

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Robot Poštár

Diplomová práca

Bratislava, 2014

Bc. Marína Madová

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Kód práce:

Robot Poštár

Diplomová práca

Študijný program : Aplikovaná informatika

Študijný odbor: 2511 APLIKOVANÁ INFORMATIKA

Školiace pracovisko: KATEDRA APLIKOVANEJ INFORMATIKY

Školiteľ: Mgr. Pavel Petrovič, PhD.

Bratislava, 2014

Bc. Marína Madová

Zadanie záverečnej práce



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Marína Madová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Robot poštár

Cieľ: Cieľom je navrhnúť a realizovať systém pre mobilného robota, ktorý sa dokáže pohybovať po budove, komunikovať s okoloidúcimi a preberať požiadavky na doručovanie pošty z jedného miesta na druhé. Využije sa pritom platforma robota, ktorý bol zostrojený na súťaž RoboTour.

Literatúra: 1. U. Nehmzow: Mobile Robotics: A Practical Introduction, Springer, 2000.
2. G. Dudek, M. Jenkin: Computational Principles of Mobile Robotics, Cambridge University Press, 2000.

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky

Dátum zadania: 17.10.2011

Dátum schválenia: 26.10.2011

doc. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

študent

vedúci práce

Čestné prehlásenie

Čestne prehlasujem, že som diplomovú prácu vypracovala samostatne s použitím uvedenej literatúry.

Pod'akovanie

Ďakujem svojmu školiteľovi Mgr. Pavlovi Petrovičovi, PhD za trpezlivosť a vysokú mieru odbornej pomoci pri vypracovávaní tejto práce.

Abstrakt

Madová, Marína: Robot Poštár. Diplomová práca. Univerzita Komenského v Bratislave. Katedra aplikovanej informatiky Fakulty matematiky, fyziky a informatiky. Vedúci diplomovej práce Mgr. Pavel Petrovič, PhD. Bratislava 2014. Počet strán: 53.

Cieľom záverečnej práce bolo navrhnúť systém pre mobilného robota, ktorý bude na orientáciu a navigáciu používať iba vstup z kamery, ktorá je jeho súčasťou. Úlohou robota bude rozvoz pošty po pavilóne I Fakulty matematiky, fyziky a informatiky.

Počas riešenia danej problematiky sme pri práci využili hlavne algoritmy na spracovanie obrazu v reálnom čase, čo umožnilo robotovi úspešne lokalizovať svoju polohu a orientovať sa v priestore pavilónu, algoritmy umelej inteligencie pri výpočte trasy robota a znalosti z oblasti vývoja webových aplikácií pri tvorbe webového rozhrania, pomocou ktorého robot prijíma požiadavky. Ako testovacieho robota pre použili už skonštruovanú platformu zo súťaže RoboTour [4].

V konečnom dôsledku sa nám podarilo položiť kvalitné základy budúcich systémov pre robotov orientovaných čisto vstupom z kamery s využitím vyspelých algoritmov.

Kľúčové slová

autonómny robot, SURF, rozpoznávanie cesty pomocou kamery

Abstract

Madová, Marína: Robot Poštár. Diploma thesis. Comenius University in Bratislava. Department of applied informatics, Faculty of mathematics, physics and informatics. Diploma thesis supervisor Mgr. Pavel Petrovič, PhD. Bratislava 2014. Page count: 53.

The goal of this thesis was to design a system for mobile robot which gains its orientation and navigation only from its build-in camera video input. The task given to the robot will be mail delivery in the premises of pavilion I of Faculty of mathematics, physics and informatics.

During our research we used mostly algorithms for real-time image recognition, which gave our robot the ability to localize itself and to have an orientation inside the pavilion, algorithms of the artificial intelligence while computing the path of the robot and our knowledge from web application development while developing the web interface from which our robot gains its assignments. For testing we used already build robot platform from RoboTour [4].

In conclusion, we managed to lay the foundations of future systems for robots using for localization and orientation only a camera while using advanced algorithms.

Keywords

autonomous robot, SURF, camera based path recognition

Obsah

1 Úvod.....	1
2 Prehľad problematiky	3
2.1 Navigačný systém pomocou detekcie cesty a GPS.....	3
2.2 Navigačný systém pomocou vodiacej čiary.....	5
2.3 Orientácia robota v exteriéri pomocou orientačných bodov	7
2.4 Algoritmy počítačového videnia pre orientáciu robota.....	8
2.4.1 SURF (Speeded Up Robust Features).....	8
2.4.2 Kalibrácia kamery	13
3 Špecifikácia.....	17
3.1 Formulácia cieľov práce.....	17
3.2 Cieľ projektu robot Poštár.....	19
3.2.1 Popis funkcionality projektu robot Poštár	19
3.2.2 Perspektíva projektu robot Poštár.....	20
4 Návrh riešenia	21
4.1 Robot smelý zajko.....	21
4.1.1 Mechanika	21
4.1.2 Systém robota	21
4.2 Návrh komunikácie robota s používateľom.....	22
4.3 Návrh lokalizácie robota v interiéri.....	23
4.4 Orientácia v mape, plánovanie cesty.....	28
4.5 Návrh testov	31
5 Realizácia.....	33
5.1 Výber technológií.....	33
5.2 Popis softvérovej architektúry.....	34
5.2.1 Modul komunikácie.....	34
5.2.2 Modul spracovania obrazu.....	35
5.2.3 Modul navigácie	36
6 Výsledky	39
7 Diskusia.....	Error! Bookmark not defined.
Záver	42
Literatúra.....	44

1 Úvod

Autonómna orientácia robotov v priestore je za posledné desaťročie aktuálnou a zaujímavou témou vzhľadom na rýchlo rastúce tempo rozvoja nie len hardvérovej techniky, ale aj optimalizácie algoritmov.

Veda a výskum sa aktuálne zaoberajú využitím a zefektívnením jednotlivých algoritmických prístupov, čím sa poskytuje priestor na ich komplexné zjednotenie, a teda aj na následné uplatnenie existujúcich poznatkov do praxe.

Súčasťou bežného života je napríklad automatizácia domácich prác, kde sa uplatňuje autonómna robotika na základe komplexného plánovania pokrytia obmedzenej plochy, ktorá zakladá na algoritmoch neurónových sietí s využitím informácie o aktuálnom umiestení robota (robotické vysávače, kosačky, či umývače okien [2]).

Robotika smeruje v dnešnej praxi oveľa ďalej, čoho príkladom je účinná asistencia robotov v medicínskej sfére s využitím precíznej a spoľahlivej vizuálnej asistencie robota v stereo-laparoskopických zákrokoch. Agregácia dát vizuálneho monitorovania, ich následné využitie a analýza prostredníctvom farebnej segmentácie v stereo-laparoskopii umožňuje lekárom ovládať robotov pomocou snímaných farebných značiek a zároveň im uľahčuje nie len diagnostiku, ale aj vykonávanie potrebných zákrokov [1].

Jedným z funkčných príkladov robota orientujúceho sa na základe snímania dráhy je robot Smelý Zajko, ktorý bol pôvodne skonštruovaný a implementovaný za účelom účasti na súťaži RoboTour, kde jeho cieľom bolo prepraviť menší náklad po cestičkách v parku [4]. Robot bol schopný sa autonómne pohybovať na dostatočne rozlíšiteľnej cestičke, pričom orientácia bola implementovaná kombináciou neurónovej siete a spracovania obrazu nasnímanej zeme so snahou detekcie cesty pred sebou a s pomocou GPS. [5]

Po konzultácií sme sa so školiteľom rozhodli myšlienku posunúť ešte ďalej na novú výzvu a cieľom našej práce bolo naučiť robota orientovať sa v interiéri, pričom by

mal byť schopný prepraviť zásielku medzi odosielateľom a adresátom nachádzajúcimi sa v budove pavilónu I. Miesto vyzdvihnutia a doručenia zásielky bude robot prijímať cez webové rozhranie, pričom odosielateľ/prijímateľ zásielky bude upozornený zvukovým signálom.

V chodbových a iných užších interiéroch problematická orientácia, zlyhávajú viaceré systémy použiteľné na otvorenom priestranstve, ako napríklad ultrazvukové senzory, keďže steny chodieb sú detekované ako prekážky a robot má tendenciu zastať. GPS taktiež nie je úplne spoľahlivé, ako aj orientácia na základe kompasu, nakoľko možná kovová konštrukcia budovy môže spôsobiť odchýlky. Naše riešenie si vyžaduje úplne iný prístup.

Pri realizácii projektu nás zaujala myšlienka tímu LEE, ktorý sa na súťaži RoboTour zúčastnil v roku 2009, kde sa robot orientoval z bodov záujmu vygenerovanými algoritmom SURF z prostredia pomocou kamery. Cesta robota bola korigovaná pozíciou týchto príznakov v rámci obrazu. [6]

V druhej kapitole opisujeme teóriu autonómnych robotov a predchádzajúci výskum o uplatnení algoritmov na orientáciu robotov. Taktiež predstavujeme technické parametre a implementáciu Smelého Zajka a objasňujeme možné algoritmy a technológie, ktoré sme mohli pri implementácii cieľa použiť.

V tretej kapitole definujeme podrobnejšie cieľ práce a projektu robota Poštára.

V štvrtej kapitole opisujeme ako sme postupovali pri návrhu riešenia a vyberáme konkrétne algoritmy.

Vo piatej kapitole objasňujeme podrobnosti z realizácie zadania.

Následne v šiestej kapitole nasleduje zhrnutie výsledkov testovania.

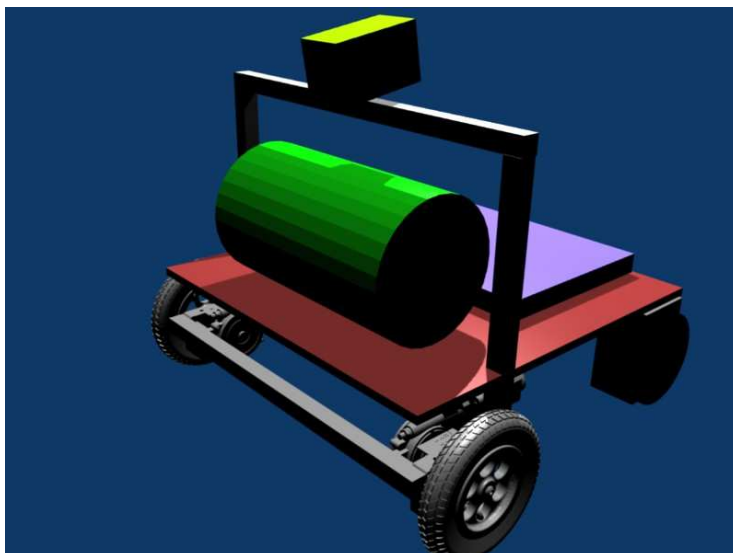
V závere zhodnocujeme prácu a určujeme ďalšie možné výzvy implementovateľné pre robota.

2 Prehľad problematiky

V tejto kapitole opisujeme doteraz implementované spôsoby orientácie robotov ako v interiéri, tak aj v exteriéri.

2.1 Navigačný systém pomocou detekcie cesty a GPS

Príkladom detekcie cesty pomocou kamery je robot Smelý zajko, ktorý bol zostrojený a implementovaný za účelom účasti na súťaži RoboTour. [5]

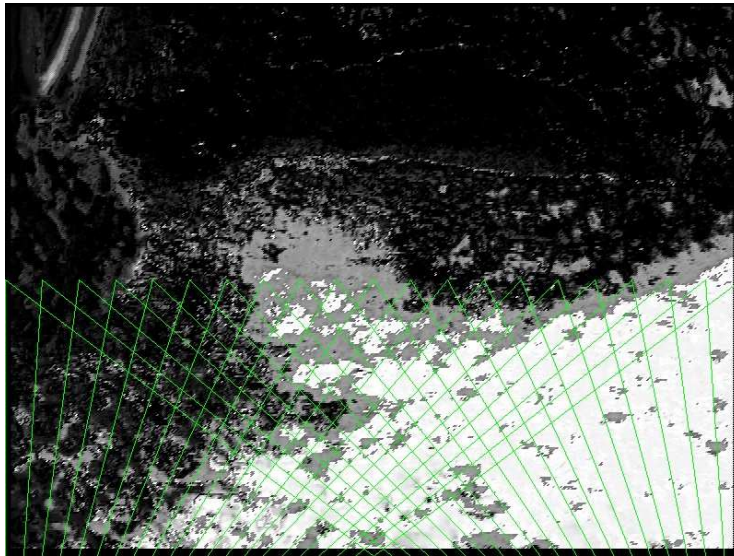


Obr. 1: Návrh konštrukcie robota Smelý zajko [5]

Algoritmus rozpoznávania cesty pracuje na jednotlivých snímkach nezávisle. Keďže robot sa mal pohybovať po rôznych druhoch ciest, rozpoznávanie cesty z obrazu je riešené pomocou MLP neurónovej siete, kde na vstupe robot dostane snímky z cesty. Pre označenie cesty je ručne vytvorená binárna maska, kde biela farba znázorňuje cestu v obraze. Táto sa následne aplikuje na pôvodný obrázok. Táto trojica obrázkov rovnako rozdelená do segmentov slúži ako vstup pre neurónové siete.

Výstupom neurónovej siete bol vektor ohodnotení výhodnosti cesty pre jednotlivé smery v rozsahu $\langle -45^\circ, 45^\circ \rangle$. Keďže sieť nebola príliš spoľahlivá pri zmenách osvetlenia, príp. výskytu tieňov alebo prekážok v ceste, boli výsledky ovplyvnené pridanými modulmi ako histogram farieb, ktorý zohľadňuje farbu cesty aj svetelné podmienky, alebo oblasť s väčšou pravdepodobnosťou výskytu cesty, ktorá predpokladá, že oblasť pred robotom je cesta.

V závere bol každý smer reprezentovaný trojuholníkom, priemerná hodnota v tejto oblasti určovala cestu.



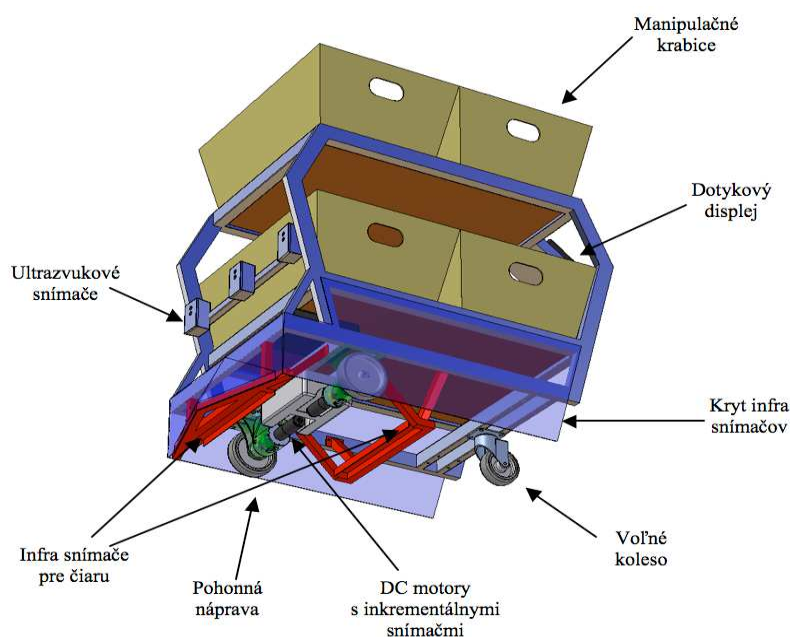
Obr. 2: Ukážka rozhodovania robota [5]

Algoritmus na rozpoznávanie cesty čiastočne rozpoznával prekážky, primárne malo tento problém riešiť použitie ultrazvukových senzorov.

Výsledný uhol pohybu bol z globálneho hľadiska ovplyvnený aj polohou robota na mape a plánovaným smerom cesty. Po lokalizácii na mape pomocou GPS bola naplánovaná trasa z mapy pomocou algoritmu prehľadávania do šírky.

2.2 Navigačný systém pomocou vodiacej čiary

Jedným z príkladov funkčnej realizácie orientácie robota v interiéri je mobilný vozíkový robotický systém ROBOCAR. [3] Úlohou je odbremeniť pracovníkov skladu od prenášania ťažkých nákladov v priestoroch skladu. Boli vyhotovené dva prototypy, z ktorých každý je usposobený na prenos štyroch kartónových krabíc s maximálnou celkovou nosnosťou 40kg.



Obr. 3 Popis konštrukcie ROBOCARU [3]

Cieľové stanovište je robotovi zadané prostredníctvom dotykového displeja. Orientácia v priestore je riešená vyznačením pevnej dráhy na podlahe skladu s určenými stanovišťami označenými pomocou čísiel. Dráha je reprezentovaná súvislou čiarou namaľovanou matnou farbou alebo označená páskou a jednotlivé stanovišťa sú reprezentované krátkymi kolmými čiarami pretínajúcimi dráhu. Podmienkou je, aby stanice boli v dostatočnej vzdialenosti tak, aby nevznikali zápchy. Predbiehanie vozíkov je realizované pomocou výhybiek.

Vyhýbanie sa kolíziám ako aj udržiavanie odstupu medzi vozíkmi je riešené pomocou ultrazvukových senzorov umiestnených v prednej časti vozíka. V prípade, že vozík detekuje prekážku, zastaví a počká kým nie je prekážka odstránená, následne pokračuje vo vykonávaní úlohy. Aby sa predišlo detekcii prekážok v blízkosti vodiacej čiary, mala by byť umiestnená v dostatočnej vzdialenosti od stien a materiálu.

Čiara je snímaná infračervenými snímačmi s cieľom obmedziť rušivé žiarenie (prípadne jeho odraz) z iných zdrojov. Dopomôcť k tomu majú aj hliníkové bočné kryty umiestnené v okolí snímačov.

2.3 Orientácia robota v exteriéri pomocou orientačných bodov

Príkladom orientácie robota pomocou kamery v exteriéri je robot tímu LEE[13], ktorý bol víťazom súťaže RoboTour v roku 2009. Robot sa v prostredí orientoval pomocou odometrie a kamery, konkrétne z bodov záujmu vygenerovanými algoritmom SURF. Z tohto dôvodu nás táto práca zaujala najviac.

Navigácia je založená na dvoch algoritmoch: SurfNav a GeNav, pričom SurfNav je primárnym algoritmom.

Algoritmus GeNav slúži na detekciu cesty a križovatiek a je spúšťaný, keď SurfNav prekročí hranicu, kedy už nie je spoľahlivý. Pracuje na základe rozpoznávania farby cesty, v HSV farebnom priestore, vo vzdialenosti 1-5 m v smere jazdy robota.

Jednotlivé trasy v mape sú reprezentované ako prepojené segmenty. Každý segment je popísaný deskriptormi detekovaných bodov záujmu, obrazovými súradnicami týchto bodov na začiatku a na konci rozpoznávania a dátami z odometrie (vzdialenosť od začiatku segmentu a natočenie robota). Za stabilný bod záujmu sa považuje bod detekovaný v päťdesiatich po sebe idúcich snímkach.

Algoritmus SURFnav koriguje smer jazdy počas behu, kedy sa snaží pozície detekovaných bodov záujmu priblížiť pozíciám naučeným počas mapovania.



Obr. 4: Robot tímu LEE.[13]

2.4 Algoritmy počítačového videnia pre orientáciu robota

V tejto kapitole popisujeme základné algoritmy, z ktorých vychádzame pri naplňaní cieľov tieto práce.

2.4.1 SURF (Speeded Up Robust Features)

Algoritmus SURF extrahuje z obrazu význačné body. Tieto body sú invariantné na rotáciu, škálovanie, šum či zmenu jasu alebo osvetlenia. Hľadanie zodpovedajúcich bodov prebieha v 3 fázach:

1. detekcia bodov záujmu
2. popis okolia bodu záujmu vektorom
3. porovnanie vektorov nájdených bodov

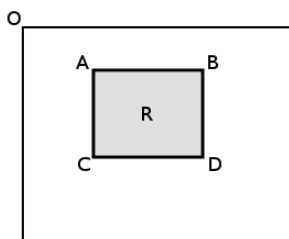
Počet dimenzií vektoru má vplyv na časové spracovanie (t.j. je žiaduce, aby bol čo najmenší). Detekcia bodov záujmu ako aj extrakcia príznakov prebieha na integrálnych obrazoch za cieľom zníženia výpočtového času.

2.4.1.1 Integrálny obraz

Integrálny obraz predstavuje efektívny spôsob výpočtu súčtov hodnôt v obraze alebo regióne. Každý bod integrálneho obrazu predstavuje súčet bodov v regióne ohraničeného počiatkom a daným bodom, pre ktorý je táto suma počítaná ako:

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

Na zistenie sumy hodnôt intenzít regiónu v obraze postačujú teda 4 prístupy do pamäte: $R = A + D - C - B$



Obr. 5: Ilustrácia regiónu v integrálnom obraze [8]

2.4.1.2 Detekcia bodov záujmu

Detektor vychádza z kombinácie Harrisovho detektora s Laplaciánom.

$$H(f(x, y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

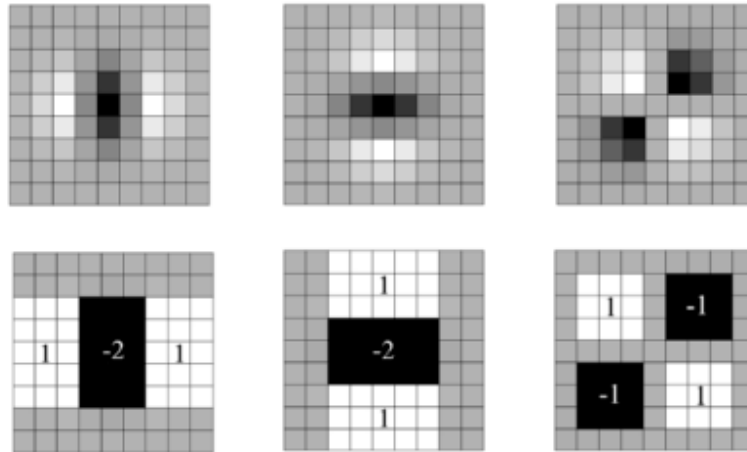
Fast-Hessian detektor

Pozícia aj škála je počítaná z determinantu Hessovej matice:

$$H(X, \sigma) = \begin{bmatrix} L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\ L_{xy}(X, \sigma) & L_{yy}(X, \sigma) \end{bmatrix} \quad \det(H_{\text{approx}}) = D_{xx} D_{yy} - (0.9 D_{xy})^2$$

Pričom $L_{xx}(X, \sigma)$ predstavuje konvolúciu druhej derivácie Gaussovej funkcie $\frac{\partial^2 H(X, \sigma)}{\partial x^2}$ obrazu v bode $X = (x, y)$ pri mierke (škále) σ (obdobne pre L_{xy} a L_{yy}),

čím získame konvolučné masky pre jednotlivé smery (Obr.6. - horná časť). Tie môžeme aproximovať pomocou celočíselných filtrov D_{xx} , D_{yy} , D_{xy} - tzv. box filter (Obrázok 6.



- dolná časť).

Obr. 6: Parciálna derivácia druhého rádu Gaussianu a jej aproximácia v smere x , y a v smere xy . Hodnota šedých regiónov je nulová. [8]

Body záujmu teda predstavujú lokálne maximá v obraze $X=(x, y)$ a škále σ . Hodnota determinantu sa tiež nazýva odozva obrazu na filter.

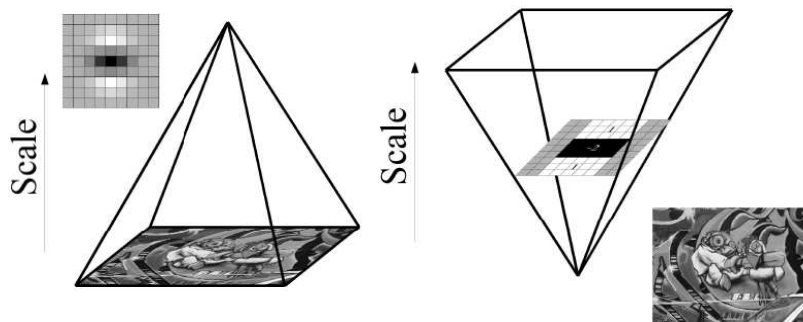
Scale-space reprezentácia

Na rozdiel od väčšiny prístupov, kde je škálovaný obraz, ktorý je vstupom do algoritmu, SURF zabezpečuje invarianciu voči škálovaníu zväčšovaním konvolučnej masky, ktorá je aplikovaná na integrálny obraz, čo výrazne napomáha urýchleniu algoritmu. Najmenšia veľkosť konvolučnej masky je 9x9, čomu približne odpovedá škála $\sigma=1.2$. V každej nasledujúcej škále je kvôli zachovaniu pomeru hrane pridaných 6 pixlov.

Keďže s rastúcim filtrom by narastala aj odozva (hodnota determinantu), je nevyhnutné výslednú hodnotu vydeliť koeficientom zväčšenia.

Odozvy zo scale-space sú najprv prahované (aj v susedných dvoch scale-space)

a následne porovnané s 3x3x3 okolím, pričom sú potlačené lokálne minimá. Nájdené body sú body záujmu.



Obr. 7: Scale-space [8]

Popis okolia bodov záujmu

Pracuje na podobnom princípe ako deskriptor algoritmu SIFT [9], nakoľko poskytuje dostatočnú rozlišovaciu schopnosť. Popis pozostáva z 2 hlavných fáz:

1. Priradenie orientácie detekovanému bodu záujmu na základe prislúchajúceho kruhového okolia, pričom polomer závisí na škále, v ktorej bol bod detekovaný.
2. Zostrojenie štvorcového regiónu pozdĺž zistenej orientácie a extrakcia SURF deskriptora z vytvorenej oblasti.

Priradenie orientácie

Za orientáciu bodu záujmu sa považuje najdominantnejší smer, ostatné sa zanedbávajú (na rozdiel od SIFT), čím je zabezpečená invariantnosť na rotáciu. Algoritmus prebieha v 2 krokoch:

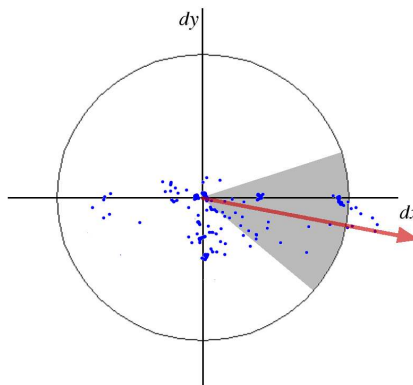
1. Aplikujú sa Haarove vlnky[12] (s dĺžkou strany $4s$) v smere x a y na kruhové

okolie detekovaného bodu (s polomerom $6s$), kde s predstavuje škálu, v ktorej bol bod detekovaný.

2. Zostrojí sa štvorec v súlade so zistenou orientáciou a extrahuje sa z neho SURF deskriptor.

Postupne aplikujeme Haarove vlnky v smeroch osí x a y . Odozvy filtrov v danej oblasti sú následne váhované Gaussovou funkciou ($\sigma=2$) centrovanou v bode záujmu.

Dominantná orientácia predstavuje maximum zo súm pre kruhové výseky veľkosti $\frac{\pi}{3}$. Algoritmus poskytuje dobré výsledky pre rotáciu o cca $\pm 15^\circ$. Z toho dôvodu je odporúčaný krok rotácie 10° .

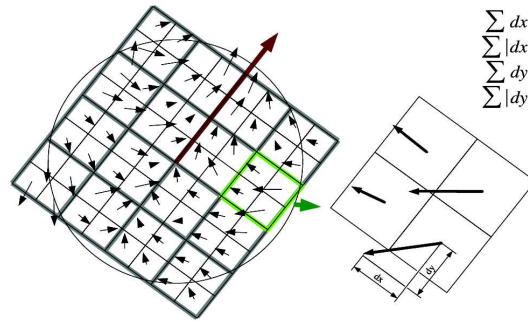


Obr. 8: Pridelenie orientácie [8]

Extrakcia deskriptorov

Po skonštruovaní štvorca veľkosti $20s$ pozdĺž získanej orientácie, je región rozdelený na pravidelné 4×4 subregióny, z ktorých každý obsahuje 25 rovnomerne rozložených vzorkovacích bodov. Následne je sčítaná odozva po aplikácii Haarovho vlnkového filtra pre smery dx a dy , pričom orientácia dx odpovedá orientácii bodu záujmu. Získané hodnoty sú váhované Gaussovou funkciou ($\sigma=3.3$) pre zvýšenie robustnosti.

Každý zo subregiónov je následne popísaný vektorom: $v = [\sum dx, \sum dy, \sum |dx|, \sum |dy|]$, čím získame 64 bitový deskriptor pre detekovaný bod záujmu.



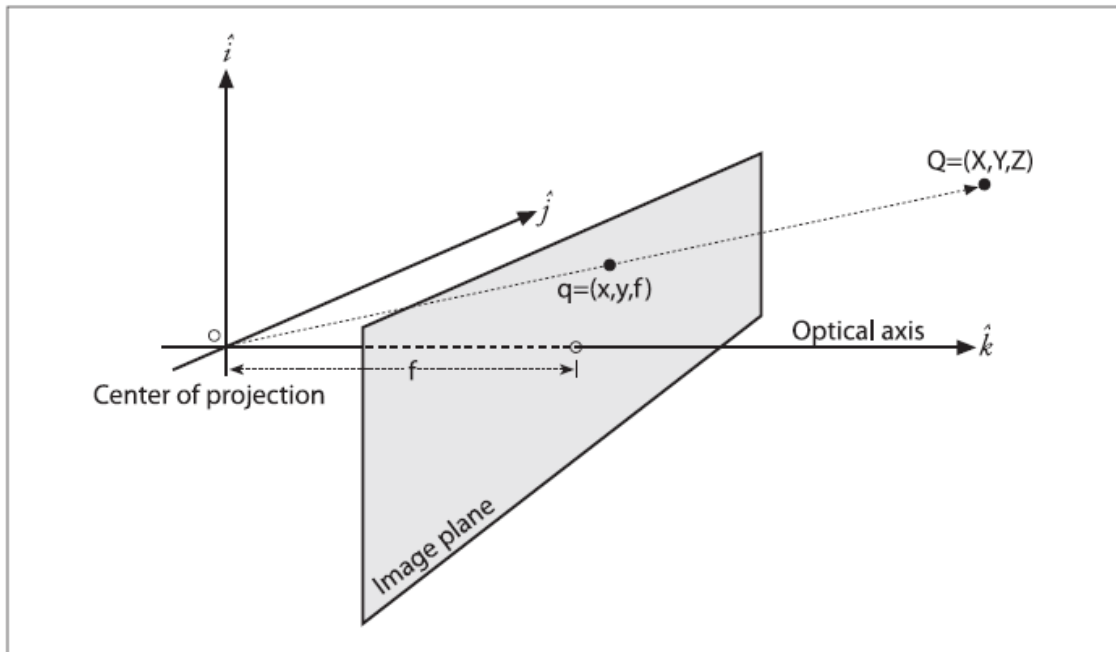
Obr. 9: Budovanie deskriptora [8]

Porovnanie zodpovedajúcich bodov

Pre porovnanie zodpovedajúcich bodov sa používa rýchle indexovanie nájdených bodov pomocou znamienka z Laplaciánu (čím znížime výpočtovú zložitosť, nakoľko bolo určené už počas detekčnej fázy). Toto znamienko rozlišuje svetlé body na tmavom pozadí a naopak. V prípade, že sa zhodujú sú porovnané deskriptory.

2.4.2 Kalibrácia kamery

V tejto kapitole si podrobnejšie popíšeme projekciu na kameru, kde za stred premietania uvažujeme počiatok súradnicovej sústavy, tj. pozícia kamery. Projekčná rovina q je rovnobežná s rovinou ij a posunutá o ohniskovú vzdialenosť f .



Obr. 10: Projekcia bodu v priestore na obrazovú rovinu. [10]

Vzťah, ktorý mapuje bod fyzikálneho sveta Q so súradnicami (X, Y, Z) na bod v projekčnej rovine q so súradnicami (x, y) sa nazýva projekčná transformácia.

Hľadaný bod q prevedieme do homogénnych súradníc, nakoľko vnútorná matica kamery M má rozmery 3×3 .

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad q = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

V ideálnom prípade, by sme bod q našli vzťahom $q = MQ$, avšak ideálny objektív bez skreslenia nie je prakticky možný.

Skreslenie

Existujú 2 druhy skreslenia:

1. *Radiálne skreslenie* vzniká ako dôsledok tvaru šošovky (guľatý alebo parabolický objektív).

2. *Tangenciálne skreslenie* vyplýva z procesu montáže kamery (zladenie objektívu a snímača).

Radiálne skreslenie

Objektívy kamier často výrazne narušujú umiestnenie pixlov v blízkosti hrán (vypuklý efekt). Lúče ďalej od stredu šošovky sú viac ohnuté v porovnaní s lúčmi, ktoré prechádzajú bližšie k stredu šošovky. Radiálne skreslenie je v strede snímača 0 a zvyšuje sa pri pohybe smerom k okraju.

Skreslený obraz sa upravuje tak, že bod obrazu sa líši od ideálneho bodu len o radiálny posun r . Čiže každý bod obrazu je podrobený transformácii, ktorá závisí len od jeho vzdialenosti od stredu obrazu r .

Stred obrazu nepodlieha skresleniu. Čím je transformovaný bod (x,y) viac vzdialený od stredu, tým viac sa jeho poloha skreslí. Radiálne skreslenie teda vieme minimalizovať rovnicami transformácie, kde k_1 , k_2 a k_3 sú parametre radiálneho skreslenia.

$$x' = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y' = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Tangenciálne skreslenie

Tangenciálne skreslenie je spôsobené výrobnou chybou, keď objektív nie je úplne rovnobežný s obrazovou rovinou. Tangenciálne skreslenie je rádovo nižšie ako radiálne skreslenie. Tangenciálna deformácia sa prejavuje vo smere kolmom na radiálnu deformáciu. Skreslenie vieme minimalizovať transformáciou, kde p_1 a p_2 sú parametre tangenciálneho skreslenia.

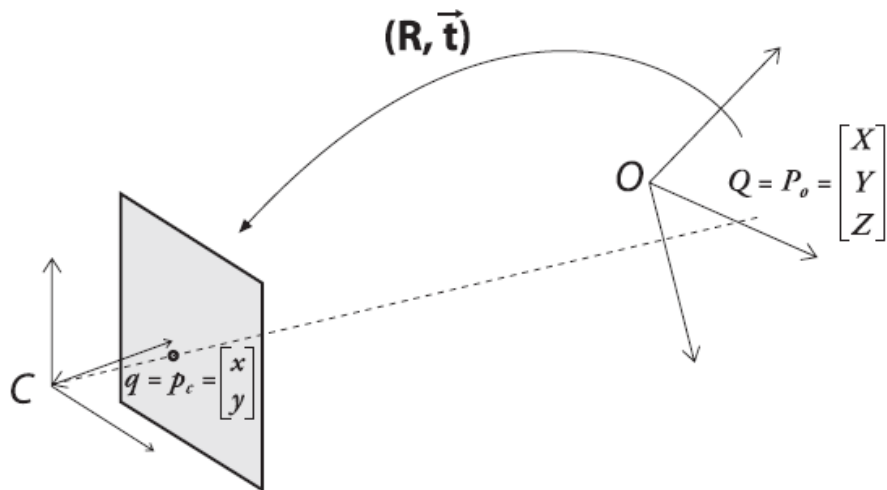
$$x' = x + [2 p_1 y + p_2 (r^2 + 2x^2)]$$

$$y' = y + [p_1 (r^2 + 2y^2) + 2 p_2 x]$$

Spomínané koeficienty skreslenia sú v OpenCv[10][11] v jednom vektore skreslenia, ktorý je reprezentovaný maticou $[K1, K2, P1, P2, K3]$.

Kalibrácia kamery

Kalibrácia v našom prípade neznamená nastavenie kamery, ale naopak určenie vnútorných parametrov kamery. Za vnútorné parametre kamery považujeme vnútornú maticu kamery a vektor skreslenia. V OpenCv je počas kalibrácie kamera zameraná na známu štruktúru, ktorá ma veľa identifikovateľných bodov ako napríklad šachovnica. Zobrazením šachovnice z rôznych uhlov, je možné vypočítať vnútorné parametre kamery, relatívnu pozíciu a orientáciu kamery na každom snímku. Pre úspešnú kalibráciu je potrebné objekt rotovať a posúvať na získanie viacerých pohľadov.



Obr. 11: Transformácia súradníc objektu na súradnicovú sústavu kamery.[10]

Týmito transformáciami vzniká matica rotácie a vektor posunutia. Prevod zo súradnicového systému objektu do súradnicového systému kamery vznikne aplikáciou matice rotácie R a vektora posunutia t na bod Q .

$$q = R(Q - t)$$

3 Špecifikácia

V tejto kapitole podrobnejšie sformulujeme ciele práce a projektu robot Poštár. Zároveň opíšeme funkcionality a perspektívu tohto projektu.

3.1 Formulácia cieľov práce

Cieľom tejto diplomovej práce je nájsť riešenie na orientáciu autonómneho robota v interiéri, pričom by mal dokázať dynamicky reagovať na nové požiadavky na zmenu trasy od používateľa.

Základnou požiadavkou je zvoliť vhodné algoritmy z počítačového videnia, ktoré umožnia lokalizáciu robota v interiéri. Ďalšou požiadavkou je zabezpečiť komunikáciu robota s používateľom, pričom robot má požiadavky prijímať priebežne počas behu.

Keďže robot má reagovať na požiadavky v reálnom čase, je nutné zabezpečiť nepretržité spojenie s používateľom. Jedna z možností je prepojenie zariadenia používateľa s robotom pomocou Bluetooth. Toto pripojenie má však krátky dosah, a keďže predpokladáme, že sa používateľ sa nachádza v inej miestnosti ako robot, je nespoľahlivé. Ako najvhodnejší spôsob sa teda javí webová aplikácia, keďže vychádzame z toho, že v budove kde sa má robot pohybovať a obsluhovať požiadavky, je zabezpečené internetové pripojenie prostredníctvom wifi.

Zároveň bude nevyhnutné vypracovať podrobnú mapu zobrazujúcu aktuálny stav a rozmery pôdorysu budovy. Rozhodli sme sa pre formát Scalable Vector Graphics (SVG), kvôli jednoduchej reprezentácii a možnosti doplniť mapu o vlastné atribúty.

Základným cieľom práce je orientácia robota z kamery, pre ktorú je nevyhnutné dáta nadobudnuté z kamery mapovať na mapu. Rozhodli sme sa ukladať konkrétne body, nadobudnuté zo SURF algoritmu namiesto celých snímok, keďže takéto riešenie by bolo neefektívne. Východzie snímky pre naplnenie cieľov tejto práce získame vo fáze mapovania, kde budeme manuálne ovládať robota pomocou ovládača po vopred naplánovanej trase, pričom jeho polohu budeme musieť v krátkych intervaloch

kontrolovať ručne, aby sme sa vyhli odchýlke ktorá vzniká pri dátach získaných z odometrie. Ak táto odchýlka nebude príliš výrazná, bude možné tento interval predĺžiť, prípadne eliminovať na konkrétny segment mapy, tj. počiatočná a cieľová poloha.

Mapovanie sme sa rozhodli riešiť algoritmom, ktorý z údajov o polohe a natočení robota bude môcť vypočítať polohu detekovaného bodu v obraze na konkrétnu pozíciu na mape. Tento bod získame ako prienik najbližšieho segmentu mapy (zohľadňujúc natočenie robota) a polohu kľúčového bodu v obraze vzhľadom na hlavný bod projekcie (principal point).

Bude teda nevyhnutné kalibrovať kameru. Na naplnenie cieľa práce, teda orientácia robota v interiéri, je nevyhnutná jeho lokalizácia vzhľadom na vyššie zvolenú reprezentáciu mapy.

Pomocou detekovaných bodov záujmu a ich vzájomnej polohy budeme vedieť určiť približnú pozíciu robota v mape. Predpokladáme, že s narastajúcim množstvom detekovaných bodov bude robot svoju pozíciu detekovať presnejšie, čomu dopomôže aj priebežné pridávanie novo detekovaných bodov. V prípade, že algoritmus pre lokalizáciu nebude dávať presné výsledky, ďalším vylepšením na spresnenie pozície robota bude zaznamenávanie jeho poslednej nájdenej polohy.

Predpokladáme, že po úspešnej lokalizácii bude možné naplniť výzvu, z ktorej táto práca vychádza, a teda nájdanie trasy medzi dvoma zadanými pozíciami v interiéri.

Túto úlohu sme sa rozhodli riešiť pomocou algoritmu A* na upravenej mape. Mapu bude nevyhnutné upraviť do mriežkovej štruktúry s konštantnou veľkosťou štvorca, ktorého rozmery budú *dĺžka robota + k*. Táto veľkosť vyplýva z faktu, že chceme, aby bol schopný prechádzať priestorom vzhľadom na jeho veľkosť. Ďalší prístup, ktorý sme zvážili, bol jednoduchý graf reprezentujúci priestor, v ktorom sa robot môže pohybovať. Tu by však bolo nevyhnutné vytvárať ďalšiu mapu na vstupe, kde by bola okrem pôdorysu znázornená aj trasa možného pohybu robota spolu s križovatkami, prípadne túto štruktúru vytvárať automaticky, čo by však bolo implementačne náročnejšie.

3.2 Cieľ projektu robot Poštár

Robot Poštár bude funkčným rozšírením platformy robota Smelý zajko, s ktorým sa bližšie oboznámime v kapitole 4 Návrh riešenia.

Rozšírenie hardvéru nie je potrebné, nakoľko súčasná konštrukcia je na naplnenie cieľov projektu postačujúca. Jedinou zmenou za účelom presnosti lokalizovaných bodov záujmu je umiestniť kameru napevno na podložku, nakoľko je v súčasnosti izolovaná penou, ktorá má znížiť otrasy kamery pri jazde v exteriéri.

Cieľom robota Poštár je doručiť poštu medzi jednotlivými miestnosťami v pavilóne informatiky (ďalej len pavilón I) na fakulte Matematiky, fyziky a informatiky Univerzity Komenského. Robot by mal preberať požiadavky odosielateľa na doručenie pošty prostredníctvom jednoduchého používateľského rozhrania.

Na naplnenie cieľov projektu bude potrebné vytvoriť presnú mapu pôdorysu pavilónu I, keďže mapa pôdorysu pavilónu I zatiaľ nie je v digitálnej podobe odpovedajúcej aktuálnemu členeniu pavilónu. Zároveň má značné odchýlky od aktuálnych rozmerov a nie sú na nej vyznačené dvere.

Ďalším krokom bude zhotovenie testovacieho videa, a to manuálne riadenou jazdou robota po pavilóne I.

3.2.1 Popis funkcionality projektu robot Poštár

Používateľ bude môcť zadávať miestnosť odosielateľa a prijímateľa a prípadne špecifikovať čas prebratia zásielky robotom. V prípade, že používateľ čas prebratia zásielky nezvolí, za čas odoslania sa považuje aktuálny miestny čas.

Jednotlivé požiadavky z grafického rozhrania, ktoré robot prijme budú vybavené v poradí zvolených časov prebratia zásielky na odoslanie. Nasledujúca požiadavka sa začne vybavovať po ukončení predchádzajúcej požiadavky. Požiadavky nebudú spracovávané súčasne.

Robot sa bude lokalizovať na mape v časovom intervale 30 sekúnd, pričom svoju polohu odošle do grafického rozhrania.

Na základe lokalizácie svojej polohy a polohy cieľovej miestnosti nájde na mape najkratšiu trasu k cieľovej miestnosti. Obdobne naplánuje a uloží trasu z miestnosti odosielateľa k miestnosti prijímateľa.

Robot ovláda pohyb kolies a zároveň svoje natočenie na základe svojej aktuálnej polohy a najbližšieho kroku po plánovanej trase. Ak lokalizovaná poloha robota je totožná s polohou cieľovej miestnosti na mape, robot zastane a vyšle zvukový signál, ktorý slúži na upozornenie používateľa na odovzdanie/prijatie zásielky.

Používateľ informuje robota o odovzdaní/prevzatí zásielky prostredníctvom grafického rozhrania. V prípade, že tento signál nevyšle do časového intervalu 2 minút, robot sa vráti so zásielkou späť k miestnosti odosielateľa.

Po prevzatí zásielky bude stav požiadavky o doručenie zmenený na „doručená“. Zmena stavu sa vizuálne prejaví aj v grafickom rozhraní na zadávanie požiadaviek, čo slúži ako informácia pre odosielateľa o úspešnom doručení. V prípade, že robot nespĺní požiadavku, zásielku doručí naspäť odosielateľovi a stav požiadavky o doručení zmení na „neúspešná“. Nespracované požiadavky sú v stave „plánovaná“.

3.2.2 Perspektíva projektu robot Poštár

Implementácia funkcionality robota by mala zaručiť jeho použitie na splnenie cieľa projektu a zároveň v prípade záujmu, rozšíriť jeho pôsobenie v akomkoľvek interiérovom prostredí zadaním rozšírenej mapy.

Robot okrem roznášania pošty v odvetví školstva môže byť použitý na prenos rôznych menších predmetov, napr. v liekov v prostredí nemocnice, či dokumentov v štátnych inštitúciách. Takéto použitie v budúcnosti môže nie len urýchliť prácu, vyhnúť sa chybám, ktoré vznikajú pri ľudskej rozptýlenosti, ale zároveň môže značne ušetriť finančné prostriedky.

Robot by tiež mohol slúžiť ako sprievodca v budove, či v múzeu, kde by bolo vhodné doplniť ho o tablet, kde by na základe polohy zobrazoval informácie o exponáte, pri ktorom sa práve nachádza.

4 Návrh riešenia

Kapitola Návrh riešenia popisuje výber technológií a návrh jednotlivých modulov projektu. Návrh riešenia projektu je členený do modulov, ktoré sú čiastočne kompatibilné s modulmi robota Smelý zajko.

4.1 Robot smelý zajko

Robot Smelý Zajko bol konštruovaný priamo za účelom účasti na súťaži RoboTour v roku 2010. Jeho cieľom bolo prepraviť náklad zo štartu do cieľa, pričom sa robot mohol pohybovať výhradne po cestách vyznačených v mape a nesmel prísť do kontaktu s prekážkou.

4.1.1 Mechanika

Samotnú konštrukciu ovplyvnili aj skúsenosti tímov z minulých ročníkov, keďže sa jednalo o 5. ročník súťaže. Pohon kolies je riešený pomocou systému "differential drive" - kolesá sa môžu točiť nezávisle od seba, čo umožňuje otáčanie robota na mieste a teda lepšiu manipuláciu v zúžených priestoroch. Robot má 3 kolesá čo zabezpečuje jeho lepšiu stabilitu.

4.1.2 Systém robota

Systém robota je členený do troch softvérových modulov: lokalizácia, plánovanie, rozpoznávanie cesty a prekážky. Vstupné dáta do jednotlivých modulov poskytujú senzory: kompas, GPS, odometria (lokalizácia a plánovanie), kamera (rozpoznávanie cesty) a ultrazvukové senzory (rozpoznávanie prekážok).

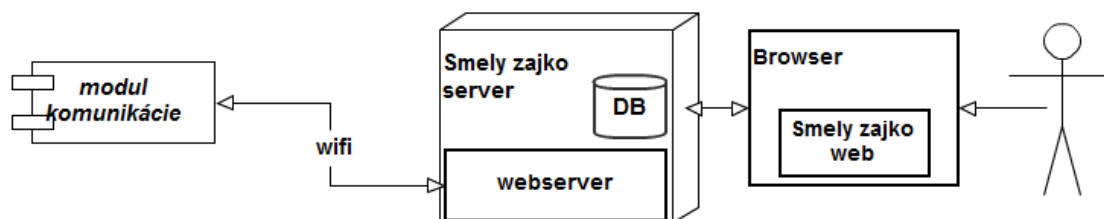
4.2 Návrh komunikácie robota s používateľom

Modul komunikácie má na starosti výmenu informácií medzi grafickým používateľským rozhraním a robotom.

Robot bude získavať informácie o zásielkach z webovej stránky pomocou knižnice libcURL. Výhodou cURL projektu je, že je open-source softvér optimalizovaný pre rôzne operačné systémy. Knižnica libcURL slúži na prenos dát pomocou rôznych protokolov. Pre našu funkcionality bude postačujúce vytvoriť jednoduchú metódu, ktorá nadviaže spojenie so serverom za cieľom výmeny informácií o polohe robota a požiadavkách, ktoré zadá používateľ. Odpoveď na požiadavku bude v jednoduchom formáte (*string*), čo zjednoduší spracovanie na strane robota.

Taktiež bude nevyhnutné zabezpečiť autentifikáciu používateľa na webovej stránke, aby sme sa vyhli zneužitiu robota.

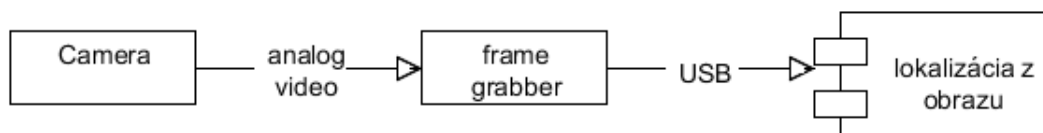
Preberanie požiadaviek a komunikácia je znázornená na obrázku *Obr. 12 Diagram rozmiestnenia elementov komunikácie*.



Obr. 12: Diagram rozmiestnenia elementov komunikácie.

4.3 Návrh lokalizácie robota v interiéri

V tejto časti navrhujeme riešenie na kalibráciu kamery a detekciu bodov záujmu pomocou SURF algoritmu. Všetky snímky budú mať pred ich spracovaním odstránené skreslenie z kamery.



Obr. 13 Diagram rozmiestnenia elementov lokalizácie.

Kalibrácia

Kalibrácia kamery bude riešená pomocou knižnice OpenCv. Výsledkom kalibrácie bude matica, ktorá bude uložená v XML súbore a načítavaná na začiatku behu robota. Z každého obrazu bude pred spracovaním odstránené skreslenie, čím sa vyhneme prípadným odchýlkam.

Nájdenie rohov šachovnice

V OpenCv je funkcia `cvFindChessboardCorners`, ktorá slúži na detekciu rohov šachovnice a má nasledujúce argumenty, ktoré navrhujeme nastaviť:

- *corners* - smerník na pole, v ktorom sú zaznamenané pozície rohov
- *image* - obrázok zachytávajúci šachovnicu
- *pattern_size* - počet vnútorných rohov v každom riadku a stĺpci
- *corner_count* - voliteľný argument, ktorý definuje počet nájdených rohov
- *flags* - príznaky pre jeden alebo viacero filtračných krokov na pomoc nájdenia rohov na šachovnici
 - *CV_CALIB_CB_ADAPTIVE_THRESH* predvolene prahuje obrázok na základe priemerného jas. Ak je tento príznak nastavený, bude použité adaptívne prahovanie.

- *CV_CALIB_CB_NORMALIZE_IMAGE* zabezpečí, že pred prahovaním bude snímka normalizovaná pomocou ekvalizácie histogramu.

Korekcia pozície rohov

Rohy vrátené funkciou *cvFindChessboardCorners* sú len orientačné. Polohy bodov sú s presnosťou jedného pixlu. Funkcia *cvFindCornerSubPix* vypočíta presné umiestnenie rohov s presnosťou subpixlov. Ak zanedbáme volanie tejto metódy, prvé nájdenie pozície rohov môže spôsobiť značné chyby v kalibrácii.

Vstupnými argumentmi tejto funkcie sú:

- *win* - polovica veľkosti vyhľadávacieho okna
- *criteria* - kritériá pre ukončenie iteračného procesu korekcie rohov
 - *CV_TERMCRIT_ITER* - algoritmus sa zastaví po určitom počte iterácií
 - *CV_TERMCRIT_EPS* - algoritmus sa zastaví ak sa dosiahne požadovaná presnosť

Kalibrácia kamery

Funkcia *cvCalibrateCamera2* vypočíta vnútorné parametre kamery a parametre skreslenia. Pre kalibráciu sú minimum 2 obrázky na šachovnici veľkosti 3x3, kde sa počítajú len vnútorné rohy. V praxi, pre vysoko kvalitné výsledky budeme potrebovať najmenej desať obrázkov 7x8 alebo väčšej šachovnice. Ideme popísať argumenty tejto funkcie, kde *M* značí počet obrazov a *N* celkový počet bodov vo všetkých obrazoch. Rozhodli sme sa nastaviť nasledujúce argumenty:

- *object_points* - matice súradníc bodov objektu, 3xN. Tieto body sú v súradnicovej sústave objektu. Sú definované aj fyzikálne jednotky a štruktúra súradnicovej sústavy. Napríklad pri šachovnici sa definujú súradnice tak, že z-tová súradnica bude 0 a x-ová a y-ová súradnica sa budú merať v cm.
- *image_points* - matica 2xN obsahujúca súradnice všetkých bodov v pixloch, ktoré sú v *object_points*. Pri šachovnici sa tento argument sa skladá iba z návratových hodnôt *M* volaní funkcie *cvFindChessboardCorner*.

- *points_count* - počet bodov v každom snímku, ktorý definuje vektor veľkosti $1 \times M$.
- *image_size* - veľkosť snímok v pixloch, z ktorých boli obrazové body získané.

Vnútorne parametre kamery tvoria *intrinsic_matrix* a *distortion_coeffs* a sú hlavným dôvodom kalibrácie kamery. Môžu sa použiť aj ako vstup, potom budú mať hodnoty v týchto maticiach vplyv na vypočítané výsledky funkcie. Ktorá z týchto matíc bude použitá ako vstup bude závisieť na zvolených príznakoch.

- *intrinsic_matrix* - výstup vnútornej matice kamery 3×3 tvaru $[fx \ 0 \ cx, \ 0 \ fy \ cy, \ 0 \ 0 \ 1]$. Tento výstup ovplyvňuje príznak *CV_CALIB_FIX_ASPECT_RATIO*
- *distortion_coeffs* - výstup je vektor koeficientov deformácie 5×1 tvaru $[K1, K2, p1, p2, K3]$
- *flags* - OpenCV môže kontrolovať hľadanie parametrov pomocou nastavenia príznakov. Príznaky povoľujú kontrolu, ako presne bude vykonaná kalibrácia.
 - *CV_CALIB_FIX_ASPECT_RATIO* - ak je nastavený tento príznak, potom sa udržiava fixný pomer fx / fy , ktoré sú nastavené pri inicializácii vo vnútornej matici kamery.

Odstránenie skreslenia

Odstrániť skreslenie môžeme pomocou funkcie *cvUndistort2* alebo dvojicou funkcií *cvInitUndistortMap* a *cvRemap*, ktoré nám vyriešia skreslenie trochu efektívnejšie pre video alebo situácie, kedy máme veľa obrázkov z jednej kamery.

Základnou metódou je vypočítať mapu skreslenia, ktorá sa potom používa na korekciu obrázku. Funkcia *cvInitUndistortMap* počíta mapu skreslenia a *cvRemap* použije túto mapu na ľubovoľný obrázok. Nakoniec keď máme zoznam 2D bodov, môžeme volať funkciu *cvUndistortPoints*, aby sme ich transformovali z ich originálnych do neskreslených súradníc.

Funkcia *cvInitUndistortMap* počíta mapu skreslenia, ktorá sa týka každého bodu obrázku. Prvé dva argumenty sú vnútorná matica kamery a koeficienty skreslenia, ktoré sú výstupom z funkcie *cvCalibrateCamera*.

SURF

Implementácia SURF algoritmu bude taktiež riešená s využitím knižnice OpenCv, nakoľko má implementované čiastkové funkcie pre tento algoritmus. Výsledkom SURF algoritmu budú detekované body záujmu spolu s ich deskriptormi, ktoré budeme ukladať do dátového súboru. Pre prehľadnosť rozčleníme algoritmus do nasledujúcich štyroch krokov.

Nájdenie bodov záujmu

Pre snímok, ktorý je vstupom do algoritmu najprv z funkcie *getKeypoints* vrátime nájdené body záujmu.

- *detector.detect(image, keypoints)* - použitím detektora *SurfFeatureDetector* sa vykoná proces detekcie bodov záujmu z obrázka *image*, ktoré budú uložené v poli *keypoints*.

Extrakcia deskriptorov pre body záujmu

Pre nájdené body záujmu vo funkcii *getDescriptors* vyextrahujeme deskriptory, pomocou ktorých budeme môcť nájsť zhodné body v rámci sekvencie.

- *extractor.compute(image, keypoints, descriptors)* - použitím extraktoru *SurfDescriptorExtractor* nájdeme vektory príznakov *descriptors* pre jednotlivé body záujmu *keypoints*, konkrétne funkciou *compute*.

Nájdenie zhody detekovaných bodov záujmu s uloženými bodmi záujmu. Funkcia *getMatches* vráti zhodné body, ktoré boli detekované v obrázkoch z jednej sekvencie.

- *matcher.match(descriptors1, descriptors2, matches)* - navzájom si odpovedajúce body záujmu sú vyhľadávané pomocou *FlannBasedMatcher*, a to nachádzaním zhôd *matches* medzi vektormi príznakov *descriptors* vo funkcii *match*. *FlannBasedMatcher* je rýchlejší pri porovnávaní veľkej trénovacej sady deskriptorov ako *BruteForceMatcher*. Nájdené zhody následne odfiltrujeme podľa vzdialenosti vektorov príznakov, čím zabránime falošným zhodám.

Uloženie bodov záujmu

Nový bod záujmu uložíme volaním funkcie *saveNewKeypoint* ak bude detekovaný aspoň v 30-tich po sebe idúcich obrázkoch.

Mapa

Pre načítanie pôdorysu budovy sme sa rozhodli použiť formát SVG, nakoľko ide o jednoduchý xml formát, ktorý môžeme rozšíriť o vlastné atribúty a rovnako môže byť použitý na zobrazenie pozície robota na stránke. Pre vytvorenie mapy bude nevyhnutné presne zmerať priestor, v ktorom sa robot bude pohybovať. Aby bolo jednoduché rozšíriť tento priestor, zhotovíme program, kde na vstupe dostaneme namerané rozmery, uhly medzi jednotlivými stenami a prípadné parametre (číslo dverí, ...). Výstupom bude korešpondujúca mapa vo formáte SVG.

Pre orientáciu z kamery budeme musieť triedu reprezentujúcu vytvorenú mapu rozšíriť o detekované body záujmu, čo je popísané v nasledujúcej kapitole.

4.4 Orientácia v mape, plánovanie cesty

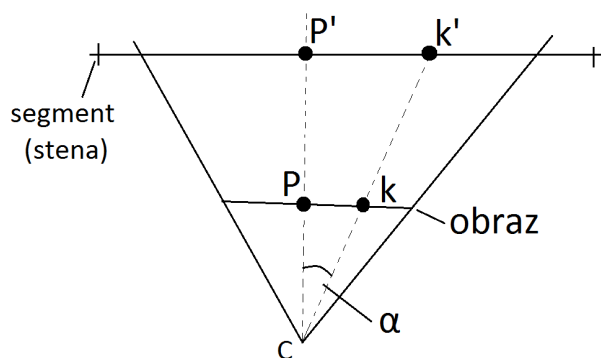
V tejto kapitole podrobne opíšeme návrh mapovania, lokalizácie robota a algoritmus pre plánovanie trasy.

Mapovanie

Cieľom mapovania je určiť pozíciu detekovaných bodov záujmu na mape. Vo fáze mapovania umiestnime robota na známu pozíciu s natočením s uhlom α . Z matice vnútorných parametrov kamery poznáme ohniskovú vzdialenosť f . Taktiež poznáme pozíciu *principal point* $P(x_1, y_1)$ a bodu záujmu $K(x_2, y_2)$, ktorý chceme namapovať na segment mapy. Z daných hodnôt vypočítame $\tan \beta = (x_1 - x_2) / f$. Pomocou funkcie atan zistíme veľkosť uhla β , pod ktorým vidíme daný bod záujmu.

Z pozície robota a súčtu výsledného uhlu ($\alpha + \beta$) skonštruujeme polpriamku. Prienik polpriamky a najbližších segmentov v smere natočenia robota nám určí pozíciu (x, y) lokalizovaného bodu záujmu.

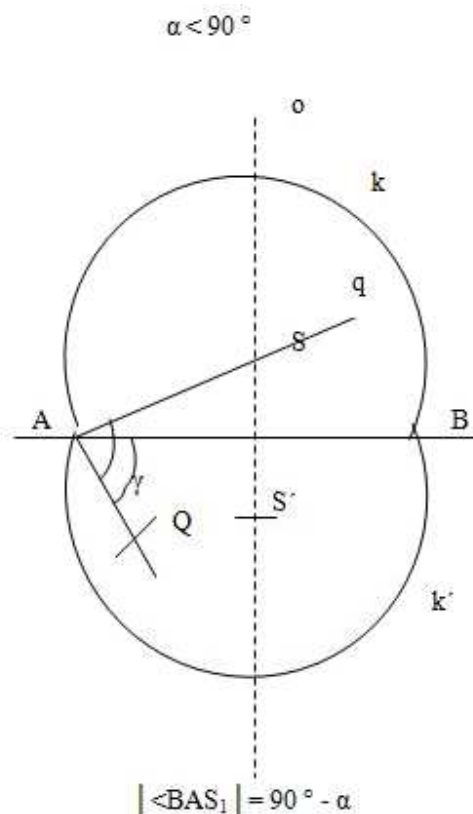
Obdobne pomocou súradnice y vypočítame výšku bodu z v priestore. V prípade, že bude hodnota z záporná/väčšia ako 3.5m (výška steny), detekovaný bod záujmu sa nachádza na podlahe/strope, a teda ho neberieme do úvahy. Výsledný bod $R(x, y, z)$, ktorý vráti funkcia *findInMap*, uložíme spolu s deskriptorom lokalizovaného bodu záujmu a príslušným *id* segmentu mapy.



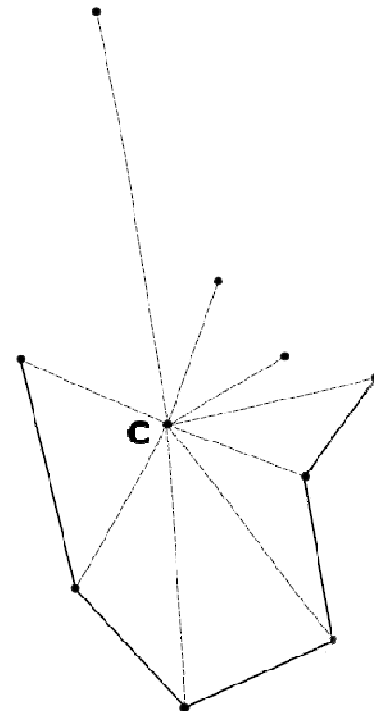
Obr. 14: Projekcia bodu záujmu K na bod K' na segmente. Ohnisková vzdialenosť je označená f , bod P označuje *principal point* a P' jeho projekciu na segment. Výsledný uhol pod ktorým robot vidí bod K , K' je alfa.

Lokalizácia

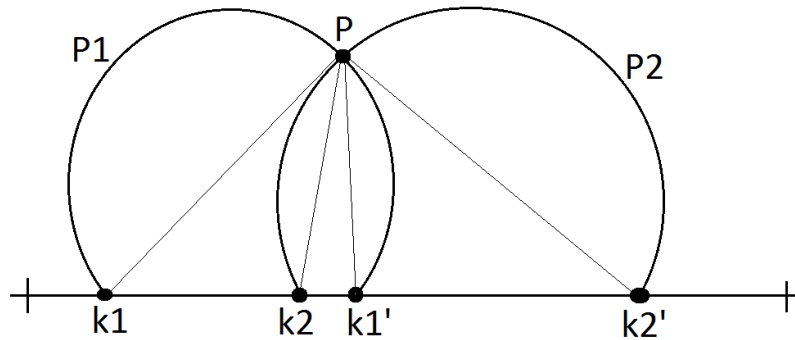
Pri lokalizácii budeme pomocou detekovaných bodov záujmu určovať pozíciu robota. Algoritmus bude fungovať podobne ako algoritmus mapovania. Zistíme, pod akým uhlom vidíme úsečku tvorenú dvoma detekovanými bodmi záujmu A, B . *Principal point* teraz nemôžeme použiť, nakoľko nevieme kam na segment sa nám premietne. Konštrukciou kružnicového oblúka nad úsečkou nájdeme množinu bodov, z ktorých vidíme úsečku pod daným uhlom α , tj. množinu potenciálnych polôh robota P_1 . Obdobne skonštruujeme množiny pre ostatné detekované body $P_2..P_n$. Prienikom týchto množín, kružnicových oblúkov, spresníme polohu robota nájdením centroidu týchto prienikov.



Obr. 15a Množina bodov, z ktorých je úsečka AB viditeľná pod rovnakým uhlom [14]



Obr. 15b Centroid prienikov kružnicových oblúkov



Obr. 16: Body k_1 a k_1' označujú zvolenú dvojicu bodov záujmu. Kruhový výsek p_1 označuje množinu potenciálnych polôh robota vzhľadom na uhol pod ktorým tieto body vidí. Obdobne pre k_2 a P_2 . Predpokladaná pozícia robota je prienik týchto množín, teda bod P .

Plánovanie trasy

Prvým krokom plánovania bude úprava mapy do mriežky, aby sme mohli na vyhľadávanie trasy použiť algoritmus A*. Veľkosť mriežky bude daná dĺžkou základne robota, tj. 45cm.

Algoritmus A* vyhľadá najkratšiu optimálnu trasu na princípe algoritmu prehľadávania do šírky. Postupne prehľadáva jednotlivé políčka, ktoré sú ohodnotené funkciou $f(x) = g(x) + h(x)$, pomocou doteraz prejdenej vzdialenosti $g(x)$ a heuristickej funkcie $h(x)$, ktorá vyjadruje predpokladanú dĺžku cesty do cieľového bodu.

4.5 Návrh testov

V tejto podkapitole uvidíme návrhy na testovanie jednotlivých algoritmov, ktoré použijeme na naplnenie cieľov tejto práce a funkcionality robota Poštára.

Testovanie komunikácie robota a používateľa

Bude nevyhnutné otestovať 2 základné funkcie: prebratie požiadavky robotom, aktualizácia polohy robota a zobrazenie na stránke.

Pri preberaní požiadavky môže nastať viacero možností: žiadna nová požiadavka nebude prijatá, bude prijatá jediná požiadavka na odoslanie zásielky alebo bude prijatých viacero požiadaviek s rovnakým časom odoslania.

Požiadavky budú zadávané manuálne pomocou web stránky, ktorá bude slúžiť na komunikáciu robota s používateľom.

Testovanie lokalizácie robota

Táto časť testovania je pre našu prácu kľúčová, nakoľko zhodnotí naplnenie cieľu tejto práce, ktorý je uvedený v kapitole 3.1. Formulácia cieľov práce.

Pri tomto teste použijeme video nasnímané kamerou Panasonic SDR-T50 počas testovacej jazdy robota Poštára, ktorého budeme ovládať manuálne. Keďže snímková frekvencia kamery je 60 fps, predpokladáme, že bude postačujúce detekovať body záujmu v každom desiatom obraze pri mapovaní a v každom tridsiatom pri lokalizácii.

V našom teste budeme zvažovať 2 prípady: lokalizácia z jedného snímku a lokalizácia „za behu“, tj. zo sekvencie obrazov.

Pri lokalizácii pomocou 1 obrazu budeme náhodne vyberať 15 obrazov, na ktorých sa pokúsime určiť polohu robota vo videu použitím SURF algoritmu. Pri lokalizácii zo sekvencie vyberieme dve sekvencie rôznej dĺžky v rozmedzí 10-30 sekúnd z rôznych častí pavilónu.

V prípade, že výsledky tohto testu nebudú uspokojivé, bude dôležité rozlíšiť či zlyhalo mapovanie alebo lokalizácia. Neúspešnosť lokalizácie sa dá otestovať

viacnásobnou lokalizáciou na jednom obraze. V prípade, že sa robot lokalizuje na pozíciách príliš navzájom vzdialených, bude zrejmé, že zlyhal náš algoritmus na lokalizáciu.

Testovanie vykonávania požiadavky

Po úspešnom vykonaní predošlých testov budeme testovať vykonanie prijatej požiadavky. Tu sa otestuje plánovanie trasy a prechod po naplánovanej trase pomocou lokalizácie počas pohybu robota. Robot začne vykonávať najbližšie naplánovanú požiadavku zadanú pomocou webovej stránky.

5 Realizácia

V tejto kapitole popíšeme vybrané technológie a implementáciu cieľov práce.

5.1 Výber technológii

V tejto časti sa zameriavame na návrh a popis technológii, ktoré je možné použiť pri implementácii projektu robot Poštár. Výber technológii je z veľkej miery ovplyvnený platformou robota Smelý zajko.

C++

Platforma robota Smelý zajko je implementovaná v programovacom jazyku C++. Pre kompatibilitu s robotom a možnosť využitia jeho implementovanej funkcionality robota sme sa rozhodli dopĺňujúce moduly programovať taktiež v C++.

OpenCv

Ako je spomenuté v kapitole 1.Úvod pre orientáciu robota v interiéri je vhodné a spoľahlivé použiť kameru. Knižnica OpenCv je podporovaná jazykom C++ a zároveň poskytuje prevažnú časť funkcií bežne využívaných pre spracovanie obrazu.

PHP

Keďže používateľ ma zadávať požiadavky a prijaté dáta majú byť priebežne prístupné, zvolili sme implementáciu prostredníctvom webového grafického rozhrania, pričom dáta budú ukladané do MySQL databázy. Ako spoľahlivé a bezpečné riešenie sme zvolili jazyk PHP, konkrétne Yii framework.

5.2 Popis softvérovej architektúry

Projekt robot Poštár je členený do modulov, ktoré medzi sebou komunikujú a sú čiastočne kompatibilné s modulmi robota Smelý zajko.

5.2.1 Modul komunikácie

Zadávanie požiadaviek je umožnené pomocou webovej stránky, ktorá je aktuálne prístupná na adrese: <http://dai.fmph.uniba.sk/projects/smelyzajko/> a zobrazená na obrázku *Obr. 16 Webové rozhranie*. Používateľovi je po prihlásení sprístupnené zadávanie počiatkovej a cieľovej miestnosti, prípadne času odoslania zásielky. Informácie o požiadavkách na zaslanie spolu s informáciou o ich stave sú prehľadne zobrazené v tabuľke. Zároveň sa na mape zobrazuje pozícia robota.

Štart	Cieľ	Čas doručenia	Stav
I 1	I 2	2014-02-10 11:24:22	Plánované
I 4	I 12	0000-00-00 00:00:00	Plánované
I 5	I 15	2014-02-12 11:00:00	Plánované
I 1	I 16	2014-02-11 00:00:00	Plánované
I 2	I 1	2014-03-01 00:00:00	Plánované
I 5	I 1	2014-03-01 00:00:00	Plánované
I 2	I 1	2014-02-28 00:00:00	Plánované

Obr. 17: Webové rozhranie pre zadávanie požiadaviek robotovi.

Spojenie medzi robotom a serverom, na ktorom je webová stránka umiestnená, prebieha cez POST requesty, ktoré sú implementované pomocou knižnice libcurl. Odpoveď na tento request je jednoduchý textový reťazec „*cislo_dveri_odosielatela cislo_dveri_prijemcu cas_odoslania*“. Metóda POST je implementovaná nasledovne:

```
CURLcode response;
CURLcode post(char* urlAddress, char* param) {
    curl_easy_setopt(curl, CURLOPT_URL, urlAddress);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, param);
    response = curl_easy_perform(curl);
    if (response != CURLE_OK) {
        printf(" POST failed: %s\n",
            curl_easy_strerror(response));
    }
    return response;
}
```

5.2.2 Modul spracovania obrazu

Modul spracovania obrazu obsahuje triedy na spracovanie obrazu a kalibráciu.

Kalibrácia

Kalibrácia je riešená pomocou šachovnice rozmerov 8x8. Program kalibrácie postupne vykonávame v troch krokoch.

1. Najprv získame rozmery šachovnice, snímky šachovnice a nájdeme všetky rohy šachovnice.

```
cvFindChessboardCorners(image, board_size, corners,
    &corner_count, CV_CALIB_CB_ADAPTIVE_THRESH |
    CV_CALIB_CB_FAST_CHECK | CV_CALIB_CB_NORMALIZE_IMAGE);
cvFindCornerSubPix(grayImg, corners, corner_count,
    cvSize(11,11), cvSize(-1,-1), cvTermCriteria(
```

```
CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, 30, 0.1));
```

2. Následne vypočítame vnútorné parametre kamery a parameter skreslenia a uložíme ich do xml súborov.

```
cvCalibrateCamera2(object_points, image_points,  
point_counts, cvGetSize(image), intrinsic_matrix,  
distortion_coeffs, NULL, NULL,  
CV_CALIB_FIX_ASPECT_RATIO);
```

3. Nakoniec odstránime skreslenie a zobrazíme neskreslenú verziu obrazu.

```
cvInitUndistortMap(intrinsic,distortion,mapx,mapy);  
cvRemap(image_clone, image, mapx, mapy);
```

SURF

Pri implementácii SURF algoritmu sme postupovali podľa návrhu uvedenom v kapitole 4.3 Návrh lokalizácie robota v interiéri, pričom *SurfFeatureDetector* inicializujeme s minimálnym prahom Hessiánu 400. Vo funkcii *getMatches* odstraňujeme nájdeme zhody, ktorých vzdialenosti vektorov príznakov prekračujú prah 100. Odfiltrované body záujmu sú hľadané nasledovným spôsobom:

```
SurfFeatureDetector detector(400);  
keypoints = getKeypoints(image);  
descriptors = getDescriptors(image, keypoints);  
matches = getMatches(descriptor1, deskriptor2, 100);  
saveNewKeypoint(descriptors, keypoints, matches);
```

5.2.3 Modul navigácie

Príklad vstupu textového súboru pre algoritmus na vytvorenie mapy, kde stena je označená *w* *dlzka_steny*, dvere *d* *dlzka_dveri* *cislo_dveri*, zmena natočenia *r* *velkost_uhla* a komentár *# text_komentáru*:

```
w 240  
d 87 I32  
w 215  
# pocitacove miestnosti  
r 90
```

Výstup SVG súboru z príkladu uvedeného vyššie a jeho grafická reprezentácia:

```
<line x1="3638" y1="570" x2="3398" y2="570" lineId="78" />
<line x1="3398" y1="570" x2="3311" y2="570" lineId="79"
doors="I32"/>
<line x1="3311" y1="570" x2="3096" y2="570" lineId="80" />
```

Obr. 18: Výstup zobrazujúci úsek mapy.

5.2.3.1 Mapovanie

Výsledkom funkcie *findInMap* je dvojica obsahujúca polohu bodu záujmu na mape a ID segmentu, na ktorom sa nájdený bod nachádza. Vstupnými parametrami je ohnisková vzdialenosť, principal point, bod záujmu, ktorý ideme mapovať, pozícia a orientácia robota a mapa.

```
pair<Point, int> findInMap(focal_length, principal_point,
    key_point, robot_position, robot_orientation, map)
```

Funkcia *getViewAngle* vracia uhol pohľadu robota na bod záujmu.

```
angles = getViewAngle(focal_length, principal_point,
    key_point);
```

Pomocou *getNearestSegments* získame najbližšie hrany, ktoré robot môže potencionálne vidieť, v rozmedzí +/- 90 stupňov od jeho natočenia.

```
final_orientation = robot_orientation + angles.getX();
segments = getNearestSegments(map, robot_position,
    robot_orientation, 180);
```

Následne nájdeme segment, na ktorom leží bod záujmu.

```
Pair<point, id> segment = getIntersection
    (robot_position, final_orientation, segments);
```

5.2.3.2 Lokalizácia

Pri implementácii lokalizácie robota sme taktiež postupovali podľa návrhu. Funkcia *getRobotPosition* vracia predpokladanú pozíciu robota. Vstupom funkcie sú body záujmu, ktoré robot z danej pozície a orientácie vidí, ohnisková vzdialenosť, ktorá je daná vo vnútorných parametroch kamery a mapa.

Nájdeme 10 prienikov kružnicových oblúkov zostrojených nad náhodne vybranými dvojicami bodov záujmu. Za výslednú pozíciu robota považujeme centroid tejto množiny prienikov.

```
Point getRobotPosition(key_points, focal_length, map) {
    for (i=0; i<10; i++) {
        key_points1 = getRandomPair(key_points);
        key_points2 = getRandomPair(key_points);
        angles1 = getViewAngle(focal_length,
            key_points1.getFirst(), key_points1.getSecond());
        angles2 = getViewAngle(focal_length,
            key_points2.getFirst(), key_points2.getSecond());
        points.push(getIntersection(
            countGSet(key_points1, angle1.getX()),
            countGSet(key_points2, angle2.getX())));
    }
    position = findCentroid(points);
}
```

6 Výsledky

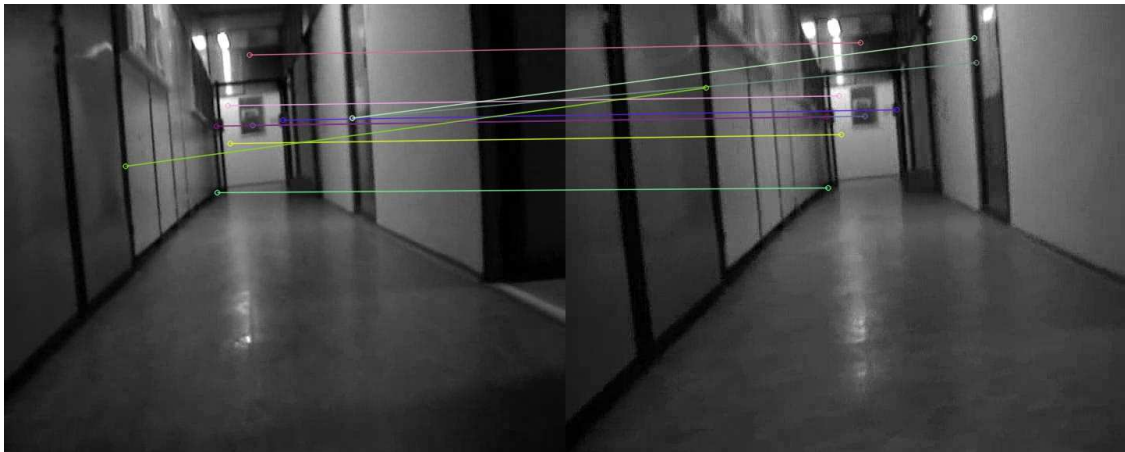
V tejto kapitole uvedieme výsledky uskutočnených testov .

Testovanie komunikácie robota a používateľa

Podarilo sa nám úspešne nadviazať spojenie so serverom. Robot dokáže prijímať požiadavky vo formáte(*start, cieľ, datum_cas_prevzatia*), ktoré následne spracuje a tiež aktualizovať svoju polohu na mape, či . Filtrovanie požiadaviek na základe stavu a dátumu prevzatia prebieha už na serveri, robot teda vždy dostane najbližšiu plánovanú požiadavku na doručenie.

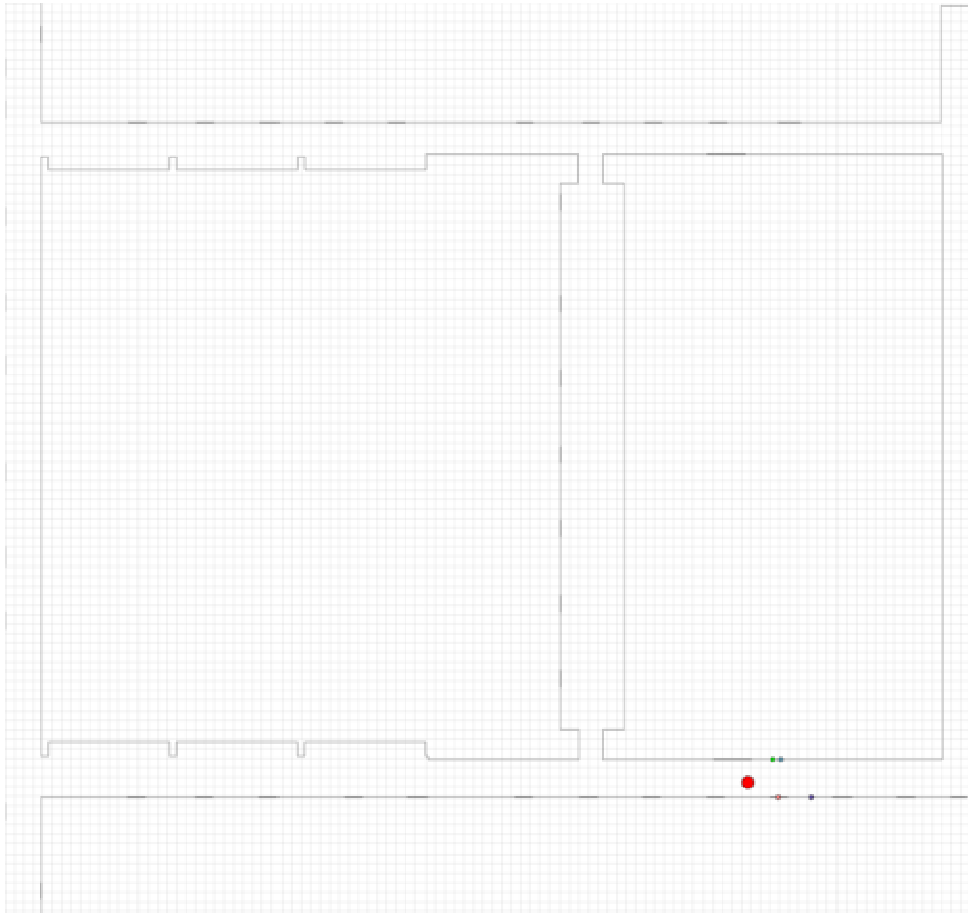
Testovanie lokalizácie robota

Algoritmus SURF nám rozpoznáva cca 200-250 bodov záujmu na jeden frame, pričom cca 20 nájde zhodu.



Mapovanie

Pri mapovaní je nevyhnutné previezť ohniskovú vzdialenosť a vzdialenosť medzi detekovanými bodmi do rovnakých jednotiek (pixlov). Rovnako je potrebné zamerať natočenie robota čo najpresnejšie nakoľko pri odchýlke jedného stupňa môžu vzniknúť odchýlky od pôvodnej polohy rádovo v centimetroch.



Obr. 18: Mapa pavilónu I. Robot (červený bod) je natočený smerom ku kratšej stene. Označené body sú namapované body z obrazu.

Lokalizácia

Pri lokalizácii nastal problém s určením finálnej pozície pomocou centroidu, keďže navrhovaný algoritmus pri bodoch záujmu z protiľahlých segmentov detekoval viacero prienikov na rozdielnych stranách, odchýlka bola príliš veľká. Tento problém je riešený nájdením zhľuku (klastra) z prvých 10 prienikov, na ktorý je následne aplikované nájdenie centroidu. Zhľuk je nájdený pomocou algoritmu na nájdenie K-najbližších susedov, nakoľko je implementovaný v OpenCV – CvKNearest.

Testovanie vykonávania požiadavky

Tento test sme zatiaľ neuskutočnili nakoľko stále vylepšujeme navrhnutý systém.

Záver

V dobe, keď sa robotom pomaly redukuje počet senzorov potrebných na ich efektívnu činnosť sme sa zamerali na orientáciu robota iba pomocou kamery. Inšpiroval nás pri tom kreatívny prístup účinkujúcich na svetových robotických súťažiach, ako aj sľubný vývoj v oblasti technológií.

Základom fungovania systému je dobrý vstup, čo sme dosiahli kvalitnou kalibráciou (získanie údajov z kamery). Nie len, že navrhnutý systém poskytuje spoľahlivý vstup. Zároveň sme sa zbavili závislosti na type, alebo veľkosti kamery, čo by pri už spomínanom sľubnom vývoji vo svete technológií mohlo priniesť prekvapivé vynálezy.

Druhým pilierom presnosti systému bolo mapovanie, ktoré v kombinácii ručne zmeraných rozmerov pavilónu I, testovacích videí a nášho algoritmu poskytovalo spoľahlivé výsledky.

Výhodou robota so vstupom z kamery je, že vďaka vstupnému obrazu sa vie spoľahlivo lokalizovať v akejkolvek časti trate bez toho, aby sme mu museli nechávať akékoľvek značky alebo pomocné predmety. Počas diskusie ohľadom budúceho vývoja sa objavila myšlienka implementovania nášho algoritmu do lietajúcich robotov v interiery aj exteriery, čo by práve vstup z kamery v kombinácií s naším systémom umožňoval, no bolo by potrebné spraviť celý prístup sofistikovanejším ohľadom na mapovanie nie pavilónu, ale povedzme zemského povrchu, resp. záchytných bodov.

Navyše internetové rozhranie zadávania požiadaviek plne napĺňa funkčné očakávania od robota pošťára.

Hlavné prínosy práce

Hlavným prínosom práce je určite jej zameranie sa na orientáciu robota v interiery, kde sú senzory iného druhu veľmi často nepresné, alebo nepostačujúce. Video vstup v kombinácií s efektívnym využitím SURF algoritmu, A* algoritmu, algoritmov na

rozpoznávanie obrazu, mapovanie, lokalizáciu a pod. je určite cestou, ktorou by sa mali moderné roboty uberať a sme radi, že sme si mohli v časoch, keď roboty začínajú naplňať aj funkciu kuriérov, presnejšie zanalyzovať aj takýto prístup k ich orientácii a využitiu.

Literatúra

1. Wei, Guo-Qing, Klaus Arbter, and Gerd Hirzinger. "Real-time visual servoing for laparoscopic surgery. Controlling robot motion with color image segmentation." *Engineering in Medicine and Biology Magazine, IEEE* 16.1 (1997): 40-45.
2. Yang, Simon X., and Chaomin Luo. "A neural network approach to complete coverage path planning." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 34.1 (2004): 718-724.
3. Svetlík, J., Sukop. M.: Robocar - mobilný robotický vozíkový systém. In: Acta Mechanica Slovaca. Roč. 12, č. 2-A (2008), s. 617-624. - ISSN 1335-2393
4. Nadhajský, Miroslav, and Pavel Petrovič. "Robotour Solution as a Learned Behavior Based on Artificial Neural Networks."
5. Nadhajský, Miroslav. Robotour (2011)
6. J. Šváb, T. Krajník, J. Faigl, and L. Přeučil. FPGA-based Speeded Up Robust Features. In 2009 IEEE International Conference on Technologies for Practical Robot Applications, pages 35{41, Boston, 2009. IEEE
7. Bay, Herbert, et al. "Speeded-up robust features (SURF)." *Computer vision and image understanding* 110.3 (2008): 346-359.
8. Lowe, David G.: Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision 2*. pp. 1150–1157. (1999)
9. Bradski, Gary, and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.
10. opencv dev team: *OpenCV 2.4.9.0 documentation* (2011-2014). <http://docs.opencv.org/>, [1.5. 2014]
11. HAAR, Alfréd: Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*. 1910, roč. 69, čís. 3, s. 331–371. ISSN 0025-5831

13. Krajník Tomáš, Chudoba Jan, Faigl Jan, Fišer Ondřej, Vonásek Vojtěch, Salvucci Valerio: 2009 SurfNav-GPU. <http://imr.felk.cvut.cz/robotour/> [1.4.2014]
14. PaedDr. Šimová, Elena: Množiny bodov s danou vlastnosťou. http://www.oskole.sk/?id_cat=50&clanok=17047 [1.5.2014]