VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH A REALIZACE VIZUÁLNÍHO LOKALIZAČNÍHO SYSTÉMU PRO AUTONOMNÍ MOBILNÍ ROBOT

DESIGN AND REALIZATION OF VISUAL LOCALIZATION SYSTEM FOR AUTONOMOUS MOBILE ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — SEBASTIÁN FILIP
AUTHOR

VEDOUCÍ PRÁCE — ING. MICHAL RŮŽIČKA
SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2014/15

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Sebastián Filip

který/která studuje v **bakalářském studijním programu**

obor: **Základy strojního inženýrství (2341R006)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Návrh a realizace vizuálního lokalizačního systému pro autonomní mobilní robot

v anglickém jazyce:

Design and realization of visual localization system for autonomous mobile robot

Stručná charakteristika problematiky úkolu:

Hlavním cílem práce je návrh a realizace vizuálního lokalizačního systému pro autonomní mobilní robot pohybující se ve vnitřních prostorách budov. Navržený systém musí být odolný vůči dynamickým překážkám v blízkosti robotu a musí zajišťovat jeho lokalizaci v předem určených oblastech. Práce je součástí vývoje většího projektu, který se touto problematikou zabývá.

Cíle bakalářské práce:

1) Seznamte se s dostupnými principy metod lokalizace ve vnitřních prostorách budov na základě zpracování obrazu.
2) Navrhněte metodu zpracování obrazu pro určení polohy a natočení robotu ve známém prostředí.
3) Metodu prakticky realizujte.
4) Proveďte praktické experimenty pro ověření funkce systému.

Seznam odborné literatury:

1) A. Davison. Real-time simultaneous localization and mapping with a single camera. In IEEE Conference on ICCV'03, 2003.
2) Simon, G., Fitzgibbon, A.W., Zisserman, A., Markerless Tracking Using Planar Structures in the Scene, Proceedings of International Symposium on Augmented Reality (ISAR 2000), pp. 120-128, 2000

Vedoucí bakalářské práce: Ing. Michal Růžička

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/15.

V Brně, dne 26.11.2014

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
Děkan

Abstrakt

Tato práce je zaměřena na návrh a realizaci vizuálního lokalizačního systému pro vnitřní prostory v předem známém prostředí. První dvě části jsou věnovány teoretickému úvodu do problematiky, a to lokalizaci a zpracování obrazu. Ve třetí části je popsán navrhovaný systém. Poslední dvě části jsou věnovány realizaci a testování navrženého systému.

Abstract

This thesis is focused on design and realization of indoor visual localization system in a known environment. The first two parts are dedicated to the theoretical introduction to the problem and cover localization and image proccesing. In the third part a designed system is described. The last two parts are dedicated to the system realization and its testing.

Klíčová slova

Lokalizace, zpracování obrazu. OpenCV.

Keywords

Localization, image proccesing, OpenCV.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Návrh a realizace vizuálního lokalizačního systému pro autonomní mobilní robot jsem vypracoval samostatně pod vedením ing. Michala Růžičky a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

V Brně dne 28. května 2015 ……..

FILIP, S. *Návrh a realizace vizuálního lokalizačního systému pro autonomní mobilní robot.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 54 s. Vedoucí bakalářské práce Ing. Michal Růžička.

Poděkování

Touto cestou bych rád poděkoval mým rodičům za velkou podporu ve studiu. Dále bych chtěl poděkovat vedoucímu mé bakalářské práce ing. Michalu Růžičkovi za jeho věnovaný čas a odbornou pomoc při řešení problému týkající se této bakalářské práce.

Obsah

1 Úvod

V dnešní době se velmi rozmáhá trend autonomních mobilních robotů, tedy robotů, které v reálném čase řídí software a pracují s určitou mírou autonomie. Pojem mobilní představuje možnost robota pohybovat se volně v prostoru. Využití připadá v úvahu především tam, kde náklady na výrobu a provoz robotu konkurují ceně v práci lidské, anebo v prostředí pro člověka špatně dostupných.

Pro správnou činnost takového robotu je zapotřebí mnoho systému. Jedním z nich je lokalizace, tedy nutnost vědět v jakém prostředí se bude pohybovat, a kde přesně se nachází. K tomu je robot vybaven lokalizačním systémem, který určuje polohu na základě dat získaných ze senzorů. Ve venkovních prostorech lze použít konveční metody jako je GPS *(Global Positioning System)*. Ve vnitřních prostorech jsou signály ze satelitů oslabovány a rozptylovány střechami, stěnami a dalšími objekty, díky kterým je výsledná pozice často dost nepřesná na to, aby byla vůbec použitelná. Proto jsou pro lokalizaci ve vnitřních prostorách vyvíjeny jiné způsoby. Jedním takovým je zpracování obrazu.

Lokalizace pomocí zpracování obrazu je obvykle založena na principu rozpoznání pasivních orientačních prvků, které jsou v prostoru umístěny na předem známých pozicích, tak aby byla zajištěna přímá viditelnost mezi prvkem a robotem. Vstupním signálem jsou obrazová data poskytnuté kamerou, v kterých se prvek vyhledává pomocí algoritmů vytvořených pro práci s těmito daty. Cílem moji bakalářské práce je seznámit se s danou metodou lokalizace a možnostmi pro práci s obrazovými daty a na základě nabytých informací následně navrhnout a realizovat takový lokalizační systém pro autonomní mobilní robot a otestovat jeho funkčnost a spolehlivost.

2 Lokalizace

Lokalizace je jedním z fundamentálních problémů mobilní robotiky a měla by bezpečně zajistit určování polohy robota. Je nutná k autonomnímu chování mobilních robotů a základ pro řešení dalších robotických problémů jako je navigace a plánování trasy. K určení polohy se používají informace získané ze senzorů robota.

2.1 Senzory

Senzory jsou zařízení, které poskytují využitelný signál odpovídající měřené veličině. Podle výstupních veličin se dají dělit na optické a elektrické, ty se dále dělí podle fyzikálního principu na odporové, indukčnostní, kapacitní aj. Na trhu se nachází široké spektrum různých senzorů. V následující podkapitole bych rád stručně shrnul nejpoužívanější senzory pro lokalizaci.

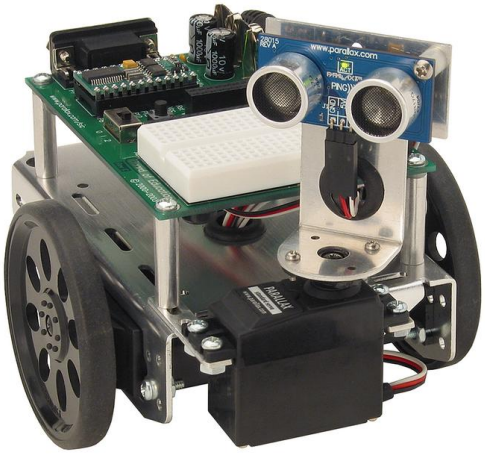
2.1.1 Infračervené senzory pro měření vzdáleností

Jsou schopny detekovat překážky a měřit vzdálenosti k nim. Skládají se z infračerveného vysílače, který vysílá pulzní modulaci infračerveného světla. Druhou součástí je infračervený příjmač. Měření vzdáleností od překážky je provedeno pomocí principu optické triangulace, což je řešení obecného trojúhelníka, u kterého známe délku jedné strany a k ní přilehlého úhlu. Ukázku infračerveného senzoru je možno vidět na obr.1. [2]

Obr. 1 Infračervený senzor [3]

2.1.2 Ultrazvukové sonary pro měření vzdáleností

Jsou schopny měřit vzdálenosti na principu odrážení mechanických kmitů (zvukových vln). Podobně fungují radary, akorát místo mechanických kmitů jsou zde elektromagnetické kmity (rádiové vlny). Doba je závislá na prostředí v jakém se signál pohybuje. Sonary jsou relativně levné a jednoduché na zpracování signálu, jejich nevýhodou je poměrně krátký dosah a možný výskyt nežádoucích odrazů. Využití nacházejí zejména ve venkovní navigaci (např. námořnictví), ve zdravotnictví a pro detekci překážek (obr. 2).



Obr. 2 Sonar umístěný na mobilním robotovi [4]

2.1.3 Kamery

Jsou elektronické přístroje schopné zachytit více snímku v krátkém časovém úseku. Obrazová data jsou v podobě dvourozměrných matic pixelů, které představují nejmenší body obrazu. Pixely jsou definované podle charakteru jejich barevného formátu (RGB, HSV, CMYK). Pro šedotónový obraz postačí k definici pixelu pouze jeden parametr (jas). Pro určení barvy je třeba parametrů více, např. u barevného modelu RGB je výsledná barva pixelu reprezentována třemi barevnými složkami (červenou, zelenou, modrou).

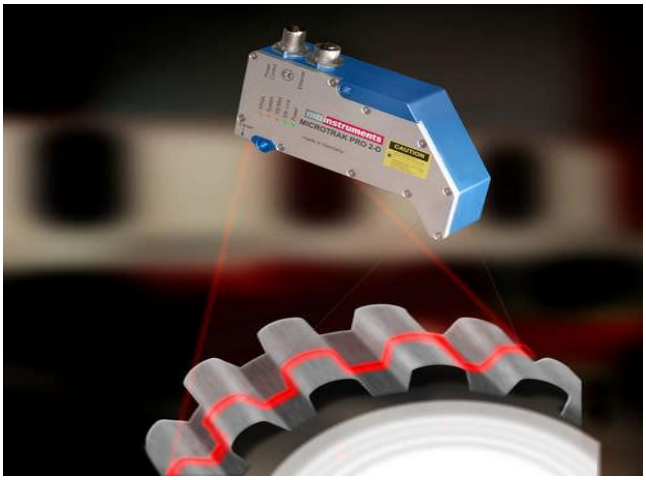
Jeden ze základních parametrů kamer je rozlišení, to se udává buď jako počet pixelů, které tvoří sloupce a řádky matice pixelů např. 1024x768 nebo pomocí megapixelů (MPx), což je počet pixelů, které je kamera schopna zachytit. Jeden MPx odpovídá 2^{20} pixelů, takže například rozlišení 1024x768 je přesně 0,75 MPx. Kamery jsou v dnešní době velmi rozšířené a bývají často součásti fotoaparátů i mobilních telefonů. Zajímavým produktem jsou 3D kamery (obr. 3), které mají své využití i v průmyslu, protože dokážou zpracovat profil výrobku. [7]

Obr. 3 3D kamera JVCGS-TD1 jedna z prvních na trhu
se dvěma objektivy a dvěma záznamovými čipy [8]

2.1.4 Laserové senzory pro měření vzdáleností

Laser funguje na principu stimulované emise fotonů, které jsou vyzařovány z elektronů v metastabilní hladině. Laserové senzory měří dobu letu vysílaného paprsku, který se pohybuje rychlostí světla. Použiti nacházejí pro bezkontaktní přesná měření (obr 4).

Rozšířením bodových laserů jsou laserové skenery, které jsou schopné vyprodukovat velké množství dat v krátkém časovém úseku. Nasnímaný objekt je obvykle zobrazen ve formě bodového mračna. Laserové skenery jsou poměrně drahé zařízení, nejenom díky složitého fyzikálního principu, ale i z důvodu, že musí být opatřeny velmi přesnými měřícími zařízeními. Jejich využití je kromě 3D modelování a vizualizaci také např. i v kalibrování a kontrole přesnosti rotačních os obráběcích strojů.



Obr. 4 The Microtrak Pro 2D Laser Displacement Senzor
Ukázka využití 2D skeneru pro kontrolu přesnosti ozubeného profilu [5]

2.2 Metody určování polohy mobilního robotu

K určování polohy mobilního robotu lze použít širokou škálu lokalizačních technik. Vhodná metoda záleží na dané aplikaci mobilního robota a na okolních podmínkách prostředí. Podle těchto kritérií se dá lokalizace rozlišovat na:

- **Relativní a absolutní lokalizace**
  Rozdělení podle charakteru měření. Prostředky relativní lokalizace měří nebo odhadují danou aktuální pozici vůči počáteční poloze. Většinou pomocí posunutí a natočení robotu. Naopak absolutní lokalizace umožňují zjistit nebo odhadnout danou polohu bez návaznosti na počáteční polohu a předchozí stavy robotu. Obvyklé je, že robot má dané prostředí zmapované a jeho poloha je dána hustotou pravděpodobnosti. Aby se dosáhlo uspokojivější výsledků, tak se v praxi velmi často tyto metody kombinují. [13]

- **Lokalizace ve statickém a dynamickém prostředí**
  Charakterizuje prostředí, ve kterém se robot pohybuje. V dynamickém se můžou pohybovat další objekty nebo měnit parametry prostředí jako např. intenzita osvětlení. Statické prostředí je ideálně neměnné a na libovolné jedné pozici budou příjmané signály v čase konstantní. [13]

2.2.1 Odometrie

Je metoda zjišťující relativní lokalizaci, která se používá u mobilních robotů nebo vozidel. Principiálně jde o odhad změny pozice a orientace prostřednictvím naměřených údajů o natočení jeho hnacích nebo běžných kol pomocí rotačních enkodérů. Nutnou podmínku pro fungování odometrie je schopnost vypočítat celkový pohyb robota z natočení kol. Nevýhoda této metody je, že nedokáže zjistit informace o svoji původní poloze. Dále jsou výpočty zatížené chybami způsobené například nedokonalostí kruhového průřezu kola, rozdíl v obvodu kol jedné nápravy atd. Tyto chyby mají kumulativní charakter a dá se s nimi při výpočtu počítat. Horší jsou takzvané nesystematické chyby, jako prokluz jednoho z kol, nerovnost povrchu a podobně. Tyto chyby se dají jen velmi složitě předpovídat. Odometrie je poměrně levná metoda, a proto je pro přiměřeně rovné povrchy v praxi nejpoužívanější metodou relativní lokalizace. [6]

2.2.2 Lokalizace na základě umělých orientačních bodů

Umělé orientační body jsou takové, které nějakým způsobem modifikují prostředí, ve kterém má probíhat lokalizace. Příkladem je rozmístěný majáku v operačním prostoru mobilního robota. Majáky představují aktivní orientační prvky a slouží buď jako příjmače nebo vysílače signálů, které jsou následně vyhodnocovány k určení

absolutní polohy robota. Nevýhoda je nutnost údržby a napájení majáků. K bodové lokalizaci je zapotřebí obvykle 3 majáků, ze kterých se triangulací nebo trilaterací určuje poloha.

Jsou-li takovými majáky satelity na oběžné dráze Země, tak se jedná o lokalizaci globální, která umožňuje absolutní lokalizaci na jakémkoli místě pokryté satelitním signálem. Jelikož jsou satelity daleko od příjmače je třeba k bodové lokalizaci přidat 4. maják, který dopočítává přesný čas. Asi nejznámějším takovým systémem je GPS, jehož obvyklá přesnost se v dnešní době pohybuje okolo pěti metrů.

Pro aplikace, kde je třeba přesnější určení polohy, nebo které jsou ve vnitřních prostorách je využíváno lokálních lokalizačních systémů. Ty mohou být principiálně různých druhů např. infračervené lokalizační systémy. Jedním z takových systémů je Virtual Wall Lighthouse, který využívají robotické vysavače Roomba od firmy iRobot. Princip této technologie spočívá ve vytvoření virtuálních zdí pomocí infračervených vysílačů (obr. 5), ty vymezují oblast, v které má robot operovat a zároveň ho dokážou v ní lokalizovat. [14,2]

Obr. 5 Vysavač Roomba využívající technologii virtuálních zdí [15]

2.2.3 Lokalizace založena na zpracování obrazu

Lokalizace pomocí zpracování obrazu řeší zpravidla absolutní lokalizaci pomocí detekce pasivních orientačních bodů v obrazových datech. Pojmem pasivní se rozumí, že body nevysílají ani nepřijímají žádný signál. Tyto body bývají rozmístěny na předem známých místech a tvoří pomyslnou mapu, ve které se robot může lokalizovat.

Podle charakteru se dají dělit na umělé a přirozené. Umělé modifikují pracovní prostředí a jsou navržené tak, aby byly dobře rozpoznatelné např. čárové kódy nebo kontrastní geometrické obrazce. Přirozené jsou už součástí daného prostředí např. dveře, zásuvky, rohy místnosti atd. Tyto body jsou pochopitelně náročnější na implementaci softwaru. Jejich hlavní výhodou je, že není třeba modifikovat pracovní prostředí.

Při řešení problému lokalizace musí být obrazová data zpracovávány v reálném čase. To bývá většinou výpočtově náročné kvůli charakteru obrazových dat. Ke správné činnosti je tedy zapotřebí výkonných výpočetních systémů, které jsou poměrně drahé, což má za následek časté zvolení jiné metody absolutní lokalizace. Hlouběji o tomto tématu pojednává literatura [1].

3 Zpracování obrazu

Zpracování obrazu je souhrn operací prováděných na obrazových datech pro dosažení určitého cíle. Vstupními parametry jsou obrazová data v digitálním formátu. Mezi základní operace zpracování obrazu patří: bodové transformace, lineární filtry, detekce hran a další. Kombinováním těchto operací a vytváření různých algoritmů lze vytvořit aplikace, jako jsou například: detekce objektů na vstupních obrazových datech okem nepozorovatelných, vyhledávání oblastí zájmů v obrazových datech (např. zpracování otisků prstů) nebo hledaní specifického prvku přímo v obrazu, což jak zmíněno v části 2.2.3 může být použito pro řešení problému lokalizace. V dnešní době existuje mnoho různých způsobů, jak obraz zpracovávat. V následujících podkapitolách bude přiblíženo několik známějších z nich.

3.1 Zpracování obrazů s využitím SDK

Zkratka SDK pochází z anglického *Software Development Kit*. Většinou to bývají nástroje nebo knihovny rozšiřující rozhraní daného programovacího jazyka. Příkladem SDK jsou knihovny obsahující balíčky funkcí a algoritmů pro práci s obrazovými daty. V následující podkapitole budou ty nejznámější z nich stručně popsány.

3.1.1 IPP

Je rozsáhlá knihovna od firmy Intel, která se zabývá výrobou procesorů. Název vznikl z anglického *Integrated Performance Primitives*. Hlavní výhodou této knihovny je bezkonkurenční rychlost díky vysoce optimalizované technologii, která podporuje procesory Intel řady i3 a vyšší. Knihovna obsahuje funkce pro zpracování dat a vývoj multimédií. Toto SDK má rozsáhlou dokumentaci, což usnadňuje rychlou orientaci ve funkcích knihovny. Nevýhodou je vysoká cena.[17]

3.1.2 OpenCV

Je knihovna funkcí zaměřená na zpracování obrazu v reálném čase. Název pochází z *Open Source Computer Vision*. OpenCV podporuje rozhraní jazyků C++,C, Python a Java a je kompatibilní pro operační systémy Windows, Linux, Mac OS, iOS a Android. Některé metody této knihovny jsou připraveny na běh s CUDA architekturou (*Compute Unified Device Architecture*), což je technologie od výrobce grafických karet NVIDIA, která urychluje výpočty díky využívání grafických výpočtových jednotek GPU (*Graphics Processing Units*). OpenCV má velice rychlý vývoj otevřených zdrojových projektů a zároveň je z velké části bezplatná, což má za následek fakt, že je jedna z nejrychleji se rozvíjejících knihoven a v dnešní době představuje nejpopulárnější knihovnu pro práci s obrazovými daty, a to i přes svoji slabší dokumentaci. [11]

3.1.3 EmguCV

Je volně šiřitelná knihovna pro zpracování obrazu v reálném čase. Vychází z knihovny OpenCV, jejíž je wrapper a rozšiřuje používání jejích funkcí pro programovací jazyky C#, VB, VC++ a IronPython. Podobně jako OpenCV je kompatibilní pro operační systémy Windows, Linux, Mac OS X, iOS, Android a Windows Phone. Některé metody této knihovny rovněž provádějí výpočty pomocí grafických karet s CUDA architekturou. Nevýhoda je možná nepřehlednost datových typů a pomalejší výpočtová rychlost, díky vytváření kódů ve vyšších programovacích jazycích. Na druhou stranu jsou tyto programovací jazyky obecně uživatelsky přívětivější. [16]

3.1.4 JavaCV

Je rozhraní, které poskytuje přístup z běžně používaných knihoven přímo z JVM *(Java Virtual Machine).* Mezi tyto knihovny patří OpenCV, OpenKinect, videoInput, ARToolKitPlus a další knihovny počítačových vidění. JavaCV je bezplatná a multiplatformní s kompatibilitou systémů Mac OS X, Windows a Linux. Jediným podporovaným programovacím jazykem je Java, která dosahuje nižších výpočtových rychlostí než nativní jazyky jako C++.

3.2 Zpracování obrazu v prostředí MATLAB

MATLAB je výkonné programovací prostředí určené pro technické programování. Vyniká uživatelsky přívětivým prostředím, kde problémy a řešení jsou vyjádřeny známými matematickými notacemi a základním datovým elementem je matice. Odtud vznikl i název maticová laboratoř *(MATrix LABoratory).* Typické použití zahrnuje matematické výpočty, rozvoj algoritmů, zpracování dat, modelování, simulace a mnohé další, mezi kterýma je i vizualizace a zpracování obrazu. Zpracování obrazu v MATLABU je možné pouze s Image Processing Toolboxem, což je balíček funkcí obsahující základní algoritmy a aplikace pro analýzu a vizualizaci obrazu. Obrovskou nevýhodou je pomalá výpočetní rychlost a poměrně vysoká cena, která se pro jednoho uživatele pohybuje kolem $150. [9]

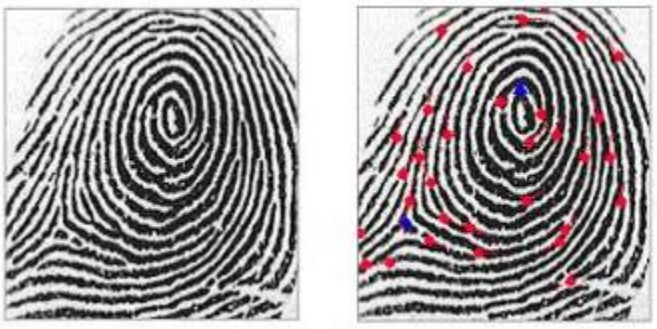
3.3 Zpracování obrazu v praxi

V následujících podkapitolách bude uvedeno a stručně popsáno několik typických příkladů použití zpracování obrazu v praxi.

3.3.1 Zpracování otisků prstů

Verifikace otisků prstů je nejrozšířenější metoda identifikace osoby pomocí biometrických znaků člověka. Spolehlivé a rychlé určení je vyžadováno jak pro osobní, tak i pro komerční účely. Pomocí zpracování obrazu se zautomatizovalo porovnání otisků prstů i jejich analýza.

Nejrozšířenější metoda porovnání otisků prstů je pomocí detekce markantů (obr. 6), což jsou charakteristické znaky papilárních linií prstů. Mezi markanty patři např. začátek, konec, vidlice aj. Tyto znaky jsou v obrazu vyhledány pomocí sofistikovaných algoritmů a následně uloženy jejich souřadnice, typ a orientace. Dalšími metodami je následně hledána maximální shoda mezi vstupní a referenční sadou markantů.



Obr. 6 Analýza otisku prstu. Vlevo je vstupní snímek a vpravo jsou nalezené markanty pomocí zpracování obrazu.[18]

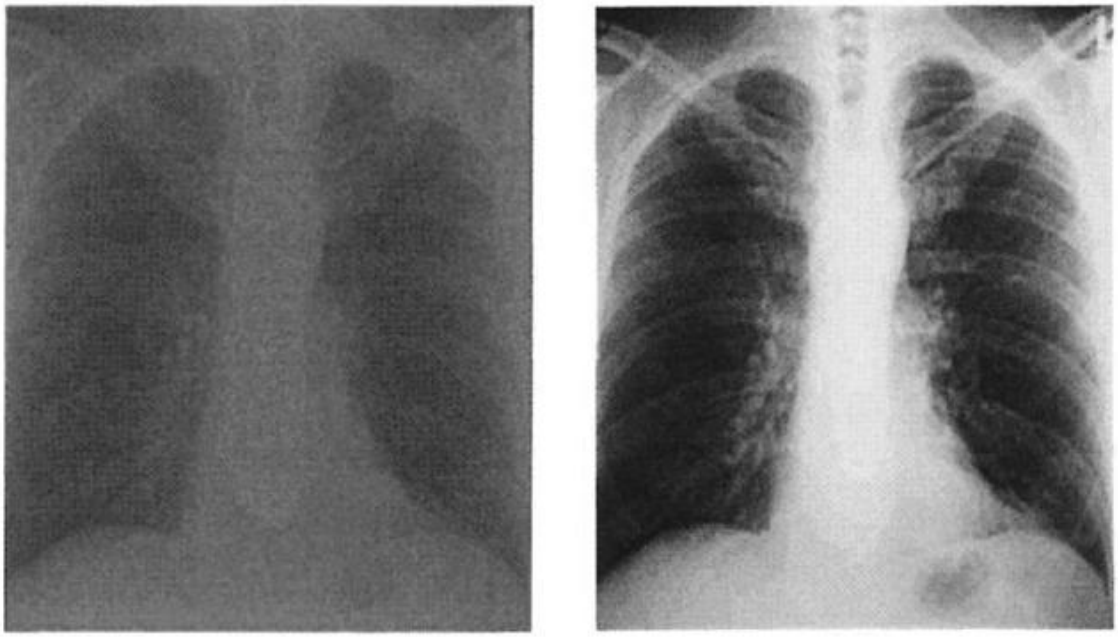
3.3.2 Detekce vozidel projíždějící na červenou

Je jedním z mnoha dopravních aplikací, kde si zpracování obrazu našlo svoje využití. Výhoda takového systému je jeho automatický provoz tj. zpracování a sběr dat v reálném čase s následnou prokazatelností o přestupku. Jeden takový systém byl vyvinut v České republice a je vyráběn firmou CAMEA, spol. s.r.o. a nese název UnicamREDLIGHT. V současnosti se tímto systémem monitoruje více než 80 jízdních pruhů po celé ČR.

Systém se skládá ze dvou typů kamer, z nichž první je přehledová, která je naprogramovaná tak, aby dokázala zachytit fázi signalizačního zařízení (semaforu), díky tomu není třeba, aby byl systém propojený s řadičem křižovatky, a tím je ulehčena instalace a údržba. Dalším úkolem přehledové kamery je zachytit přestupek (vjezd vozidla do křižovatky na červenou). Druhým typem je detailová kamera, která zabírá právě jeden jízdní pruh na druhé straně křižovatky a v případě přestupku „přečte“ SPZ auta a udělá snímek auta i s obličejem řidiče. Software pracuje v reálném čase a SPZ je k dispozici ihned po detekci přestupku. [19]

3.3.3 Zpracování obrazu v medicíně

Velké využití našlo zpracování obrazu v medicíně. Příkladem je například počítačová tomograie (CT), magnetické rezonance (MR), rentgen, ultrazvuk aj. Výstupem těchto lékařských vyšetřovacích metod jsou obrazová data, která je nutno často modifikovat, aby byly lépe pozorovatelné výsledky. Většina medicínských obrazů je v odstínech šedi, jedná se tedy o jednokanálovou matici pixelů. Časté zpracování spočívá na základě práce s histogramem, který zobrazuje četnost pixelů o dané intenzitě. Intenzita je totiž často závislá na fyzikálních vlastnostech látek (např. hustotě tkáně). Každá z metod vyžaduje samozřejmě vlastní přístup pro analýzu (obr. 7). [20]



Obr. 7 Zlepšení rentgenového snímku pomocí úpravy kontrastu [21]

4 Návrh řešení

Cílem moji bakalářské práce je navrhnout a realizovat lokalizační systém pro autonomní mobilní robot, který má řešit problém pomocí zpracování obrazu. Vstupními daty tedy budou obrazová data pořízené kamerou umístěnou na robotovi. Následně by měl být vytvořený systém otestován a vyhodnocen.

Prvním krokem bylo zvolení softwarového nástroje pro implementaci systému. Po nastudování možných způsobů pro zpracování obrazu (viz kapitola 2) jsem si zvolil využití programovacího jazyka C++, který je nejlepší volbou z hlediska výpočtové rychlosti, která je základním předpokladem pro zpracování obrazu v reálném čase. Jako SDK byla zvolena knihovna OpenCV. Hlavním kritériem při výběru byl fakt, že knihovna OpenCV je primárně určena právě pro zpracování obrazu v reálném čase a obsahuje spoustu sofistikovaných funkcí a algoritmů.

Dalším krokem byl samotný návrh lokalizace. Pro ten byl zvolen přístup detekce pasivních orientačních bodů. Pro rychlejší výpočet jsou ideální body umělé, které budou rozmístěny v operačním prostoru robota na předem určených pozicích. Podoba bodů neboli značek, jak budou dále označovány, by měla být jednoduchá a výrazná, aby byla zajištěna jejich snadnější detekce. Přímo se tedy nabízí použití kontrastních geometrických obrazců. Ukázka možné podoby takové značky je na obr. 8. Počet navrhovaných značek je závislý na velikosti daného prostoru, ve kterém má probíhat lokalizace. Pro tuto práci je uvažovaná středně velká místnost a jejich počet by tedy měl být v řádu desítek. K tomuto předpokladu je tedy třeba přizpůsobit tvary a barvy značek, protože každá značka by měla být unikátní a měla by jí odpovídat právě jedna specifická pozice. Umístění značek je zvoleno ve stropním prostoru místnosti ze dvou důvodů, z nichž první je fakt, že stropní omítky bývají barevně jednolité, což je příhodné pro zvětšení kontrastu mezi značkou a pozadím. Druhým důvodem je, že lícní strana značek bude téměř vždy kolmá na objektiv kamery, to má za následek menší zkreslení samotné značky a lehčí výpočet dané pozice.

Obr. 8 Možná podoba značky
(pasivního umělého orientačního prvku)

4.1 Návrh algoritmu pro detekci značek

V této podkapitole bude stručně popsán navrhovaný algoritmus zpracování obrazu, na konci kterého by měla být úspěšně detekována značka ve vstupním obraze a známé její parametry jako je typ, pozice těžiště a barva.

Pro řešení problému detekce značky jsem si zvolil dvě metody, a to z důvodu zvýšené spolehlivosti. Když jedna metoda najde pouze část značky a druhá metoda na stejném místě detekuje tutéž část a k tomu navíc třeba zbytek značky, tak se s celkem velkou pravděpodobností dá tvrdit, že se značka nachází na tomto místě. První zvolenou metodou je separace barev a druhou je detekce hran. Obě budou podrobněji představeny v následujících podkapitolách a oběma bude předcházet předzpracování obrazu pomocí morfologických operací, kterými by se měl minimalizovat šum na vstupním obrazu.

4.1.1 Detekce hran

Hrany jsou dány vlastnostmi daného pixelu a jeho okolí. Z matematického hlediska je obraz diskrétní jasová funkce a hrany se dají určit diferencí pixelu a jeho okolí, kde náhlá změna hodnoty obvykle odpovídá hraně. Pro tuto metodu je zásadní tzv. prahování, ve kterém je třeba rozhodnout, jak moc velká změna má odpovídat hraně a jak malá změna nemá být vyhodnocena jako hrana. Špatné nastavení prahu by mohlo mít za následek vyhodnocení šumu jako hrany nebo vynechání podstatných hran.

4.1.2 Detekce barev

Podmínkou tohoto přístupu je možnost detekovat určitý rozsah barevného spektra. Každý pixel je definovaný právě jednou barvou v daném barevném formátu, z nichž nejznámější je barevný model RGB, kde je barva rozdělena do tří složek na červenou, zelenou a modrou o různé intenzitě. Různými kombinacemi zastoupení těchto tří barev lze vytvořit celé barevné spektrum. Při změně intenzity osvětlení dochází ke změně hodnot intenzity barev daného pixelu, např. jasně červenou můžeme v šeru pozorovat jako rudou, proto je třeba hledat danou barvu v určitém intervalu. Čím bude kontrast mezi hledanou barvou a pozadím větší, tím větší může být interval, což má za následek zvětšení pravděpodobnosti, že bude daná barva detekována. Výhodou detekce barev oproti detekci hran je, že lze ke detekované oblasti připojit informaci o její barvě.

4.1.3 Detekce geometrických tvarů

Výstupem z výše zmíněných metod však ještě není samotná značka, ale pouze binární (černobíle) obrazy vzniklé prahováním, kde bílá odpovídá hledané oblasti zájmů (barvě nebo hraně). Pro detekci geometrických tvarů je tedy třeba dalších operací. Mezi tyto operace patří nalezení kontur z binárních obrazců, což jsou hranice, které definují dané oblasti zájmů. Smysl vytvoření kontur spočívá ve snadnější práci s nalezenými oblastmi, protože kontura je tvořena vektorem bodů na rozdíl od maticové povahy

binárních obrazců. Počet vytvořených kontur je závislý na tvaru a počtu nalezených oblastí.

Dalším krokem je práce se samotnými konturami. Je předpokládáno, že značka bude relativně velká vůči rozměrům snímku kamery, takže kontury, které budou hodně malé, představují pouze nějaký šum nebo jiné objekty, které nejsou oblasti zájmu. Proto budou tyto kontury vyfiltrovány a ponechány budou jenom ty s větším rozlohou.

Po vyfiltrování vektoru kontur je následující operací samotná detekce geometrických tvarů, ale díky šumu nebo nekvalitnímu vstupnímu obrazu se může např. čtverec jevit jako n-hranou konturou místo čtyř-hranné. Proto je třeba dané kontury aproximovat jinými, a tím zredukovat tyto možné odchylky. Následně se můžou konturám podle počtu vrcholů přiřazovat geometrické tvary. Tyto jednotlivé kontury je třeba vyhodnotit, zdali jsou opravdu hledanými geometrickými tvary a případně je uložit, aby byly jednoduše dostupné pro další práci.

4.1.4 Třídění kontur

Jako vstupní data jsou k dispozici uložené kontury, které mají podobu hledaných geometrických tvarů. Nyní je třeba zjistit, jestli se shodují některé kontury, které byly nalezeny metodou 1 (detekcí hran) a metodou 2 (detekcí barev) a případné shody spárovat. Je třeba si uvědomit, že i když daná kontura z každé metody reprezentuje tentýž objekt na snímku, tak nebude stejná, díky předchozím operacím se zkresluje její tvar a tudíž i její parametry. Třídící algoritmus musí tedy projít jednotlivé kontury a zkontrolovat, jsou-li těžiště blízko u sebe, plochy přibližně stejné a samozřejmě zdali se shoduje geometrický tvar. V případě, že se najde shodná kontura z obou metod, tak se dá tvrdit s celkem velkou pravděpodobností, že objekt reprezentovaný těmito konturami se skutečně na daném místě nachází.

4.1.5 Hledání značky

Posledním krokem je samotný návrh algoritmu, který z databáze značek bude hledat shodu s vstupními daty (nalezené páry kontur). Tato úloha je závislá na konkrétní podobě značek. Na úkor spolehlivosti detekce lze navrhnout, aby algoritmus částečně pracoval i s nespárovanými konturami. Následným výstupem by měly být souřadnice nalezených značek, které budou reprezentovány jejich těžišti.

4.2 Lokalizace

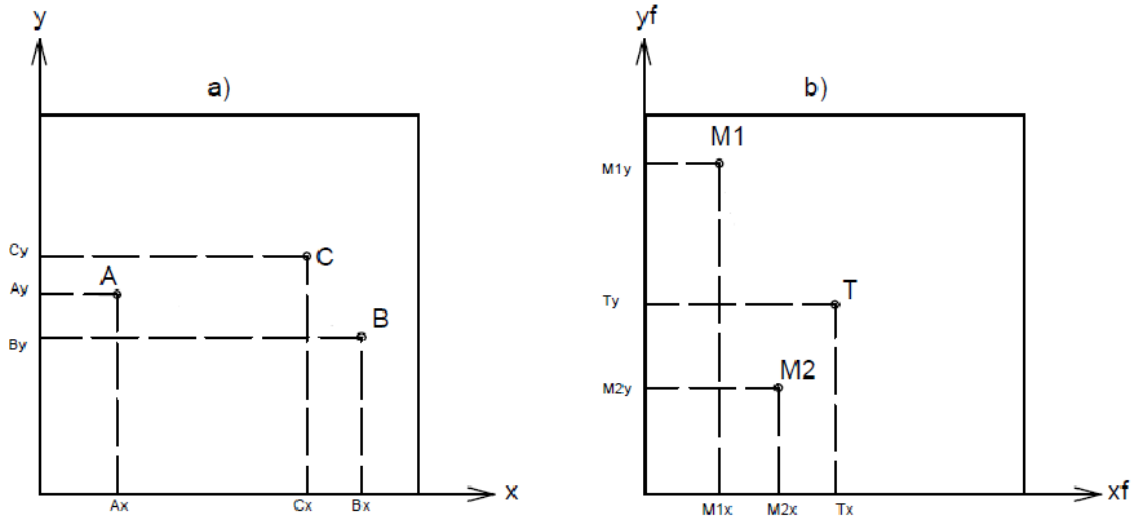
Pro určení polohy je třeba vytvořit ze značek v daném operačním prostoru robota pomyslnou mapu. Mapa bude mít tvar půdorysu místnosti a každé místo na ní bude odpovídat souřadnicím [x , y]. Těžiště každé značky *i* bude přirazen právě jeden bod $[x_i , y_i]$ v této mapě. V následujících podkapitolách bude navrženo řešení pro výpočet polohy a natočení.

4.2.1 Určení polohy

Vstupními daty pro výpočet jsou pozice těžišť detekovaných značek v obrazových datech *[Mix , Miy]*. Tyto souřadnice odpovídají souřadnému systému snímku. Dalším známým parametrem je střed obrazu *T[Tx , Ty]*, který odpovídá momentální poloze robotu (vyplývá z předpokladu, že objektiv je kolmý na stěnu).

Problém je znázorněn na obr. 9, kde 9a přestavuje referenční mapu s osami *x*, *y* a body *A*, *B* odpovídají poloze značek ve vytvořené mapě. Jednotka této osy potom je v metrech nebo v jiných délkových jednotkách. Obr. 9b znázorňuje snímek kamery a jeho obrazová data s osami y_f a x_f, které tvoří souřadný systém snímku. Jednotky os snímku představují jednotlivé pixely a bod *T* představuje střed snímku kamery. Body M_1 a M_2 odpovídají těžištím nalezených značek *A* a *B* právě v tomhle pořadí.

Cílem je najít k bodu *T* bod *C[Cx , Cy]* v referenční mapě. Souřadnice tohoto bodu *C* potom odpovídají hledané poloze. Pro jednoznačné určení pozice je nutno znát alespoň dvě detekované značky, jak je znázorněno v obrázku 9. Jedna detekovaná značka může totiž poskytnout pouze hrubý odhad pozice, principiálně jde o to, že zdali je značka *i* detekována a robot ví, kde přesně se tato značka nachází (souřadnice $[x_i , y_i]$), tak se musí také on nacházet v blízkosti této značky *i*. Speciálním případem je, když se jediná detekovaná značka nachází právě na místě středu snímku *T*. Pro tento případ, by byla pozice *C* rovná pozici odpovídající značky v referenční mapě. Pro obecné řešení je však nutno počítat se dvěma nalezenými značkami.



Obr. 9 a) Znázornění referenční mapy se souřadným systémem
b) Znázornění snímku kamery se souřadným systémem

Jedno z možných řešení tohoto problému je skrze řešení podobnosti trojúhelníků Δ ABC a Δ M_1M_2T. Poměry délek stran trojúhelníků se rovnají podobnostní konstantě k, která může být vyjádřena následovně (1):

$$k = |AB| / |M_1M_2| = |BC| / |M_2T| = |CA| / |TM_1| \quad (1)$$

Délku úseček mezi známými body lze snadno vyjádřit pomocí Eukleidovské vzdálenosti. Pro vzdálenost $|M_1M_2|$ platí vztah (2):

$$|M_1M_2| = \sqrt{(M_1x - M_2x)^2 + (M_1y - M_2y)^2} \quad (2)$$

Podobným způsobem lze spočítat i ostatní vzdálenosti známých bodů. Vyjádřením délek $|AB|$ a $|M_1M_2|$ lze dopočítat podobnostní konstantu k z rovnice (1), z které lze vyjádřit vztahy na spočítání vzdáleností bodů A a B k bodu C (3) (4) následovně:

$$|AC| = |M_1T| \times k \quad (3)$$

$$|BC| = |M_2T| \times k \quad (4)$$

Nalezení bodu C se dá vyřešit numerickým výpočtem průniků dvou kružnic k_1 a k_2 (viz obr. 10a), kde $r_1 = |AC|$ a $r_2 = |BC|$. Tato úloha bude mít dvě řešení, která jsou označena C_1 a C_2. Problém je znázorněn na obr.10a. Bod S rozděluje úsečku AB, jejíž délka je označena d, na dvě úsečky AS a SB. Úsečka AS je označena jako e a úsečku SB jako f, přičemž platí: $d = e + f$. Bod S je situován tak, aby vzniklé trojúhelníky Δ ASC_1 (obr. 10b) a Δ ASC_2 byly oba pravoúhlé. Vzdálenost $|SC_1|$ se pak rovná $|SC_2|$ a obě tvoří výšky v Δ ASC_{12} , proto je úsečka $|SC_1|$ označena v. Z Pythagorovy věty pak platí vztahy pro Δ ASC_1 (4) a pro Δ BSC_1 (5).

$$r_1^2 = e^2 + v^2 \quad (4)$$

$$r_2^2 = f^2 + v^2 \quad (5)$$

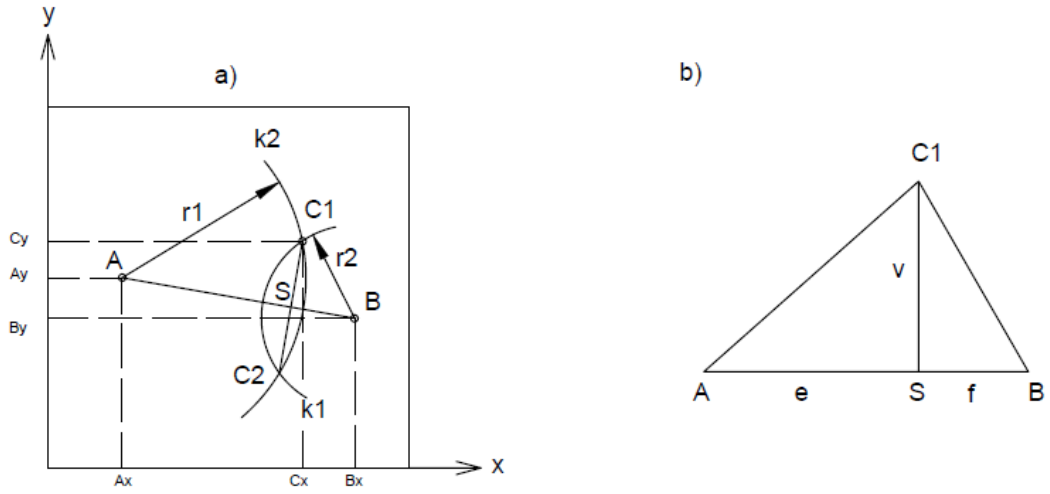
Po odečtení a následné úpravě těchto rovnic se s využitím vztahu $d = e + f$ dá vyjádřit vztah pro velikost f (6)

$$f = \frac{r_1^2 - r_2^2}{2d} + \frac{d}{2} \quad (6)$$

Výška v trojúhelníku se pak vyjádří vztahem (7) :

$$v = \sqrt{r_1^2 - f} \quad (7)$$

Podmínkou použitelnosti této metody je, že Δ ABC musí být ostrý, to znamená že všechny vrcholové úhly musí být menší než 90 °.



Obr. 10 a) Řešení problému pomocí průniků kružnic
b) Pomocný výpočtový trojúhelník

Lineární interpolací lze spočítat souřadnice bodu S[Sx,Sy] (8) (9).

$$Sx=Ax+(f/d)\times(Bx-Ax) \tag{8}$$

$$Sy=Ay+(f/d)\times(By-Ay) \tag{9}$$

Z analytické geometrie se dá vyjádřit vztah pro nalezení bodů C_1 a C_2 (10) (11).

$$Cx_{12}=Sx \pm (v/d) \times (Ay-By) \tag{10}$$

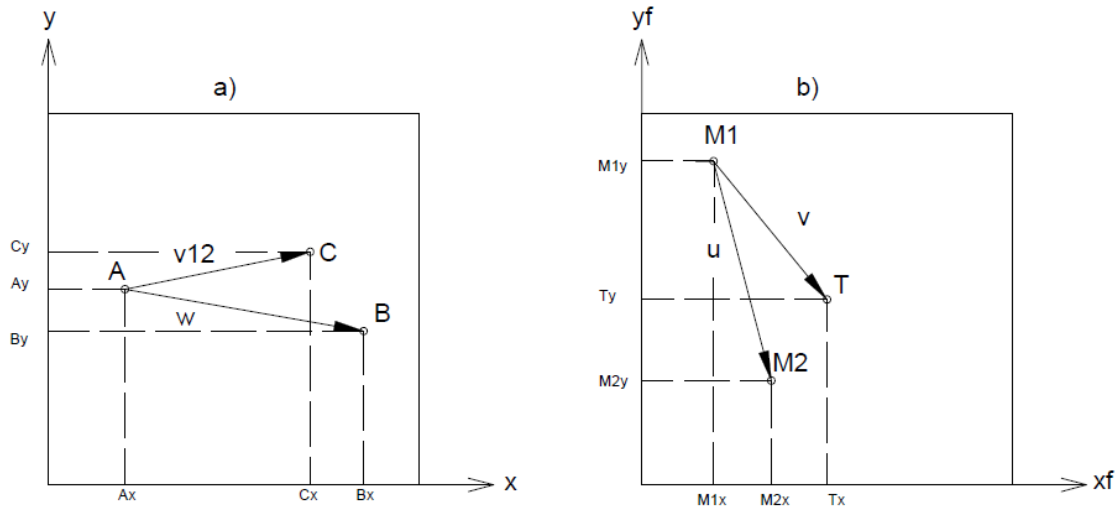
$$Cy_{12}=Sx \pm (v/d) \times (-Ax+Bx) \tag{11}$$

Nyní jsou známy souřadnice obou průsečíků kružnic k_1 a k_2, je třeba však rozhodnout, který je hledanou polohou C robotu. K tomu lze použít vektorového součinu. Jelikož všechny body, které tu byly zmíněny, leží v rovině *xy*, tak jejich vektorový součin je vektorem se souřadnicí v ose *z* (pozn.: výsledek vektorového součinu je vektor kolmý na oba tyto vektory). Pro jednotkový vektor *k* této osy *z* platí vztah (12) za předpokladu, že vektory *u,v* leží v rovině *xy*.

$$k = u_x v_y - v_x u_y \tag{12}$$

Vektory v souřadném systému snímku jsou označeny dle obr. 11b následovně: $u=M_1M_2$, $v=M_1T$ a vektory v referenční mapě dle obr. 11a jsou: $w=AB$, $v_{12}=AC_{12}$. Správný bod *C* bude stejně orientován k vektoru *w* jako je vektor *u* k vektoru *v*. Matematicky vyjdou velikosti jednotkových vektorů *k* vektorového součinů $(u\times v)$ a $(w\times v_{12})$ buď oba záporně nebo kladně. Proto lze psát podmínku výběru správné pozice C (13). Správný bod C bude takový, který splňuje tuto podmínku.

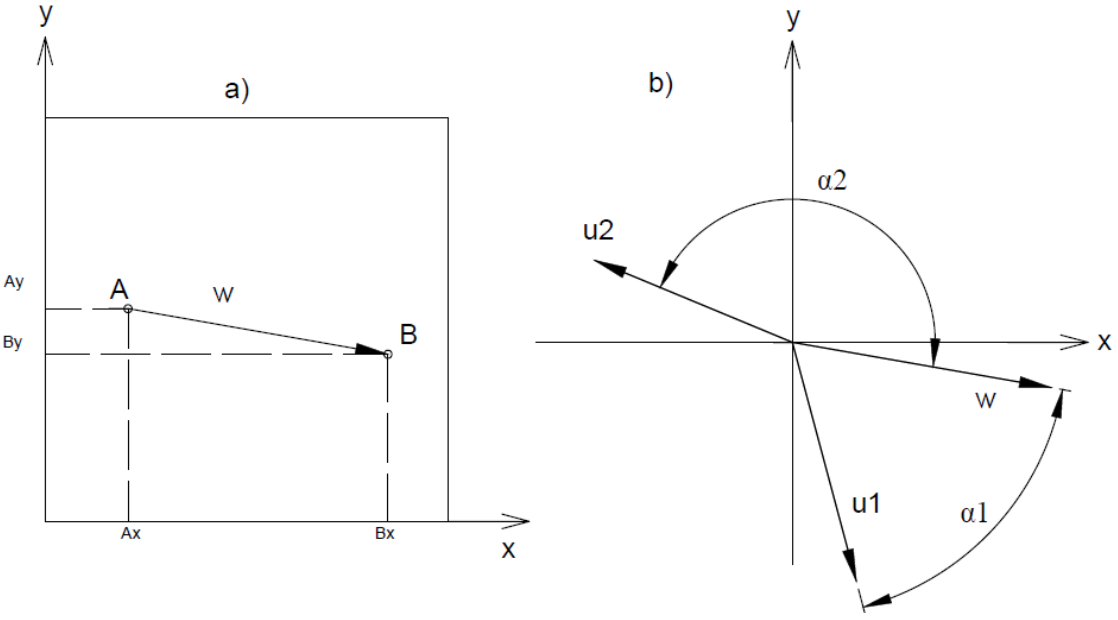
$$(u\times v)\cdot(w\times v_{12}) > 0 \tag{13}$$



Obr. 11 a) Označení vektorů v referenční mapě
b) Označení vektorů ve snímku

4.2.2 Určení natočení

Vstupními parametry pro řešení problému natočení robota je poloha dvou nalezených značek M_1, M_2 a k nim odpovídající poloha 2 značek A, B se známou polohou na myšlené referenční mapě (viz obr. 12a). Z těchto dvou dvojic bodů jsou vytvořeny vektory, pro které platí symbolické rovnice: $w=B-A$ a $u=M_2-M_1$. Tyto vektory jsou své podstaty volné a můžou být překresleny do počátku souřadných os, jak je naznačeno na obr. 12b. Na tomto obrázku jsou dva vektory u odpovídající, dvěma různým natočením stejných značek.



Obr. 12 a) Referenční mapa se značky A,B
b) Znázornění vektorů ve společném souřadném systému

Pro jednoznačné řešení je nutné nadefinovat, jak se bude natočení vnímat. Řekněme tedy, že natočení robota bude úhel φ, o který se musí robot natočit ve smyslu běhu hodinových ručiček tak, aby se souřadný systém snímku shodoval se souřadným systémem mapy. Pro polohu dvou vektorů je v analytické matematice vztah (14).

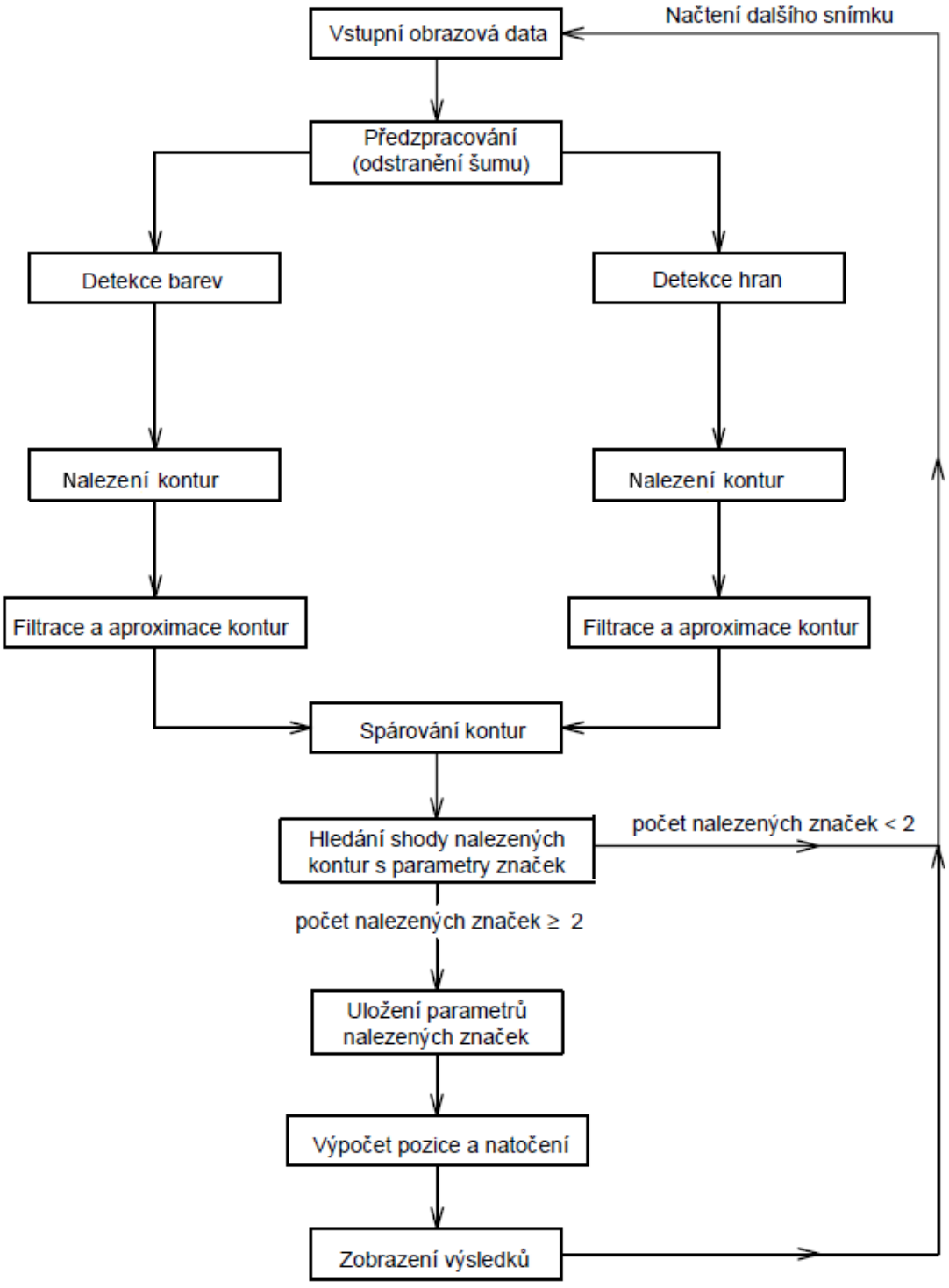
$$\cos\alpha = \frac{w \cdot u}{|w| \cdot |u|} \tag{14}$$

Kde v čitateli je skalární součin vektorů a ve jmenovateli je součin velikostí vektorů. Tento úhel α nabývá hodnot v intervalu 0° až 180° a jeho velikost pro ilustrační příklady je patrná z obr. 12b. Pro vektor u_2 je úhel α_2 zároveň hledaným úhlem φ. Pro vektor u_1 to už neplatí, protože z definice úhlu φ je to úhel doplňkový do 360°, teda $\varphi=360-\alpha$. Pro rozhodnutí, kdy můžeme úhel α prohlásit hledaným natočením φ nám postačí vektorový součin, podobně jak je použit v kap. 4.2.1 a to následujícím způsobem:

Je-li $(w \times u)<0$ potom $\varphi = (360 - \alpha)$ při nesplnění této podmínky pak platí: $\varphi = \alpha$

4.3 Blokové schéma

V obr. 13 je zobrazen celý navržený algoritmus pro lokalizaci pomocí blokového schématu.



Obr. 13 Algoritmus návrhu řešení

5 Praktická realizace

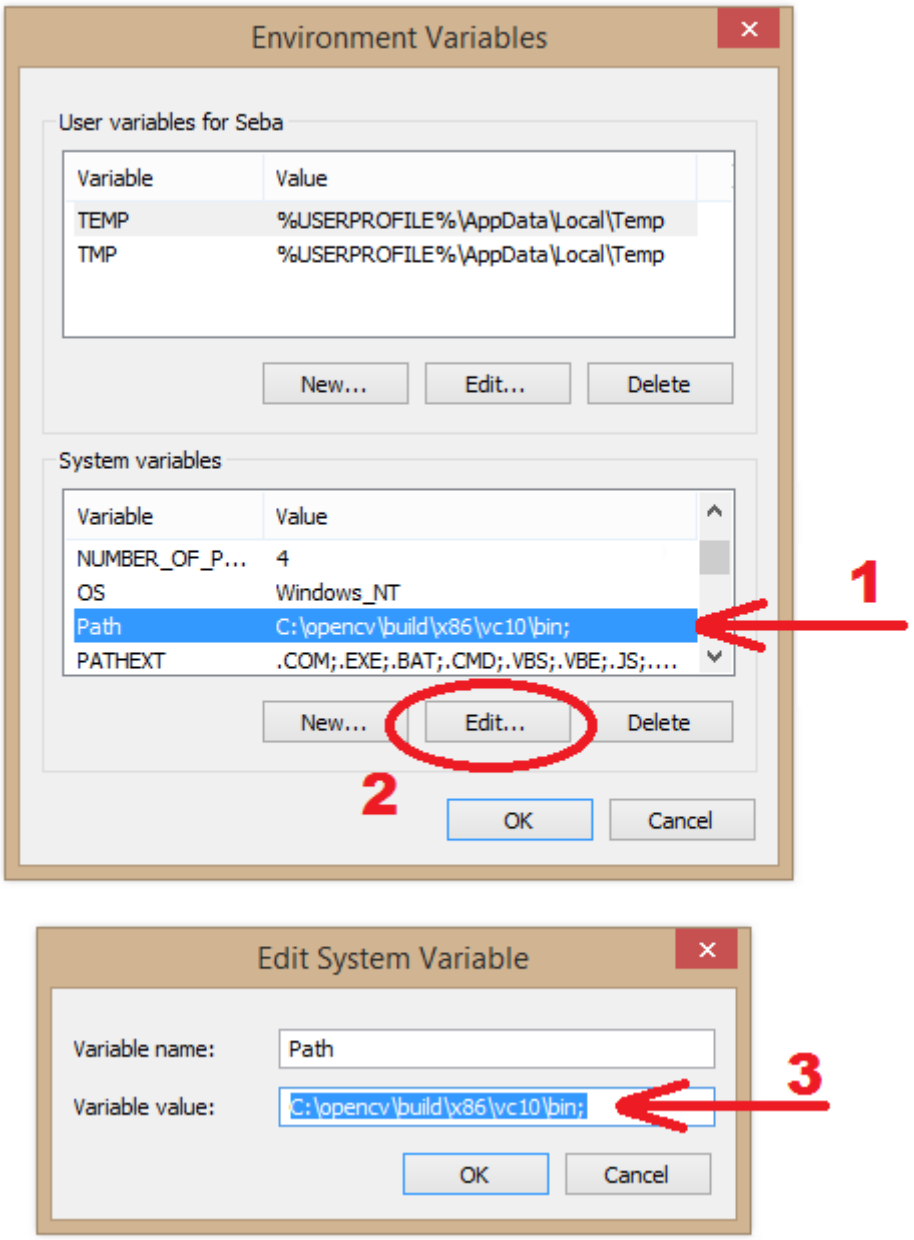
V této kapitole bude popsána praktická realizace navrhovaného systému a to včetně problémů, které se vyskytly. Velká část bude také věnována popisu použitých funkcí knihovny OpenCV a popisu nastavování jejích parametrů k optimalizaci pořízených výsledků.

5.1 Instalace OpenCV 2.4.10

Instalace a konfigurace bývá u OpenCV poměrně pracná záležitost. Na internetu je spousta návodu, avšak funkčních je jen málo z nich [12], proto je v následující podkapitole udělán návod na instalaci a následnou konfiguraci OpenCV verze 2.4.10 v Microsoft Visual c++ 2010 Express, což je vývojové prostředí od Microsoftu.

Prvním krokem je stáhnutí OpenCV 2.4.10 z oficiálního webu [11] a následné rozbalení a nainstalování do zvolené složky např. *C:\opencv*. Složka v dané lokaci nyní obsahuje všechny hlavní soubory, knihovny, příklady kódů atd. Knihovna se nyní musí importovat do prostředí systému i do vývojového prostředí.

Druhým krokem je přidání cesty *C:\opencv\build\x86\vc10\bin* do našeho systému. Přes **Control Panel → System → Advanced system settings → Advanced** Tab**. → Enviroment variables** dále postup dle obr. 14.

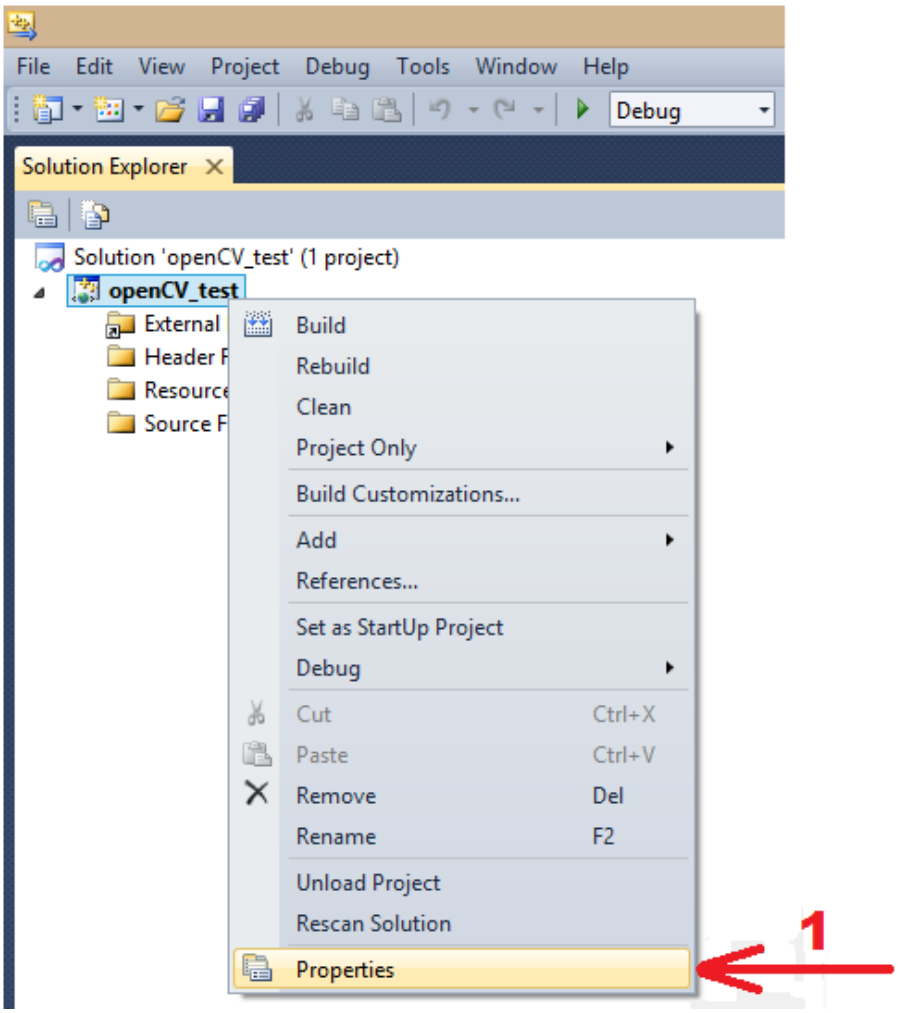


Obr. 14 Konfigurace
Path(1), Edit(2), vložení cesty do Variable value(3)

5.2 Vytvoření nového projektu a nastavení Visual C++

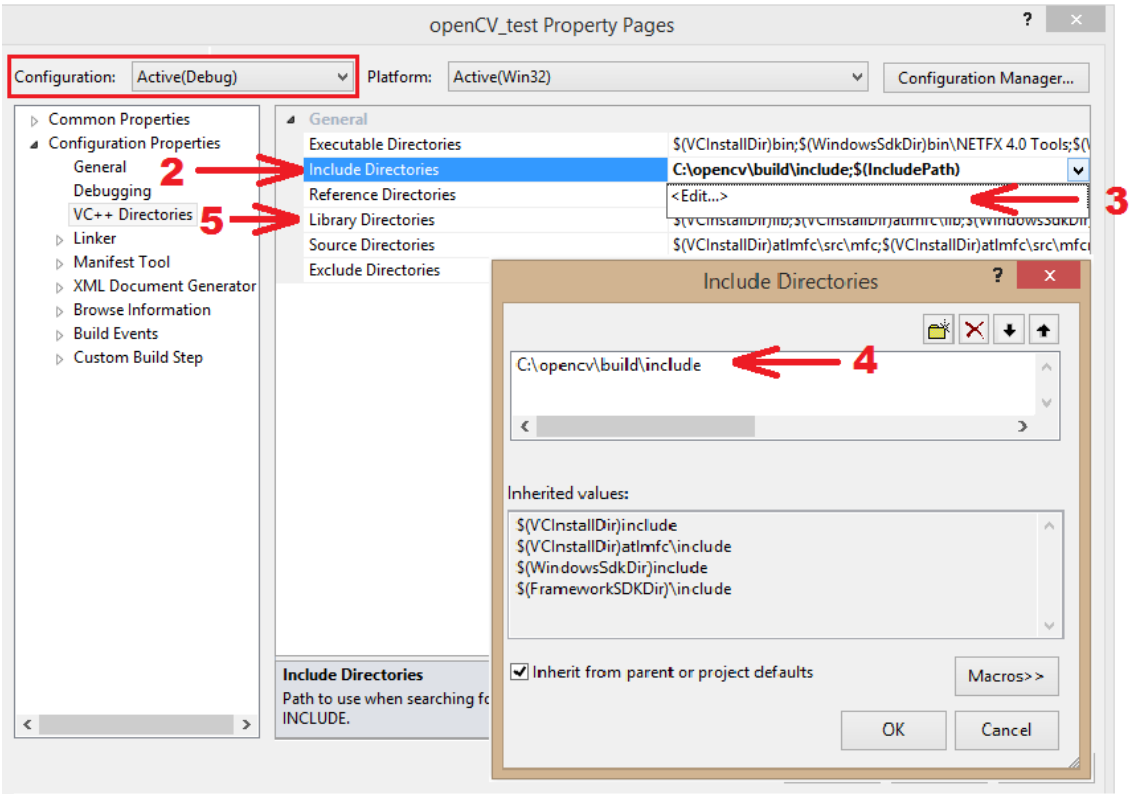
Zaprvé je třeba si stáhnout a nainstalovat Visual C++ 2010 Express, což je jedno z nejpopulárnějších vývojových prostředí pro programování. Prostředí je zcela zdarma a může být nalezeno přímo na oficiálních stránkách Microsoftu [22].

Dalším krokem je otevřít Visual C++ a založit nový projekt: **File →New →Projetct →Visual C++ →Empty Project**. Pojmenován může být jakkoli např. *openCV_test*. Přes pravé tlačítko myši po najetí na název projektu (obr. 15) je třeba otevřít **Properties (1)**



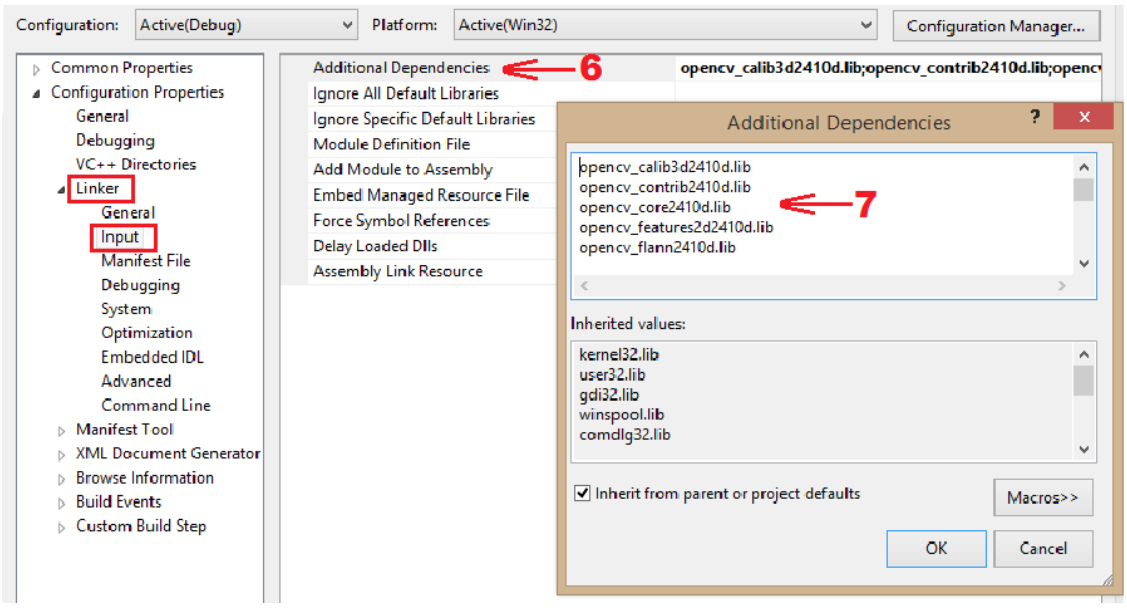
Obr. 15 Zobrazení vlastností projektu

Je třeba ověřit zdali je v *Configurations:* označený *debug* (obr. 16) a dále najít **VC++ Directories** a označit **Include Directories (2) → Edit (3) → vepsat** *C:\opencv\build\include* **(4)**
Dalším krokem je vybrat **Library Directories (5)** a podobným způsobem přes **Edit** vepsat *C:\opencv\build\\x86\vc10\lib* nakonec se musí vše potvrdit tlačítkem apply.



Obr. 16 Properties window

Dalším krokem je najet zpátky do okna vlastností (obr. 17) a označit **Linker → Input → Additional Dependencies (6) → Edit → vložit následující knihovny (7)**



Obr. 17 Additional Dependencies

opencv_calib3d2410d.lib
opencv_contrib2410d.lib
opencv_core2410d.lib
opencv_features2d2410d.lib
opencv_flann2410d.lib
opencv_gpu2410d.lib
opencv_highgui2410d.lib
opencv_imgproc2410d.lib
opencv_legacy2410d.lib
opencv_ml2410d.lib
opencv_nonfree2410d.lib
opencv_objdetect2410d.lib
opencv_photo2410d.lib
opencv_stitching2410d.lib
opencv_ts2410d.lib
opencv_video2410d.lib
opencv_videostab2410d.lib

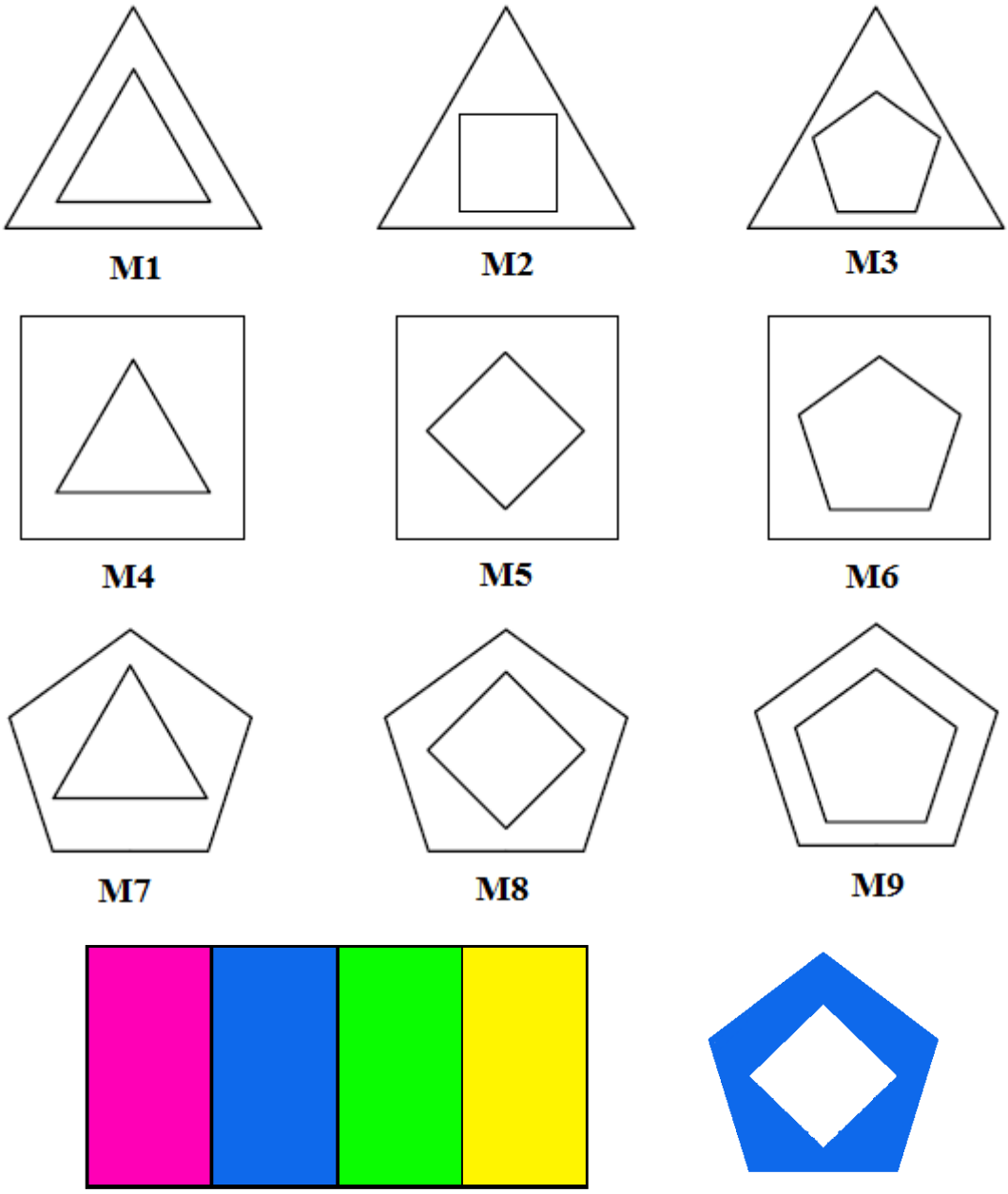
Poznámka: čtyřčíslí 2410 označuje verzi OpenCV pro verzi 2.4.9 by to bylo trojčíslí 249. Písmeno *d* za čísly vyjadřuje *debug*. Optimálně se ještě může udělat stejný postup pro *release*, kopírované knihovny by pak byly ve formátu:

o*pencv_calib3d2410.lib*
atd…

Nastavování je nyní dokončeno a nový c++ file může být vytvořen. Knihovny s velkým množstvím funkcí a algoritmů pro práci s obrazem by měly být nyní plně dostupné. Nový c++ file se vytvoří stisknutím pravého tlačítka myši na názvu projektu a označením: **Add → New Item → Visual C++ → C++ File** .

5.3 Volba podoby značek

Pro řešení problému detekce značky si je první nutné navrhnout podobu značek. Pro tuhle práci je uvažováno 9 tvarů značek označených M1 až M9 (viz obr. 18), jejichž podoba byla záměrně navrhnuta velice jednoduše. Značky obsahují pouze 3 základní geometrické tvary a těmi je: trojúhelník, čtyřúhelník a pětiúhelník. Každá značka obsahuje právě 2 tyto tvary. Pro zvětšení celkového počtu značek je uvažováno 4 barevných provedení pro každou z nich. Celkově je tedy 9×4=36 specifických značek. Navrhované barvy společně s ukázkou značky M8 v barevném provedení jsou zobrazeny na obr. 18. Barvy jsou záměrně voleny velice pestré, aby byly rozpoznatelné i při snížení intenzity osvětlení.



Obr. 18 Návrh podoby značek a jejich barev.
Vpravo dole uvažovaná podoba modré značky M8

5.4 Funkce použité pro implementaci softwaru

V této kapitole budou uvedeny použité funkce knihovny OpenCV a krok po kroku názorně ukázán postup při detekci značek. Popis parametrů funkcí je podle oficiální online dokumentace OpenCV [10].

5.4.1 Bilaterální filtr

K potlačení šumu je použito bilaterálního filtru, jehož hlavní výhodou je, že zachovává hrany, což je vzhledem k nadcházejícím operacím (detekce hran a barev) příhodné. V knihovně OpenCV je vytvořena funkce s názvem *bilateralFilter*. Výsledek po aplikování tohoto filtru na vstupní obraz (obr. 19) je vidět na obr. 20. Nevýhodou tohoto filtru je poměrně velká výpočtová náročnost, ta se dá redukovat rozsahem velikosti filtrace.

Obr. 19 Původní podoba obrazu

Obr. 20 Výsledek po aplikování bilaterálního filtru

5.4.2 Detekce hran

Pro detekci hran byl zvolen Cannyho hranový detektor, který je v OpenCV vyjádřen funkcí *Canny*. Hlavními přednostmi této detekce hran je malá chybovost (detekuje pouze hrany, které v obraze skutečně jsou) a dobrá lokalizace (minimální rozdíl mezi detekovanou a skutečnou hranou). Vstupní parametry této funkce jsou vysvětleny v tabulce 1.

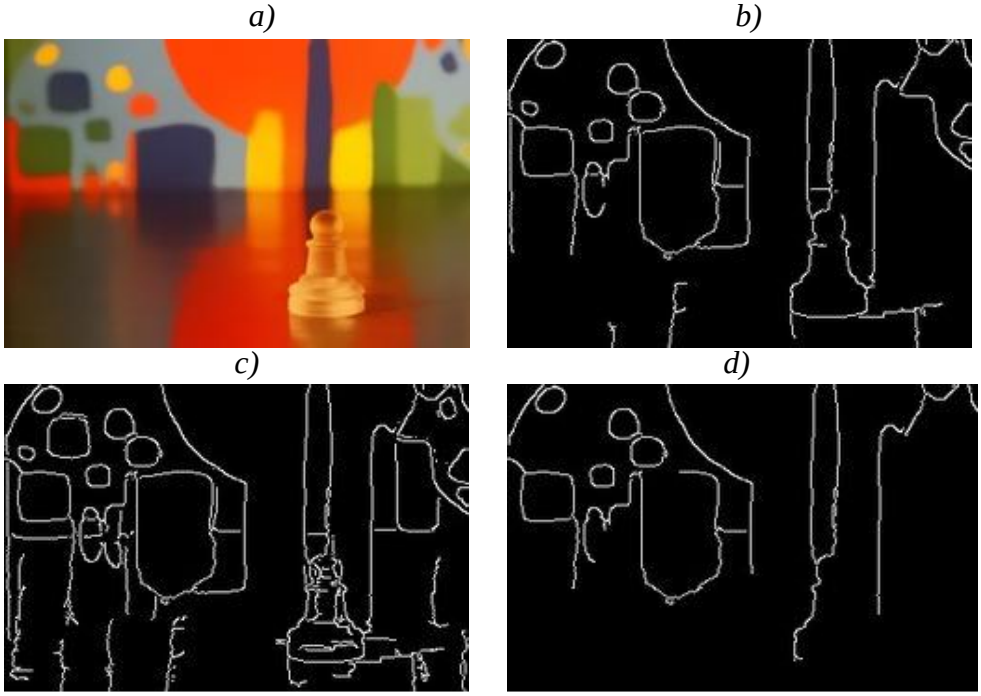
Tab. 1

Canny Edge Detector

```
Canny(source, detected_edges, lowThreshold, upThreshold, kernel_size);
```

`Mat source;`	matice se vstupním obrazem
`Mat detected_edges;`	matice pro výstupní obraz
`int lowThreshold;`	spodní hranice prahování
`int upThreshold;`	horní hranice prahování
`int kernel_size;`	velikost předzpracovávané oblasti

V případě vlastního předzpracování obrazu je argument *kernel_size* nepotřebný a může se vynechat. Hranice prahování mají zásadní vliv na výsledek detekce hran, proto je třeba, aby se prahovací hodnoty experimentálně vyzkoušely pro dané provozní podmínky. Obecně je hodnota závislá na velikosti rozlišení vstupního obrazu, kde horní hranice prahu se doporučuje volit jako trojnásobek spodní. Různé výsledky pro různé hodnoty spodní hranice můžete vidět na obr. 21.



Obr. 21 Detekce hran a) Vstupní obraz b) Ideální nastavení spodního prahu c) Nízká hodnota spodního prahu d) Vysoká hodnota spodního prahu

5.4.3 Detekce barev

Pro detekci barev je použito funkce *inRange*. Operace, která této funkcí předchází je převod z barevného modelu RGB do HSV a to z důvodu stabilnějšího přechodu barev. V RGB jsou barvy „míchány“ a je velice obtížné jednoznačně vymezit odstíny jedné barvy.

Barevný model HSV více odpovídá lidskému vnímání barev. Barvy jsou podobně jako u RGB rozloženy do tří složek a to: Hue (barevný tón), Saturation (sytost) a Value (hodnota jasu). Název HSV vznikl podle názvů těchto tří složek a barevný převod je realizován pomocí funkce *cvtColor*. Vstupní argumenty funkce *inRange* a *cvtColor* jsou vysvětleny v tabulce 2.

Tab. 2

Color Detection

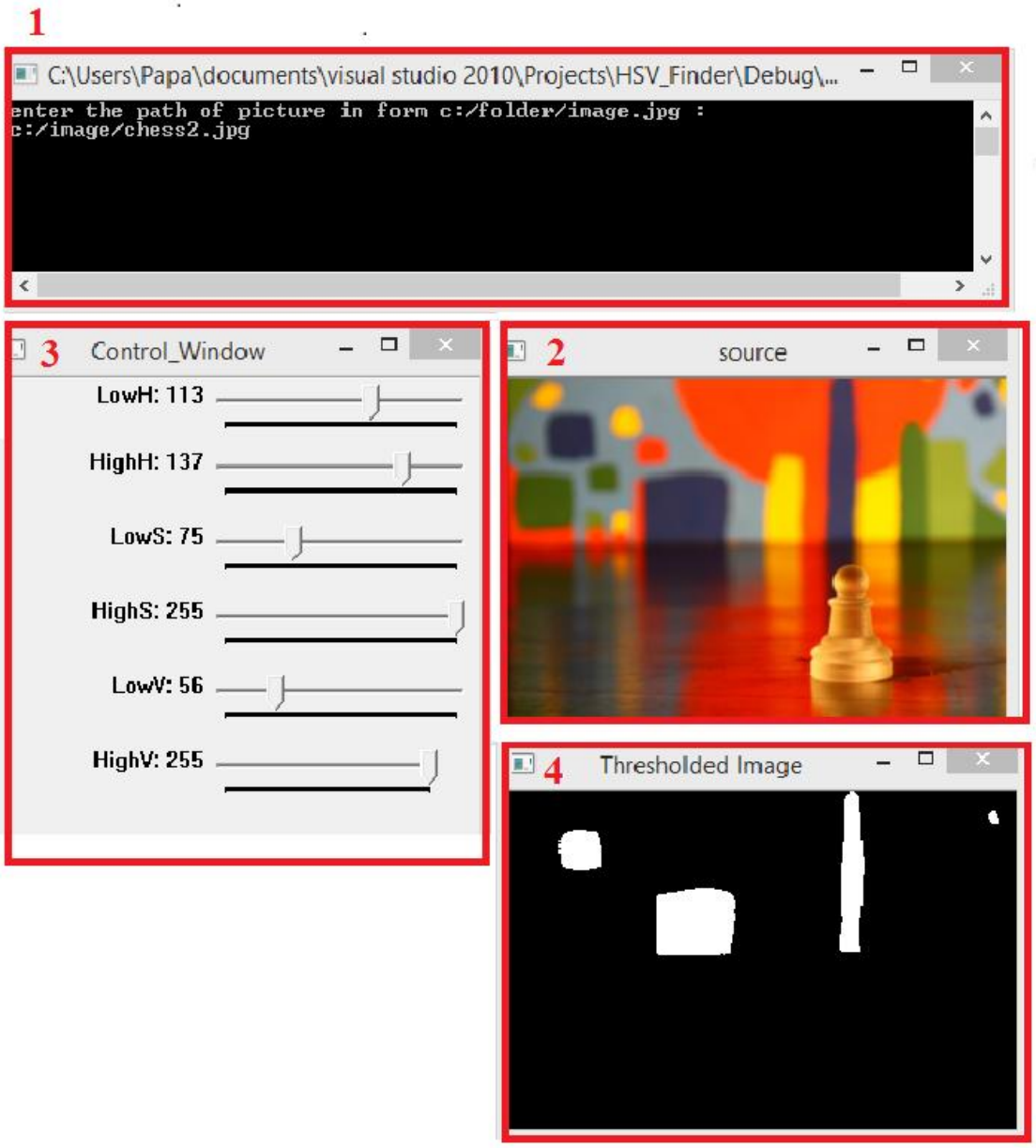
```
cvtColor(source, sourceHSV, COLOR_BGR2HSV);

inRange(sourceHSV, Scalar(lH, lS, lV), Scalar(hH, hS, hV), Thresholded);
```

`Mat source;`	matice se vstupním obrazem
`Mat sourceHSV;`	výstupní matice funkce *cvtColor*
`COLOR_BGR2HSV`	parametr označující převod z RGB do HSV
`int lH;`	spodní hranice barevného tónu H
`int lS;`	spodní hranice sytosti S
`int lV;`	spodní hranice hodnoty jasu V
`int hH;`	horní hranice barevného tónu H
`int hS;`	horní hranice sytosti S
`int hV;`	horní hranice hodnoty jasu V
`Mat Thresholded;`	výstupní matice funkce *inRange*

Důležitým faktorem pro správnou funkci detekce je zjistit intervaly hodnot H, S, V. Pro tento účel byl vytvořen software, v kterém lze měnit spodní a horní hranice hodnot H, S, V, a kde je možné výstup metody *inRange* ihned pozorovat a díky toho snadno a rychle nalézt, pro dané podmínky, správné intervaly hodnot. Tento program je možné najít na přiloženém CD pod názvem HSV_Finder. Podoba rozhraní programu je patrná z obr. 22, ve kterém je vyobrazená detekce modré barvy v ilustračním obrázku.

Postup po zapnutí programu HSV_Finder je následující (obr. 22): Do příkazového řádku (1) je nutné vepsat cestu umístění vstupního obrazu, který se po správném zadání zobrazí v okně source (2). V okně Control_Window (3) pak lze kurzorem myši měnit intervaly hodnot H, S, V a výsledek funkce *inRange* pozorovat v okně Thresholded Image (4).



Obr. 22 Podoba rozhraní programu HSV_Finder na zjištění optimálních hodnot H, S, V pro danou hledanou barvu

5.4.4 Nalezení a filtrace kontur

Pro nalezení kontur je v knihovně OpenCV vytvořena funkce *findContours*, jejíž parametry jsou vysvětleny v tabulce 3.

Tab 3.

Contours

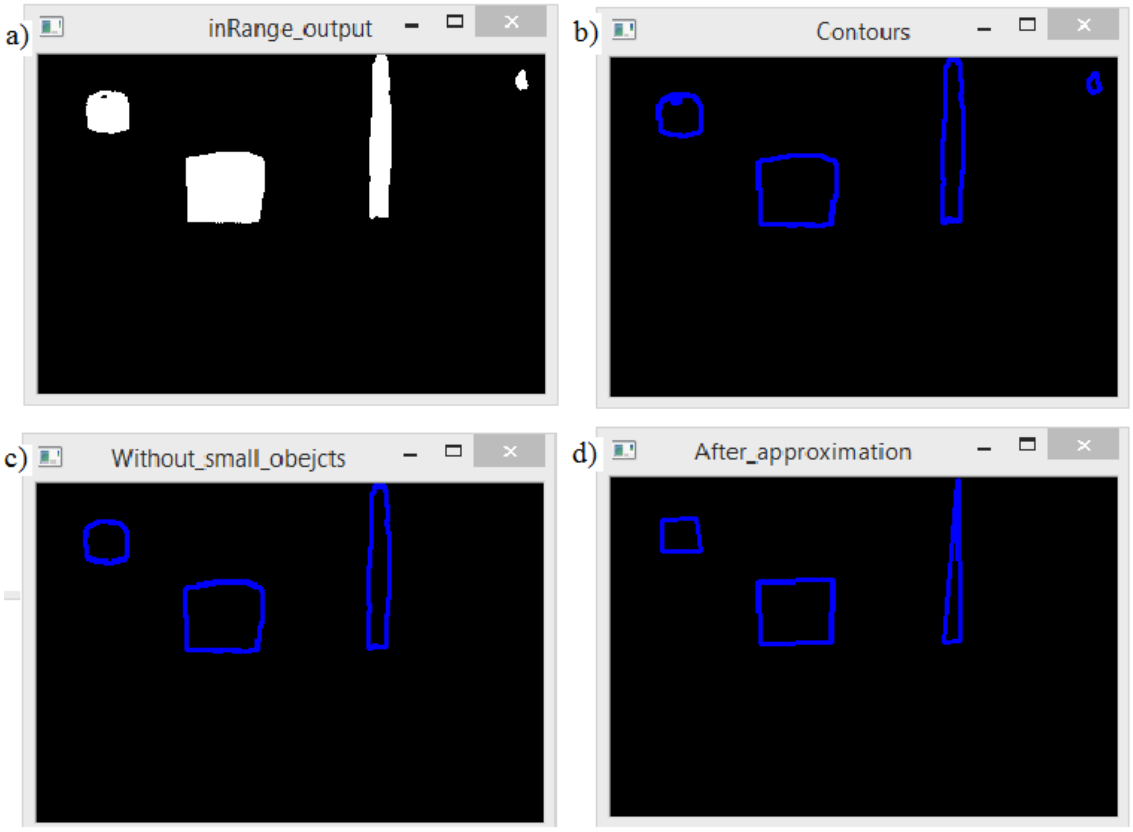
```
findContours(bin_im, contours, hiearchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE,
    Point(0,0));
```

`Mat bin_im;`	matice se vstupním binárním obrazem
`vector<vector<Point>>contours;`	výstupní vektor kontur
`vector<Vec4i>hiearchy;`	volitelný výstup vektoru, který obsahuje informace o topologii snímku.
`CV_RETR_TREE`	argument, který určuje vrácení všech kontur a rekonstrukci plné hierarchie vnořených kontur
`CV_CHAIN_APPROX_SIMPLE`	argument specifikující nalezení bodů kontur následovně: pro vertikální diagonální a horizontální rozměry budou zachovány jen jejich koncové body.
`Point(0,0)`	Je offset, o který se má posunout každý bod kontury

Výstupem je vektor vektorů bodů *contours*, který lze pomocí cyklu *for* prohledávat, a tím pracovat s každou konturou individuálně. Prvním krokem v cyklu je podmínka minimální velikosti kontury pomocí funkce *contourArea,* kde hranici nejmenší plochy je nutno odhadnout a experimentálně ověřit, aby nedocházelo k přeskočení významných kontur. Následně jsou kontury aproximovány novými pomocí funkce *approxPolyDP*, kde je taky důležité zregulovat míru aproximace, aby se dosáhlo optimálních výsledků. Dalším krokem je podmínka konvexního tvaru kontury (navrhované geometrické tvary značek jsou všechny konvexní), to je zrealizováno funkcí *isContourConvex*. Posledním krokem je klasifikace geometrických tvarů kontur, toho lze docílit pomocí vyjádření velikosti kontury, která odpovídá počtu bodů neboli vrcholů, které jí tvoří. Pro navrhované značky jsou oblasti zájmů pouze tři tvary a to: trojúhelník (3 vrcholy), čtyřúhelník (4 vrcholy) a pětiúhelník (5 vrcholů). Vykreslené kontury po jednotlivých úpravách je možno pozorovat na obr. 23.

I přes všechny tyto předchozí operace je možné, že dojde ke špatné klasifikaci tvaru a to hlavně z důvodu špatné aproximace, kde např. jedna hrana trojúhelníku bude rozdělena na dvě, které budou svírat úhel blízky 180 °, takže tento trojúhelník bude vyhodnocen na základě počtu vrcholů jako čtyřúhelník. K eliminaci tohoto problému byla vytvořena funkce *Angle*, která počítá hodnoty cosinů úhlů sousedících bodů kontur. Následně je pro dané geometrické tvary vymezena maximální a minimální hodnota velikosti cosinu tohoto úhlu. Např. úhly trojúhelníků navrhovaných značek jsou všechny 60 ° a po zvážení možných deformací těchto úhlu (záběrem kamery a následnými operacemi s obrazem) byla nastavena hodnota maximálního cosinu úhlu na 0.7 (přibližně 45 °) a minimální na 0.25 (přibližně 75 °). Obdobně byl tento postup aplikován i na čtyřúhelník a pětiúhelník. Díky tohoto postupu je spolehlivost klasifikace hledaných geometrických velmi vysoká.

Pro následnou práci s konturami byla vytvořena třída s názvem *Box*, která obsahuje kromě kontury také informací, zdali kontury přišly z metody detekce hran nebo barev, kde pro detekci barev je připojena i informace o barvě. Dále třída obsahuje informaci o tvaru kontury a také o těžišti kontury, které lze vyjádřit pomocí momentů.



Obr. 23
a) Vstupní binární obraz b) Nalezené kontury funkcí findContours
c) Vynechání malých kontur d) Aproximované kontury

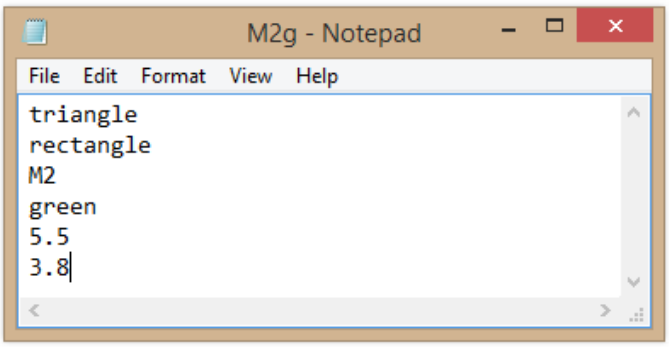
5.4.5 Detekce značky, lokalizace

Před detekcí značky je třeba kontury, které přišly z metody detekce hran a z detekce barev protřídit a v případě shody vytvořit odpovídající páry. Pro tento účel byla vytvořena nová třída s názvem *Couples*. Celý proces třídění využívá standardních funkcí a příkazů jazyku C++, proto nebude podrobněji popsán.

Dále jsou vyhledané páry prozkoumány, jestli netvoří nějaká jejich dvojce některou z hledaných značek, přičemž je kontrolována vzdálenost mezi jednotlivými konturami, aby nedošlo k přiřazení kontur, které jsou na odlišných místech. Jsou-li detekovány aspoň dvě značky, tak jsou souřadnice polohy těžišť nalezených značek uloženy a slouží jako vstupní parametry pro lokalizaci, která je vypočítána dle návrhu (viz kapitola 4.2).

5.5 Zásady tvorby referenční mapy

Databáze značek je zrealizována v podobě textových souborů, které jsou obsaženy ve složce s názvem database, která je umístěna ve složce projektu. Textový soubor má na prvním řádku název tvaru většího geometrického tvaru dané značky, na druhém řádku je tvar menšího, třetí řádek obsahuje název značky, čtvrtý barvu značky, pátý a šestý řádek je vymezen pro x-ovou a y-ovou souřadnici polohy značky v tomto pořadí. Po vytvoření mapy prostředí je třeba pro každou použitou značku zapsat souřadnice těžiště na tyto poslední dva řádky dokumentu. Příklad dokumentu je patrný z obr. 24.

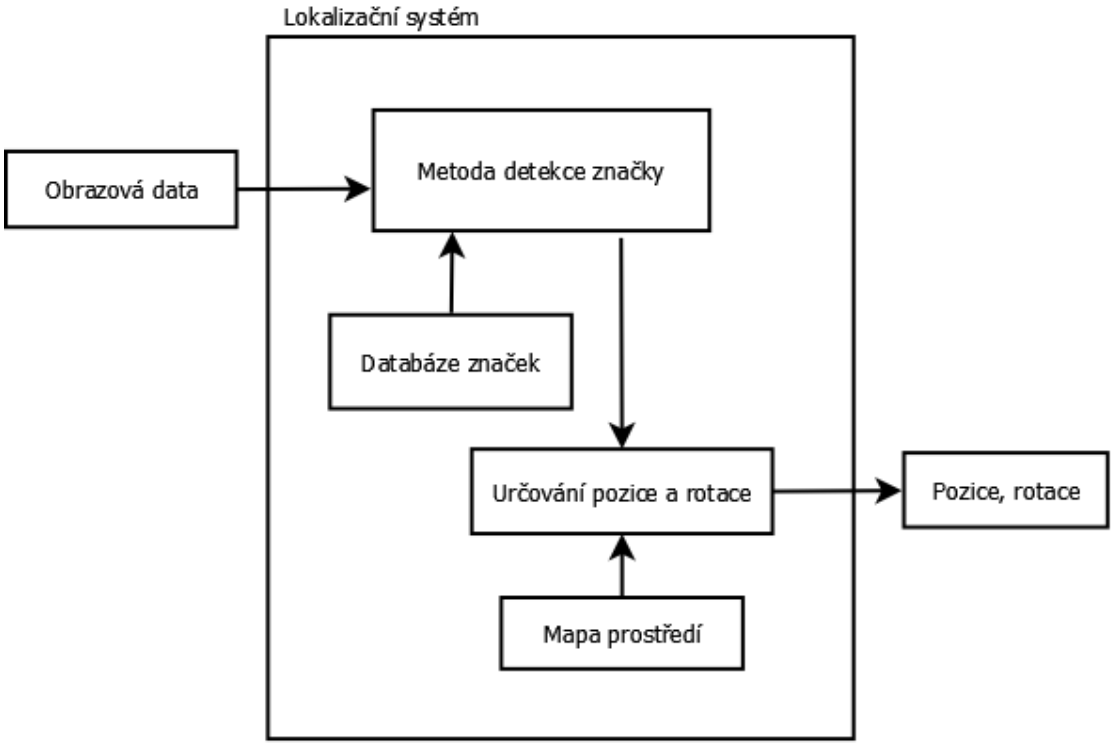


Obr. 24 Ukázka textového dokumentu pro zelenou značku M2

Je třeba mít na paměti, že na jedné poloze může být maximálně jedna značka a desetinná čísla se dělí desetinou tečkou. Při navrhování referenční mapy by se mělo vycházet z levotočivého souřadného systému, jinak by docházelo ke špatné lokalizaci z důvodu opačného smyslu souřadného systému referenční mapy a souřadnic snímku. Při rozmísťování značek v prostoru je třeba dbát takových vzájemných odstupů mezi značkami, aby na záběru kamery byly vždy aspoň dvě značky, jinak by docházelo ke špatné funkci navrhovaného lokalizačního systému.

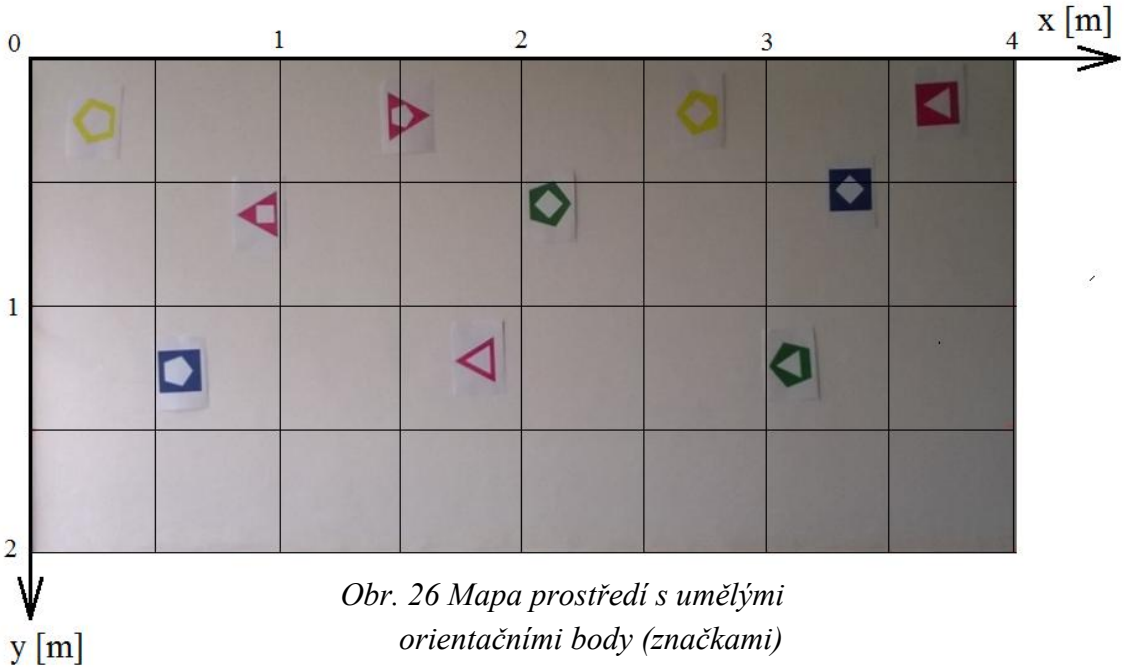
5.6 Popis navrženého systému

Navržený systém je schopen z obrazových dat určovat pozici a natočení v mapě prostředí. Ke správné funkci systému je nutné modifikovat prostředí systematickým rozmístěním značek (viz kapitola 5.5), kterých může být až 36 (viz kapitola 5.3). Pozice značek musí být vepsány do textových souborů dle kapitoly 5.5 a objektiv kamery by měl být kolmý na lícní stranu značek. Blokové schéma systému a jeho interakce jsou patrné z obr. 25.

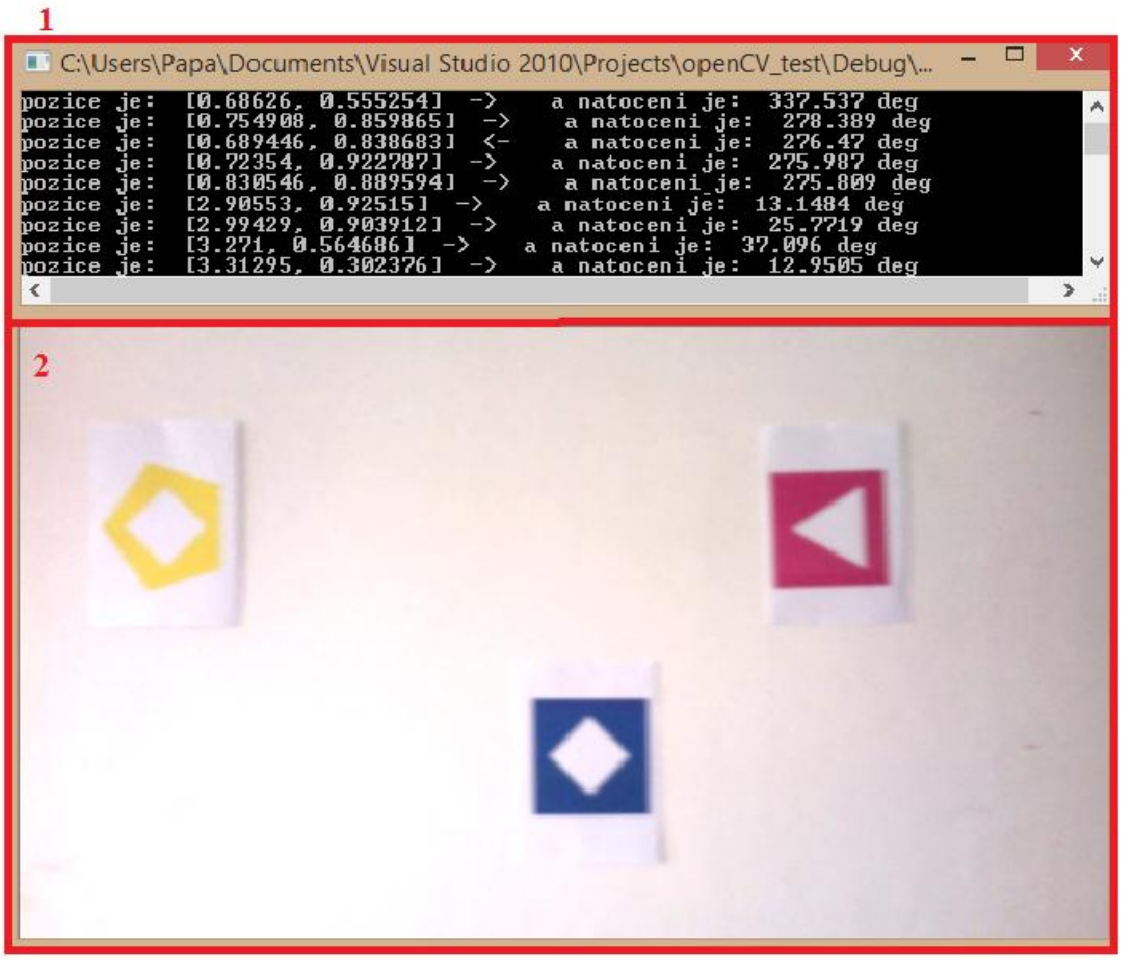


Obr. 25 Blokové schéma systému

Na obr. 26 je mapa prostředí i s naznačeným souřadným systémem. Na obr. 27 je reálná podoba vytvořeného softwaru, v okně (1) jsou tisknuty výsledky lokalizace, které odpovídají pozici v referenční mapě a příslušnému natočení souřadného systému snímku od souřadného systému mapy. Znaky šipek jsou připojeny pro rychlejší orientaci zdali x-ová souřadnice roste (->) nebo klesá (<-). Spodní řádek okna (1) odpovídá pozici v mapě (obr. 26) pro dané vstupní obrazová data, která jsou na obr. 27 (2).



Obr. 26 Mapa prostředí s umělými orientačními body (značkami)



Obr. 27 Reálná podoba navrženého softwaru

6 Testování systému

Tato kapitola popisuje základní poznatky nabyté po otestování navrženého systému v reálném prostředí a také návrh experimentu na určení pravděpodobnosti detekce značky. Systém byl testován na notebooku Lenovo IdeaPad G510, jehož parametry jsou:

- Procesor : Intel® Core™ i5-4210M
- Grafika: AMD Radeon R7 M265 - 2GB
- Paměť: 4GB (1x 4GB) DDR3L 1600 MHz

Před měřením byly rozmístěny značky v prostředí a vytvořen souřadný systém, podle kterého bylo ke každému středu značky přiřazena souřadnice, která byla vepsána do příslušného textového dokumentu (viz kapitola 5.5). Dále byl simulován provoz mobilního robotu pohybem kamery namířenou kolmo na lícní stranu značek. Výsledné pozice a natočení byly kontrolovány, zdali odpovídají poloze v mapě.

Provoz systému v reálném čase byl na první pohled neplynulý, a to z důvodu relativně pomalé výpočtové rychlosti. Bylo zjištěno, že v závislosti na rozlišení a hloubky předzpracování obrazových dat, se pohybuje výpočtová rychlost v rozmezí 4 až 10 snímků za sekundu. Samotná lokalizace byla korektní a odpovídala reálné pozici a natočení, je však zatížená chybami vzniklých při zpracování obrazu. Tyto chyby nemají kumulativní charakter a jejich velikost by se pro dané provozní podmínky musela určit experimentálně.

Během testování systému nedošlo ani jednou ke špatné identifikaci značky, a to hlavně díky požití dvou metod zpracování obrazu. Z toho lze vyvodit závěr, že systém je spolehlivý, co se týče identifikace značek (jejich tvaru i barvy).

6.1 Experiment na určení pravděpodobnosti detekce značky

Na výpočet pravděpodobnost detekce jedné značky byl navržen experiment, kde se ze vstupních dat spočítalo, kolikrát byla detekována daná značka. Tento počet se pak vydělil počtem snímků, na kterých se značka vyskytovala a výsledkem byla pravděpodobnost detekce dané značky pro dané provozní podmínky. Aby se dosáhlo objektivnějších výsledků, tak byl tento experiment prováděn na různých značkách o různých barvách a v různé intenzitě osvětlení. Výsledná pravděpodobnost detekce jedné značky je pak dána průměrem vypočtených dílčích pravděpodobností.

Roli samozřejmě hraje i typ použité kamery a rozlišení vstupních obrazových dat, proto byly použity dvě kamery a to:

- Fotoaparát v mobilu Nokia Lumia 620, parametry videa 1 280 × 720 px, 30 FPS
- Webová kamera s rozlišením 640 × 480 px

Data byla ukládána do textových souborů a jejich další zpracování proběhlo v softwaru Excel. Výsledek experimentu je, že v 29 % případů byla značka na snímku detekována.

6.2 Pravděpodobnost lokalizace

Aby mohl vytvořený systém spočítat pozici a natočení, tak musí být detekovány alespoň dvě značky. Z toho vyplývá, že pravděpodobnost lokalizace je přímo úměrná pravděpodobnosti detekce dvou značek. Je-li předpokládáno, že značky jsou v mapě prostředí rozmístěny právě tak, aby na snímku byly vždy právě dvě, tak pravděpodobnost detekce dvou značek je součinem pravděpodobnosti detekce jedné (viz kapitola 6.1).

Výsledná pravděpodobnost lokalizace je tedy přibližně 8,4 % při každém zpracovaném snímku. Při požadavku lokalizace jednou za vteřinu by tedy muselo být zpracováno 12 snímků za sekundu. Tento výsledek je třeba považovat pouze za orientační. Obecně je pravděpodobnost lokalizace určená provozními podmínkami, parametry kamery a nastavením parametrů systému (hranice prahování, velikost aproximací aj.).

7 Závěr

Tato práce se zabývá návrhem lokalizačního systému na základě metod zpracování obrazu. První 2 kapitoly této práce jsou rešeršního charakteru, přičemž první shrnuje lokalizační metody a senzory využívaných pro lokalizaci. V Druhé části jsou popsány základní způsoby zpracování obrazu.

Lokalizační systém se podařilo následně vytvořit v prostředí Microsoft Visual Studio 2010 Express za použití programovacího jazyka C++ a pomocí knihovny OpenCV, jejíž stěžejní funkce použité v realizaci systému jsou popsány v kapitole 5.4. Vytvořený systém je schopný určovat absolutní polohu a natočení mobilního robota v předem známém prostředí. Díky aplikování dvou různých přístupů zpracování obrazu se podařilo dosáhnout velké spolehlivosti identifikace umělých orientačních bodů. Experimentálně bylo zjištěno, že detekce těchto bodů byla v 27 % případů.

Zpracování dat v reálném čase je výpočtově náročné a pro plynulý chod je zapotřebí výkonné výpočtové techniky. Do budoucna se může připojit k navrženému systému relativní lokalizační technika využívající zpracování obrazu (např. optický tok).

Seznam použité literatury

[1] DAVISON, Andrew J. Real-time simultaneous localisation and mapping with a single camera. In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003. p. 1403-1410.

[2] RŮŽIČKA. M *Návrh konstrukce majáků pro indoor navigaci mobilních robotů*..Brno: Vysoké učení technické v Brně. Fakulta strojního inženýrství. 2010. 49s. Vedoucí bakalářské práce Ing. Stanislav Věchet. Ph.D..

[3] Obr. 1 *sgbotic* [online]. [cit. 2015-04-04]. Dostupné z: http://www.sgbotic.com/index.php?dispatch=products.view&product_id=22

[4] Obr. 2 *ic0nstrux* [online]. [cit. 2015-04-04]. Dostupné z: http://www.ic0nstrux.com/PING%29%29%29-Ultrasonic-Sensor-Mounting-Bracket#.VSBCcuE_a1B

[5] obr. 4 *Direct INDUSTRY* [online]. [cit. 2015-04-04]. Dostupné z: http://www.directindustry.com/prod/mti-instruments/laser-displacement-sensor-harsh-environments-19330-1087331.html

[6] SKALKA, Marek Skalka. Srovnání lokalizačních technik v robotice: Prostředky relativní lokalizace. *Http://marek.sk.sweb.cz* [online]. 1. 8. 2011, 11. 3. 2015 [cit. 2015-04-05]. Dostupné z: http://marek.sk.sweb.cz/lokalizace/kapitola3.html

[7] Strojové vidění & Průmyslové kamery. BLUMENBECKER PRAG S.R.O. *Http://www.blumenbecker.cz* [online]. [cit. 2015-04-05]. Dostupné z: http://www.blumenbecker.cz/cs/kompetence/inzenyring/strojove-videni/

[8] obr. 3 *video-zone* [online]. [cit. 2015-04-04]. Dostupné z: http://www.video-zone.cz/recenze/jvc-gs-td1-t9.html¨

[9] GONZALEZ, Rafael C.; WOODS, Richard Eugene; EDDINS, Steven L. *Digital image processing using MATLAB*. Pearson Education India, 2004.

[10] OpenCV Tutorials. *OpenCV* [online]. [cit. 2015-05-25]. Dostupné z: http://docs.opencv.org/doc/tutorials/tutorials.html

[11] OpenCV. [online]. [cit. 2015-04-05]. Dostupné z: http://opencv.org/

[12] Stackoverflow. FLOWFREE. *Stackoverflow: questions* [online]. 2012 [cit. 2015-04-05]. Dostupné z: http://stackoverflow.com/questions/10901905/installing-opencv-2-4-3-in-visual-c-2010-express

[13] SKALKA, Marek Skalka. Srovnání lokalizačních technik v robotice: Lokalizace v mobilní robotice. *Http://marek.sk.sweb.cz* [online]. 1. 8. 2011, 11. 3. 2015 [cit. 2015-04-05]. Dostupné z: http://marek.sk.sweb.cz/lokalizace/kapitola1.html#kapitola1_3

[14] SKALKA, Marek Skalka. Srovnání lokalizačních technik v robotice: Prostředky absolutní lokalizace. *Http://marek.sk.sweb.cz* [online]. 1. 8. 2011, 11. 3. 2015 [cit. 2015-04-05]. Dostupné z: http://marek.sk.sweb.cz/lokalizace/kapitola4.html#kapitola4_1_1

[15] obr. 5 *Vacuum Cleaner Reviews Today* [online]. [cit. 2015-04-04]. Dostupné z: http://www.vacuumcleanerreviewstoday.com/irobot/irobot-560/

[16] EmguCV. [online]. [cit. 2015-04-05]. Dostupné z: http://www.emgu.com/wiki/index.php/Main_Page
[17] intelDeveloperZone. [online]. [cit. 2015-04-05]. Dostupné z: https://software.intel.com/en-us/intel-ipp
[18] obr. 6 *Biometrics Access Control* [online]. [cit. 2015-04-04]. Dostupné z: http://sphinxtech.com.sg/AboutBio.htm
[19] CAMEA: Detekce jízdy na červenou. SHERWOOD MEDIA S.R.O. [online]. [cit. 2015-04-25]. Dostupné z: http://www.camea.cz/cz/reference/detekce-jizdy-na-cervenou/
[20] SOUKUP, Tomáš Soukup. Vidět znamená vědět. O práci s obrazovými daty v medicíně. [online]. [cit. 2015-04-25]. Dostupné z: http://vtm.e15.cz/videt-znamena-vedet-o-praci-s-obrazovymi-daty-v-medicine
[21] obr.7 *what-when-how* [online]. [cit. 2015-04-04]. Dostupné z: http://what-when-how.com/embedded-image-processing-on-the-tms320c6000-dsp/contrast-stretching-image-processing/
[22] Microsoft. [online]. [cit. 2015-04-05]. Dostupné z: www.microsoft.com