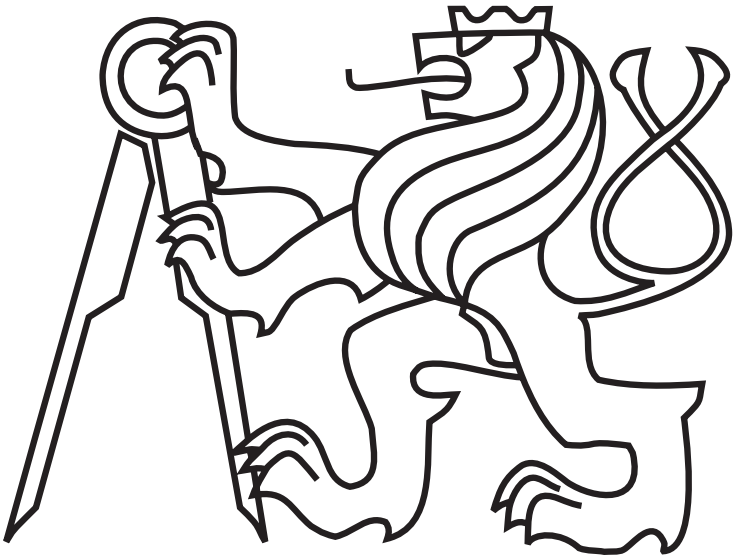
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačů

Bakalářská práce



Filip Zoubek

Robustní vizuální navigace mobilních robotů

Vedoucí práce: **Ing. Tomáš Krajník, Ph.D.**

Květen, 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zoubek** Jméno: **Filip** Osobní číslo: **457122**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Robustní vizuální navigace mobilních robotů

Název bakalářské práce anglicky:

Robust Visual Navigation of Mobile Robots

Pokyny pro vypracování:

Kód Cílem práce je rozšíření stávajícího systému [1] pro visuální navigaci mobilních robotů tak, aby byla zlepšena jeho robustnost vůči nepřesnosti odometrického systému.
1) Seznamte se s robotickým operačním systémem (ROS).
2) Seznamte se s metodami používanými v robotice pro zpracování obrazu a s principy stereo vidění.
3) Seznamte se s principy visuální navigace mobilních robotů.
4) Seznamte se se s navigačním systémem `Bearnav' [1].
5) Navrhněte rozšíření výše uvedeného systému tak, aby využíval informaci o hloubce snímané scény.
6) Navržené rozšíření naimplementujte.
7) Seznamte se s ovládáním robota ECA CAMELEON.
8) Navržené rozšíření otestujte na výše uvedeném robotu.
9) Navrhněte sérii experimentů umožňujících porovnat vaše rozšíření s původním systémem a s metodou ORBSLAM [2].
10) Proveďte experimentální ověření systému na reálném robotu.
11) Shrňte výsledky experimentů, proveďte porovnání.
Výstupy: Kód implementovaného systému na určeném depositáři.

Seznam doporučené literatury:

[1] Krajnik, T. - et al.: Navigation without localisation: reliable teach and repeat based on the convergence theorem, In IROS, 2018.
[2] Mur Artal, R. - et al.: ORB-SLAM: a Versatile and Accurate Monocular SLAM System., IEEE TRO, 2015.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Krajník, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **24.01.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Tomáš Krajník, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

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..... .....

Poděkování

Velké díky rozhodně patří mému vedoucímu práce, který měl se mnou trpělivost a neměl nikdy problém mi s čímkoliv poradit a cokoliv vysvětlit. Současně bych rád poděkoval své rodině, která mně při práci na bakalářské práci celou dobu podporovala.

Abstrakt

Bakalářská práce se zabývá rozšířením stávajícího systému *STRoLL BearNav* pro visuální navigaci mobilních robotů. Navržené řešení zlepšuje jeho robustnost vůči nepřesnosti odometrického systému za pomoci principu stereo vidění, z kterého dokáže získat vzdálenost objektů před sebou. Následným vypočítáním souřadnic získá 3D orientaci v okolí, ve kterém se robot nachází a díky tomu také je schopen určit svoji pozici vůči předchozí uložené jízdě.

Abstract

The bachelor thesis is focused on the extension of the existing visual navigation system for mobile robots named *STRoLL BearNav*. The extension improves its robustness against the inaccuracy of the odometric system using the principle of stereo vision from which the robot obtains the distance of objects in front of itself. Subsequent calculation of the coordinates provides a 3D orientation in the surroundings where the robot is located, and thus it also determines its position relating to its previous stored ride.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Dnešní svět doslova žije autonomními vozidly. Vývoj těchto strojů zaznamenáváme velmi často v médiích. Velké společnosti, jako například Google, Apple nebo Tesla, se snaží vyvinout auta, která by nás dokázala odvést od našeho domu až na druhou stranu kontinentu, aniž bychom museli nějak zasahovat do řízení. Na druhou stranu, existuje zde i vývoj, který se zabývá mobilními roboty. Takového mobilního robota si můžeme představit například jako malé zařízení se čtyřmi koly a kamerou nasměrovanou vpřed. I když se to na první pohled nemusí zdát, takoví roboti dokáží být také velmi užiteční. Například je mohou využít logistické společnosti ve svých skladech pro přepravu zboží anebo armáda pro mapování území, ve kterém se nachází.

Obrázek 1: Příklady mobilních robotů [1]

Ať už se jedná o automobily pro běžnou dopravu nebo mobilního robota, pohybující ho se ve skladu, každý z těchto strojů musí mít odpovídající navigaci, kterou se řídí při svém pohybu. Tato navigace většinou přijímá data ze senzorů, jako jsou odometrické systémy nebo kamery, zachycující okolí v reálném čase. Navigace následně zpracovává informace z těchto senzorů a podle výsledků rozhodne, jak se přizpůsobí následující cesta vozidla.

V článku [4] se autoři zmiňují o problémech, které musí taková navigace řešit. Ty shrnuli do tří následujících otázek, které by si systém měl říci: "Kde jsem ?", "Kam jedu ?" a "Jak se tam dostanu ?". První otázku lze interpretovat tak, že robot potřebuje vědět, kde se právě nachází a jak vypadá prostředí kolem něho. Tato informace je velmi důležitá, protože bez ní by navigace nevěděla, co má dělat, a jak má pokračovat. Zbylými dvěma otázkami je naznačeno, že si systém musí poradit s naplánováním cesty, aby se dostal do cíle a splnil úkol. Jinými slovy, můžeme říci, že navigace musí vyřešit tři hlavní úkoly, aby splnila svůj účel, a těmi jsou: lokalizace, plánování a navigace.

Existují dva přístupy, jak robot může získat informaci o tom, kde se právě nachází. První možnou variantou je, že navigace zná přesné souřadnice své polohy. Toho lze například dosáhnout pomocí známého amerického systému GPS nebo evropského systému Galileo. Druhá eventuální možnost je, že robot neví, jaká je jeho přesná lokalizace, ale ví pouze informaci o tom, kde se nachází vůči své stanovené nebo jemu jinak známé cestě.

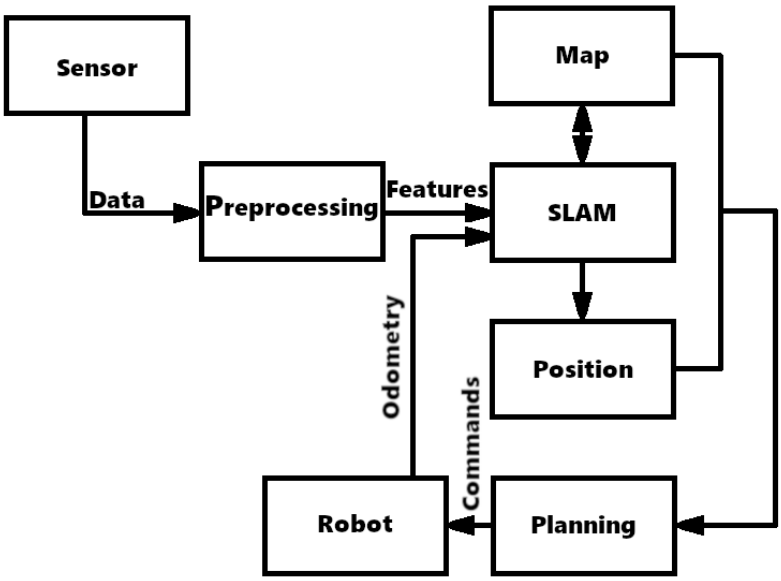
Pokud systém dokáže lokalizovat sám sebe, může přejít k dalšímu kroku, který označujeme plánování. V něm musí navigace vyřešit problém, jak se dostat ke svému cíli. Možným populárním řešením je, že systém má uložené mapy okolí, ve kterém se nachází, a ze kterých si vybere cestu ke svému cíli. Toho lze dosáhnout například předchozím projetím trasy robotem s lidským operátorem, kdy si robot vytvoří lokálná mapu nebo nahráním již vytvořených map do paměti stroje. Další možností může být, že necháme robota zjistit, jak prostředí kolem něho vypadá a následně ho samovolně nechat rozhodnout, jaká cesta se mu jeví jako nejlepší.

Jakmile navigace vyřeší předešlé úkoly, může se spustit samotné navigování, při kterém se robot řídí přicházejícími příkazy od algoritmu. Systém při tom využívá data ze svých senzorů, díky kterým dokáže zjistit změnu své polohy a reagovat na měnící se prostředí. V této části už podstatně závisí na vývojářích, jak si s tímto úkonem poradí, a jak dobře své řešení dokáží vymyslet a vyladit.

Představme si nyní mobilního robota, který má za úkol ujet určenou vzdálenost. Ví, kde se právě nachází, má naplánovanou cestu i určený cíl. Robot se může při své cestě lokalizovat pomocí odometrického systému, ze kterého zjistí informace o vzdálenosti, kterou ujel. Co se však stane, pokud začnou kola na mokrém povrchu podkluzovat? Nebo pokud se zastaví o překážku, která mu znemožní další pohyb? Robot si bude myslet, že stále pokračuje v cestě, jeho reálná poloha však může být jiná nebo dokonce může stát na jednom místě. To je rozhodně důvod k zamyšlení, zdali by tu neměla být další cesta, kterou by si svoji polohu minimálně potvrzoval.

Cílem mé práce je rozšířit stávající navigační systém *STRoLL BearNav* [1] tak, aby byla zlepšena jeho robustnost vůči nepřesnostem z odometrického systému. Jelikož, jak jsem již naznačil výše, nemusí být tyto informace z odometrie spolehlivé a mohou vzniknout chyby, je dobré je kontrolovat i z jiného zdroje. Problémem však zůstává, z jakého zdroje. Nápad je takový, generovat data z okolí pomocí kamery a ty kontrolovat s již uloženými. Kamera zachytí zajímavé body z pohledu před robotem, který je pak porovnává s již uloženými. Tento nápad již byl částečně v uvedeném systému naprogramován. Ukládá a porovnává však pouze přímky - tedy vektory s dvěma souřadnicemi. Mé rozšíření spočívá v zavedení třetí souřadnice, která bude znázorňovat vzdálenost mezi kamerou a objekty před ní. Tím pádem systém získá více informací o svém aktuálním okolí, se kterými může následně pracovat, měnit případný směr své cesty a celková přesnost pohybů bude tak zdokonalena.

Na obrázku 2 je zobrazen příklad toku dat navigace, která využívá vizuálních senzorů. Data prostředí okolo robota jsou pomocí kamery pořízeny do 2D obrazu. Tento obraz je v zápětí poslán v podobě dat a zpracován do klíčových bodů. Body se následně porovnávají s lokální mapou, kterou má robot uloženou. Po těchto úkonech navigační systém zjistí,



Obrázek 2: Tok dat vizuální navigace

kde se právě nachází a může plánovat svoji další cestu, přičemž vygeneruje sadu příkazů, které má robot vykonat. Podle tohoto schématu jsem svoji práci rozdělil do šesti částí. Nejdříve se zmíním o lokalizaci a jaké jsou dnes možnosti určení polohy (*State of the art*). Následně budu pokračovat vznikem obrazu (*Vznik obrazu*) a kalibrací. Další kapitola se bude zabývat zpracováním pořízené scény (*Zpracování obrazu*) a poté navážu s vysvětlením pojmu *STRoLL BearNav* (*Popis navigačního systému*) i s tím, jak celý systém funguje. Na závěr zmíním, jak jsem navrhl svoje řešení (*Řešení*) a provedené testování (*Testování*).

2 State of the art

Navigování provází lidstvo už od počátku vzniku civilizace. Souviselo hlavně s migrací lidí a objevování nových zemí, ať už pro obchod nebo pro získání nového bohatství z ještě neobjevených částí světa. Lidé se často navigovali především pomocí osobního dorozumívání. Postupně v historii pak vznikaly mapy terénu, ve kterém se lidé pohybovali, následně pak mapy celých kontinentů, a to hlavně v důsledku mořeplavby. S mořeplavbou souvisí i vynález kompasu, kterým námořníci určovali svoji přibližnou polohu na moři. Průlom přišel s vynálezem počítačů, kdy se začaly tyto informace sjednocovat a sdílet. S nimi také souvisí vypouštění prvních satelitů na oběžnou dráhu a pořizování snímků naší Země z výšky. Nejen díky tomu začaly mapy být až na několiv metrů přesné. V dnešní době většina lidí využívá tyto vymoženosti dennodenně hlavně prostřednictvím aplikací ve svých chytrých telefonech, které jim již dokáží říci přesnou cestu cíli. I tyto aplikace však vždy na začátku musí vyřešit již v úvodu zmíněné tři zásadní otázky podle [4] a to "Kde jsem ?", "Kam mířím ?" a "Jak se tam dostanu ?".

2.1 Lokalizace

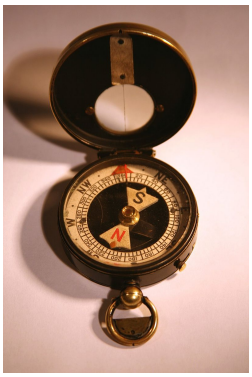
Lokalizace slouží k vyřešení otázky "Kde jsem ?" a je to tedy proces pro určení aktuální polohy. Existují dvě metody pro získání lokalizace [5] - globální a lokální. Pokud robot nemá žádnou informaci o své předchozí jízdě, jedná se o globální metodu. Pokud robot má schopnost poskytnout pozici na základě přechozí jízdy, jinými slovy kdy má robot daný počáteční bod a při své jízdě dokáže odhadnout, jakou dráhu od tohoto bodu již urazil, jedná se lokální metodu lokalizace.

2.1.1 Globální metoda lokalizace

Problém ztraceného robota nebo také problém s rozjezdem, tedy pokud robot nezná svoji předchozí cestu, nám může pomoc vyřešit například globální družicový polohový systém jako je americká *GPS* (*Global Positioning System*) nebo evropská *Galileo*. Systém se skládá z několika desítek satelitů, které obíhají naší zeměkouli. Funguje na principu *trilaterace* [6], kdy na zjištění polohy stačí tři vysílací body - satelity. Komunikace probíhá pomocí vysokofrekvenčního signálu, přičemž satelity posílají informaci o jejich aktuální poloze. Po získání informací od tří satelitů lze vypočítat zeměpisnou výšku, šířku i nadmořskou výšku. Na rozdíl od evroského systému *Galileo* americká *GPS* funguje několik desítek let. Původně sloužila pro vojenské účely, dnes je využita širokou veřejností. Její přesnost je kolem deseti metrů, to se má změnit se zavedením nového čipu, který odchylku změní na pouhých 30 centimetrů [7]. Další možností pro zjištění polohy globální metodou lokalizace je využití magnetického kompasu nebo principu *triangulace*, kdy senzory na robotovi přijímají signály od okolních vysílačů pod různými úhly a systém si tak dokáže vypočítat svoji polohu.

Nakonec zmíním i poslední alternativu s označením *LIDAR*. Light Detection and Ranging (*LIDAR*) [8] je americký mapovací systém, který dokáže zachytit velmi přesná digitální terénní data bez pomoci GPS. Využívá k tomu laserové měřidlo vzdálenosti, které je zachyceno na podvozeku letounu. Letadlo následně letí nad určeným terénem podél čáry a laserová jednotka vydává proud světelných pulzů z jedné strany na druhou od letadla. Data jsou zachycena v reálném čase a pro zpracování pak využívá triangulaci. Po dokončení triangulace jsou provedeny finální korekce za pomoci fotogrammetrických metod. Tento systém je hlavně určený pro lokalizaci letadel.

(a) Global Positioning System [9]



(b) Kompas používaný za První světové války

2.1.2 Lokální metoda lokalizace

Lokální lokalizace může využívat ke svému splnění úkolu mnoho způsobů. Mezi nejznámější rozhodně patří odometrie. Ta spočívá v tom, že systém získává data ze senzorů, která snímají kola vozidla. Metoda je jednoduchá, na druhou stranu systém musí znát přesnou aktuální rychlost a data mohou být při jízdě zkreslena, a to například prokluzováním kol o překážku nebo jízdou na mokrém povrchu. Kvůli těmto omezením je doporučena pouze pro navigaci na krátkou vzdálenost. Lokalizaci lze také vyřešit pomocí inerciální navigace [10] (akcelerometr, gyroskop), která pracuje se zrychlením.

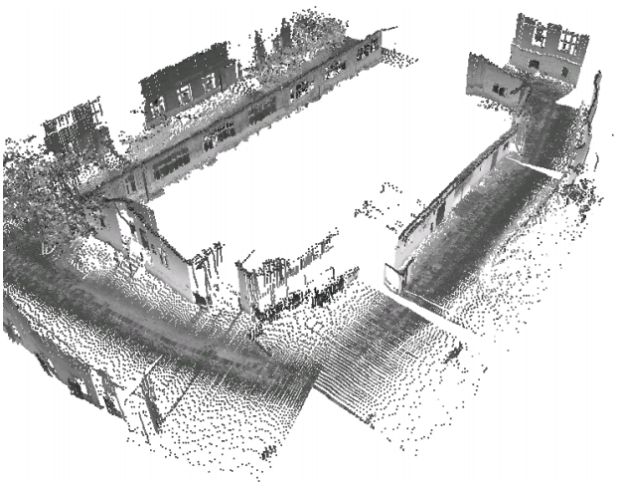
2.2 SLAM

Simultaneous localization and mapping (*SLAM*) [11] je problém s lokalizací, který se ptá na to, zda je možné, aby mobilní robot byl umístěn v jemu neznámém místě a prostředí , přičemž by dokázal určit svoji polohu v rámci uložené mapy a zároveň na ní dokázal navázat. Vyřešení tohoto problému znamená, že mobilní robot by byl zcela autonomní. Ačkoliv *SLAM* byl formulován a dá považovat na teoretické úrovni za vyřešený, při implementaci v mnoha formách však přetrvávájí významné praktické problémy.

S pojmem *SLAM* také souvisí i pojem *DRIFT*. Ten značí chybu, která nastává při sledování orientačních bodů při jízdě robota, který je ukládá. Každý orientační bod je totiž lehce nepřesný a vykazuje tak malou chybu. Akumulace těchto chyb však na dlouhé trajektorii jízdy může způsobit jednu velkou a tím zásadně ovlivnit další pohyb robota.

2.3 Visuální lokalizace

Visuální lokalizace spočívá v práci s okolím, ve které se mobilní robot nachází. Robot zachytí 3D scénu pomocí kamery a vytvoří z ní obrázek ve 2D. Pořízená scéna však obsahuje vysoký podíl šumu, který systém nevyužije a navíc ohrožuje kvalitu lokalizace. Proto se kvůli tomu snaží najít takzvané klíčové body. Pod klíčovým bodem si můžeme představit místa na pořízeném obrázku s velkým kontrastem a nebo dokonce i celé předměty. Funkcím, které nám dokáží vygenerovat ze scény tyto klíčové body, říkáme extraktory. Jakmile systém má tyto klíčové body vygenerované, může si je uložit do své 3D lokální mapy, kterou si postupně staví z jednotlivých pořízených snímků při své cestě. Tímto způsobem dokáže robot zmapovat okolí trasy z hypotetického bodu A do bodu B.



Obrázek 4: Příklad 3D lokální mapy reprezentované body [1]

Při autonomní jízdě se robot řidí podle své uložené lokální mapy. Je stejně jako při mapování závislý na aktuálním pohledu z kamery. Jakmile jsou vygenerované klíčové body, nahrají se i ty uložené. Následně probíhá porovnání (*matching*), ve kterém se algoritmus snaží najít shodné dvojice klíčových bodů pomocí určeného kritéria. Funkce, které porovnávají tyto význačné body mezi sebou, se nazývají deskriptory. Tyto deskriptory pracují právě s klíčovými body, nikoliv s celým obrazem tak, aby se zajistila rychlost a kvůli výše zmíněnému šumu, který k lokalizaci nepotřebujeme. Výsledek těchto algoritmů je pak poslán navigátoru, který na změněnou situaci patřičně reaguje, například změnou směru jízdy.

3 Vznik obrazu

Při vizuální lokalizaci se robot lokalizuje pomocí scény kolem sebe. Aby však z této scény získal potřebná data, musí využívat kameru, která mu je dokáže zprostředkovat. Ta zachytí prostředí před sebou, které je ve 3D, a převede ho do obrázku, tedy do 2D. Při této tranformaci však dochází ke zkreslení, které dokáže výsledný obrázek negativně ovlivnit. Nejen kvůli tomu je dobré se zabývat principem kalibrace kamery.

Kalibrace kamery je proces určování geometrických a optických charakteristik vnitřní kamery, neboli tzv. vnitřních parametrů, a/nebo 3D orientaci kamery k určitému světovému souřadnicovému systému, neboli tzv. vnějších parameterů. Jinými slovy, výsledek kalibrace můžeme rozdělit na vnitřní a vnější parametry kamery. Vnitřní parametry slouží například k úpravě pořízeného obrazu tak, aby eliminoval přirozenou deformaci, kterou způsobuje zachycení obrazu objektivem. Vnější parametry kamery slouží k určení polohy a natočení kamery vzhledem k nějakému bodu. V mnoha případech zavisí výkon systému pracujícího se strojovým viděním právě na přesnosti kalibrace.

3.1 Vnitřní parametry kamery

Jak jsem již naznačil výše, vnitřní parametry kamery určují optické charakteristiky, které ve výsledku ovlivňují pořízený výsledek. Mezi vnitřní parametry patří:

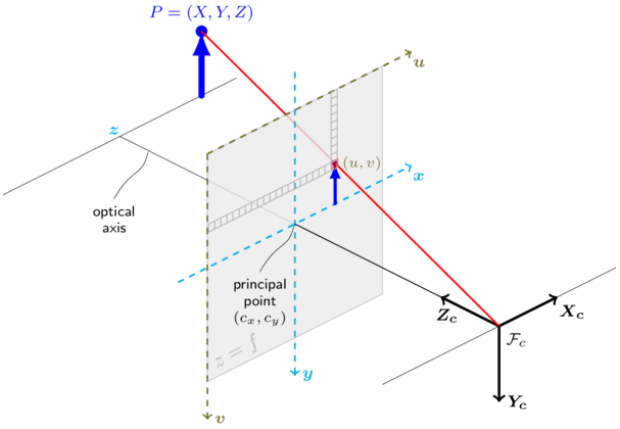
- Ohnisková vzdálenost
- Poloha hlavního snímkového bodu
- Součinitel zkosení, který definuje úhel mezi osami x a y
- Radiální a tangenciální deformace neboli součinitel zkreslení obrazu

Jakákoliv scéna je zachycená kamerou v prostoru, tedy v $\mathbb{R}^3$, ale rovina, kterou objektiv následně snímá, je v $\mathbb{R}^2$. Tomuto zobrazení zachycující obraz kamery z $\mathbb{R}^3$ do $\mathbb{R}^2$ se říká perspektivní projekce. U této projekce je scéna definována jako komolý jehlan [12]. Pokud je objekt o určitých rozměrech blíže, zabírá větší část plochy, něž kdyby byl objekt dále od ohniska. Pro znázornění využijeme perspektivní model kamery (Pinhole camera model) jako rovnici 1 a nebo rozepsanou rovnici 2, kde (x, y, z) jsou souřadnice bodů v trojrozměrném prostoru v určitých světových souřadnicích, (u, v) jsou souřadnice projekčního bodu v pixelech, $\mathbf{A}$ je matice kamery nebo jinak také matice vnitřních parametrů, (c_x, c_y) je hlavní bod, který býva obvykle v centru obrazu a f_x, f_y jsou fokální délky.

$$\alpha \boldsymbol{m} = \mathbf{A}[\mathbf{R}|\boldsymbol{t}]\boldsymbol{m'} \tag{1}$$

$$\alpha \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

Pokud je obraz z kamery měněn jakýmkoliv faktorem, parametry z matice vnitřních parametrů **A** se budou měnit také. Avšak zůstávají vždy stejné, pokud jde o závislost na sledované scéně. Proto jakmile je matice jednou odhadnuta, není již potřeba se jí dále zabývat. Matice vnějších parametrů $[\mathbf{R}|\boldsymbol{t}]$ v sobě skývá translaci a rotaci, která popisuje pohyb kamery kolem statické scény nebo naopak. Překládá tak souřadnicové body (x, y, z) do souřadnicového systému. Obrázek 5 ilustruje tento model graficky.



Obrázek 5: Pinhole camera model [2]

Nechť P je bod v prostoru se souřadnicemi $[x_c, y_c, z_c]$ v referenčním rámci kamery. Pak X_n je normalizovaná projekce obrazu:

$$X_n = \begin{bmatrix} x_c/z_c \\ y_c/z_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

$$r^2 = x^2 + y^2 \tag{4}$$

Po zahrnutí zkreslení objektivu je nová souřadnice normalizovaného bodu X_d definována:

$$X_d = \begin{bmatrix} x_{d1} \\ x_{d2} \end{bmatrix} = (1 + \boldsymbol{kc_1} r^2 + \boldsymbol{kc_2} r^4 + \boldsymbol{kc_5} r^6) X_n + \boldsymbol{d_x} \tag{5}$$

kde $\boldsymbol{d_x}$ je vektor tangenciálního zkreslení:

$$\boldsymbol{d_x} = \begin{bmatrix} 2\boldsymbol{kc_3} xy + \boldsymbol{kc_4}(r^2 + 2x^2) \\ \boldsymbol{kc_3}(r^2 + 2y^2) + 2\boldsymbol{kc_4} xy \end{bmatrix} \tag{6}$$

5-vektorový ***kc*** tedy obsahuje jak radiální, tak i tangenciální koeficienty zkreslení. Za zmíňku stojí, že tangenciální zkreslení je způsobeno "decentrováním" nebo nedokonalým centrováním součástí čoček a jiných výrobních vad ve složené čočce. Jakmile se použije zkreslení, konečné pixelové souřadnice projekce P na obrazové rovině jsou:

$$x_p = f_x(x_{d1} + alpha * x_{d2}) + c_x \tag{7}$$

$$y_p = f_y x_{d2} + c_y \tag{8}$$

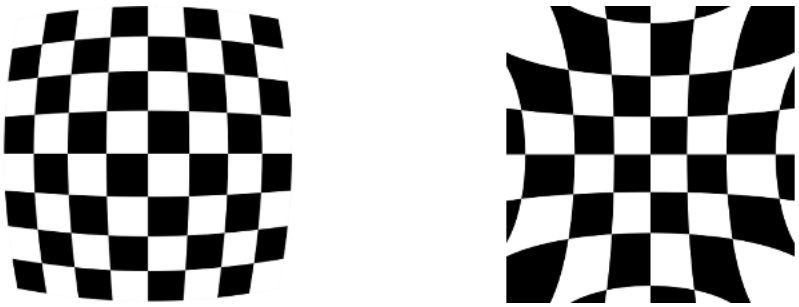
Vektor souřadnic pixelů a normalizovaný (zkreslený) vektor souřadnic jsou tedy navzájem spojeny lineární rovnicí:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathbf{R} \begin{bmatrix} x_{d1} \\ x_{d2} \\ 1 \end{bmatrix} \tag{9}$$

kde $\mathbf{R}$ je již zmíněná matice kamery, která má v sobě ale navíc zahrnutou $alfu$ značící součinitel zkosení, neboli úhel mezi osami x a y :

$$\mathbf{R} = \begin{bmatrix} f_x & alfa * f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Obrázky 6a a 6b ještě ukazují dvě nejčastější radiální zkreslení - positivní a negativní.



(a) Positivní radiální zkreslení (b) Negativní radiální zkreslení

Obrázek 6: Radiální zkreslení [2]

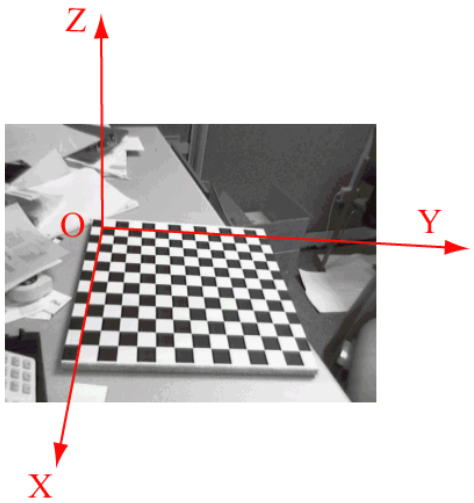
3.2 Vnější parametry kamery

Mezi vnější parametry kamery patří:

- Rotace

- Translace

Definujeme si rotaci jako set rotačních maticí 3x3 $\mathbf{R}_1$, $\mathbf{R}_2$ ..., $\mathbf{R}_{20}$ a translaci jako vektory 3x1 $\boldsymbol{t_1}, \boldsymbol{t_2}, ...\boldsymbol{t_{20}}$.



Obrázek 7: Referenční rámec (O, x, y, z) připojený ke kalibrační mřížce [3]

Nechť O je bod se souřadnicemi $[x, y, z]$ na referenčním rámci na obrázku 7. Nechť $\boldsymbol{x_c} = (x_c, y_c, z_c)$ je souřadnicový vektor O v referenčním rámci kamery. Pak O a $\boldsymbol{x_c}$ jsou navzájem propojeny pomocí rovnice 11.

$$\boldsymbol{x_c} = \mathbf{R}_i * O + \boldsymbol{t_i}, i = 1, ..., 20 \tag{11}$$

Jakmile jsou souřadnice referenčního bodu vyjádřeny v rámci kamery, mohou bý promítnuty do obrazové roviny pomocí vnitřních parametrů.

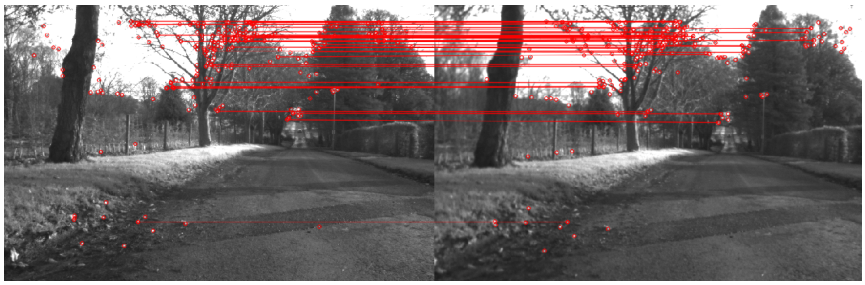
3.3 Využití pro navigační systém

V mé práci kalibrace kamery hraje významnou roli, jelikož pomocí vnitřních parametrů kamery přiřazuji klíčovým bodům souřadnice v 3D prostoru. S těmi následně počítám translaci a rotaci současného pohledu objektivu s lokálním uloženým pohledem, neboli využívám vnějších parametrů.

Důležité informace o kalibraci kamery při zpracovávání kapitoly jsem čerpal z článků a prácí [13], [14], [15], [16], [17] a webové stránky [2], a[3].

4 Zpracování obrazu

Po pořízení obrazu prostřednictvím kamery putují data ke zpracování. Nejdříve se generují klíčové body pomocí extraktorů, které se následně ukládájí do lokální mapy okolí anebo se porovnávají (*matching*) a vyhodnocují s již uloženými body pomocí deskriptorů.



Obrázek 8: Příklad probíhajícího *matchingu* systému

V dnešní době existuje spousta extraktorů i deskriptorů. Mezi zajímavé extraktory patří například *Hessian-Laplace Region* (*SURF*) [18], *Features from Accelerated Segment Test* (*FAST*) [19] nebo *Oriented FAST and Rotated BRIEF* (*ORB*) [20]. Mezi zajímavé deskriptory naopak patří třeba *Speeded Up Robust Features* (*SURF*) [18], *Scale Invariant Feature Transform* (*SIFT*) [21], *Oriented FAST and Rotated BRIEF* (*ORB*) [22] nebo *Binary Robust Independent Elementary Features* (*BRIEF*) [23]. Některé extraktory i deskriptory kombinují již předchozí vytvořené algoritmy. Za zmínku také stojí algoritmus *GRIEF* (*Generated BRIEF*) [24], který je uzpůsoben změnám prostředí, ve kterém se robot pohybuje. Těmi se rozumí změna světla v prostředí (den / noc) nebo střídání ročního období.

4.1 AGAST

Adaptive and Generic Accelerated Segment Test (*AGAST*) je rohový extraktor [25], který pracuje na stejném kritériu jako *Accelerated Segment Test* (*AST*), tedy i jako *FAST*. Pracuje na principu rozhodovacího stromu, který je však od *FAST* rozdílný. Je vyškolen na datové sadě, která zahrnuje všechny možné kombinace pixelů v kruhu [26]. To zajišťuje, že algoritmus funguje v jakémkoliv prostředí. Navíc má dynamický algoritmus, který automaticky přepíná mezi stromy - homogenním a heterogenním. Díky tomu se výkonnost zvyšuje u náhodných scén a celkově je rychlejší.

4.2 SURF

Speeded-Up Robust Features (*SURF*) je už starší, ale stále velmi populární extraktor i deskriptor. V práci [27] autoři uskutečnili zrychlení detektoru pomocí integrálních obrazů. Výsledky ukázaly, že výkon jejich Hesenské aproximace je přinejmenší srovnatelný, často však lepší než moderní detektory ve sledované době, tedy v roce 2008. Důležité ovšem je,

že rychlost detektoru je i bez speciálních optimalizací téměř real-time bez ztráty výkonu, což představuje výhodu pro on-line počítačové aplikace pracující s viděním. Stejně jako detektor, tak i deskriptor *SURF* překonal většinu tehdejších moderních metod. Autoři tvrdí, že jejich algoritmus je konkurenceschopný z hlediska rychlosti. Přestože *SURF* už existuje delší dobu, stále je to velmi používaný nástroj. V práci [28] autoři srovnávají deskriptory *SURF*, *SIFT*, *FAST* a *ORB* z hlediska přizpůsobení tranformace a deformace. *SURF* vynikl ve speciálním případě, kdy se kamera otočila o téměř 90 stupňů oproti původní pozici. V práci z roku 2015 [29] autoři analyzovali *SURF* a napsali, že tento algoritmus je stále výkonnostně srovnatelný s dnešními nejmodernějšími metodami, které porovnávají obraz.

4.3 U-SURF

Upright SURF (*U-SURF*) je extraktor i deskriptor, který funguje jako běžný *SURF*, ale ignoruje výpočet dominantní orientace a následné rotace sousedství klíčových bodů [30]. Jelikož není rotačně invariantní, je užitečný pro detekci a popis bodů v obraze, které se liší posunem či rotací v rovině. V práci [31] navíc autoři píší, že *U-SURF* je odolnější vůči osvětlení než původní *SURF* nebo *SIFT*.

4.4 BRIEF

Binary Robust Independent Elementary Features (*BRIEF*) je deskriptor, který používá jednoduché binární testy mezi pixely ve vyhlazeném obrazu. Opírá se o relativně malý počet těchto testů rozdílů intezity. Spojování klíčových bodů je zde rychlejší než u většiny moderních deskriptorů a navíc lze dosáhnout výsledků i na stroji s omezeným výkonem v reálném čase [23]. V [32] autoři píší, že výkonnostně podobný jako SIFT a to v mnoha ohledech - robustnost osvětlení a zkreslení perspektivy. Je však velmi citlivý na rotaci v rovině.

4.5 Convolutional Neural Networks

Některé z projektů, které vyvíjejí visuální navigační systémy, se často zmiňují o metodě *Convolutional Neural Networks* (*CNN*) nebo jí přímo aplikují. Jedná se o neuronovou síť, která se samovolně učí. Tuto síť je možné využít jako extraktor, tak i jako deskriptor. Před použitím se však musí vytrénovat nad miliónem testovacích dat.

V [33] se zmiňují o *CNN* jako o velmi dobrém systému pro detekci objektů v okolí, kterému nevadí změna světla ani prostředí. Píší, že detekce je tak dobrá, že jí lze využít pro nalezení klíčových bodů, podle kterých se robot může orientovat. Systém rozezná stůl, židli nebo deštník, a to i po jejich otočení nebo nahrazení jiným duplikátem. V této práci využívají algoritmus YOLO-2 CNN. Testy však probíhaly pouze ve vnitřních prostorách

na drobném létajícím dronu. V práci [34] autor píše o *CNN* jako o robustním řešení pro měnící se prostředí. Píše ale také o tom, že toto řešení je výpočetně náročnější než binární deskriptory. V [35] se autoři zmiňují o novém přístupu k teach and repeat úloze. Ta využívá pro řešení úkolů rovněž neuronovou síť. Experimenty naznačují, že síť je schopná se naučit zadanou trasu, a dokonce jí předpovídat. V práci také zmiňují výhody *CNN*, a to ty, že neuronová síť má při práci fixní paměť a časovou složitost při inferenci. Dále je snížena celková složitost systému a je snadno přizpůsobitelná jiným modalitám. Testovací robot však umí pouze tři příkazy (vpřed, doleva, doprava), nevyužívá visual servoing a testování probíhalo hlavně ve vnitřní prostorách. V [36] autoři ohodnotili jednotlivé extraktory s různými deskriptory při změnách prostředí. Byla mezi nimi i neuronová síť jakožto deskriptor. Nejlépe dopadly z hlediska robustnosti následující dvě dvojice (extraktor/deskriptor): *SpG*/*CNN* a *STAR*/*GRIEF*.

Obrázek 9: Ukázka létajícího drona [1]

Neuronové sítě se staly v posledním době velkým trendem a dostaly se i k vizuálnímu navigačnímu systému. Jak je vidět, zatím si nevedou špatně a pro vývojáře jsou atraktivní. Bude rozhodně zajímavé sledovat, kam až tento trend dospěje.

5 Popis navigačního systému

Představte si situaci, kdy chceme zajít s kamarády na pivo do restaurace. Tato restaurace je ale příliš daleko a my jsme líní cestovat hodinu hromadnými dopravními prostředky. Při představě, že bychom tuto trasu měli absolvovat nazpět, se nám rázem nikam nechce. Zároveň nechceme použít ani automobil, protože máme chuť na ty dvě piva a nechceme riskovat kvůli tomu řidičský průkaz. Co kdyby ale existoval způsob, jak přijet do restaurace automobilem a zpátky se nechat odvést domů, aniž bychom museli platit za řidiče. Jednoduše byste při cestě do restaurace stiskli tlačítko pro natočení trasy, po které aktuálně pojedete, a po skončení akce, by Vás automobil odvezl poté samé cestě zpět až k Vašemu domovu. Nebo si představme, že v továrně na automobily potřebujeme naučit robota svářet dveře od vozidla. Robotovi řekneme, ať uvolní klouby a zapisuje si pohyby, které s ním profesionální svářeč vykoná. Stroji následně přikážeme, ať tento pohyb automaticky opakuje stále dokola. Přesně těmito myšlenkami je inspirovaný následující navigační systém *STRoLL BearNav*.

STRoLL BearNav je jednoduchá vizuální navigace založená na metodě teach-and-repeat. Je přizpůsobená vlivům rušení měnícího se světla a přirozených změn prostředí, jako jsou kupříkladu různá roční období. Systém je dále schopen opravovat si aktuální cestu, pokud zjistí, že nejede podle své stanovené trajektorie. K tomu využívá visuální scény, které vidí před sebou prostřednictví stereo kamery. Nepotřebuje kalibraci kamery a je výpočetně efektivní. Díky tomu dokáže běžet i se senzory, jejichž pořízení není finančně náročné. Další výhodou je jednoduchá implementace a výpočetní nezávislost prostředí, ve kterém se nachází.

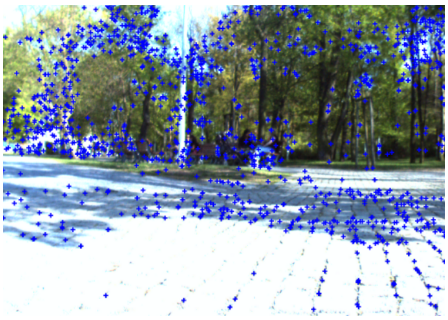
5.1 Extrakce bodů

Navigační systém *STRoLL BearNav* pracuje také s význačnými body. Tyto body se extrahují pomocí detektoru, který v našem případě může být buď *AGAST*, *SURF* nebo *upSURF*. Extrakce bodů probíhá ve dvou krocích. Nejdříve se detekují klíčové body, které jsou v dostatečném kontrastu s okolím, a následně se udělá popis jejich blízkosti. Tyto význačné body se pak dále porovnávají a spojují do párů podle shody pomocí deskriptoru *BRIEF* nebo *SURF*. Kvalita výsledku pak závisí na kvalitě vstupního obrazu z kamery.

5.2 Teach and repeat

Jak jsem již zmínil výše, systém využívá metodu teach-and-repeat. Metodu můžeme rozdělit na dvě fáze - fáze učení a fáze autonomní navigace. Fáze učení spočívá v tom, že robot s lidským operátorem projedou cestu a uloží si potřebné parametry pro pohyb a lokalizaci do svého lokálního úložiště. Zjednodušeně můžeme říci, že se systém učí kudy a jak má jet. Ve druhé fázi si robot nahraje svá uložená data a jede po té samé trajektorii, kterou před tím projel s operátorem, zcela sám.

(a) Vnitřní scéna



(b) Venkovní scéna

Obrázek 10: Extrakce bodů [1]

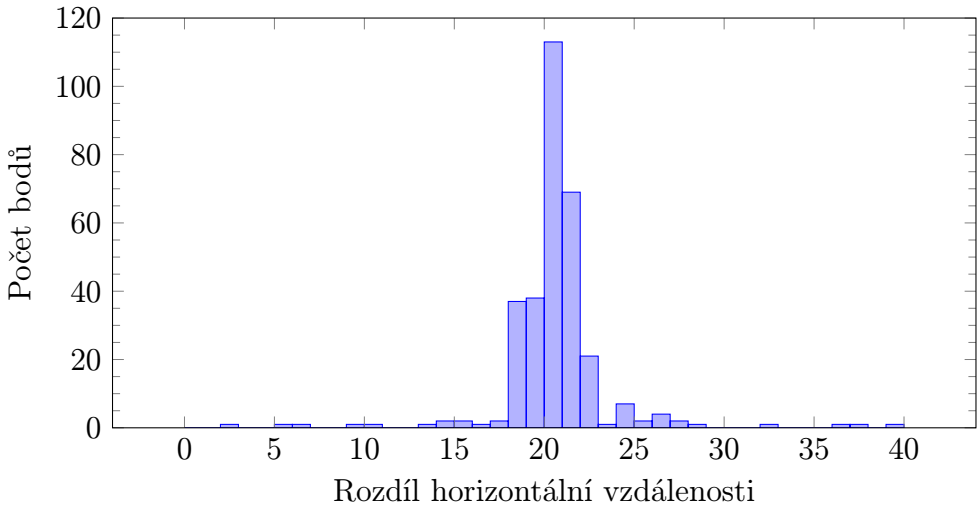
5.3 Fáze učení

Ve fázi učení je robot navigován operátorem pomocí joysticku nebo jiného ovládacího zařízení, se kterým projede určenou cestu. Systém na začátku zaznamená počáteční extrahované význačné body a nastaví ujetou vzdálenost na nulu. Následně si po celou dobu počítá tuto distanci, kterou urazil od počátečního místa. Při každé změně dopředné nebo úhlové rychlosti distanci zaznamená i s příkazy, které byly vykonány, do takzvané *lokální mapy*. Čili si vytváří takzvaný *profil cesty*, který se skládá z těchto map. Navíc k tomu ještě každých 0,2 m detekuje extrahované body z aktuálního obrázku, a ukládá je s indexem ujeté vzdálenosti.

5.4 Autonomní navigace

Jakmile je učení dokončeno, může začít fáze autonomní navigace, kdy je robot postaven na počáteční místo a operátor určí mapu, kterou se má řídit. Robot vzápětí načte počáteční *lokální mapu* a s ní i příkazy, které má provést. Následně již postupuje podle těchto příkazů a kdykoliv, kdy ujede určenou distanci, načte novou *lokální mapu* podle indexu vzdálenosti a jede podle ní dál.

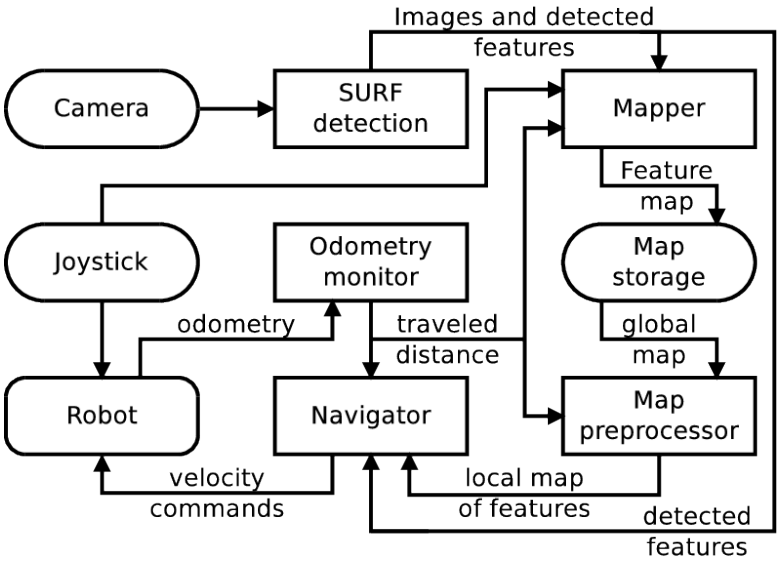
Při vykonávání tohoto procesu však může docházet k nepřesnostem při otáčení robota, pohybu vpřed nebo už i při samotném položení robota na počáteční místo. Nikdy stroj nepoložíme dvakrát přesně na to samé místo. Po pár iteracích té samé cesty mohou zmíněné nepřesnosti být velmi viditelné. Kvůli tomu při jízdě robotovi přicházejí z kamery obrázky s vyextrahovanými význačnými body, které porovnává s body, které uložil ve fázi učení. Následně se vygeneruje histogram horizontální vzdálenosti, jehož maximum pak používá k upravení své úhlové rychlosti a přiblížení se k původní naučené cestě. Pokud není jednoznačně určeno maximum toho histogramu, stroj pokračuje po své trajektorii s původními příkazy.



Obrázek 11: Příklad histogramu vygenerovaného nad testovacími daty.

5.5 Implementace systému

STRoLL BearNav je naiplementován v *Robotickém operačním systému* (*ROS*) a používá verzi Kinetic. Celý systém je napsán v jazyce C++ a je volně dostupný na webových stránkách [37]. Jeho schematické zobrazení je možné vidět na obrázku 12, na kterém jsou vyobrazeny jednotlivé uzly (*nodes*). Mezi těmito uzly, také jinak moduly, probíhá pomocí komunikačních kanálů (*topics*) posílání zpráv (*messages*). Tyto zprávy mohou obsahovat různé informace od aktuální rychlosti stroje až po celý obrázek, který byl právě pořízen z objektivu.



Obrázek 12: Struktura navigačního systému STRoLL BearNav

Jednotlivé uzly byly naimplementovány jako akční servery, pro které umožňuje *ROS* ukázat jejich aktuální stav nebo nastavovat parametry přímo za provozu. Lze si také zobrazit, jaké obrázky a jaké extrahované body na nich se posílají mezi moduly. Uzly se mohou též vypnout nebo zapnout dle potřeby. Dokonce je možné změnit jejich kód, přeložit ho a restartovat pouze daný modul, aniž bychom museli vypínat celý operační *ROS*. Jednotlivé uzly navigačního systému *STRoLL BearNav* i s jejich komunikací jsou vysvětleny v následujících dvou odstavcích.

Kamera robota zaznamenává obrázek, který pošle do *feature extraction* (na obrázku nazvaný *SURF detection*). Zde se detekují jednotlivé význačné body. Tyto body se pak následně posílají do mapovacího (*Mapper*) a navigačního uzlu (*Navigator*). Uzel pro sledování vzdálenosti (*Odometry monitor*) dostává informace z odometrie od robota a počítá najetou vzdálenost. Následně je rozesílá k mapovacímu, navigačnímu a *Map preprocessor* uzlu. Také posílá speciální zprávu, pokud robot má uložit extrahované body neboli ujel v našem případě 0,2 m. *Mapper* zpracovává přijaté zprávy s význačnými body a ukládá je lokálně do mapy podle zpráv se vzdáleností od *Odometry monitor*. Ukládání však neprobíhá stále, ale pouze po přijetí speciální zprávy z předchozí věty. *Mapper* také ukládá *profil cesty*. Tedy pokud operátor změní při fázi učení směr nebo rychlost robota, *Mapper* zapíše dopřednou a úhlovou rychlost do svého lokálního úložiště.

Map preprocessor umožňuje načíst lokální mapu a profil cesty. Ty následně posílá do navigačního uzlu v závislosti na ujeté vzdálenosti. Navigační uzel obdrží uložené význačné body. Tyto body následně porovná s těmi, které mu pošle *feature extraction*. Tedy z toho obrázku, co právě robot pořídil z kamery. Vytvoří z nich histogram a následně vypočítá dopřední a úhlovou rychlost. Výsledek se pošle robotovi, který zopakuje příkazy z fáze učení a zároveň koriguje své řízení tak, aby zůstal na určené cestě.

Při zpracování kapitoly jsem čerpal z článků [38], [39], [40] a z webové stránky [37].

5.6 Nevýhody systému

Navigační systém *STRoLL BearNav* má významnou nevýhodu spočívající ve skutečnosti, že zanedbává vzdálenost klíčových bodů. Aby upřesnil svoji polohu, musí velmi často zatáčet. Tento handicap se dá odstranit tím, že všem vygenerovaným bodům naleznu z-souřadnici a tím pádem zjistím, jak daleko tyto body jsou. Budu moct tedy určit vzdálenost robota od místa pořízení uloženého pohledu a dokonce i jeho natočení oproti své předchozí jízdě. Tímto problémem se budu zabývat v následující kapitole, kde vysvětluji navrhované řešení.

6 Řešení

Navrhované řešení spočívá v zavedení třetí souřadnice, která bude znázorňovat vzdálenost kamery od objektů před ní. K výpočtu této souřadnice využívá princip stereo vidění, který spočívá v tom, že dva objektivy zachytí v jeden okamžik obraz předmětu, který je před robotem, z rozdílnéno místa. Každý z objektivů vidí předmět pod jiným konvergenčním úhlem. Čím je objekt blíže kameře, tím je úhel mezi nimi více rozdílný.

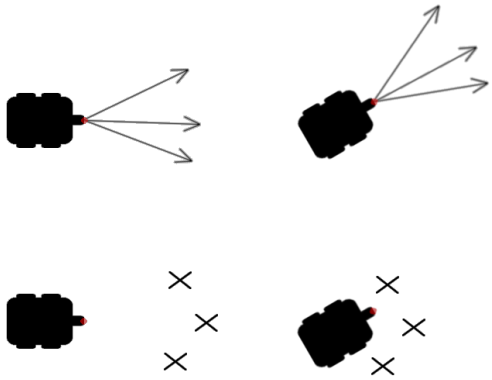
Obrázek 13: Ukázka stereo kamery [1]

V popisovaném případě je místo předmětu před kamerou klíčový bod, který je vygenerován pomocí detektoru. Těchto klíčových bodů systém vygeneruje velké množství, a to jak z pravého, tak i levého objektivu. Ty následně porovná a najde shodné páry, u kterých vypočítá z-souřadnice pomocí rovnice 12, kde x_1, x_2 znamenají x-ové souřadnice klíčového bodu z prvního a druhého obrázku a c je konstanta násobku $z * x$. Tuto konstantu jsem si vypočítal tak, že jsem před kameru vkládal předměty, které měly velký kontrast a byly tak dobře zjistitelné pro extraktor. Následně jsem změřil vzdálenost pomocí metru mezi předměty a kamerou a zjistil si, jaký je rozdíl v x-ových souřadnicích v konkrétních případech.

$$z = \frac{c}{abs(x_1 - x_2)} \tag{12}$$

Jak jsem již naznačil dříve, momentální řešení, které využívá *STRoLL BearNav*, generuje význačné body pouze se dvěma souřadnicemi x a y získanými z obrázku. Kvůli tomu systém může porovnávat pouze úhly mezi těmito body. Když však robot bude k terénu, který si před tím zaznamenal, špatně otočen, porovnání mohou selhávat. Díky řešení třetí souřadnice je tato nevýhoda eliminována. Na obrázku 14 je tento rozdíl ilustrován.

Díky zmíněnému řešení, mohu současně lépe určit, jak přesná odchylka nebo vzdálenost mezi současnou a původní trasou je. Toho dosáhnu pomocí kalibrace, o které jsem již psal v samostatné kapitole. Tedy nejdříve si převedu obrázkové souřadnice x a y do stejného souřadnicového systému jako je z, tedy do prostorového, a to za pomoci transformační matice stereo kamery. Jakmile mám souřadnice jak z lokální uložené mapy, tak i ze současného pohledu, mohu z nich vytvořit dvě množiny, které následně dosadím do rovnice 13 a vypočítám matici vnějších parametrů, která se skládá z rotace a translace. Výsledek následně



Obrázek 14: Porovnávání význačných bodů se dvěma a třemi souřadnicemi, před a po chybném natočení.

lze použít ke srovnání robota s jeho původní uloženou cestou nebo zjištění vzdálenosti od místa pořízení mapy.

$$\begin{bmatrix} \alpha * z' \\ \alpha * x' \\ \alpha \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ -a_1 & a_0 & a_3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ x \\ 1 \end{bmatrix} \tag{13}$$

Vydělením matic na levé straně rovnice hodnotou α a mi vzniknou jednotkové souřadnice. Navíc je pak možné rovnici zjednodušit vynecháním již nepotřebného posledního řádku.

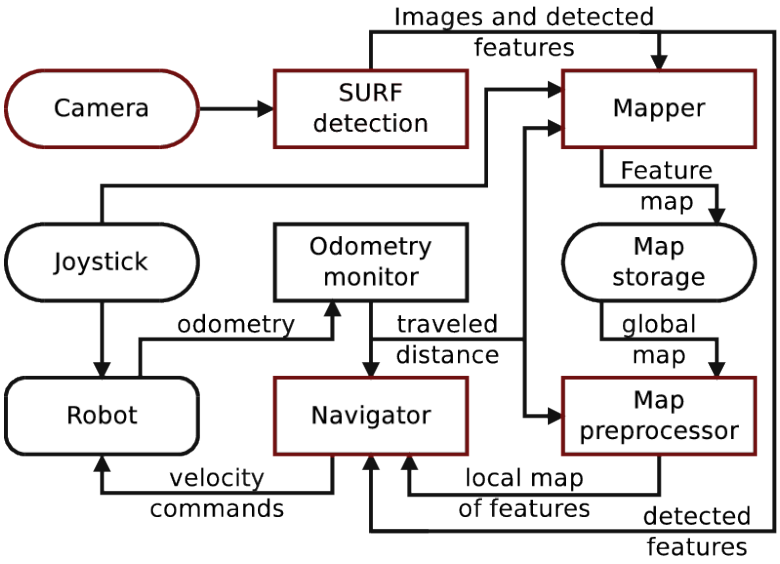
$$\begin{bmatrix} \alpha * z' \\ \alpha * x' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ -a_1 & a_0 & a_3 \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix} \tag{14}$$

6.1 Implementace

Při své implementaci do stávajícího systému jsem rozšířil celkově pět uzlů, jmenovitě *Camera*, *feature extraction*, *Mapper*, *Map preprocessor* a *Navigator*. Pro ilustraci uvádím obrázek 15.

V uzlu *Camera* jsem dopsal část kódu, který posílá scény z obou objektivů kamery spojené do jednoho obrázku. Ve *feature extraction* jsem rozšířil metodu, která generuje význačné body, o část kódu, která porovnává tyto body a následně je spojuje do párů pomocí deskriptoru. U těchto bodů se pak vypočítává jejich x, y a z souřadnice. Klíčové body se na konci algoritmu posílají do *Mapperu* a *Navigatoru*.

Uzly *Mapper* a *Map preprocessor* jsem rozšířil pouze o funkce, které mi ukládájí souřadnice vedle význačných bodů do dlouhodobé paměti stroje a nebo naopak z ní načítají .



Obrázek 15: Obrázek navigačního systému *STRoLL BearNav* rozšířený o uzly s implementací označeně červeně.

V posledním uzlu *Navigator* jsem dopsal metodu, která hledá nejlepší transformační matici. K hledání matice jsem využil algoritmus *Random sample consensus* (*Ransac*), který funguje na principu naprosté náhody. Algoritmus si náhodně vybere dvě hodnoty z množiny bodů z aktuálního pohledu, pro které spočítá transformační matici a následně nad ní vyzkouší i všechny ostatní. Při tom si počítá ty, které po vynásobením výslednou maticí odpovídají bodu z lokální mapy s určitým možným prahem odchýlení. Po skončení vrátí tu matici, která byla nejlepší.

7 Testování

Svoji práci jsem testoval pomocí samostatné stereo kamery. Nejdříve jsem natočil tzv. *rosbag*, což je balík, ve kterém jsou uložena data z jednotlivých *topics* za určitý čas, a následně jsem pomocí těchto testovacích dat svoji práci otestoval.

7.1 Testovací data

Testovací data jsem nasbíral na dvou místech - ve venkovním a vnitřním prostředí. Vždy jsem začínal pozicí, od které jsem následně otáčel kameru pod daným úhlem a/nebo která měla odlišnou vzdálenost od objektů před kamerou. Každá ze změn má vlastní *rosbag*. Jelikož mé rozšíření porovnává pouze změnu vzhledu vůči uloženému, stačí, aby každý *rosbag* měl pouhou sekundovou stopáž.

7.1.1 Vnitřní prostředí

Testovací data byla natáčena uvnitř domu směrem k parapetu a oknu. Při měření jsem si i poznamenal jejich vzdálenost. Podrobné informace uvádím v tabulce 1.

	Vzdálenost od parapetu	Vzdálenost od okna	Změna vzdálenosti vůči 1.	A	B	C	D	E
1.	50cm	76cm	0cm	0°	10°	15°	-10°	-15°
2.	40cm	66cm	10cm	0°	15°	-10°		
3.	60cm	86cm	-10cm	0°	10°	-15°		
4.	80cm	106cm	-30cm	0°				

Tabulka 1: Popis testovacích dat ve vnitřním prostředí

Obrázek 16: Ukázka testovacích dat uvnitř budovy

7.1.2 Vnější prostředí

Testovací data byla natáčena venku na zahradě u domu směrem k živému plotu. Více informací uvádím v tabulce 2.

	Změna vzdálenosti vůči 1.	A	B	C	D	E
1.	0cm	0°	10°	-10°	15°	-15°
2.	15cm	0°	15°	-15°		
3.	-15cm	0°	10°	-10°		

Tabulka 2: Popis testovacích dat ve venkovním prostředí

Obrázek 17: Ukázka testovacích dat ve venkovním prostředí

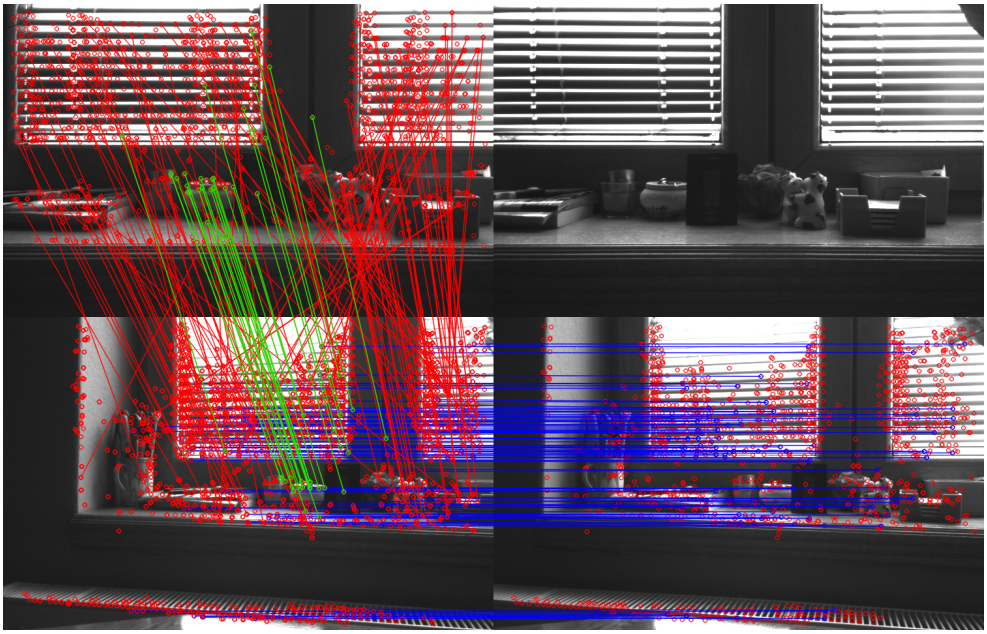
7.2 Provedené testování

Testování jsem prováděl nad získanými daty a výsledky předkládám v tabulkách 3 pro vnitřní a 4 pro venkovní prostředí. Z obou tabulek lze vyčíst očekávaný úhel a změnu vzdálenosti vůči uložené mapě. Následně je zde pořízený úhel i zmíněná vzdálenost, kterou navigační systém vygeneroval. Důležitá je poslední část tabulky, která říká, kolik bodů je pozitivních z celkových nalezených souřadnic. Jinými slovy pro které vygenerovaná tranformační matice dokáže převést současné body na ty uložené s určitým prahem odlišnosti.

Při testování jsem nechal vždy vygenerovat 3000 klíčových bodů a využil algoritmus SURF jako extraktor i deskriptor. Práh odlišnosti jsem nastavil na 7 centimetrů. Vždy jsem se snažil najít tranformační matici s nejvíce pozitivními body. Pokud jsou v řádku nuly, znamená to, že systém nedokázal najít žádnou odpovídající tranformační matici. Jinými slovy se nedokázaly najít a spojit alespoň dva páry klíčových bodů, které by měly podobné souřadnice vůči uloženým po tranformaci, kvůli velkému natočení kamery a odlišné vzdálenosti od lokální mapy.

	Očekávaný úhel	Očekávaná vzdálenost	Pořízený úhel	Pořízená vzdálenost	Positivních bodů	Celkově bodů
1-A	0°	0cm	0°	0cm	464	464
1-B	10°	0cm	0.22°	+5.0cm	5	73
1-C	15°	0cm	1.94°	5.6cm	4	83
1-D	-10°	0cm	-12.13°	-4.9cm	7	90
1-E	-15°	0cm	-14.04°	-2,6cm	5	77
2-A	0°	+10cm	-3.33°	13cm	11	134
2-B	15°	+10cm	23.60°	26,7cm	5	69
2-C	-10°	+10cm	-12.87°	-2.2cm	5	73
3-A	0°	-10cm	-0.47°	-16.5cm	5	73
3-B	10°	-10cm	-32.44°	-12.5cm	4	68
3-C	-15°	-10cm	-29.76°	-9.6cm	5	87
4-A	0°	-30cm	-33.23°	-16.2cm	8	133

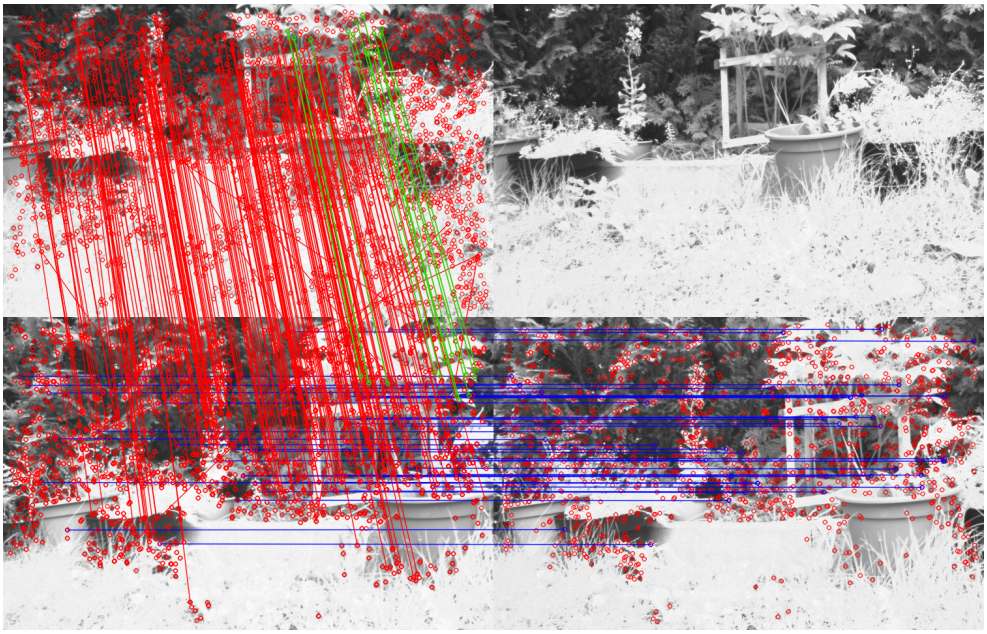
Tabulka 3: Výsledky z testovacích dat pořízené uvnitř budovy.



Obrázek 18: Ukázka testování uvnitř budovy: modře *matching* bodů pro výpočet souřadnice z, červeně chybně určený *matching* současného a uloženého pohledu a zeleně správně určený *matching* současného a uloženého pohledu.

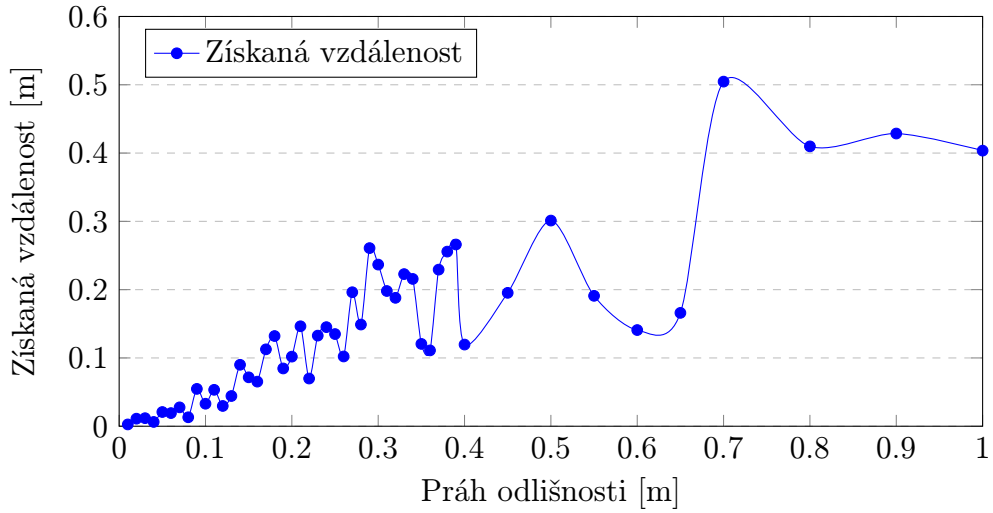
	Očekávaný úhel	Očekávaná vzdálenost	Pořízený úhel	Pořízená vzdálenost	Positivních souřadnic	Celkově nalezených souřadnic
1-A	0°	0cm	-0.12°	0.8cm	37	159
1-B	10°	0cm	10.59°	-4.4cm	2	63
1-C	-10°	0cm	-10.58°	+11.5cm	2	62
1-D	15°	0cm	0°	0cm	0	0
1-E	-15°	0cm	0°	0cm	0	0
2-A	0°	+15cm	-0.05°	+17.8cm	2	35
2-B	15°	+15cm	0°	0cm	0	0
2-C	-15°	+15cm	0°	0cm	0	0
3-A	0°	-15cm	-1.56°	-0.5cm	2	30
3-B	10°	-15cm	2.79°	189.4cm	2	33
3-C	-10°	-15cm	-7.28°	-14.8cm	1	32

Tabulka 4: Výsledky z testovacích dat pořízené ve venkovní prostředí.



Obrázek 19: Ukázka testování uvnitř budovy: modře *matching* bodů pro výpočet souřadnice z, červeně chybně určený *matching* současného a uloženého pohledu a zeleně správně určený *matching* současného a uloženého pohledu.

Při testování jsem používal práh odlišnosti, kterým se určovaly pozitivní body pro vygenerovanou matici od ostatních bodů. Práh jsem si po delším zkoumání nastavil na 7cm, jelikož získané hodnoty nejsou při tomto čísle nikterak významně odlišné. V grafu 20 jsou ilustrovány výsledky, které jsem při zkoumání naměřil. Graf reprezentuje závislost získané vzdálenosti z tranformační matice při nulové změně scény od lokální mapy vůči změnám prahu odlišnosti, tedy maximální možné odchylky mezi aktuálním a uloženým bodem po transformaci. Při testování jsem bral v potaz vždy největší hodnotu, co mi transformační matice poskytla. Jak je z grafu zřejmé, při zvyšování prahu odlišnosti se zvětšuje i obsažený šum a rozdíl mezi vygenerovanými vzdálenostmi je tím pádem více kolísavý.



Obrázek 20: Závislost získané vzdálenosti při nulové změně scény vůči prahu odlišnosti.

7.3 Zhodnocení testování

Z výsledků testování je zřejmé, že algoritmus se velmi dobře dokáže lokalizovat vůči své uložené mapě, pokud se nachází přesně na stejném místě. Při rotaci nebo změně vzdálenosti od lokální mapy se už úspěšnost snižuje. Hlavním důvodem je malý počet pozitivních bodů, pro které je tranformační matice vygenerována. To souvisí i s tím, že při extrahování a spojování bodů vzniká šum, který dokáže ovlivnit výslednou matici. Algoritmus *Ransac* dokáže pracovat i s pouze 10 procenty správných výsledků, aby nevypisoval chybu. Jelikož však v mnoha případech se ani nenalezne těchto 10 procent a navíc není ani celkových spojených bodů se souřadnicemi dostatečné množství, nedokážeme v těchto případech vyvrátit, zda se nespočítala transformační matice pro šum.

Také musím podotknout, že testování ve vnitřním prostředí bylo úspěšnější než ve venkovním. To lze hlavně vyčíst z počtu pozitivních bodů, ale i z toho, že v některých případech nebylo možné získat tranformační matici. Tento jev bude způsoben odlišností scény před robotem, a to hlavně vzdáleností od objektů, která je úmyslně větší.

8 Závěr

Cílem mé práce bylo navrhnout řešení pro zlepšení robustnosti stávajícího vizuálního navigačního systému *STRoLL BearNav* vůči odometrii, kterou mobilní roboti disponují. Díky kameře, kterou jsou vybavení, získávám informace o prostředí, ve kterém se pohybují. Na základě těchto informací jsem navrhl řešení, při kterém využívám tato data k identifikaci klíčových bodů z pohledu před robotem. Jakmile je identifikuji, snažím se z nich získat informaci o třech souřadnicích, které tyto klíčové body reprezentují ve 3D prostoru. Tuto informaci mi samotný objektiv neposkytne. Díky tomu mohu pak určit přibližnou vzdálenost vygenerovaných bodů, kterou pak porovnávám s uloženými daty a mohu vygenerovat tranformační matici, která mi ukáže odchýlení současného pohledu od toho původního. Po vyhodnocení výsledků může robot změnit svůj směr tak, aby se srovnal přesně s původní trasou. Navíc pokud je vzdálenost po delší dobu stejná, lze konstatovat, že se robot nepohybuje vpřed v určené trajektorii.

Celý navigační systém je naimplentovaný v Robotickém operačním systému a je rozdělen do uzlů, které mezi sebou komunikují. Moje řešení je přímo implementováno do tohoto stávajícího systému rozšířením pěti uzlů o získání souřadnic a transformační matice.

Svoji práci jsem otestoval pomocí stereo kamery v budově a ve venkovním prostředí. Z výsledků plyne, že rozšíření funguje, pokud je aktuální pohled stejný jako uložený. Selhává však při větší rotaci či translaci. Důvodem je hlavně nedostatečné množství spojených bodů, které mají uložené souřadnice. Kvůli tomu také v určitých případech nedokážeme stanovit, zda je výsledná transformační matice správná nebo zda byla ovlivněna šumem, který přirozeně vzniká.

Reference

[1] Tomáš Krajnık. *Large-scale mobile robot navigation and map building.* PhD thesis, Ph. D. thesis, Czech Technical University in Prague, 1999, Draft, 2011.

[2] Camera calibration and 3d reconstruction. `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`. Accessed: 2019-04-05.

[3] Description of the calibration parameters. `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html`. Accessed: 2019-04-05.

[4] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3):376–382, 1991.

[5] David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003.

[6] Lucie Halodová. Údržba map pro dlouhodobou navigaci mobilních robotøu. B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2018.

[7] Samuel K Moore. Super-accurate gps coming to smartphones in 2018 [news]. *IEEE Spectrum*, 54(11):10–11, 2017.

[8] Kevin E Murphy. Light detection and ranging (lidar) mapping system, March 23 2004. US Patent 6,711,475.

[9] How does gps work? `https://spaceplace.nasa.gov/gps/en/`. Accessed: 2019-12-05.

[10] Marek Bílỳ. Přesnỳ inerciální navigační systém kategorie" tactical grade". Master's thesis, České vysoké učení technické v Praze., 2015.

[11] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[12] Nastavení projekce. `https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=22411`. Accessed: 2019-04-05.

[13] Janne Heikkila, Olli Silven, et al. A four-step camera calibration procedure with implicit image correction. In *cvpr*, volume 97, page 1106, 1997.

[14] Jaroslav Lištvan. Kalibrace kamery pro robotické pracoviště. B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2017.

[15] Erik Král. Analỳza obrazové informace kamerového systému. Master's thesis, Univerzita Tomáše Bati ve Zlíně, 2006.

[16] J Zahrádka. Rozšířené uživatelské rozhraní. *Dostupné na URL: http://www.fit.vutbr.cz/study/DP/DP.php*, 2011.

[17] M HASMANDA and ING KAMIL ŘÍHA. Detekce a korespondence vyznamnỳch bodøu v obraze, 2010.

[18] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[19] Deepak Geetha Viswanathan. Features from accelerated segment test (fast), 2009.

[20] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[23] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[24] Tomáš Krajník, Pablo Cristóforis, Matias Nitsche, Keerthy Kusumam, and Tom Duckett. Image features and seasons revisited. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–7. IEEE, 2015.

[25] Agast corner detector. `http://www.i6.in.tum.de/Main/ResearchAgast`. Accessed: 2019-12-05.

[26] Hongmou Zhang, Jürgen Wohlfeil, and Denis Grießbach. Extension and evaluation of the agast feature detector. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 3(4), 2016.

[27] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[28] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*, 2017.

[29] Edouard Oyallon and Julien Rabin. An analysis of the surf method. *Image Processing On Line*, 5:176–218, 2015.

[30] Christoffer Valgren and Achim J Lilienthal. Sift, surf & seasons: Appearance-based long-term localization in outdoor environments. *Robotics and Autonomous Systems*, 58(2):149–156, 2010.

[31] Philippe Dreuw, Pascal Steingrube, Harald Hanselmann, Hermann Ney, and G Aachen. Surf-face: Face recognition under viewpoint consistency constraints. In *BMVC*, pages 1–11, 2009.

[32] Jie Xu, Hua-wen Chang, Shuo Yang, and Minghui Wang. Fast feature-based video stabilization without accumulative global motion estimation. *IEEE Transactions on Consumer Electronics*, 58(3):993–999, 2012.

[33] Amirmasoud Ghasemi Toudeshki, Faraz Shamshirdar, and Richard Vaughan. Robust uav visual teach and repeat using only sparse semantic object features. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 182–189. IEEE, 2018.

[34] Nan Zhang. *Towards Long-Term Vision-Based Localization in Support of Monocular Visual Teach and Repeat*. PhD dissertation, University of Toronto Institute for Aerospace Studies, 2018.

[35] Tristan Swedish and Ramesh Raskar. Deep visual teach and repeat on path networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1533–1542, 2018.

[36] Tomáš Krajník, Pablo Cristoforis, Keerthy Kusumam, Peer Neubert, and Tom Duckett. Image features for visual teach-and-repeat navigation in changing environments. *Robotics and Autonomous Systems*, 88:127–141, 2017.

[37] Filip Majer, Lucie Halodová, Tomáš Vintr, and Tomáš. Krajník. Stroll bearnav, simple and robust visual teach-and-replay navigation. `https://github.com/gestom/stroll_bearnav`. Accessed: 2019-01-6.

[38] Filip Majer, Lucie Halodová, and Tomáš Krajnık. A precise teach and repeat visual navigation system based on the convergence theorem. In *Student Conference on Planning in Artificial Intelligence and Robotics (PAIR)*, 2017.

[39] Tomáš Krajník, Filip Majer, Lucie Halodová, and Tomáš Vintr. Navigation without localisation: reliable teach and repeat based on the convergence theorem. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1657–1664. IEEE, 2018.

[40] Filip Majer, Lucie Halodová, Tomáš Vintr, Martin Dlouhỳ, Lukáš Merenda, Jaime Pulido Fentanes, David Portugal, Micael Couceiro, and Tomáš Krajník. A versatile visual navigation system for autonomous vehicles. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 90–110. Springer, 2018.

Příloha

Obsah přiloženého CD

Jméno adresáře	Popis obsahu
maps	lokální uložené mapy
test-inside	testovací data pořízené uvnitř budovy
test-outside	testovací data pořízené ve venkovním prostředí

Tabulka 5: Obsah CD