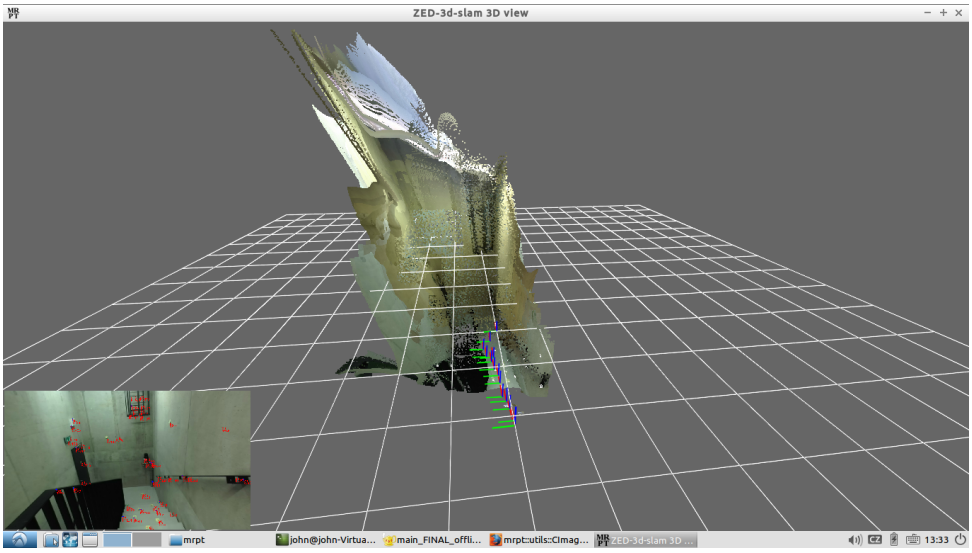
Velikost mapy představuje problém při tvorbě větších map. V tomto experimentu jsme vytvořili z třiceti snímků model místnosti, který měl 12 miliónů bodů. Měření probíhalo v místnosti, která je zobrazená na obrázku 5.1.

Na obrázcích 5.7, 5.8 a 5.10 můžeme pozorovat vytvořenou mapu. Jsou zde vidět stejné chyby způsobené stereokamerou, které jsme zmínili i v sekci 5.1. Výraznější je okolí stropního světla, které je zobrazeno mimo mapu. Dále můžeme pozorovat chybu u zobrazení zábradlí (obrázek 5.8), které bylo během měření příliš blízko kameře (tzn. ve vzdálenosti menší než 1 metr).

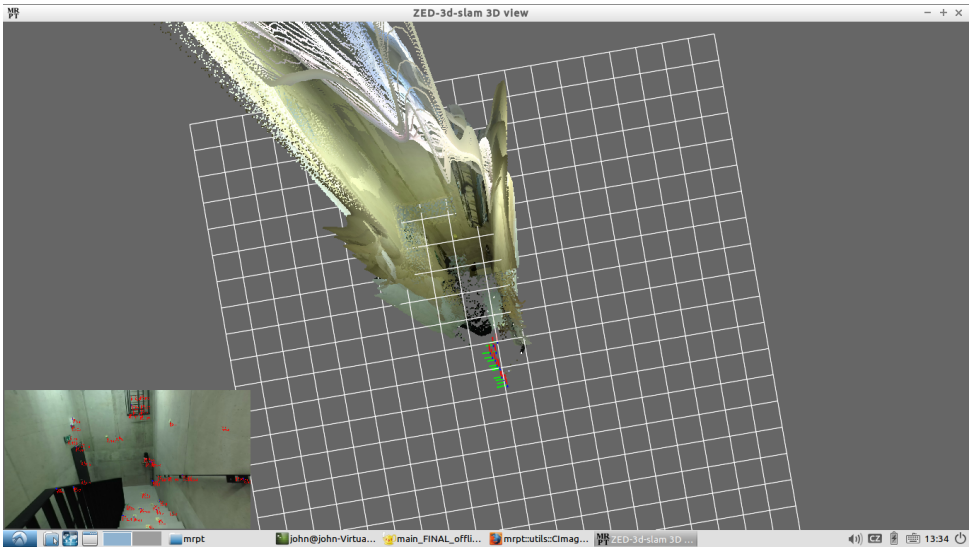
To způsobilo špatný výpočet u všech bodů zábradlí, kterým byla přiřazena podobná vzdálenost.

Dalším problémem stereokamery je výpočet vzdálenosti u lesklých povrchů. To zde můžeme pozorovat u schodů, kde jsme problém řešili s pomocí umělých výrazných prvků, jak lze vidět na obrázku 5.9. Toto řešení pomohlo při výpočtu vzdálenosti schodů, jsou zde ale zobrazeny jako rovná plocha, jak můžeme pozorovat na obrázku 5.10.

Můžeme zde pozorovat i několik chyb způsobených použitými metodami. Na obrázku 5.8 lze vidět vypočítanou cestu robotu znázorněnou posunutými čarami, které symoblizují počátky souřadného systému kamery. Vyznačená cesta není přesná, ve skutečnosti jsme u kamery pouze změnili náklon.

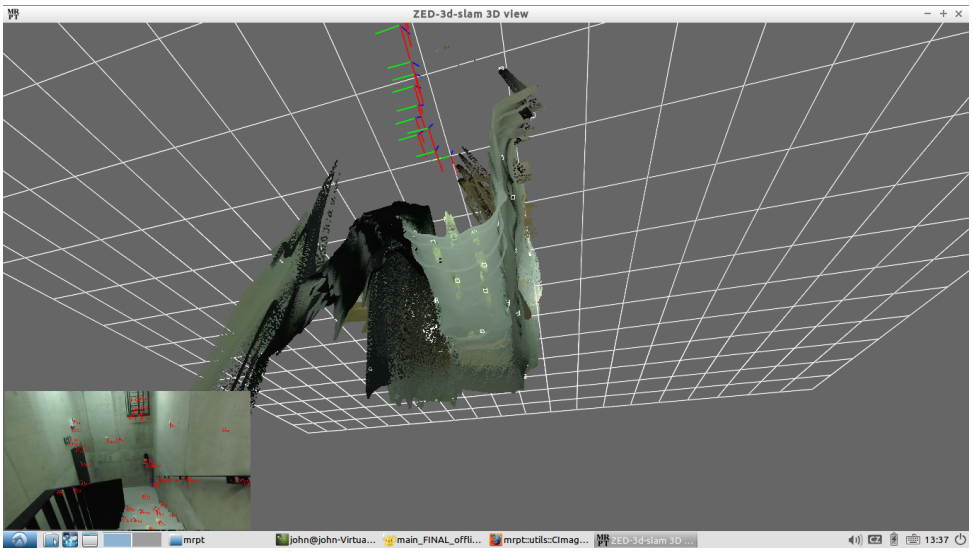


Obrázek 5.7: Výsledná mapa snímaného prostoru



Obrázek 5.8: Pohled shora na mapu

Obrázek 5.9: Snímek s vykreslenými význačnými prvky



Obrázek 5.10: Pohled na schody zespodu mapy

5.3 Mapa vytvořená za letu

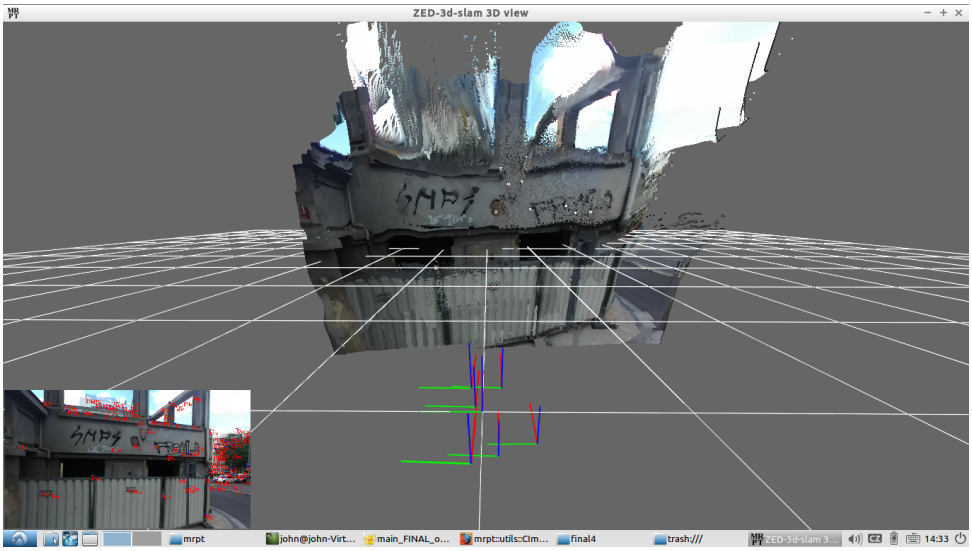
Venkovní prostředí poskytuje lepší možnosti sledování význačných prvků, kterých je více než uvnitř budov. Další výhodou je lepší osvětlení snímaných objektů, což zabraňuje vzniku rozmazaných snímků.

Na obrázku 5.11 je vidět snímaná scéna. U tohoto měření se překrývalo vždy pouze pár snímků následujících za sebou. Mapa na obrázcích 5.12 a 5.13 je vytvořena z deseti snímků. Další snímky se s posledním z těchto snímků dostatečně nepřekrývaly a sledovat jsme mohli pouze pár korespondencí nevhodných pro nalezení podmnožiny RANSACem. Vykreslené význačné

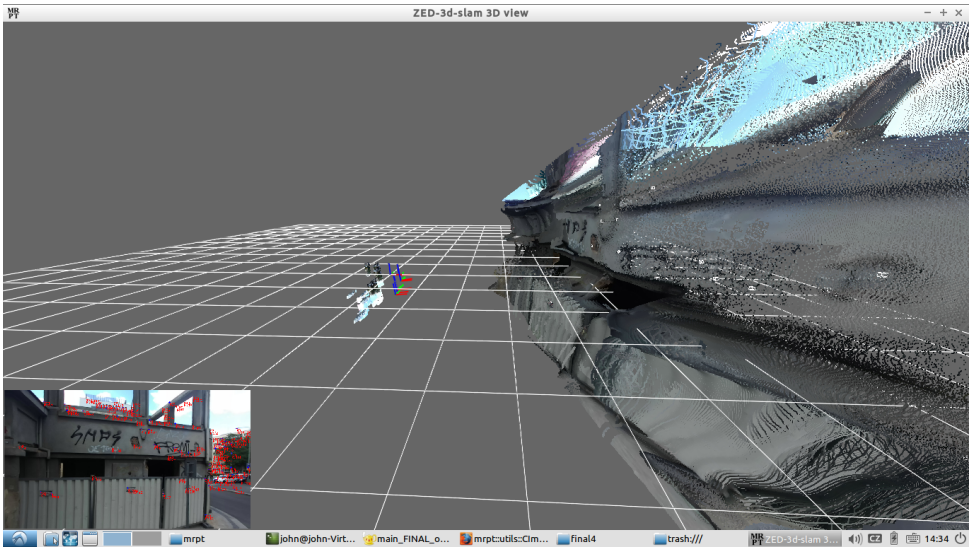
prvky navazujících snímků jsou vidět na obrázcích 5.14 a 5.15.

Jak můžeme vidět na obrázku 5.13, kamera odhaduje vzdálenost předmětů vzdálenějších než 20 metrů (např. oblohu prosvítající mezi sloupy) špatně a jejich vzdálenost odhadne podle bližších předmětů. Obloha má proto na obrázku stejnou vzdálenost jako pozorovaný objekt.

Obrázek 5.11: Snímaná scéna



Obrázek 5.12: Vytvořená mapa



Obrázek 5.13: Pohled ze strany na mapu

Obrázek 5.14: Snímek s vykreslenými features

Obrázek 5.15: Následující snímek s vykreslenými features

5.4 Zhodnocení experimentů

Náš algoritmus pro správnou funkci vyžaduje překryv po sobě jdoucích snímků. Pokud tato podmínka není splněna, stavba mapy selže. Let kvadrokoptéry proto musí být pomalý a stabilní. Na tento problém jsme naráželi hlavně v experimentu 5.3.

Pokud se snímky dostatečně nepřekrývají, algoritmus RANSAC není schopen najít vhodnou podmnožinu bodů pro výpočet nové polohy robotu. Tento problém by se dal vyřešit úpravou parametrů tohoto algoritmu, což by ale mohlo způsobit prodloužení doby výpočtu nebo nestabilitu programu.

U RANSACu se počáteční množina bodů určuje náhodně. U některých experimentů proto výpočet polohy selhal. Částečným řešením bylo spuštění programu vícekrát, dokud se počáteční množina neurčila tak, aby výpočet proběhl, nicméně pro real time aplikace tato možnost není použitelná. Jednou z možností by bylo zvýšit počet iterací RANSACu, čímž bychom dosáhli stejného výsledku.

Dalším problémem, který se vyskytl v experimentu 5.2, je prostředí bez většího množství výrazných prvků. Stereokamera pak může v tomto prostředí špatně vypočítat vzdálenost objektů nebo algoritmus RANSAC nemusí najít novou polohu a celý program skončí. Problém jsme řešili přidáním umělých vizuálních prvků do snímané scény.

Použitá stereokamera ZED odhaduje ke každé vypočítané vzdálenosti bodu i její konfidenci. Tu můžeme použít k vymazání těch částí mapy, které mají nízkou konfidenci. V některých případech však konfidence není spolehlivá. To jsme mohli pozorovat v experimentu 5.1. Vedle případů uvedených v tomto experimentu (okolí světla, příliš blízké objekty a pomezí zorného pole stereokamery) stejný problém nastává i u lesklých povrchů, monotonního prostředí (bíle zdi) a oken.

Jak již bylo řečeno, při stavbě mapy body do mapy pouze přidáváme a to včetně těch, které už v mapě jsou. Takto roste velikost mapy a využití paměti, kterou tato mapa zabírá. Za následek to má značné zpomalení algoritmu při tvorbě větších map, které navíc často končí chybou. Předejít tomu můžeme spojováním blízkých bodů nebo používáním snímků s menším rozlišením.

Kapitola 6

Závěr

Práce se zabývá tvořením 3D modelu prostředí. Využívá k tomu stereokameru ZED od firmy Stereolabs připevněnou na mobilní platformě a algoritmus pro SLAM. Tento algoritmus je založený na aplikaci pro 3D kameru Kinect od firmy Microsoft napsané pro knihovnu MRPT. Funkce algoritmu je založená na sledování význačných bodů v pořízených snímcích s využitím KLT trackeru a výpočtu polohy z této informace s použitím RANSACu a Hornovy metody.

Experimentální výsledky ukazují na skutečnost, že pro reálnou aplikaci algoritmu je třeba jej optimalizovat - jsme omezení maximální možnou velikostí mapy a do budoucna je třeba implementace funkce pro spojování stejných nebo blízkých bodů.

Další možností pro zlepšení funkce algoritmu je vylepšit spolehlivost map konfidencí. Použitá stereokamera nedokáže odhalit všechna nejistá místa, což způsobuje velké chyby v tvořené mapě.

Již nyní byla provedena různá opatření pro autonomní let naší kvadrokoptéry. Cílem do budoucna pravděpodobně bude zajištění plné autonomity této kvadrokoptéry. Hlavní úpravy se budou týkat interpretace tvořené mapy, která by měla být více přizpůsobená pro orientaci robota.

Literatura

[1] O. Ton *Orientace v prostředí, bakalářská práce* České vysoké učení technické, fakulta elektrotechnická, 2006.

[2] Sudeep Pillai, Srikumar Ramalingam and John J. Leonard *High-Performance and Tunable Stereo Reconstruction* ICRA, 2016

[3] Weingarten, J., *Feature-based 3D SLAM. PhD Thesis*, Swiss Federal Institute of Technology Lausanne, EPFL, no 3601, Dir. Roland Siegwart, 2006

[4] Mázl Roman, *Lokalizace pro autonomni systemy, disertační práce*, Praha 2007

[5] P.J. Besl, H.D. McKay, *A method for registration of 3-D shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992

[6] M.A. Fischler and R.C. Bolles. *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.* Communications of the ACM, 1981

[7] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[8] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991

[9] B. K. Horn. Closed-form solution of absolute orientation using orientation using unit quaternions, *Journal of the Optical Society of America, 4:629-642*, 1987

[10] `http://www.citr.auckland.ac.nz/`

[11] `http://docs.opencv.org/3.1.0/`

[12] Larry Li Time-of-Flight Camera - An Introduction *Sensing Solutions, Texas Instruments*, 2014

[13] Montemerlo, M., Thrun, S., Koller, D. a Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem.

Příloha A

Obsah CD

Přiložené CD obsahuje:

- tuto práci v souboru *bp_jan_cabicar_2016.pdf*
- implementovaný algoritmus pro SLAM v souboru *main_SLAM.cpp*
- program pro ukládání dat z kamery v souboru *main_ZED.cpp*
- dva soubory *CMakeLists.txt* pro vytváření souborů makefile k oběma programům