VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LOKALIZACE ROBOTA V BYTĚ S VYUŽITÍM JEDNÉ KAMERY

MOBILE ROBOT LOCALIZATION IN AN APPARTMENT USING MONO-CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE TOMASZ KONDERLA
AUTHOR

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

BRNO 2016

Abstrakt

Táto práce se zabývá lokalizací robota pomocí analýzy obrazu z kamery umístěné na robotovi. Také se budu zabývat tvorbou 3D modelu prostředí, který je tvořen z datové sady fotografii pořízené kamerou. 3D model budu tvořit pomocí metody SFM. Lokalizovat kameru budu pomocí předem známého 3D modelu. V práci popíšu teorii zpracování obrazových dat potřebnou pro tento projekt. Nakonec zhodnotím výsledky úspěšnosti lokalizace.

Abstract

This bachelor is thesis deals with the localization of the robot by means of image analysis from the camera placed on the robot. I will also deal with the formation of a 3D model of the environment which is created of a data set of photos taken with the camera. The 3D model I will create using the method of SFM. I will locate the camera using the pre-set 3D model. In my bachelor is thesis I will further describe the theory of processing of the image data necessary for this project. Finally, I will evaluate the results of the success of the localization.

Klíčová slova

Lokalizace kamery, klíčové body, model prostředí, SURF metoda, RANSAC

Keywords

Camera localization,key points, environment model, RANSAC

Citace

KONDERLA, Tomasz. *Lokalizace robota v bytě s využitím jedné kamery.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Beran Vítězslav.

Lokalizace robota v bytě s využitím jedné kamery

Prohlášení

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

. . . . . .
Tomasz Konderla
18. května 2016

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Vítězslavu Beranovi, Ph.D. za cenné rady, připomínky a hlavně za trpělivost. Dále své manželce a dětem, že mi dali podporu a moc nezlobily. A v neposlední řadě bych chtěl poděkovat mým rodičům.

Obsah

Kapitola 1

Úvod

Robotika je velice mladé a rychle se rozvíjející odvětví informačních technologií, zabývá se mnohá problémy od autonomního řízení robota, až po jejich dizajn a výrobu. Táto práce se konkretně zabývá zpracováváním obrazových dat pro lokalizaci robota v místnosti. Bohužel je to velmi náročný proces. V dnešní době existuje několik způsobu jak tento problém vyřešit. Jedním ze způsobu je využití 3D modelu prostředí a lokalizace snímku z kamery. O některých způsobech pojednávám v kapitole 2.1 .
Táto práce se zabývá tvorbou 3D modelu vnitřního prostředí a následné lokalizace kamery v tomto prostředí na základě porovnání lokálních příznaku obrazu a vyhledání referencí z modelem. V rámci práce bylo vytvořeno několik datových sad, o kterých se pojednává v kapitole 4.2. Tyto datové sady byly využité k tvorbě 3D modelu vnitřního prostředí, o kterém se hovoří v této kapitole 4.2. Dále v této práci byly provedeny experimenty, které jsou popsané v kapitole 4.5. Celý návrh lokalizačního systému je popsán v kapitole 3.

Kapitola 2

Zpracovaní obrazu a 3D techniky

Počítačové vidění a lokalizační metody se stále rozvíjí, a proto je velmi mnoho pojmů a různých metod. Hodně často jsou tyto metody vykládané různě, podle toho jak je autor práce potřeboval. V této kapitole bych chtěl vysvětlit metody a pojmy, které jsou důležité pro pochopení této práce.

2.1 Běžně používané postupy a existující řešení

Klíčové body

Klíčové body, nebo také významné body, oba tyto možné jména vyjadřují totéž, můžeme chápat jako důležitá místa na fotografii. Tyto body můžou být například rohy, hrany, průsečíky přímek, skvrny. Určování, co bude za významný bod na fotografii, závisí jaký detektor jsme zvolili. Detektor vybíráme podle toho jakou úlohu budeme řešit. Tyto detektory nám najdou klíčové body a popíšou nám je do takzvaného deskriptoru. Důležitou vlastností všech těchto detektorů je stabilita nalezených klíčových bodů. To znamená, že bod, který je popsán danou metodou na jednom snímku, bude opakovaně nalezen i na ostatních snímcích. Tyto body by mělo být možné nalézt i po působení geometrických a foometrických změn. Z toho důvodu významných bodů můžeme hodnotit kvalitu. Táto kvalita úzce souvisí z daným detektorem, proto touto kvalitou popisujeme vlastností detektoru.
Tady je výčet vlastností:

1. Invariantnost vůči natočení, zkosení a jiným transformacím.
2. Robustnost náchylnost detekce v obraze šum, rozmazání ...
3. Výkonnost zda-li je detekce dostatečně rychlá..
4. Rozlišitelnost jednoznačné matematické vyjádření, které zajisti porovnatelnost v databází příznaků.

Tady uvedu par detektoru, které podporuje knihovna OpenCV. Toto jsou nejběžněji užívané detektory, každý má jiné vlastností [15]:

1. "FAST"– hledání FAST bodů. Vhodné pro tracking.

2. "SIFT"– hledání SIFT bodů.
3. "SURF"– hledání SURF bodů.
4. "ORB"– hledání bodů ORB, neboli Oriented Brief.
5. "GFTT"– hledání rohů v obraze, které splnují určité požadavky na kvalitu. Vhodné pro tracking.

Existují řešení

Jako jednu z prvních věci co bychom měli udělat při řešení problému je nastudovaní už vyřešených postupu, které nějakým způsobem souvisí z naším problémem. Tyto systémy nám můžou napovědět směr, kterým by jsme měli jít při vytváření našeho návrhu. V problematice lokalizace robota jsou dva významné typy *SLAM*[1]- simultánní lokalizace a mapovaní a druhy je *PTAM*[2] -Paralelní trakování a mapování. Oba systémy lze zařadit mezí systémy, které se zabývají mapováním prostředí a souběžnou lokalizaci. Pracují tak že, robot se pohybuje v neznámém prostředí a toto prostředí zkoumá a vytváří aktuální mapu prostředí. Díky tomu se může zorientovávat (lokalizovat) v neznámém prostředí. Bohužel, aby se robot přesně lokalizoval musí mít přesnou mapu. V procesu vytváření map v metodách *SLAM* a *PTAM* dochází k malým chybám(nepřesnostem) a tyto chyby se můžou akumulovat.

SLAM: v angl. Simultaneous localization and mapping. SLAM je to spíše koncept než nějaký algoritmus, protože se skládá z několika části, které si každý implementuje po svém a přizpůsobují svému hardwaru. Jak už z nazvu vyplývá SLAM je metoda, které pomoci kamer a senzoru vytváří mapu neznámého okolí a zároveň se v té mapě lokalizuje, postupně tu mapu zpřesňuje. Z toho vyplývá že slam se skládá z těchto části:

1. Snímaní okolí všemi senzory
2. Vytváření mapy
3. Lokalizace
4. Zpřesňovaní

Jak jsem výše uvedl SLAM, využívá kromě kamery i mnoho jiných senzoru. Tyto senzory můžou byt nějaké laserové dálkoměry, sonary 3D kamery. Protože mé zadaní jednoznačně určuje, že budu moct využit jen jednu kameru. Tato metoda proto není moc vhodná pro inspirací pro projekt, i když je ve světe hojně vyžívaná.

PTAM: v angl. Parallel tracking and mapping. Oproti metodě SLAM je metoda PTAM založená na tom že využívá jednu kameru (i když jak už to bývá v IT světe jsou už i modifikace , které využívají více kamer). Metoda PTAM rozděluje mapovaní a lokalizace kamery do dvou separátních častí. Často se využívá více-jádrový procesor, kde každé jádro obsluhuje jinou činnost. Ale jednu věc máji společnou a to je, že mapu vytvářejí postupně. U teto metody jíž jsem se mohl víc inspirovat, protože využívá jedné kamery, ale o proti ní jsem si zvolil že model prostředí již budu mít kompletně vytvořeny před lokalizaci. Protože u metody PTAM je vidět výhodnost rozložit metodu do více častí a paralelně je zpracovávat,

ale nastávají problémy, kolem paralelního zpracovaní (synchronizace dat, zabíraní zdrojů...) [4].

2.2 SURF

SURF (Speeded-Up Robust Features) je metoda, která dokáže popsat obrázek pomocí deskriptorů. Metoda SURF byla inspirovaná metodou SIFT, ale oproti ní je rychlejší. Jejím hlavním záměrem, jak už název napovídá, je výpočet rychlého a stabilního deskriptoru. Metoda SURF vynalezená proto, aby byla schopná pracovat v reálném čase.

Heledání klíčových bodu

Hledání klíčových bodu v metodě SURF se děje přímo pomocí determinantu Hessianovy matice. Rychlý výpočet determinantu je umožněn aplikací integrálního obrazu. Pomocí integrálního obrazu je možné získat údaj o intenzitě oblasti obrázku v konstantním čase s potřebou znát jen krajní body oblasti. Integrální obraz je struktura vybudovaná nad vstupním obrazem umožňující rychlé spočtení součtu hodnot uvnitř libovolné obdélníkové oblasti vstupního obrazu. Může být vyjádřen následujícím předpisem:

$$I_{\sum}(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j) \tag{2.1}$$

kde $I(i,j)$ je vstupní obraz a $I_{\sum}(x,y)$ představuje obraz integrální
Součet hodnot obdélníkového regionu uvnitř vstupního obrazu lze pak jednoduše vyčíslit dosazením do vztahu:

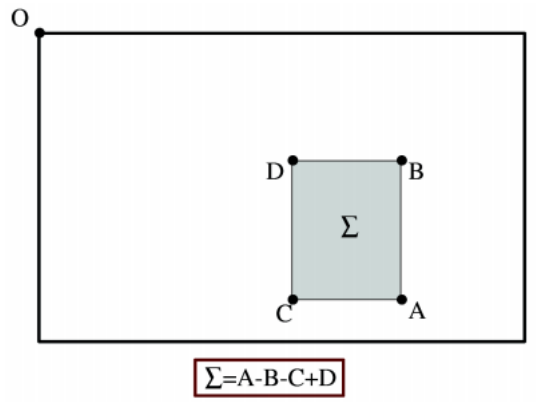
$$\sum = A - B - C + D \tag{2.2}$$

kde $\sum$ značí hledaný součet a A, B, C, D jsou hodnoty integrálního obrazu v daných souřadnicích.

Metoda SURF pro výpočet prvků Hessianovy matice využívá obdélníkové funkce. Tyto funkce aproximují druhou derivaci v diskrétní formě a jedná se tak o konvoluci obrazu s filtry. Je zřejmé, že implementace integrálního obrazu umožňuje velice rychlé spočtení odezvy na dané obdélníkové filtry. Hessianova matice bod x=(x,y) v obraze I je $H(x,o)$ v x s měřítkem o definovaná takovým předpisem:

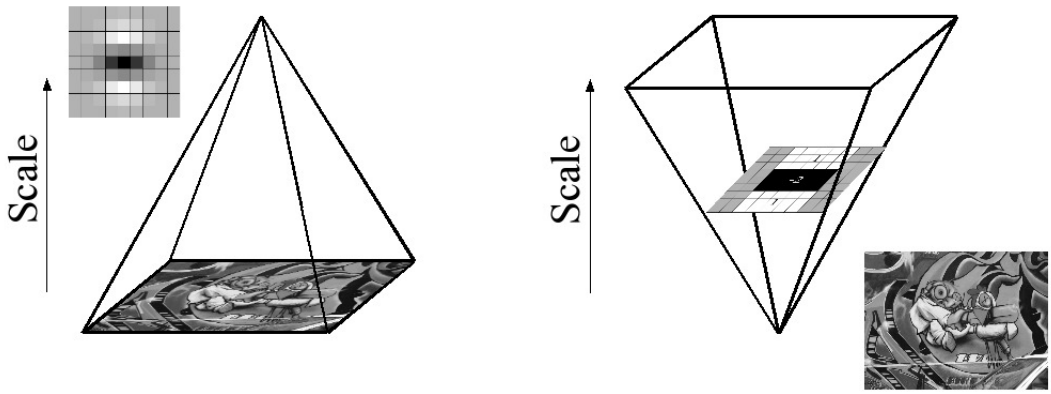
$$H(x,o) = \begin{bmatrix} L_{xx}(x,o) & L_{xy}(x,o) \\ L_{xy}(x,o) & L_{yy}(x,o) \end{bmatrix} \tag{2.3}$$

[1] http://www.openslam.org/
[2] http://www.robots.ox.ac.uk/ gk/publications/KleinMurray2007ISMAR.pdf



Obrázek 2.1: Výpočet součtu hodnot v regionu $\sum$ pomocí integrálního obrazu [7]

Klíčové body musejí vykazovat stabilní detekci i při změně měřítka obrazu. Proto je nutné takové body lokalizovat uvnitř měřítkové nezávislé reprezentace. V metodě SURF využívá teorie scale-space. Metoda konstruuje scale-space z jednotlivých obrazů vzniklých filtraci, díky implementaci integrálního obrazu nedochází k nárůstu výpočetní složitosti při zvětšení filtračního jádra. Z tohoto důvodu se větších měřítek ve scale-space dosahuje konvolucí vstupního obrazu s filtračním jádrem zvyšujících se rozměrů. Odpadá tedy nutnost kaskádovitého generování scale-space, čímž je dosaženo ještě většího zrychlení oproti metodě SIFT.



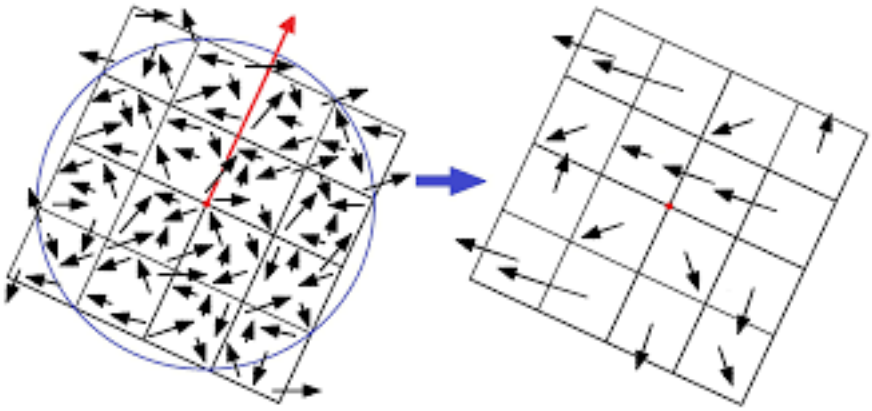
Obrázek 2.2: Namísto redukce velikosti obrazu, jak lze vidět na levém obrázku, lze díky integrálnímu obrazu zvyšovat měřítko filtru (vpravo) [6].

Jak již bylo zmíněno výše, SURF využívá k detekci významných bodů determinant Hessianovy matice. Z výsledných odezev na filtraci je tedy postupně spočtena hodnota tohoto determinantu, čímž je získán obraz $H_{Det}(x, y, o)$. Právě tyto obrazy představují finální měřítkové nezávislou reprezentaci vstupního obrazu, ve které jsou následně lokalizovány významné body.

Popis klíčových bodů

Metoda SURF se každému klíčovému bodu snaží přidělit jeho dominantní orientací, proto výsledný deskriptor bude invariantní vůči rotací. Děla to tak, že zkoumá odezvu na tzv. Haarovu vlnku v kruhovém okolí daného bodu. Haarova vlnka je opět aproximovaná pomocí obdélníkových filtrů ve směru osy x a y, což umožňuje zapojení integrálního obrazu do procesu výpočtu odezvy. Výsledky konvoluce obrazu s Haarovu vlnkou se označuje jako d_x a d_y

Přímo dělaní deskriptoru, se děla v několika krocích. Jako první krok se kolem klíčového bodu vytvoří čtvercová oblast, ta se následovně rozdělí rovnoměrné 4x4 podoblasti. V každé podoblasti je určeno pět pravidelně rozmístěných bodu. Pro tyto body je následovně vypočtená odezva na Haarovu vlnku ve směru osy x a y. Odezvy jsou značeny d_x a d_y. Následně pro každou podoblast jsou spočítány hodnoty $\sum d_x$ a $\sum d_y$ a aby byla zajištěna odolnost na změnu osvětlení musíme spočítat $\sum |d_x|$ a $\sum |d_y|$. Tyto 4 hodnoty pak tvoří vektor $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ pro každou podoblast. Tyto hodnoty jsou následovně sečteny pro všechny podoblastí a vytvoří první sadu pro deskriptor (vektor příznaku). Sečtení všech 16 podoblastí dostaneme vektor o délce 64 [7].



Obrázek 2.3: Odezvy na Haarovu vlnku v okolí významného bodu (vlevo) podle dominantní orientace (červená šipka) a výsledný deskriptor metody SURF (vpravo) [7].

2.3 Porovnávaní klíčových bodů

Nalezení klíčových bodů na fotografii by nám bylo k ničemu, kdyby jsme ty body nemohli najít na jiném snímku. Každá metoda nám daný klíčový bod nějak popíše do deskriptoru, tyto deskriptory musíme umět nějak porovnat. K tomu nám slouží pár algoritmu. Metoda SURF má navržený deskriptor tak, aby porovnání klíčových bodů mohla sloužit eukleidovská vzdálenost. Čim jsou si body podobné, tím je táto vzdálenost menší. Tady vám popíšu dvě základní metody:

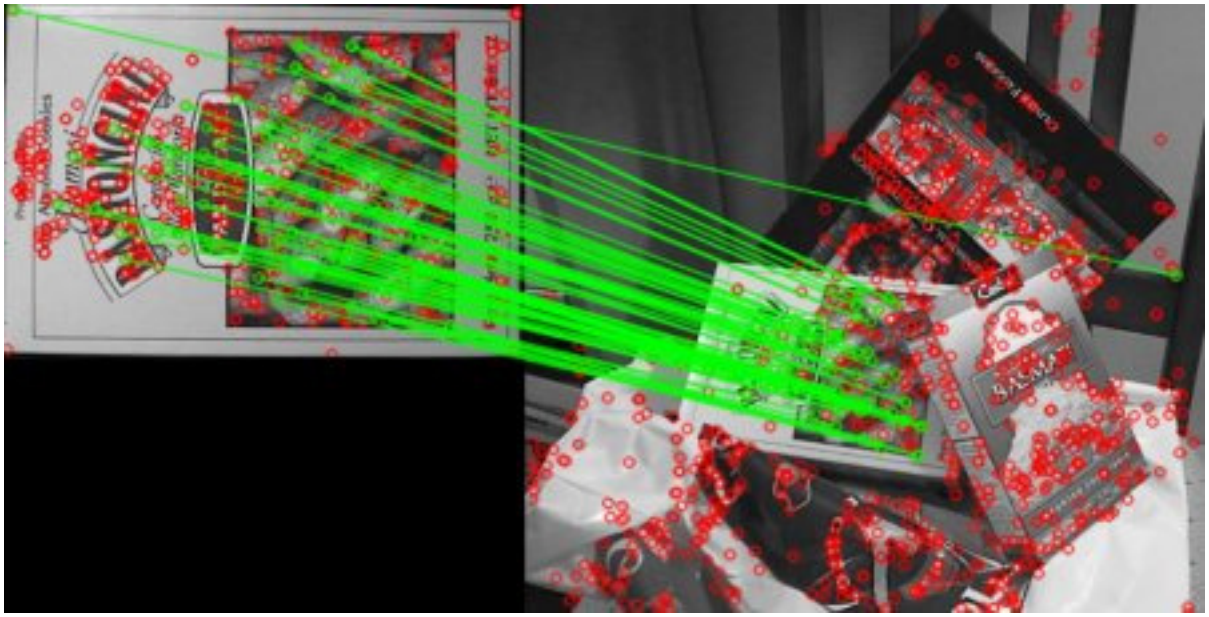
Brutální silou

Je to velice triviální algoritmus pro porovnávaní klíčových bodu. V podstatě jde o to, že se vezme první deskriptor ze vstupního setu. Pak se ten deskriptor porovnává ze všemi

deskriptory z druhého setu a vypočítavá se jejích vzdálenost. A nejbližší z nich je vrácena [14]. Tyto výsledky se zapisují do výstupního setu, kde jsou odkazy na ty dva klíčové body, které jsou si nejvíce podobné. Tuto techniku jde upravit tak, že ji udáme prahovou hodnotu. Do výstupního setu se budou zapisovat pouze výsledky do dané prahové hodnoty.

Nejbližší soused

Vyhledávaní podle nejbližšího souseda je už víc propracovaný algoritmus. Je založen na stromové struktuře. Nejprve ze sady klíčových bodů je vytvořená stromová struktura, následně táto stromová struktura je využita při hledání dvojic nejvíce podobných klíčových bodů. Opět je možno využit prahové hodnoty pro vyhledávaní dvojic podobných bodů. U knihovny OpenCV pro vyhledávaní podle nejbližšího souseda je využívána struktura FLANN.

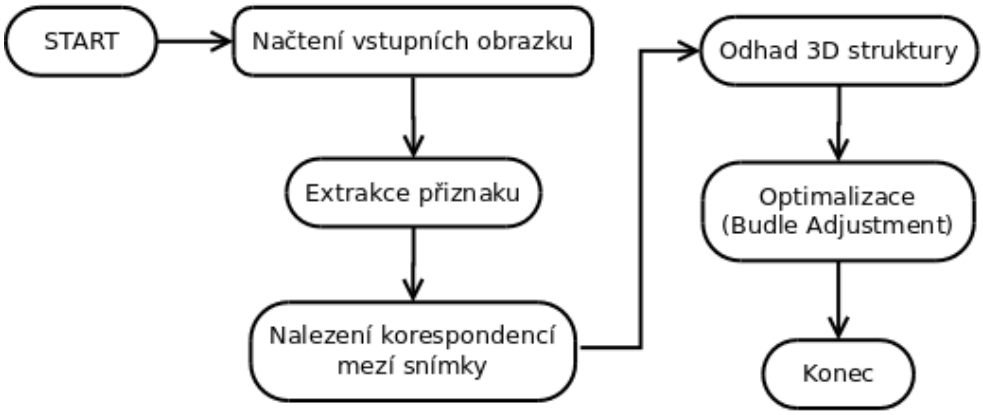


Obrázek 2.4: Naleze schody klíčových bodu [14]

2.4 Structure from motion

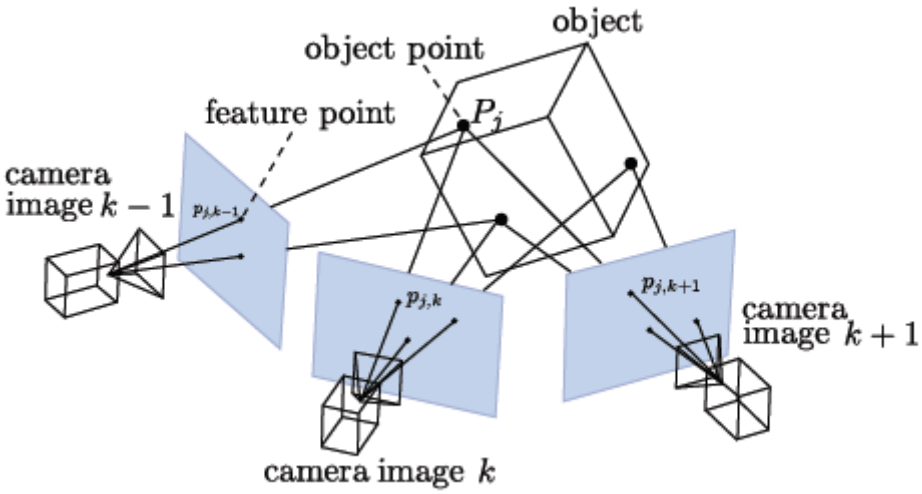
Structure from motion je velice významný pojem v počítačovém vidění. Řada programů je využívá k tvorbě 3D modelů.

Structure from motion je řada zobrazovacích technik, které se zabývají odhadováním 3D struktur ze sady fotografií, a které mohou být ve spojení se signály o lokalitě pohybu. Tyto principy jsou uplatňovány v robotice a zejména v počítačovém vidění [20]. Životní cyklus SFM na obrázku 2.5.



Obrázek 2.5: Životní cyklus SFM [2]

Structure from motion pracuje tak, že načte vstupní fotografie, pak určí klíčové body. Potom nalézá korespondence mezí snímky, to provádí pomocí porovnání viz více 2.2. Potom pomoci triangulace rekonstruuje 3d scénu viz více 2.7. V práci pomoci SFM vytváříme 3D model. Průběh jak pracuje SFM při hledání klíčových bodů, si můžeme prohlédnout na obrázku 2.6.



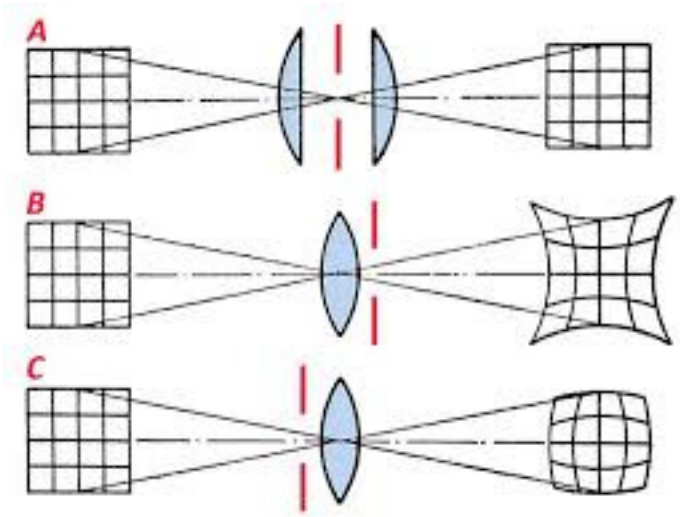
Obrázek 2.6: Nalézaní klíčových bodu [16]

2.5 Lokalizace

V této kapitole se budu věnovat teorii kolem lokalizace kamery a co vše k tomu budeme potřebovat. Představíme si pár technik, které využíváme při lokalizování kamery. Tyto techniky jsou nezbytné pro lokalizaci kamery.

Kalibrace kamery

Kalibrace kamery nám slouží k dvěma věcem. První věc je, že nám kalibrace kamery určí zkreslení kamery. Kromě toho, že nám kalibrace kamery určí zkreslení, můžeme také určit vztah mezi přirozenými jednotkami fotoaparátu (pixely) a skutečným světem (například milimetrech). Ve 20 století se ve velkém začaly používat dírkové kamery a zjistilo se, že na nich začalo vznikat zkreslení [13]. Tyto kamery obsahují nějakou soustavu čoček, na kterých vzniká zkreslení. Máme dva druhy zkreslení na čočkách kamery. Radiální zkreslení je způsobeno přítomnosti čoček (viz obazek 2.7) a je nutno ho kompenzovat. Tangenciální zkreslení způsobeno nepřesnou centrací čoček, v dnešní době toto zkreslení výrobci kamer už skoro odstranili, je velice nepatrné [9].



Obrázek 2.7: Typy Radiálního zkreslení [9]

Naštěstí toto zkreslení je konstantní a existuje několik metod, které nám ho určí a pomohou jeho vliv odstranit.
Tyto chyby se popisuji pětici [13]:

$$Distortion_{coefficients} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3) \tag{2.4}$$

Matice kamery

K popisu kamery potřebujeme vytvořit kalibrační matici. Kalibrační matice se skládá ze dvou matic. Tyto matice se nazývají vnější parametry a vnitřní parametry kamery.

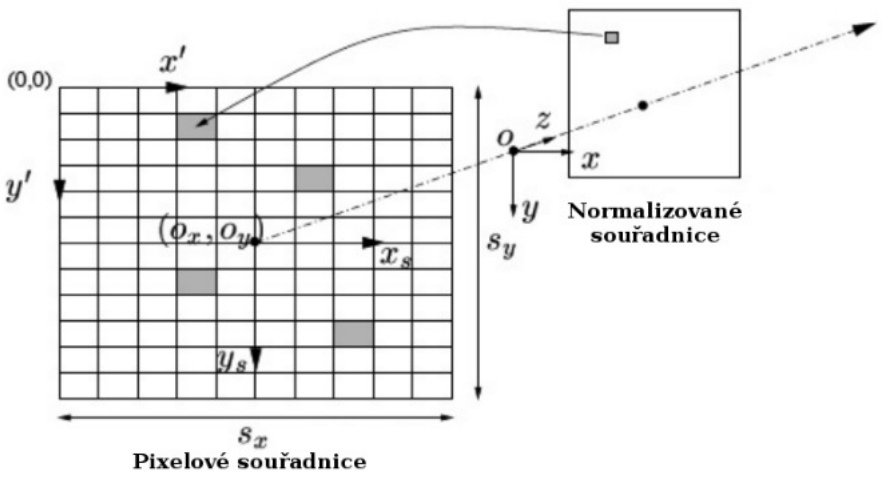
Vnější parametry kamery závisí na orientaci kamery v prostoru vzhledem k světovým souřadnicím (O_w) – matice R a t [12].

$$cam = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{pmatrix} = (R|t) \tag{2.5}$$

Vnitřní parametry kamery jsou uloženy v kalibrační matici K a popisují vlastnosti kamery nezávisle na vnějších parametrech. Tyto vnitřní parametry se pro danou kameru nemění (viz popis výše) , pravě z toho hlediska nám stačí je získat pouze jednou oproti vnějšímu parametru, které se z každým snímkem mění. Zde vidíme matici K která nám reprezentuje vnitřní parametry (jinak taky intristické parametry).

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

Na matici fx a fy jsou ohniskové vzdálenosti, cx a cy jsou souřadnice základního bodu.



Obrázek 2.8: Transformace z obrazových souřadnic do pixelu [10]

2.6 RANSAC

RANSAC(RANdom SAmple Consensus) je iterační metoda pro odhad parametrů matematického modelu ze sady pozorovaných dat, která obsahuje odlehlé hodnoty(outliers). Čím víc iterací provedeme, tím je metoda přesnější. Z metodou jsou spjaty dva pojmy outliers a inlier [11].

outlier - je to vzorek dat, který nepatří do modelu.
inlier - je to vzorek dat, který patří do modelu.

RANSAC se provádí pomocí následujících kroků:

1. Náhodný výběr podmnožiny množiny dat.
2. Montáž modelu vybraných podskupin.
3. Stanovení počtu odlehlých hodnot.
4. Opakování kroků 1-3 pro předepsaný počet iterací.

Pro více informací o této metodě, lze získat zde [11], odkud byli čerpané informace.

2.7 Rekonstrukce 3D

Protože ve většině záležitostí kolem počítačového vidění potřebujeme nějakým způsobem zrekonstruovat 3D scénu, máme několik způsobu jak na to jít. V teto kapitole vysvětlím pár důležitých pojmů a pár metod jak zrekonstruovat 3D scénu. V praxi se pák vetšinou používá kombinace vice metod pro dosažení požadovaného cíle.

Triangulace

Triangulace je zjištění 3D polohy na základě promítaní paprsku. Máme dva základní typy triangulace *aktivní* a *pasivní*.

Aktivní

Aktivní technika triangulace fotogrammetrické rekonstrukci snímaného objektu. Což znamená že objekt je při 3D rekonstrukci nasvícen nějakým externím zdrojem světla a zároveň je snímán kamerou(neboli CCD snímačem). Zdroj světla spolu se snímačem a osvětleným bodem na zkoumaném objektu tvoří triangulační trojúhelník viz 2.11. Spojnici světelný zdroj - snímač nazýváme triangulační bází . Na straně zdroje je úhel svíraný s triangulační bází neměnný, kdežto na straně snímače je úhel určen proměnnou pozicí vysvíceného bodu kamery. Z velikosti tohoto úhlu lze určit hloubku objektu.

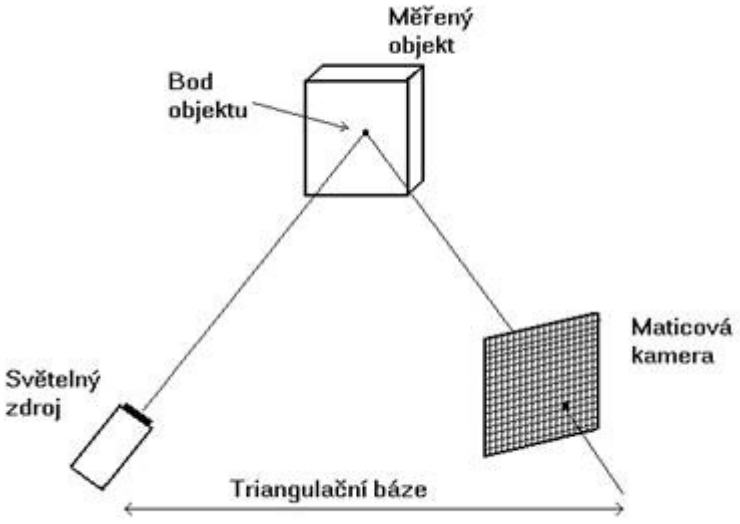
Tato aktivní metoda se děli na 2D triangulaci a 3D triangulací. Rozdíl mezí těmito dvěma metodami je takový, že u 2D triangulace máme lokální zdroj, který je postupně přemisťující po objektu zatím u 3D triangulace nasvítíme celý objekt nějakým světelným vzorem.

Pasivní

Pasivní metoda znamená, že není uvažováno o geometrickém uspořádání osvětlení. Metoda používá tyto základní koncepty:

1. Více kamer se samo-kalibrací.
2. Více kamer se známou orientací.
3. Jedna kamera v různých polohách se samo-kalibrací.

Tato metoda se využívá pro dynamické systémy, často se v ni využívá více kamer se znalosti relativních poloh nebo samo-kalibrujících se metod. Pro statické scény se využívá jedna kamera, která získá snímky ze dvou a více různých pohledů. Proto tato metoda se využívá při tvoření modelu. U technik se samo-kalibrací nemusí být dopředu známa poloha kamery (kamer), ale přímo ze snímků (více pohledů jedné kamery nebo více kamer) je určeno relativní umístění kamery (kamer) vzhledem k měřenému objektu (scéně) či vzájemná



Obrázek 2.9: Triangulace [8]

poloha kamer. Také do této metody patří stereo vidění. Táto metoda je založena na principu naších očí, což znamená spolupráce dvou kamer [8].

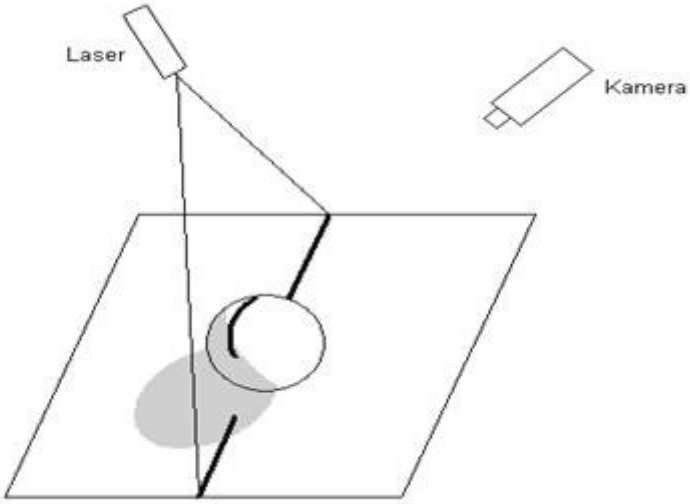
Projekční matice

Projekční matice se využívá při výše popsané triangulace. Pravě ona je schopná převést stereo korespondenci do výsledného modelu mračna bodu. Projekční matice P je matice, která zobrazuje 3D bod X do jemu odpovídajícího 2D bodu x [5].

Epipolární geometrie

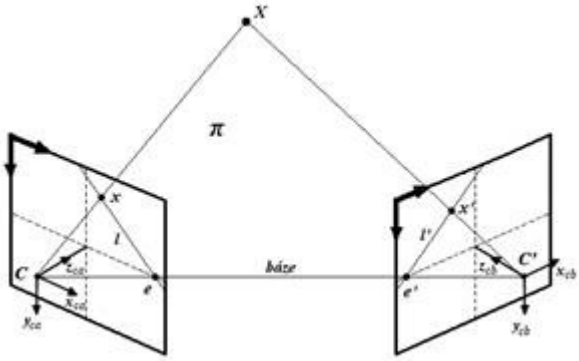
Rekonstrukce 3D scény ze dvou a více rovinných průmětů je klasická úloha počítačového vidění. Elementárním problémem je určení pozice bodu v prostoru pomocí dvojice kamer. Průmětem 3D scény do obrazových rovin těchto kamer vzniká dvojice obrazů nazývaných stereo pár. Teoretickým základem pro určení vzájemného vztahu stereo páru je epipolární geometrie, která je nezávislá na struktuře scény a závisí pouze na vnitřních parametrech kamer a jejich relativní pozici. Pomocí epipolární geometrie lze bod z prvního obrazu mapovat na přímku v druhém obrazu, což značně zjednodušuje nalezení korespondujících bodů (bod z prvního snímku není nutné hledat na celé ploše druhého snímku, ale jen na přímce nacházející se na této ploše).

V zásadě existují dva přístupy k rekonstrukci prostorové scény, tedy k získání fundamentální matice. První možností je určení projekčních matic kamer. Tento přístup ovšem vyžaduje znalost vnitřních parametrů kamer, jež lze nabýt kalibračními technikami. Druhým přístupem k rekonstrukci 3D scény je využití projektivity (homografie) snímků. Detekcí vzájemně si odpovídajících bodů určíme projektivitu snímků, aniž bychom znali vnitřní parametry kamery [3].



Obrázek 2.10: 2D triangulace [8]

V obou případech je důležité zvolit model kamery, který co nejpřesněji vystihuje kameru reálnou.



Obrázek 2.11: Epipolární geometrie [3]

V prostoru se nachází bod X, jenž je snímán dvojicí kamer, jejichž matice jsou P a$P^{'}$. Bod X se promítá na průmětny kamer jako bod x a x bod tak, že platí $x = PX$ a $x^{'} = PX$. Spojnice mezi středy kamer C a $C^{'}$ se nazývá báze. Průsečík báze s rovinou průmětny tvoří epipól e a $e^{'}$ (epipól nemusí nutně ležet v zobrazovací výseči kamery). Přímka procházející body e a x, popř. $e^{'}$ a $x^{'}$ je označována jako epipolární přímka a značí se l, popř. $l^{'}$. Středy kamer, epipóly a body X, x, $x^{'}$ jsou koplanární, to znamená, že leží v jedné epipolární rovině π viz 2.11 [3].

Fundamentální matice

Epipulární geometrii muže matematicky popsat. Jeho nejdůležitější výstup je mapování $x \rightarrow l^{'}$. Což můžeme matematicky vyjádřit pomocí fundamentální matici F.
Fundamentální matice je čtvercová matice 3 x3 a má 7 stupňů volnosti [3]. Tady máme tvar

matice:

$$F = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{23} & f_{33} \end{pmatrix} \tag{2.7}$$

2.8 Model prostředí

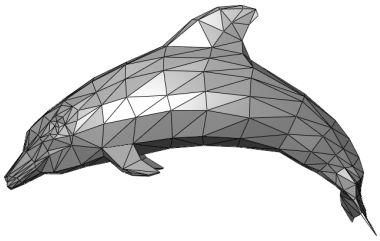
V teto kapitole se budeme věnovat modelu prostředí. Především se budeme věnovat 3D modelu prostředí, které jsou nezbytně důležité pro tuto prací. Tyto modely uchovávají informace o zrekonstruované 3D scéně prostředí. Proto je velmi důležité jakou reprezentaci dat si zvolíme, protože nám to může velice pomoct pro budoucí zpracovaní dat. V této kapitole popíšu jaké existují druhy modelů a jak se dělají.

Druhy modelu

Tady popíšu pár reprezentaci modelů. Některé z těchto modelu můžou po vykreslení dokonce velice dobře rozeznat rekonstruovanou scénu, ale některé jsou pouze pro zpracovaní důležitých údajů. Čerpal jsem z těchto zdrojů [19], [17], [20].

Mračno bodů
Mračno bodů jak už název napovídá je množina bodů v nějakém souřadnicovém systému. V trojrozměrném souřadnicovém systému, tyto body jsou obvykle definovány X, Y a Z, a často jsou určeny pro reprezentaci vnějšího povrchu předmětu. Mračna bodů jsou často vytvářena pomocí 3D skeneru. Tyto skenery jsou založeny většinou na metodě triangulace viz 2.7. Tyto přístroje měří veliký počet bodu na povrchu objektu či scény a údaje o těchto bodech zapisují do souboru. Tyto body mohou byt přímo vykreslené , ale u většiny vizuálních aplikaci jsou nepoužitelné. Proto existuje spousta metod jak převést mračno body například na 3D síť, která je mnohém vhodnější pro 3D aplikace.

3D síť
3D síť(Polygon mesh) je velice jednoduché vyjádření 3D rekonstrukce pomocí geometrické reprezentace. 3D síť je reprezentovaná pomocí kolekce vrcholu, hran a plochami, které dohromady tvoří polygony. Tyto polygony potom popisují cely objekt či scénu. Polygony vetšinou jsou trojúhelníky, čtyřúhelníky, nebo jiné jednoduché konvexní polygony. Regulární síť je speciální typ 3D sítě, který je tvořený pouze jedním druhem polygonu. Většinou to bývají trojúhelníkové polygony. Tento model se hojně využívá při modelovaní.



Obrázek 2.12: Regulární 3D síť tvořena trojúhelníky [21]

Oktalový strom

Táto struktura je známa názvem octree. Je to velice efektivní struktura pro ukládaní scény. Kořen stromu obsahuje celou scénu. Ta se postupně děli na 8 stejně velkých podčásti (voxelu). Tyto části pak mají za otce kořen. Podčásti můžeme dále dělit a vytvářet syny. Když ve scéně narazíme na objekt, uložíme ho do listu, který je potomkem daného uzlu. Je to klasická struktura založena na stromové architektuře. Dělení stromu probíhá do té doby dokud není cela scéna popsaná pouze objekty nebo, dokud není splněná podmínka. Podmínky můžou být různé, ale nejčastější je to výška stromu, počet uzlu.

Kapitola 3

Návrh systemu

V této kapitole bych se věnoval konkretně už mému lokalizátoru a to především jeho návrhu. Před samotným programováním je důležité si stanovit co konkrétně chci udělat a jakým způsobem k tomu mi slouží návrh systému. Aby lokalizace byla úspěšná musím práci rozdělit do několika dílčích části, které popíšu v této kapitole.

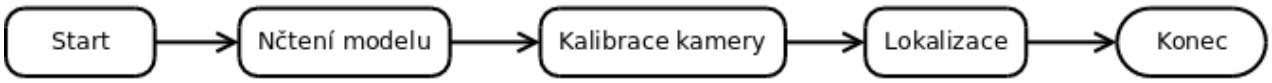
3.1 Upřesnění zadaní

Cílem této práce je lokalizovat robota pomocí jedné kamery. Z toho nám vyplývá, že můžeme lokalizovat ne robota, ale kameru, která je připevněná na něm. Proto se budeme zabývat lokalizaci kamery. Protože máme jenom jednu kameru nejlepší varianta bude ji lokalizovat v předem známém prostředí. Předem známým prostředím se myslí nějaký 3D model prostředí. Co budeme potřebovat mít v tom 3D modelu prostředí, jakým způsobem bude model reprezentovat z pohledu datových struktur. Na tyto otázky najdeme odpovědi, až budeme mít návrh lokalizačního systému v předem známém prostředí.
Proto bude nejprve vhodné navrhnout lokalizační system a na základě požadavku, které nám vyjdou z návrhu lokalizačního systému navrhnout 3D model.
Celý systém jde rozdělit do několika kroku:

1. Vytvořit 3D model prostředí
2. Lokalizovat kameru na robotovi

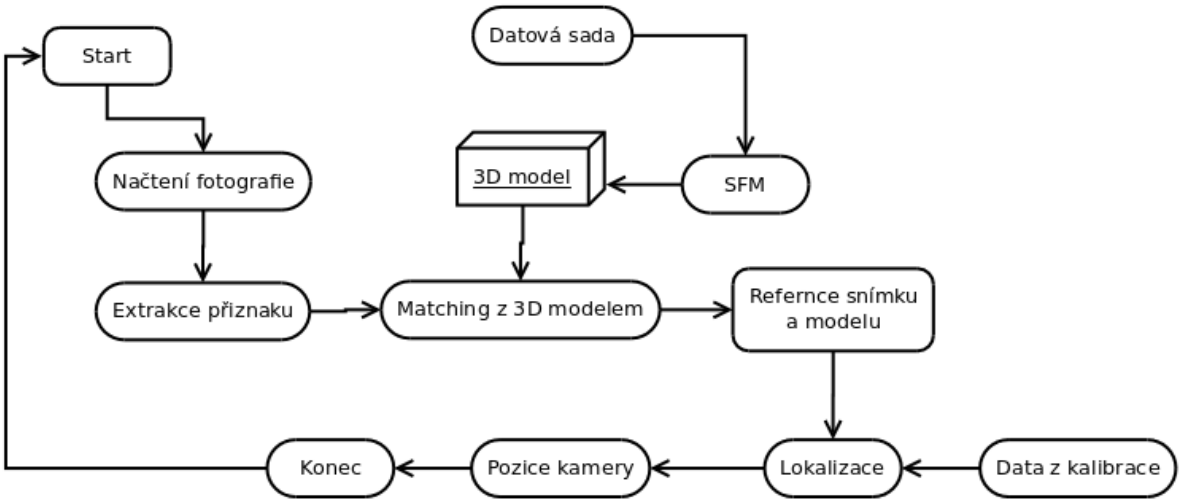
3.2 Návrh celého systému

V této kapitole popíšu návrh celého systému. Na obrázku 3.1 můžeme vidět jak vypadá návrh celého systému pro lokalizaci jednoho snímku. Při lokalizaci robota málokdy bychom potřebovali lokalizovat pouze jeden snímek proto musíme tento návrh upravit.



Obrázek 3.1: Návrh celého systému

Při lokalizaci robota budu brát video z jeho kamery. Robot se bude hýbat a bude průběžně chtít znát svoji lokalizaci. Proto budu muset v nějakém časovém intervalu, který si zvolím, brát snímky z videa od pohybujícího se robota a tyto snímky lokalizovat. Model ani kalibrace kamery se nebude měnit, budu muset pouze lokalizovat nové snímky. Na obrázku 3.2 vidíme jednotlivé kroky lokalizace. Model a kalibrační data si vytvořím jen jednou před lokalizací, proto tuto část systému je přípravná. Lokalizace jednotlivých snímků probíhá v reálném čase a stále do kola jak můžeme vidět na snímku 3.2.



Obrázek 3.2: Postupné kroky celého systému

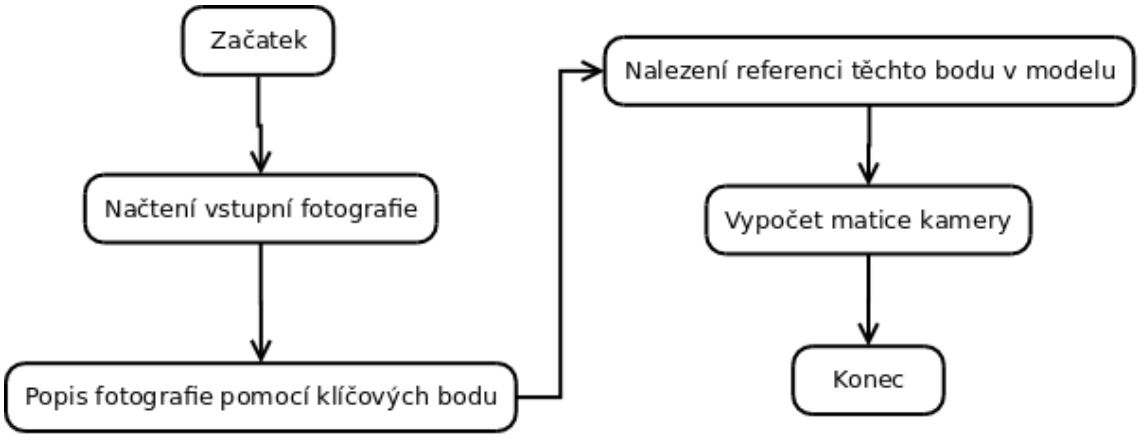
3.3 Návrh lokalizátoru

Tedka si představíme návrh lokalizátoru, pomocí metod popsaných 2.5. Lokalizátor by měl byt stabilní a robustní. Lokalizátor navrhnu tak aby mi lokalizoval vždy jeden snímek. Pokud nastane potřeba lokalizovat vice snímku po sobě kroky lokalizace se pouze zopakují. Lokalizace se děla v těchto krocích:

1. Popis fotografie pomocí klíčových bodu
2. Nalezení referenci těchto bodu v modelu
3. Výpočet matice kamery

Pro lokalizaci kamery ze snímku, musíme provést pár kroku. Jako první krok, který musíme provést, je nalezení klíčových bodů na snímku. Co to jsou klíčové body a jak se nalézají je popsáno v kapitole 2.1. Tyto body pak musíme nalézt v 3D modelu. Tento krok můžeme provést takovým způsobem, že z datové sady kterou jsme vytvořili, 3D modul, najdeme nejpodobnější snímek. A následně na podobností těchto snímku nalézt reference v 3D modelu.Tato technika je nejednodušší, ale také velice zdlouhavá. Musíme porovnat všechny snímky v datové sadě z naši fotografie. Účinnější by bylo hledat snímky z využitím vyhledávacího stromu. Posledním krokem je vypočtení matice kamery. Proto, aby jsme mohli vypočítat maticí kamery, musíme znát reference 2D do 3D a vnitřní parametry

kamery. Tyto vnitřní parametry kamery získáme pomocí kalibrace kamery. Protože výpočet matice kamery záleží na mnoha vzorcích a ne všechny tam patří, nadbytečné vzorky můžeme vyloučit pomoci metody RASNAC.



Obrázek 3.3: Lokalizace

3.4 Návrh modelu

Musím vytvořit 3D model. Model tvořím pomocí 3D rekonstrukce z datových sad. Datová sada je určitý počet snímku nějakého prostoru (optimálně se tento počet pohybuje kolem 90 snímku místnosti), které se musí z častí překrývat. Každý bod v modelu musí mít jednoznačnou pozicí v prostoru x,y,z. Dale musím umět tyto body najít i na jiných fotografiích, proto musí model nést i nějaký popis těchto klíčových bodů.
Pro tento účel postačí i model skládající se z mračná bodů. Jaké informace budu potřebovat od našeho modelu jsem se dozvěděl z návrhu lokalizátoru. Tyto údaje musí být někde uložené v nějaké datové struktuře. Lokalizační systém bude očekávat, že nálezené reference z modelem. Proto model bude muset mít tyto udaje:

1. Souřadnice 3D bodu.
2. Klíčové body fotografií
3. Deskriptory

Tyto souřadnice 3D bodů by měly být spjaty minimálně z dvěma deskriptory, které je popisuji. Jinak nebude možné nalézt reference mezi fotografii a modelem (2D bodem a 3D bodem). Výrobu modelu z série snímku je nejlepší SFM metodou, táto metoda je popsaná v kapitole 2.5. Různé nástroje pro výrobu modelu popisuji zde 4.2.

3.5 Možná vylepšení

Protože robot bude průběžné posílat snímky, můžeme předpokládat, že některé částí snímku budou totožné. Proto by bylo vhodné do lokalizace dát systém pro sledovaní pohybu kamery, takzvaný tracking. Také bychom mohli při lokalizaci si vypomáhat z udají interním od robota např kolik ujel, otáčení....

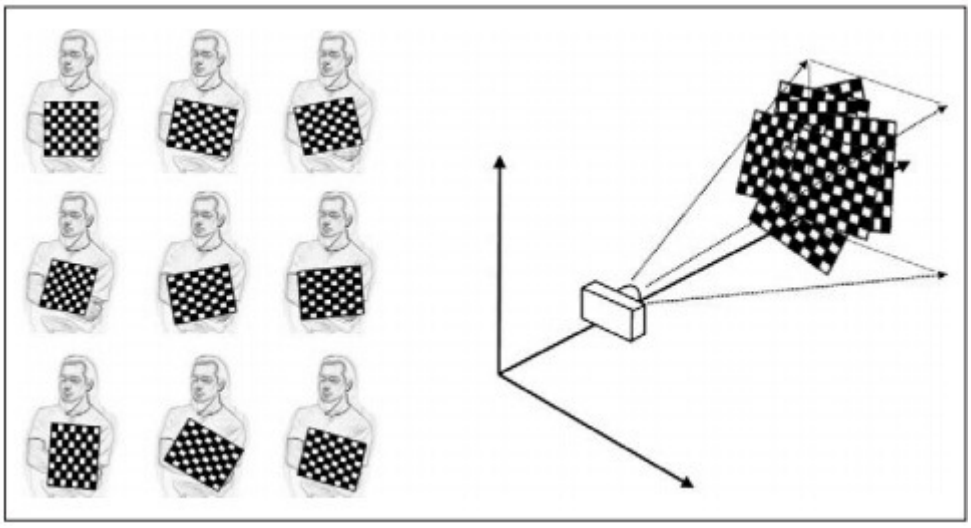
Kapitola 4

Realizace lokalizátoru

V této kapitole popíšu jak návrh blíže podaný v kapitole 3 převedu do funkčního stavu. Celý projekt byl napsaný v c++ a využíval jsem knihovnu OpenCV ve verzi 2.4.9. Nejprve popíšu kalibraci jak jsem ji dělal pro tento projekt a jaký program jsem využíval. Jako jeden z důležitých kroku je model. V kapitole o tvorbě modelu popíšu a shrnu zkušenosti, jak se tvoří datová sada fotografií. Dále přiblížím nástroje pro tvorbu modelu, u nástrojů shrnu jejich vlastnosti. U lokalizace projdu jednotlivé kroky systému, co dělají, co potřebují, jak fungují. Dále popíšu implementací celého systému. Pak se budu věnovat experimentům, na kterých jsem testoval účinnost lokalizačního systému.

4.1 Kalibrace

Protože navrženy lokalizační systém potřebuje jako jeden ze vstupu vnitřní parametry kamery. Víc o kalibraci najdete v odstavci 2.5. Protože kalibrační data potřebujeme zjistit jen jednou a to před lokalizaci.



Obrázek 4.1: Návod jak postupovat při kalibraci kamery [22]

Tyto data načtu na začátku lokalizace ze souboru, do kterého zapíšu udaje o provedené kalibraci kamery. Ke kalibraci využiji program, který poskytuje sama knihovna OpenCV.

U tohoto programu vkládáme jako vstupní parametr jméno soubor, do kterého nastavíme všechny potřebné údaje ke kalibraci.

Tady máme výčet některých důležité údajů, které se nacházejí ve vstupní souboru ke kalibraci:

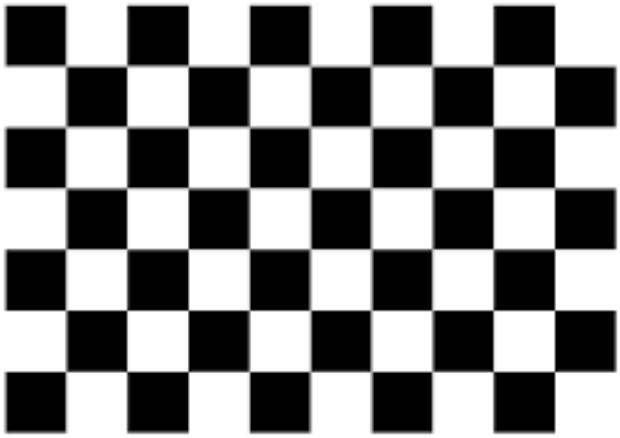
1. Z kterou kameru budeme kalibrovat.
2. Počet kalibračních snímku k vyhodnocení.
3. Kalibrační vzor.
4. Vstupní soubor, ve kterém se nacházejí kalibrační fotografie.
5. Výstupní soubor pro zápis výsledku kalibrace.

Táto aplikace má mnoho nastavení. Můžeme ji kalibrovat jak přímo konkretně zapojenou kameru, tak může tyto kalibrační údaje vypočítat, už ze snímků, které máme uloženy dříve. Pokuď máme všechny údaje vyplněné, začne samotná kalibrace. Program začne na snímkách vyhledávat udaný kalibrační vzor. Aby kalibrace byla úspěšná musíme měnit uhel snímání kamery. Jak máme měnit úhel kamery vidíme na obrázku 4.1.

Kalibraci můžeme provést i ze dvou snímku ale nedoporučuje se to taková kalibrace muže byt nepřesná. Počet kalibračních snímku by měl být 7 a více. Opencv doporučuje tyto kalibrační vzory:

1. Klasická černá-bílá šachovnice
2. Symetrický kruh vzoru
3. Asymetrické kruh vzoru

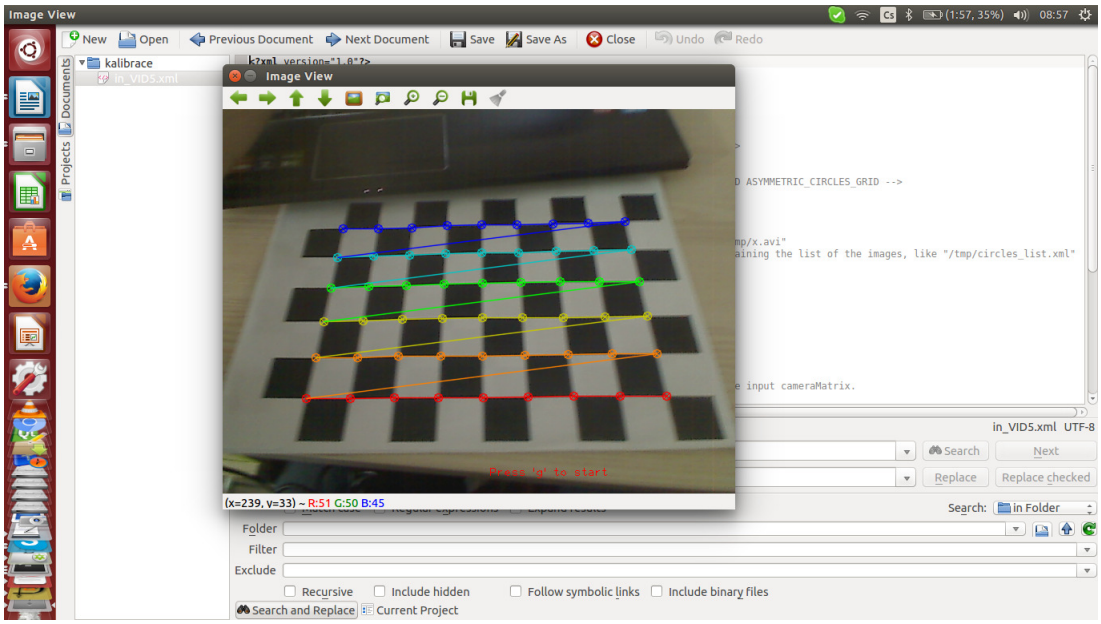
Já jsem použil vzor šachovnice. tento vzor můžete vidět na obrázku 4.2.



Obrázek 4.2: Kalibrační vzor šachovnice

Aplikace nám vypočítá vnitřní parametry kamery a matice zkreslení. Dále nám vypočte průměrnou projekční chybu všechny tyto udaje nám zapíše do vstupního souboru.

Na obrázku 4.3 mužem vidět jak kalibrují kameru a jak už aplikace nalezla vzor šachovnice.



Obrázek 4.3: Ukázka kalibrace v praxi

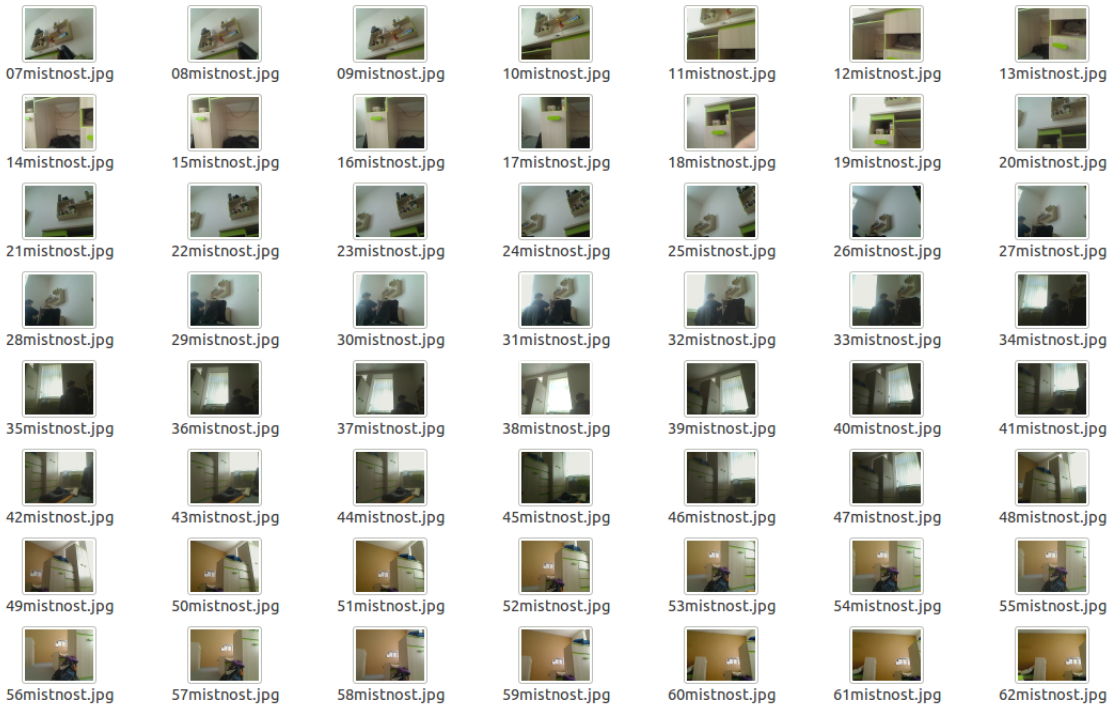
4.2 Model

Model je velice důležitou součástí lokalizace jak můžeme vidět při návrhu. Budu zde popisovat tvorbu modelu, která vychází ze zkušeností co jsem nabral při práci na projektu. Model v téhle práci jsem zvolil mračno bodu. Konkretně výstup z modelu je soubor o jménu data.yaml, kde jsou zápasený deskriptory snímku, klíčové body a hlavně se tam nachází 3D souřadnice bodu. Model tvořím pomocí programu Exploring SfM With OpenCV v kterém jsem si upravil aby pracoval pomocí metody SURF a zapsal mi potřebné údaje do souboru.

Tvorba datové sady

V průběhu prací jsem vytvořil několik datových sad fotografii, pro tvorbu modelu. Taky jsem stáhl dvě datové sady z internetu, které byly speciálně určená pro tvoření modelu pomocí SFM. Nyní bych vám chtěl shrnout poznatky, které jsem nabral v průběhu prací na datových sadách. Můj první poznatek je takový, že snímky v datové sadě by se měli překrývat minimálně v 25% fotky následující. Můžeme použit takzvané překrytí, což je vyfotografování jednoho místa více krát. Vice o teto problematice bude věnováno v kapitole experimenty 4.5. Dále pokrytí prostoru v datové sadě by mělo být rovnoměrné, tím lze dosáhnout lepší kvality výsledného modelu. Také není k zahození fotit totéž místo z odlišné vzdáleností čím uděláme rozdíl mezí pixelmi na fotografii a mm v reálném světe.

Bylo vytvořeno 22 datových sad o velikosti snímku od 5 do 90. Datové sady o malém počtu snímku byly určeny k ručnímu laděni programu a k vychytávaní much systému.
Pro tvoření datových sad jsem vytvořil speciální program, kterým pomocí kalibrované web kamery tvoří snímky. Datovou sadu, kterou jsem vytvořil můžete vidět na obrázku 4.4.



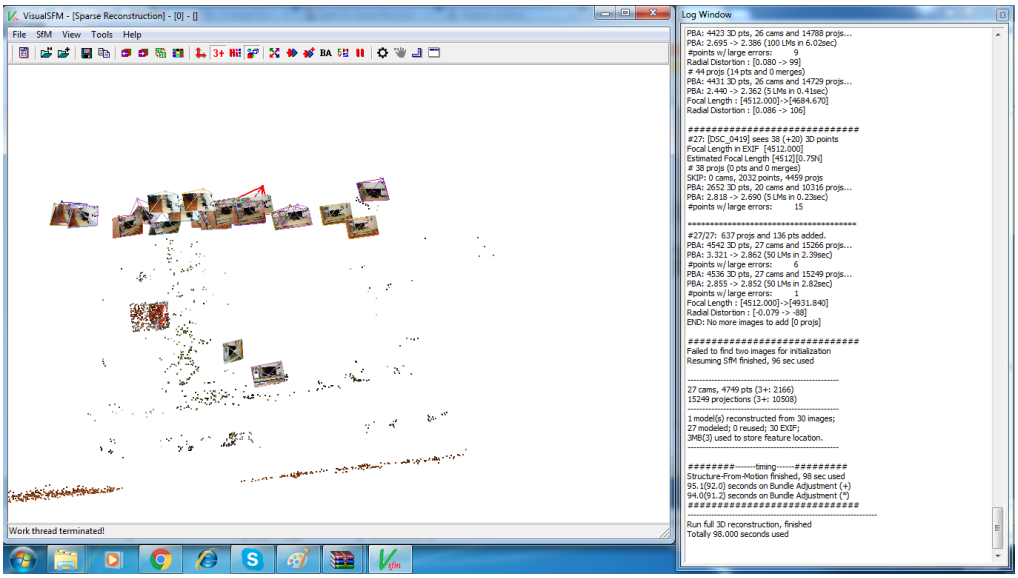
Obrázek 4.4: Ukázka datové sady fotografii pro model

Nastroje

VisualSFM je grafický desktopová aplikace pro vytváření 3D rekonstrukci s použitím struktury z pohybu(SFM) vice o teto problematice pojednávám v kapitole 2.5. Tento nástroj v sebe integruje různé projekty, které jsou potřebné k chodu projektu nebo zrychlují výpočet modelu. Tento nástroj využívá metodu SIFT a více jádrový paralelismus pro detekci, párování a nastavení svazku. VisualSFM umožňují několik výstupu: bud máme klasický model mračna bodu, nebo takzvanou hustou rekonstrukci. Hustá rekonstrukce je převedení modelu mračna bodu na nějakou, která jde lip graficky zobrazit. VisualSFM, aby mohl zobrazit hustou rekonstrukcí a musí mít v sebe integrované tyto další nastroje: execution of Yasutaka Furukawa's PMVS/CMVS, CMP-MVS, SURE, MVE [23].

VisualSFM je multiplatformní aplikace , ale bohužel instalace pro Linux je velíce složitá. A některá vylepšeni jedou jen na určitým hardwaru.
Tento nastroj má velice příjemné a přehledné GUI na tvorbu modelu. Také velice jednoduché načítaní vstupních fotografii. Bohužel jsem tento nastroj nemohl využít při mé prací, protože je to uzavřená aplikace. Má svojí upravenou metodu na detekcí klíčových bodu, která nejde měnit a taky svůj výstupní format dat. Pro tyto dvě závažné nedostatky jsem nemohl tento nástroj využit u mé práce.

Exploring SfM With OpenCV
v průběhu mého projekt jsem nalezl prací, která pomocí SFM technologie děla model mračná bodu. Tato práce je napsaná pomocí knihovny OpenCV , kterou využívá i já v mé prací. Protože kód je veřejně dostupny tak jsem mohl do něho zasahovat což je přesně to co jsem hledal. O tomhle tom projektu je vice napsáno zde [1]. Protože jsem mohl zasahovat do



Obrázek 4.5: Mračno bodu vytvořeno nástrojem VisualSFM

Obrázek 4.6: Hustá rekonstrukce vytvořeno nástrojem VisualSFM [18]

kódu tak jsem si u tohohle projektu napsal funkci, která mi uloží do souboru data, které potřebují k mému projektu. Dale tahle práce mi umožnila abych si jí přeprogramoval na můj zvoleny detektor klíčových bodu. Zvolil jsem metodu SURF.

4.3 Lokalizátor

Program lokalizátor je navrženy tak, aby budoucí uživatel si ho mohl přizpůsobit podle svojí potřeby. Program je implementován pomocí modulů, které mají své účely, o tom více je pojednáno v podkapitole 4.4. Program lokalizátor funguje v několika krocích.
Tady je výčet těchto kroku:

1. Načtení vstupních parametrů
2. Načtení dat z kalibrace a modelů
3. Načtení snímků nebo videa (popis klíčkovými body)
4. Nalezení referenci z modelů
5. Výpočet matice kamery
6. opakovat od 3 bodu pokud existují další snímky

Všem krokům se budeme postupně věnovat. Popíšeme jak fungují a co ke své prací potřebují.

Načtení údaje kalibrace a modelů

Kalibrace kamery a model prostředí jsme vytvořili před samotnou lokalizaci. Tyto údaje jsou předávaný pomocí souborů ve formátu *YAML*[1] nebo taky můžeme *XML*[2]. Knihovna OpenCV podporuje ukládání a načítaní datových struktur, toto zajišťuje třída *FileStorage*. Konfigurační soubor pro model se jmenuje *data.yaml* a nese informace o cestě k datové sadě, seznam snímků, klíčové body a jejich deskriptory, 3D souřadnice bodů. Konfigurační soubor pro kalibraci nese tyto informace maticí kamery a Distortion Coefficients ,což je matice 5 zkreslení viz více [13]. Konfigurační soubor nese více údajů ale, ostatní už nevyužívám. Tento soubor nese jméno *out _camera_data.yaml*.
Načtení kalibračních údajů v prací řeším třemi různými způsoby:

1. Můžu je zadat manuálně jako vstupní parametr
2. Pomocí konfiguračního souboru
3. Odhad ze snímku nejméně přesné (nedoporučují pouze jako krajní řešení)

[1] https://cs.wikipedia.org/wiki/YAML
[2] https://cs.wikipedia.org/wiki/Extensible_Markup_Language

Načtení snímku

Program jako první parametr příjme cestu k snímku. Pak pomocí knihovny OpenCV se načte a zpracuje. Knihovna OpenCV pro rychle zpracovaní obrázku nabízí tyto třídy *FeatureDetector*, *DescriptorExtractor*. Tyto třídy mi zajišťují nalezení a popsaní klíčových bodu. Další výhodou použití těchto tříd je rychlá výměna detektoru dle potřeby, aniž bych musel složitě zasahovat do zdrojového kódu. O detektorech se více dočteme v kapitole 2.1

Pro projekt byl vybrán detektor SURF, protože je to velmi stabilní, robustní a rychlý detektor. Tento detektor jsem zvolil i u programu co tvoří 3D model. Oba tyto programy musejí používat stejné detektory. Pokud by jsme nepoužili stejný detektor pro oba je velice pravděpodobně že nám nenalezne stejné klíčové body a body do deskriptoru popíše jinak viz více kapitola o klíčových bodech 2.1.

Nalezení referenci

Abych mohl vypočítat matici kamery musím najít souřadnice 2D body a k nim odpovídající 3D body na modelu. To se děla tak , že máme 3D body a k ním mám minimálně 2 deskriptory, které ho popisují, také mám skupinu 2D bodů a k ním deskriptory. Pak už jen porovnávám. K vyhledávaní stejných bodu využívám knihovnu *FLANN*[3] , která je implementovaná v knihovně OpenCV. Knihovna *FLANN* pro vyhledávaní využívá metodu nejbližšího souseda viz více 2.3. Dále obsahuje spoustu algoritmu pro optimalizaci hledání.

Výpočet matice kamery

Lokalizaci provádím pomocí výpočtu matici kamery. K výpočtu využívám funkcíi *solvePnPRansac* táto funkce mi vrátí translační a rotační vektor. Funkce *solvePnPRansac* je založena na PnP algoritmu, který využívá ransacu, proto je velmi odolná proti outlier viz vice v kapitole 2.6. Z translačního a rotačního vektoru pak už jen stačí vytvořit matici kamery. Rotační vektor musíme upravit pomocí funkce *Rodrigues*. Tato funkce nám transformuje vektor rotační na maticí R 3x3. Čtvrtý sloupec matice dotvoříme pomocí doplnění translačního vektoru. Výsledná matice je ve tvaru viz. vzorec 2.5.

4.4 Popis implementace

V této podkapitole budu popisovat veškeré programy, které se podílejí na veškeré funkčnosti celého projektu. Budu popisovat jak se spouští jednotlivé programy a u lokalizátoru popíšu i vnitřní implementaci modulu.

Kalibrační program

Kalibrační aplikace jak jsem už psal je zabudovaná již v knihovně OpenCV. Táto aplikace se spouští jen jedním parametrem a to je jméno souboru, v kterém jsou všechny další nastavení kalibrace.

ukázka spuštění: *./camera_calibration in_VID5.xml*

Program pro tvorbu modelu

[3] http://www.cs.ubc.ca/research/flann/

Tento program jsem popisoval v této kapitole 4.2. Spouští se také pouze s jedním parametrem a to je cesta k datové sadě pro tvorbu modelu.

ukázka spouštění: *./ExploringSfMExec /home/tom/pokus_opencv/chotba4*

Program pro lokalizací
Tento program je na-implementován modulárně. Teď popíši všechny moduly a k čemu slouží:

1. nacist – modul, který slouží k načítaní kalibrace kamery.
2. nacitani – modul sloužící k načtení modelu.
3. vypocet – výpočet referenci mezi modelem a fotografii.
4. main – řídí a konečný výpočet matice kamery.

Protože načtení kalibraci můžu dělat různými způsoby (vstupní soubor, zadat jako parametry, odhad), proto program má 4 vstupní parametry:

1. a - f je ohnisková vzdálenost v pixelech.
2. b - cx je obraz hlavním bodem x souřadnic v pixelech.
3. c - cy je obraz hlavním bodem y souřadnic v pixelech.
4. f - je cesta ke snímku, který chceme lokalizovat.

ukázka spouštění: *./moje_BP -f templeR0030.png -a 800 -b 400 -c 225*
druhý způsob spouštění: *./moje_BP -f templeR0030.png*

4.5 Experimenty

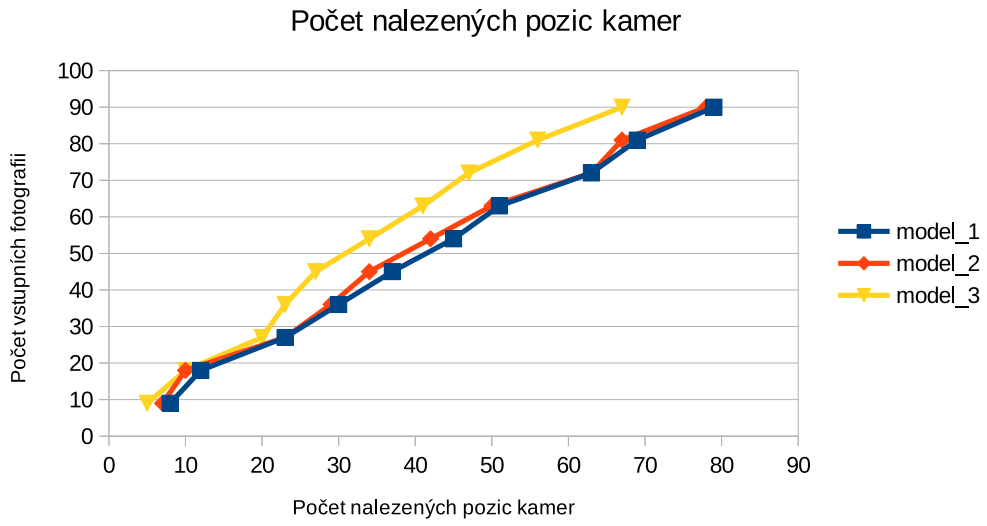
V této podkapitole chci popsat experimenty, které jsem prováděl v rámci projektu. Všechny experimenty jsem prováděl na počítači Lenovo Ideapad G500, který má tyto parametry:
Proces core i5 dva jádra o maximální frekvenci 3.20 GHz.
Pamět rám DD3 o velikostí 8 Gb.

Budu se věnovat dvěma oblastem. První oblast je testování kvality modelu a ta druhá testování rychlosti, kvality lokalizačního systému.

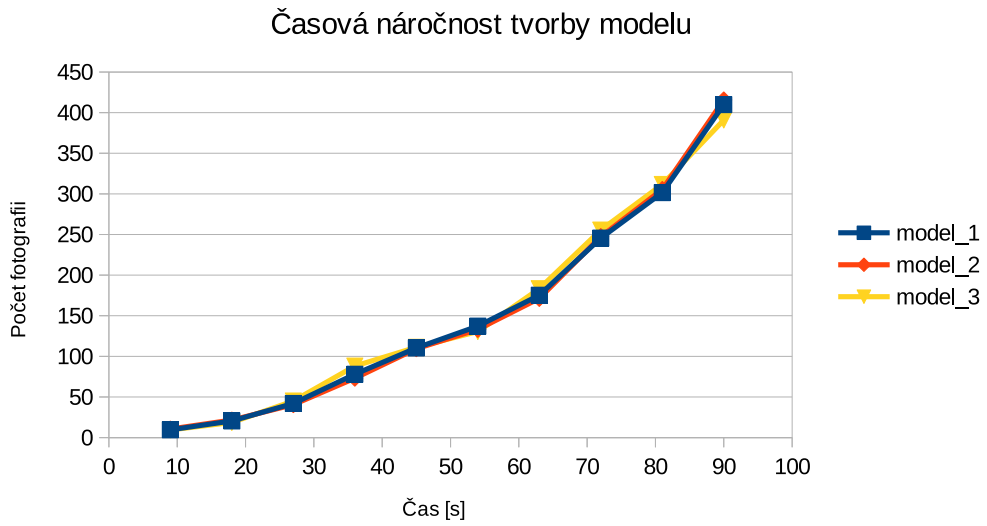
Model

U modelu jsem testoval jeho kvalitu. Navrhl jsem test, který ověří kvalitu modelu podle počtu nalezených kamer. Počet nalezených kamer, by měl odpovídat počtu vstupních fotografií. Testoval jsem 3 modely, první měl kvalitní fotografie v datové sadě a byla focena místnost z dostatkem subjektů v ni (byl to pokoj). Druhý model byla ta samá místnost, ale fotografie byly nekvalitní, jako třetí jsem testoval model místnosti z nedostatkem subjektů

(byla to chodba). Testoval jsem to tím způsobem, že jsem postupně navyšoval počet fotografii v datové sadě.



Obrázek 4.7: Počet nalezených pozicí kamer



Obrázek 4.8: Časová náročnost tvorby modelu

Z výsledku testů nám vyniká, že na kvalitě fotografií záleží, ale především záleží na počtu věci nacházejících se v místnosti. Dále vidíme na grafu 4.8, že časová náročnost tvorby modelu je podobná. Celkem mě tvorba modelu pomocí metody SURF moc neuspokojila, počet nalezených kamer byl docela malí. Dále bylo taký velice náročné vytvořit vhodné datové sady. Pokuď program nemohl dlouho najít žádné schody, nebo narazil na nějakou nekvalitní fotografii spadne. Tato chyba se objevovala především při tvorbě 3 modelu (chodbě).

Lokalizátor

Tady jsem navrhl dva různé testy, první byl takový, že jsem otestoval jak lokalizuje fotografii, která pocházela z datové sady, ale nebyla využitá při tvorbě modelu. Tato fotografie byla znehodnocená.
Druhý test byl takový že jsem simuloval činnost robota. Natočil jsem video, které simulovalo jízdu robota. Z jízdy jsem vybral fotografie, které jsem lokalizoval.

Kvalita Lokalizace
Test probíhal takovým způsobem, postupně jsem komprimoval obrázky a snažil jsem se je lokalizovat.

Výsledky testu:

komprese	**počet pokusu**	**Počet lokalizaci**	**%**
90%	10	10	100%
75%	10	10	100%
50%	10	8	80%
25%	10	5	50%

Simulace robota
Dalším experimentem na lokalizátoru bylo simulace činnosti robota. Robota jsem simuloval tak, že jsem umístil web kameru na kolečkové křeslo, kameru jsem nastavil tak jak by byla na robotovi a udělal video. Z videa jsem vybral několik snímku a tyto snímky jsem lokalizoval pomoci lokalizátoru.

Zde jsou výsledky:

model	**počet snímku**	**Počet lokalizaci**	**%**
model 1	60	55	92%
model 2	40	30	75%

Model 1 byl modelem pokoje z dostatkem vizuálních objektů, model 2 byla chodba, kde nebylo tolik vizuálních objektů. Z testu nám vyplývá, že čím více je těchto vizuálních podnětů, tím lépe pro lokalizaci. Dale z něho vyplývá, že lokalizace dopadla docela úspěšné i na neuspokojivých modelech.

Kapitola 5

Závěr

Cílem práce bylo navrhnout a naimplementovat lokalizační systém pro robota který, se bude pohybovat v bytě. Tento lokalizační system má pouze pracovat z jednou kamerou. Proto jsem zvolil lokalizací v předem známém prostředí, kde uděláme předem model prostředí.
Jako první úkol co jsem udělal, bylo seznámení se z problémem a nastudování teorie kolem lokalizace robota. Nastudování teorie je věčný koloběh, ale v teto oblastí jsem docela pokročil. V teoretické častí bakalářské práce jsem věnoval částem zpracovaní obrazových dat a rekonstrukce 3D scény, které jsou potřebné pro tuto prací.
V další kapitole jsem se věnoval návrhu lokalizátoru a celého systému.
Dále jsem se jíž věnoval realizací lokalizátoru. Popsal jsem jednotlivé častí lokalizace, tvořeni 3D modelu a program pro kalibraci kamery. Pro tvoření 3D modelu jsem popsal dva nastroje, které by šli využit pro 3D rekonstrukci. U obou těchto nástrojů jsem uvedl jejích výhody a nevýhody.
Projekt pro svůj chod musí na cílové platformě mít nainstalovanou knihovnu OpenCV. Knihovna OpenCV má velice velkou programovou základnu a jde nainstalovat na hodně platforem například i na Raspberry PI. Takže je velice vhodná pro tento projekt.
Lokalizátor je navrhnut pro robota, ale nikdy reálně, nebyl vyzkoušený na žádném robotovi. Pouze se v experimentech simulovala činnost robota což je trošičku škoda. Lokalizátor je modulárně navrhnut tak, že každý modul zodpovídá za určitou část. Důvodem toho je že, každý robot pracuje na jiné platformě. Proto může kdokoliv upravit jakoukoliv část programu aniž by to mělo nějaký fatální vliv na jinou část programu. Také jeden z důvodu byl, že některé roboty máji přistup i k jiným senzorům, než jenom ke kameře, takže kdokoliv může pozměnit program tak, aby fungoval i z využitím těchto senzoru.
Pro cely projekt jsem zvolil SURF metodu. Tato metoda měla byt podle autoru robustní a lepší po stránce výkonnosti oproti metodě SIFT. Z experimentu mi to tak jednoznačně nevyšlo. Velmi povrchně jsem jen tak vyzkoušel metodu SIFT a předpokládám, že tato metoda by byla vhodnější při 3D rekonstrukcí.
Lokalizátor by šel v praxi využít, ale jeho výsledky nejsou natolik spolehlivé. Příčinu v tom má funkce pro výpočet kamery. Tato funkce funguje tak že zkouší všechny možné umístěny kamery, ale ne vždy se vydá dobrým směrem. Pro budoucí rozvoj aplikace bych zvážil možná něco jiného. Projekt jako vstup dostává několik parametrů více, o těchto vstupních parametrech nalezneme 4.4. Projekt vypíše na standardní vystup maticí kamery. Tento projekt můžeme využít tak že jak budeme mít konkretního robota, přizpůsobíme daný projekt jeho hardwaru.

Literatura

[1] Baggio, D. L.: Mastering OpenCV with Practical Computer Vision Projects. 2003, [Online; navštíveno 20. 4. 2016].
URL https://www.packtpub.com/sites/default/files/9781849517829_Chapter_04.pdf

[2] Baillard, C.; Zisserman, A.: Automatic Reconstruction of Piecewise Planar Models from Multiple Views. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1999, s. 559–565.

[3] Beneda, M.: Homografie a epipolární geometrie. *Trilobit*, ročník 2, 2010, ISSN 1804-1795.

[4] David Murray, G. K.: Parallel Tracking and Mapping for Small AR Workspaces. 2007.

[5] HARLTEY, R. a. A. Z.: *Multiple View Geometry in Com-puter Vision.* Cambridge University Press, druhé vydání, 2003, ISBN 0521540518.

[6] Herbert Baya, A. E.: Speeded-Up Robust Features (SURF). *elektrorevue*, ročník 110, 2008: str. 346–359.

[7] Hrúz, M.: SIFT, SURF, MSER. 2015, [Online; navštíveno 16. 4. 2016].
URL http://www.kky.zcu.cz/uploads/courses/mpv/04/materialy04.pdf

[8] Kalová Ilona, H. K.: Optické metody měření 3D objektů. *elektrorevue*, ročník 23, 2005, ISSN 1213-1539.

[9] Kotek, K.: Kalibrační proces ve 3D. [Online; navštíveno 22. 4. 2016].
URL http://www.mti.tul.cz/files/obr15_fcc_kotek.pdf

[10] Kotek, K.: Image Formation and Camera Models. 2007, [Online; navštíveno 25. 4. 2016].
URL http://inst.eecs.berkeley.edu/~ee225b/sp07/lectures/lec12.pdf

[11] Martin A. Fischler, R. C. B.: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *ACM New York, NY, USA*, ročník 24, 1981: s. 381–395.

[12] Milan Sonka, V. H.; Boyle, R.: *mage Processing, Analysis and Machine Vision.* Brooks/Cole Publishing Company, druhé vydání, 1999.

[13] OpenCV: Camera calibration With OpenCV. 2011, [Online; navštíveno 22. 4. 2016].
URL http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

[14] OpenCV: Feature Matching. 2011, [Online; navštíveno 20. 4. 2016].
URL http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[15] OpenCV: FeatureDetector::create. 2011, [Online; navštíveno 11. 5. 2016].
URL http://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_feature_detectors.html

[16] OpenMVG: SFM. 2013, [Online; navštíveno 6. 5. 2016].
URL http://openmvg.readthedocs.io/en/latest/openMVG/sfm/sfm

[17] Robert F. Tobler, S. M.: A Mesh Data Structure for Rendering and Subdivision. 2006.

[18] SammysHP: Photogrammetrie und OpenStreetMap. 2013, [Online; navštíveno 12. 5. 2016].
URL http://www.sammyshp.de/betablog/post/16

[19] Szymon Rusinkiewicz, M. L.: QSplat: a multiresolution point rendering system for large meshes. *Co. New York, NY, USA*, 2000: s. 343–352.

[20] Vahalík, T.: *REKONSTRUKCE 3D MODELU PROSTŘEDÍ A LOKALIZACE KAMERY*. Diplomová práce, Vysoké učení technické v Brně, Brno, 2014.

[21] Wikipedia: Polygon mesh. [Online; navštíveno 28. 4. 2016].
URL https://en.wikipedia.org/wiki/Polygon_mesh

[22] WikiStart: Stereovision reconstruction system. 2008, [Online; navštíveno 6. 5. 2016].
URL http://apps.man.poznan.pl/trac/stereovision

[23] Wu, C.: VisualSFM : A Visual Structure from Motion System. 2011, [Online; navštíveno 15. 4. 2016].
URL http://ccwu.me/vsfm/

Přílohy

Seznam příloh

Příloha A

Obsah CD

Přiložené CD obsahuje všechny zdrojové kódy, všechny aplikace potřebné k projektu, manuál, plakát, video a technickou zprávu.

Struktura CD:

1. Kalibrace - kalibrační program.
2. Technická zprava - soubor PDF technické zprávy.
3. Code-master - Exploring SfM With OpenCV.
4. Moje_BP - Zdrojové kódy a manuál.
5. Plakat a video.