Zpracování obrazu pro řízení mobilního robotického systému

Control of mobile robot using image processing

Tomáš Urbánek

Bakalářská práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš URBÁNEK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Zpracování obrazu pro řízení mobilního robotického systému**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na téma zpracování obrazu pro řízení mobilního robotického systému.**
2. **Analyzujte současný stav vybraných nástrojů pro zpracování obrazu pro řízení mobilního robotického systému.**
3. **Navrhněte vlastní implementaci řízení mobilního robotického systému.**
4. **Demonstrujte řešení s využitím systému Lego Mindstorm NXT.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **FORSYTH, David, PONCE, Jean. Computer Vision : A Modern Approach. [s.l.] : Prentice Hall, 2003. ISBN 0130851981.**
2. **SNYDER, Wesley E., QI, Hairong. Machine Vision. [s.l.] : Cambridge University Press, 2004. ISBN 052183046X.**
3. **DUDEK, Gregory, JENKIN, Michael. Computational Principles of Mobile Robotics. [s.l.] : Cambridge University Press, 2000. ISBN 0521568765.**

Vedoucí bakalářské práce: **Ing. Erik Král**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato práce pojednává o možnostech řízení mobilního robotického systému NXT Mindstorm pomocí počítačového vidění. Teoretická část je zaměřena na rozdělení mobilních robotických systémů, jejich řízení, popis senzorů a typických úloh robotických systémů. Dále jsou v teoretické části uvedeny základy počítačového vidění, popis robota NXT Mindstorms a jeho programování. Praktická část je zaměřena na konkrétní aplikaci robota NXT Mindstorms Tribot. Je zde popsána práce s vývojovým prostředím Code::Blocks, knihovnami NXT++ a OpenCV. Vytvořil jsem vlastní implementaci řízení mobilního robota pomocí počítačového vidění. Program slouží jako demonstrační aplikace hledání světla v místnostech, může tedy sloužit např. jako pomocník nočních hlídačů.

Klíčová slova: NXT, Mindstorms, robot, C/C++, Code::Blocks, robot control, NXT++, OpenCV

ABSTRACT

This work deals with the control possibilities of the mobile robotic system NXT Mindstorms using computer vision. The theoretical part is focused on the separation of mobile robotic systems, their control, a description of sensors and the typical problems of robotic systems. Furthermore, given the theoretical basis of computer vision, description robot NXT Mindstorms and his programming. The practical part is focused on a specific application NXT Mindstorms robot Tribot. It described the work with the development environment Code::Blocks, libraries NXT++ and OpenCV. I have created my own implementation of the mobile robot control with computer vision use. The program serves as a demonstrating application for finding light in the rooms, for example therefore it may serve as an assistant for night guard.

Keywords: NXT, Mindstorms, robot, C/C++, Code::Blocks, robot control, NXT++, OpenCV

Touto cestou bych chtěl poděkovat mému vedoucímu bakalářské práce Ing. Eriku Královi za odborné vedení, za poskytování pomoci a rad během mé práce. Dále bych chtěl poděkovat svým rodičům za morální podporu a pomoc během psaní této bakalářské práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Práce se zabývá aplikací algoritmů počítačového vidění na řízení robota NXT Mindstorms. V teoretické části bakalářské práce jsou zmíněny základy mobilních robotů, jejich rozdělení, senzory, řízení a jejich praktické využití. V další kapitole teoretické části je zpracována základní část teorie počítačového vidění a zpracování obrazu pro řízení mobilního robotického systému.

V praktické části je uvedena implementace mého vlastního programu pro řízení robota NXT Mindstorms. Pro tyto potřeby byla použita knihovna OpenCV. Aplikace slouží pro vyhledávání světla v tmavých místnostech. Hlavní funkce je navržena jako demonstrace vyhledávání např. ohně nebo pachatelů trestné činnosti na základě vyhledávání zdroje světla. Z těchto důvodů bude možno aplikaci použít jako pomocný nástroj pro střežení objektů nočními hlídači.

I. TEORETICKÁ ČÁST

1 MOBILNÍ ROBOTI

Jsou roboti, kteří se mohou pohybovat na kolech, na pásech nebo například vzduchem.

Z pohledu řízení se dělí na :

- autonomní : robot rozhoduje sám za sebe, bez vnějšího zásahu
- vzdáleně řízené : robot je řízen vzdáleně člověkem nebo programem

1.1 Možnosti pohybu mobilních robotů

V současné době je k dispozici mnoho návrhu pro pohyb mobilních robotů, ať už se jedná o pohyb na souši, na vodě nebo třeba i ve vesmíru.

- pozemní pohyb : jsou roboti, kteří mají kontakt se zemí a mají prospěch z

gravitace, pohybují se převážně pomocí kol, ale je také možné aby např. chodili, šplhali nebo používali koleje.

- pohyb ve vodě : jsou roboti, kteří jsou určeni pro vodní prostředí. Mnoho těchto

robotů využívá tryskový pohon popř. vrtule.

- pohyb ve vzduchu : jejich konstrukce napodobuje letadla popř. ptáky. Stejně jako

roboti zkonstruovaní pro pohyb ve vodě potřebují dodávat energii, aby zůstali ve stacionárním stavu.

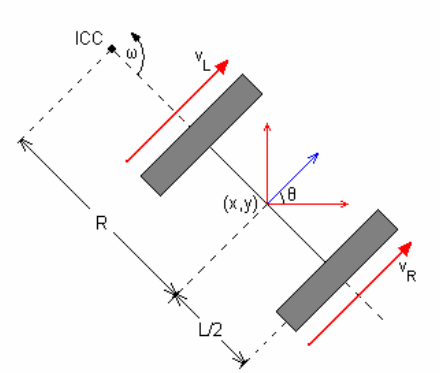
- pohyb ve vesmíru : jsou uzpůsobeni pro operování v mikrogravitaci a ve vesmíru.

1.1.1 Koloví mobilní roboti

Pohon mobilních robotů pomocí kol je nejčastější a ekonomicky nejlepší řešení pohybu mobilních robotů. Díky jejich nenáročnosti a jednoduchosti návrhu. Kola mohou být aktivní nebo pasivní. Aktivní kolo poskytuje robotovi pohon popř. možnost rotace, kdežto pasivní kolo slouží jako stabilizace robota.

- diferenciální řízení : je to nejjednodušší možný způsob řízení mobilního robota.

  Toto řízení se skládá ze dvou kol (každé ovládá jiný motor) a jednoho nebo dvou koleček pro udržení stability robota.[1] Tato konfigurace umožňuje robotovi rotaci na místě po vertikální ose a tím poskytuje větší manévrovatelnost. [5]



Obrázek 1 : Schéma diferenciálního řízení

- synchronní řízení : u tohoto typu řízení je použito více než 3 kol. Všechna kola jsou mechanicky propojena, tzn. všechna kola se pohybují stejným směrem a stejnou rychlostí. Toto řízení zvyšuje přesnost hrubého odhadu polohy tím, že všechna kola vytváří paralelní hnací sílu a redukují prokluzovaní kol. [5]

- trojkolový podvozek : tři kola, obvykle dvě kola vzadu zajišťují pohon a kolo vepředu zajišťuje zatáčení.[1]

- Ackermanovo řízení : řízení, které se objevuje u aut. Obvykle čtyři kola – dvě přední zajišťují zatáčení, dvě zadní zajišťují pohon.[1]

1.2 Senzory mobilních robotů

Senzory podávají mobilním robotům informace o okolním světě a tím umožňují jejich rozhodování.

- kontaktní senzor (nárazník) : používá se pro zaznamenání kolize mobilního a jiného objektu. Většinou je řešen pomocí mikrospínače, který podává binární informaci o stavu vypnuto/zapnuto.

- vnitřní senzory : do této skupiny patří senzory poskytující robotovi informaci o jeho rychlosti, směru a podobně. Zástupci této skupiny jsou akcelerometry, gyroskopy, kompasy a inclinometry.

- infračervené senzory : základní princip je, že infračervený vysílač vyšle infračervený puls a infračervený přijímač detekuje odražený signál. Vzdálenost cíle se poté počítá ze síly odraženého signálu.

- sonar : zvuková navigace a zaměřování se používá taktéž na měření vzdálenosti, ale na rozdíl od např. radaru se využívá jako vysílaný puls ultrazvuková vlna. Na základě doby navrácení, fáze signálu a jeho frekvence se určuje vzdálenost cíle.

- radar : využívá se k měření vzdálenosti. Jako emitovaný signál se používá signál o vysoké frekvenci. Stejně jako u sonaru se vzdálenost počítá z odraženého signálu od objektu.

- laserový dálkoměr : dělí se na tři druhy dle využitých vlastností laseru :
    - vyměřování : využívá geometrických vazeb mezi vyzářeným a přijatým zářením.
    - doba letu : využívá doby odrazu záření od objektu
    - fáze : využívá se měření fáze mezi vysílaným a odraženým zářením

- satelitní systém (GPS) : je založen na zjišťování pozice pomocí satelitů, které obíhají na orbitě Země. Využívá se 21 orbitálních satelitů, které posílají signál jednotce a ta pak měří a počítá dobu letu signálu. Na základě těchto informací lze určit přesnou polohu cíle.

- biologické senzory : do této skupiny senzorů patří :
    - vizuální senzory : jsou reprezentovány kamerou např. webovou kamerou. Vidění je uskutečňováno pomocí obrazové funkce, na kterou se aplikují algoritmy počítačového vidění.
    - magnetické senzory : používá se přirozené magnetické pole Země.
    - pachové senzory : používají se pro pachové rozpoznávání chemických látek nebo sledování stop.[1]

1.3 Praktické využití mobilních robotů

Mobilní roboti se využívají v mnoha odvětvích každodenního lidského života, kde obstarávají nebezpečnou práci. Práci na místech, kde by člověk nemohl pracovat např. z důvodů radiace, chemického znečištění nebo výbušného prostředí. [1]

- doručování
- inteligentní vozidla
- asistent řidiče
- systém konvoje
- autonomní řídící systém
- autonomní dálnicový systém
- roboti pro průzkum a inspekci
- důlní roboti
- vesmírní roboti
- autonomní letadla
- zneškodňování bomb a min
- podmořská inspekce
- těžba dřeva
- pomoc postiženým osobám
- zábavní průmysl
- roboti starající se o úklid

1.4 Lego NXT Mindstorms

Robot Lego NXT Mindstorms se skládá ze 3 částí. Je to řídící jednotka, senzory a akční členy. Lego NXT Mindstorms je možné připojit k počítači pomocí USB kabelu popř. pomocí bezdrátové technologie Bluetooth.

- řídící jednotka : Je mozkem robotů NXT Mindstorms.
    - tři řídící porty pro motory
    - čtyři porty pro senzory
    - USB port pro připojení k PC a Bluetooth
    - reproduktor
    - ovládací tlačítka
    - LCD maticový display 100 x 64
    - zdroj : 6 AA článků

- senzory :
    - kontaktní senzor : Jednoduchý senzor vracející dvě hodnoty pravda nebo nepravda.
    - zvukový senzor : Senzor má možnost detekovat zvuk ve dvou režimech
        - adjusted decibel : přizpůsobení senzoru na frekvenční pásmo pro člověka slyšitelné.
        - standard decibel : všechny zvukové frekvence i neslyšitelné.

- světelný senzor : Senzor detekuje jas odraženého světla. Může pracovat ve dvou režimech, a to v aktivním popř. pasivním. Aktivní režim spíná červenou diodu, čímž zvyšuje odrazivost tmavých objektů.

- sonar : Senzor měřící vzdálenost v centimetrech popř. palcích. Pracuje na principu odrazu a příjmu vyslaného vysokofrekvenčního signálu. Dle doby vyslání a příjmu se měří vzdálenost od objektu. Detekovaný objekt může být ve vzdálenosti až 255 centimetrů. Překážku detekuje s přesností ± 3 cm.

- akční členy :

  - servo motor : Elektricky řízený pohon robota. Servo motory mají vlastnost přesného nastavení polohy hřídele. Servomotor u NXT dosahuje rychlosti až 200 otáček za minutu a přesnost nastavení hřídele až 1°. Každý z motorů NXT Mindstorms je vybaven rotačním senzorem pro určení rychlosti motoru.[2]

1.5 Programování robotů Lego NXT Mindstorms

Možností programovaní robota Lego NXT Mindstorms je mnoho. Knihovny existují snad pro všechny dnes běžně používané programovací jazyky jako je C/C++, Java, Python, Lua, Perl, Forth, Visual Basic, atd.

Knihovny :

- brickOS – programovací jazyk C/C++
- Lego.NET – programovací jazyk C#
- Lego::NXT – programovací jazyk Perl
- leJOS – programovací jazyk Java

- NXT_Python – programovací jazyk Python
- pbForth – programovací jazyk Forth
- ruby-nxt - programovací jazyk Ruby
- XS – programovací jazyk Lisp
- LegoLog – programovací jazyk Prolog

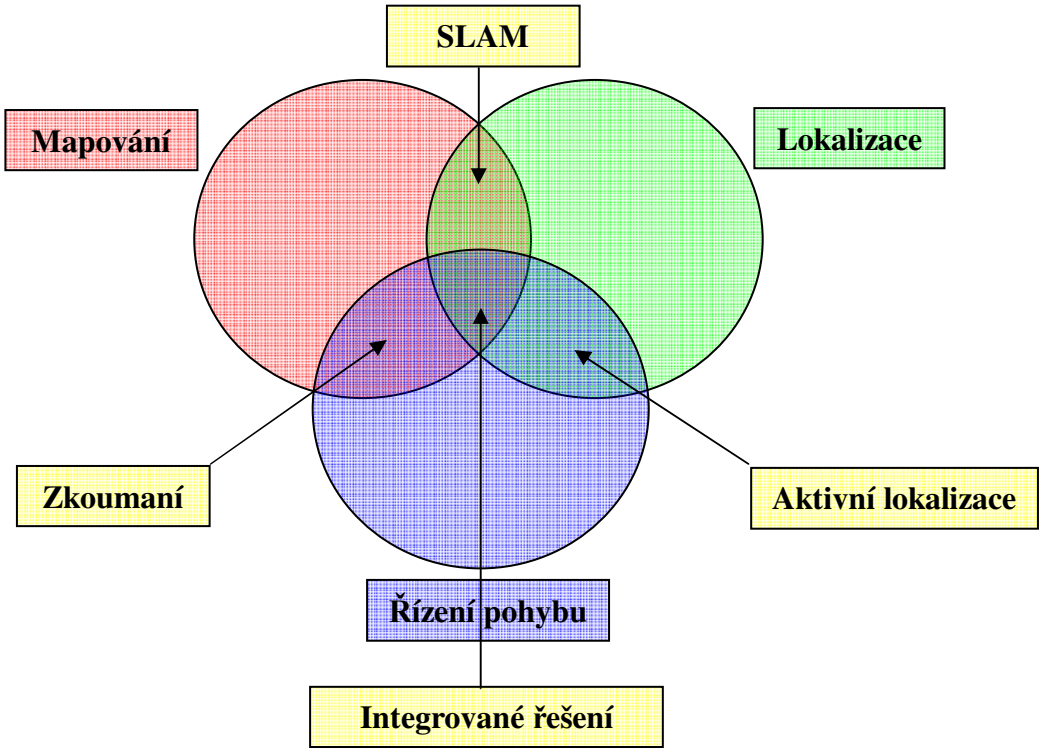
K programování robota Lego NXT Mindstorm byla použita knihovna NXT++ a programovací jazyk C/C++. [3]

Knihovna NXT++ se skládá z osmi jmenných prostorů :

- Comm : zde jsou obsaženy funkce pro práci s komunikací
- nFANTOM100 : funkce pro ovládání Fantom v.: 1.0 rozhraní
- NXT : obsahuje funkce pro práci s NXT např. otevření spojení atd.
- NXT::File : funkce pro práci ze soubory
- NXT::Module : funkce pro práci s moduly
- NXT::Motor : funkce pro práci ze servomotorem na NXT
- NXT::NxtCam : funkce pro práci s NxtCam funkcemi
- NXT::Sensor : funkce pro práci se senzory na NXT

2 TYPICKÉ ÚLOHY MOBILNICH ROBOTŮ

Typickými úlohami mobilních robotů je vyhýbaní se překážkám, lokalizace, plánování trasy, tvorba map, komunikace, atd…



Obrázek 2 : Schéma typických úloh mobilních robotů

2.1 Vyhýbání se překážkám

Opravdu autonomní roboti mají možnost při svém volném pohybu např. z místnosti do místnosti uhýbat překážkám. Jestliže detekují překážku, jsou schopni naplánovat alternativní trasu.

- navigační referenční senzory : typicky vyžadují velké úhlové pokrytí anebo přesné měření velké vzdálenosti.
- senzory pro vyhnutí se kolizi : pracují s menší vzdáleností. Poskytují dostatečné pokrytí a poskytují robotovi dostatek prostoru pro zatáčení popř. zastavení a naplánovaní alternativní trasy. [5]

2.2 Lokalizace

Je to vlastnost robota najít sebe sama uvnitř prostředí. První možností lokalizace robota je umístit robota na předem známé místo a podle jeho pohybu v prostředí určovat přibližné místo kde se právě nachází. Tato lokalizace může být vylepšena pomocí map. Pokud robot zná svou přibližnou pozici, může pomocí mapy a detekčního měření na nějaký objekt zpřesnit svou polohu.[6]

2.3 Mapování

Mapování je vlastnost robota vytvořit mapu prostředí, ve kterém se nachází. K tomuto cíli využívá robot senzorů popsaných v kapitole 1.2 . Na základě měření z těchto senzorů zapisuje robot data do mapy. Tato mapa může být reprezentována v metrickém popř. v topologickém formátu.

- metrický zápis : je to 2D prostor, kam robot zapisuje svou polohu a kolizní předměty ze senzorů. Maticový (metrický) zápis mapy je citlivý na šum, ale jednodušší na realizaci.
- topologický zápis : popisuje místa a vztahy mezi nimi. Mapa je uložena jako graf, kde uzly představují místa a oblouky popisují cestu.[8]

2.4 SLAM

Jedná se o současné mapování a lokalizaci. SLAM je spojen s problémem vytváření map v neznámém prostředí, když se robot ve stejný čas snaží navigovat v prostředí pomocí mapy.

Slam se skládá z mnoha dílčích částí jako je :

- vyhledávání orientačních bodů
- sdružování dat
- odhadnutí stavu
- aktualizace stavu
- aktualizace orientačních bodů

Existuje mnoho cest jak řešit každou z těchto dílčích částí SLAMu.[7]

3 ZPRACOVÁNÍ OBRAZU

3.1 Získání obrazu

2D obraz získaný z kamery lze zapsat pomocí tzv. obrazové funkce $f(x, y)$, kde argumenty x, y udávají souřadnice v rovině. Obvykle je v PC obraz zapsán ve formě matice o m řádcích a n sloupcích. Jestliže na monochromatický obraz (tedy na obraz ve stupních šedi) použijeme b bitů, je dán počet úrovní rovnicí $k = 2^b$, kde k je počet úrovní a b je počet bitů. Nejčastěji se používá 8 bitů, což odpovídá $k = 256$ úrovním. Z toho vyplývá, že dostaneme matici, kde každá její hodnota odpovídá jednomu pixelu a velikost této hodnoty odpovídá jasu tohoto pixelu.[4] Dle konvence je 0 reprezentace černé barvy a hodnota 255 reprezentace barvy bílé, každá hodnota mezi 0 a 255 je stupeň šedi.

3.2 Segmentace obrazu prahováním

Segmentace je operace, při níž dochází k rozdělení obrazu na objekty. Segmentace obrazu prahováním je nejjednodušší a výpočetně nejméně náročný postup segmentace obrazu. Postup je takový, že se vybere práh buď interaktivně, nebo pomocí algoritmu. Práh je hodnota, podle které se rozděluje obraz. Podle toho, zda jsme pod hodnotou prahu přiřadíme pixelu např. hodnotu 0, pokud jsme nad prahem, přiřadíme např. hodnotu 255. Aplikací tohoto algoritmu na celou plochu obrazu získáme segmentovaný obraz. [4]

3.3 Detekce hran

Detekce hran je operace, která zajišťuje nalezení hranic zkoumaného objektu v obrazové funkci. Hrana je část obrazu, kde se prudce mění hodnota jasu.

Hrany je možné hledat pomocí:

- hledání extrémů (maxim) - pomocí derivace
- prochází-li druhá derivace nulou
- řešení ve frekvenční oblasti

V prvním případě se derivace aproximuje pomocí konvolučních masek. Nejznámější jsou konvoluční masky typu Sobelův operátor, Robertsův operátor, operátor Prewittové atd.

Na druhé derivaci je založen např. velmi úspěšný Cannyho hranový detektor.[4]

3.3.1 Cannyho hranový detektor

Jedná se vlastně o sadu operací, která je navržena tak, aby dávala nejlepší možné řešení. Nejprve se obraz filtruje pomocí Gaussova filtru, který by měl odstranit velkou část šumu na obraze. Poté je aplikováno prahování, operace, při níž zůstanou na obraze významnější hrany. Následně je použito trasování významnějších hran s tím, že hledáme v okolí hrany vysoké derivace. Pokud jsme našli vysokou derivaci, označíme pixel za hranu.

4 POROVNÁNÍ NÁSTROJŮ POČÍTAČOVÉHO VIDĚNÍ

4.1 OpenCV

OpenCV je knihovna navržená firmou Intel a slouží pro oblasti počítačového vidění a interakce člověk – stroj. Knihovna nabízí hotová řešení úloh počítačového vidění jako je např. rozpoznávání tváře, sledovaní objektu a obecně zpracování obrazu. OpenCV byla uvolněna pro širokou veřejnost zdarma. OpenCV je multiplatformní knihovna a lze ji používat na mnoha operačních systémem jako je Windows, Linux. Knihovna je psaná pro jazyk C/C++, ale díky její rozšířenosti jsou k dispozici i pro jazyk např. C# a Python. Jak již bylo řečeno knihovna nabízí hotová řešení a její používání je velmi jednoduché a intuitivní.[10][11]

4.2 Matlab a Matlab Image procesing toolbox

4.2.1 Matlab

Matlab je numerické výpočetní prostředí a programovací jazyk je vyvinut firmou Mathworks a v současné době je touto firmou udržovaný. Matlab umožňuje uživateli jednoduchou práci s maticemi, vykreslování grafů funkcí, implementaci algoritmů a v neposlední řadě vytváření uživatelského rozhraní.

4.2.2 Matlab Image Processing Toolbox

Je to sada algoritmů jazyka Matlab pro zpracování obrazu, jeho analýzu a vizualizaci.

[9]

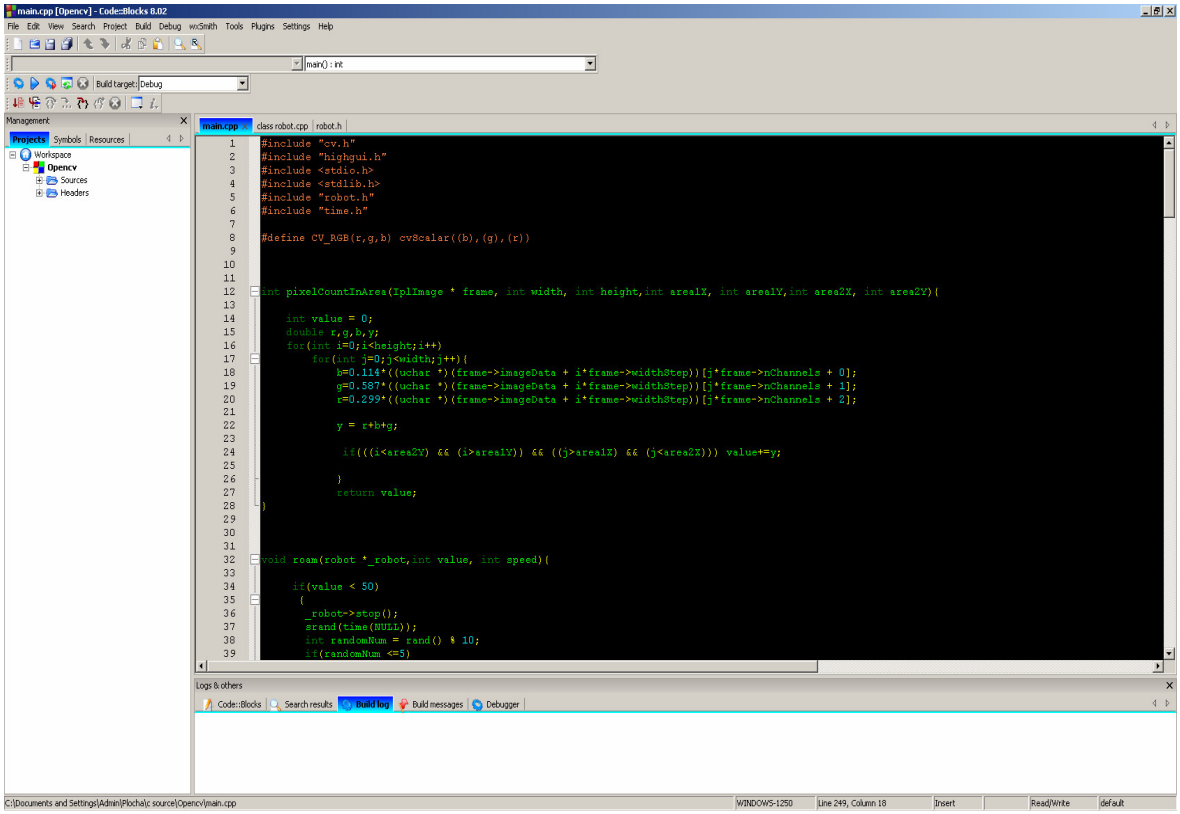
II. PRAKTICKÁ ČÁST

5 OVLÁDÁNÍ ROBOTA NXT POMOCÍ PC

K vytvoření komunikace mezi PC a Lego NXT Mindstorm je možné použít USB kabel popř. technologii Bluetooth. Programování probíhá v jazyce C/C++ a jsou použity knihovny OpenCV pro aplikaci algoritmů počítačového vidění a knihovny NXT++ pro řízení činnosti robota.

5.1 Použité vývojové prostředí

Bylo použito vývojové prostředí Code::Blocks.

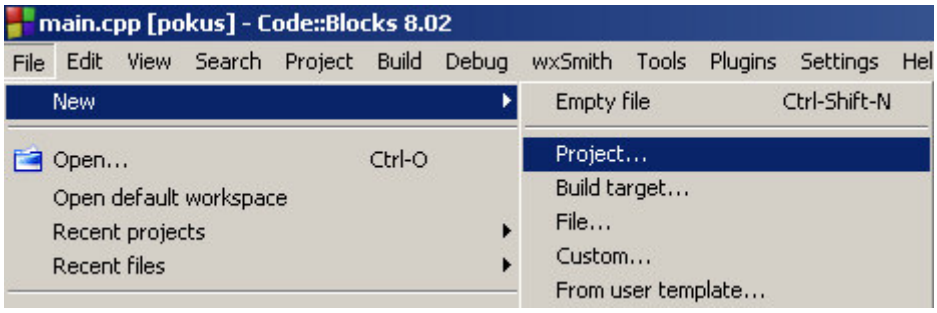


Obrázek 3 : Vývojové prostředí Code::Blocks

Je to vývojové prostředí zejména pro jazyk C++ . Je to open source a multi-platformní IDE. Code::Blocks podporuje více překladačů jako je MinGW / GCC, Microsoft Visual C++, Borland C++.

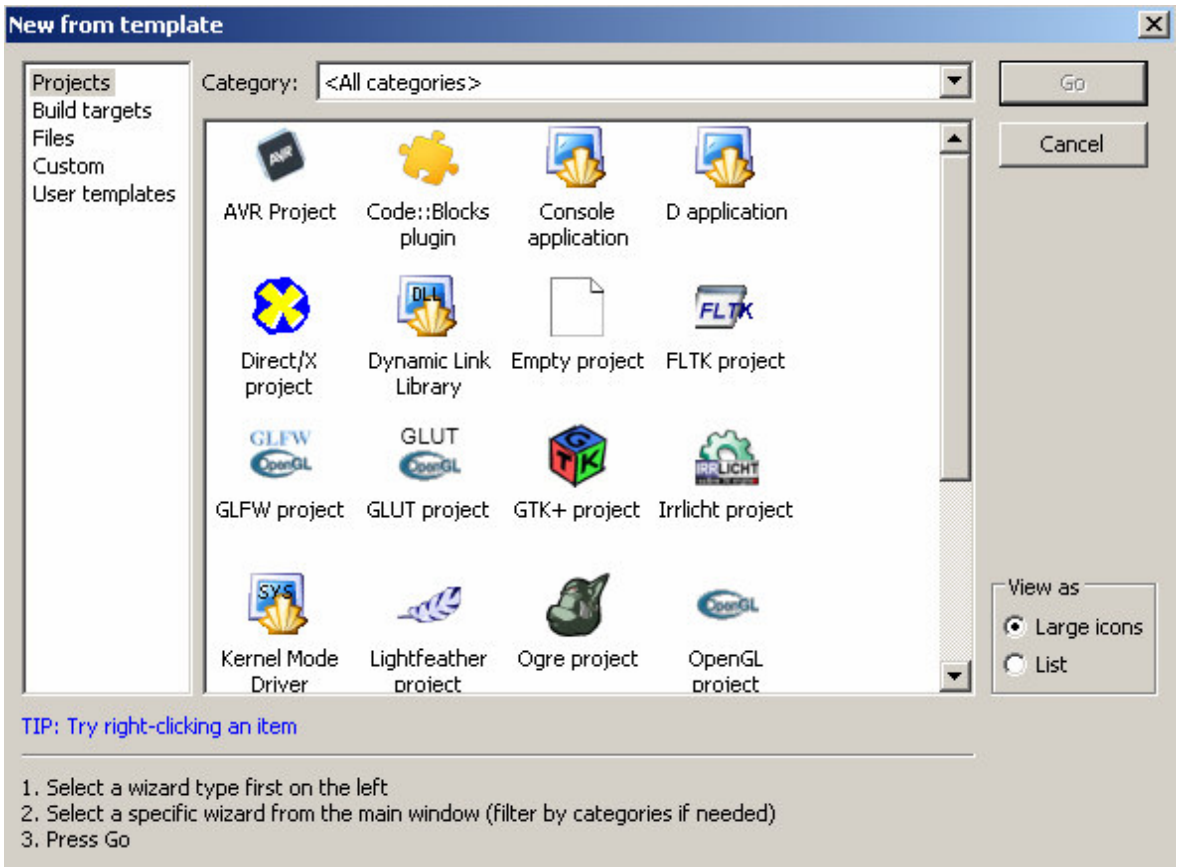
5.2 Vytvoření projektu

Klikneme na položku menu File->New->Project…



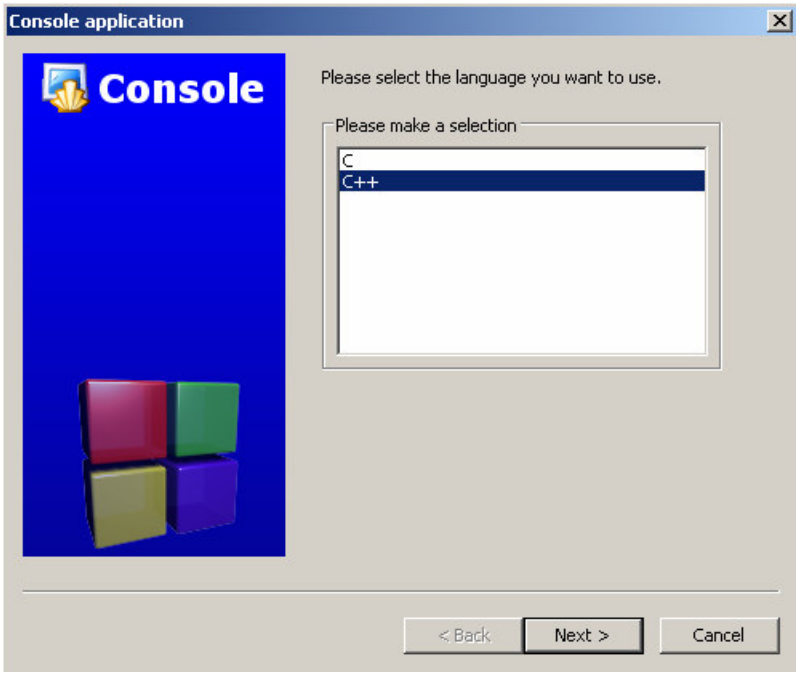
Obrázek 4 : Code::Blocks - nový projekt

Po kliknutí na položku Project… se nám otevře dialog pro nastavení projektu.



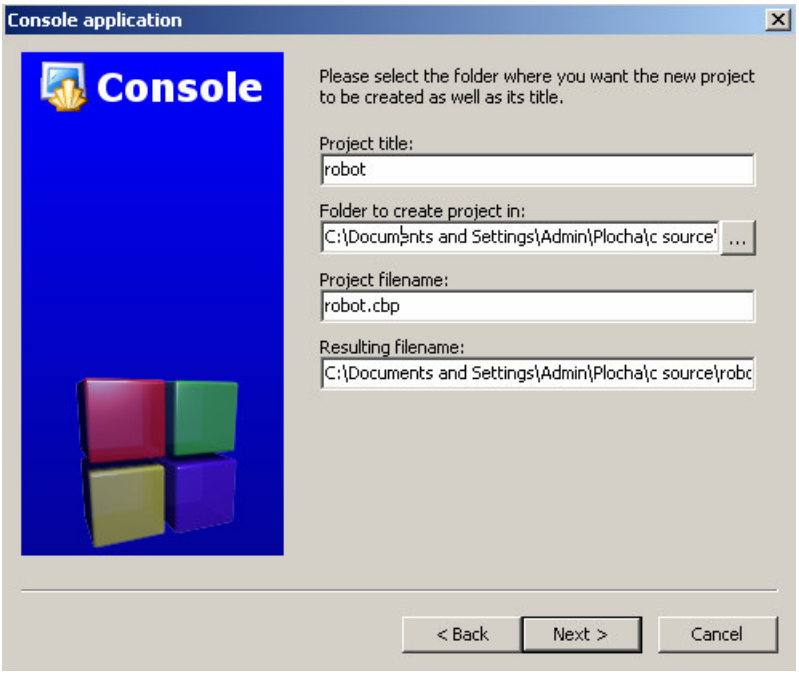
Obrázek 5 : Code::Blocks – výběr typu aplikace

Kde vybereme položku Console application. Po kliknutí na položku se nám zobrazí dialog pro výběr programovacího jazyka.



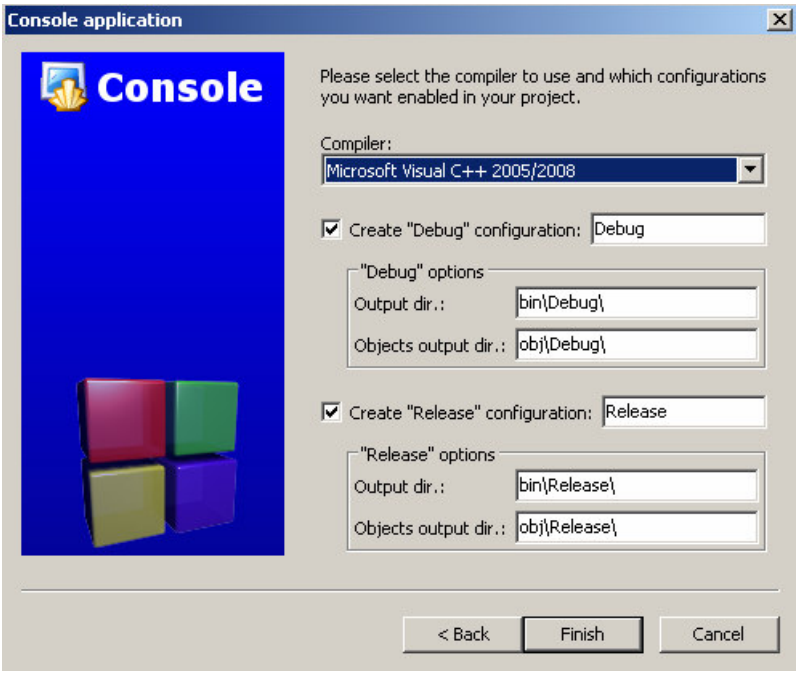
Obrázek 6 : Code::Blocks – výběr programovacího jazyka

Protože je knihovna NXT++ psaná v jazyce C++, vybereme položku C++. Následuje dialog pro výběr jména projektu a adresáře pro jeho uložení.



Obrázek 7 : Code::Blocks – výběr jména projektu

Název projektu zvolme například “robot“. Ještě určíme adresář pro uložení souborů projektu.

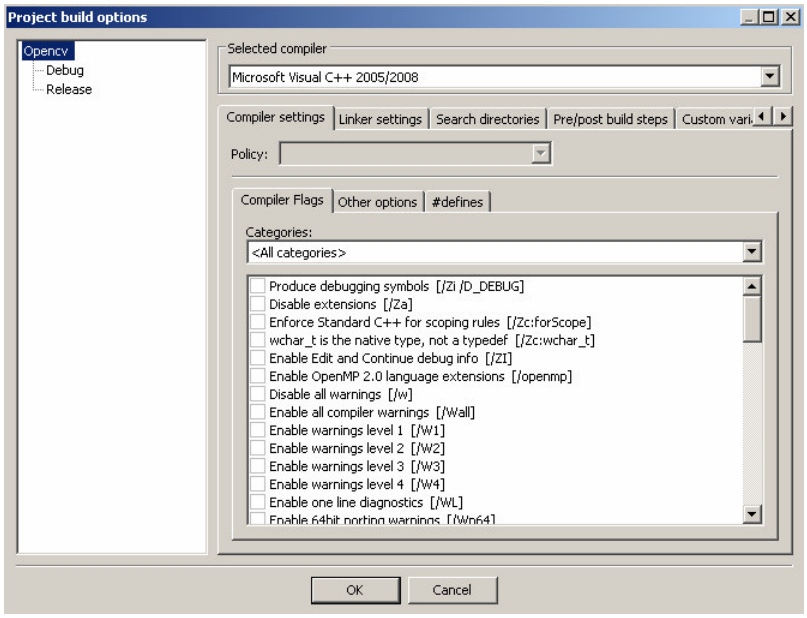


Obrázek 8 : Code::Blocks – výběr překladače

Následuje výběr překladače. Jelikož Code::Blocks podporuje více překladačů máme možnost výběru. Vybereme tedy Microsoft Visual C++ 2005/2008, protože je nutný pro překlad knihoven OpenCV a NXT++. Nastavíme adresáře, kde se uloží spustitelné soubory.

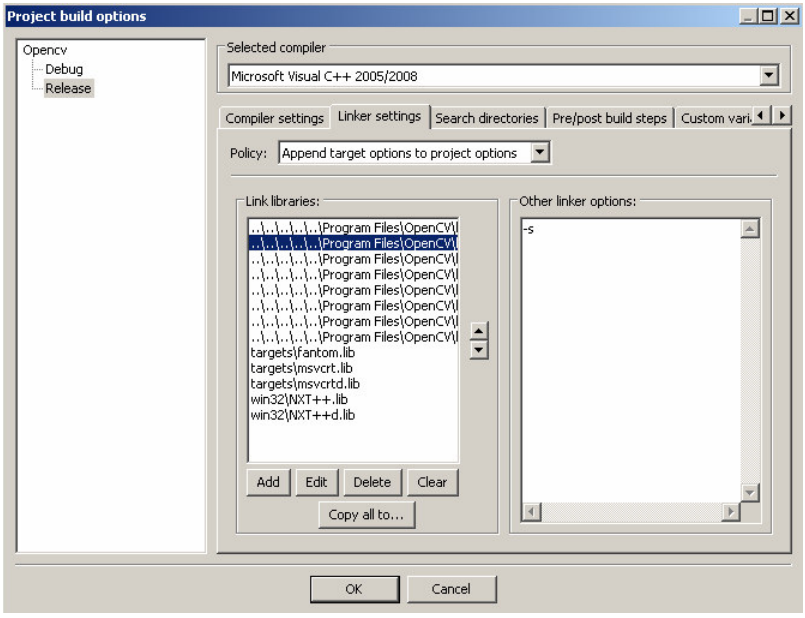
5.3 Nastavení projektu pro použití knihoven OpenCV a NXT++

Klikneme na položku menu Project->Build options… . Code::Block zobrazí dialog nastavení projektu.



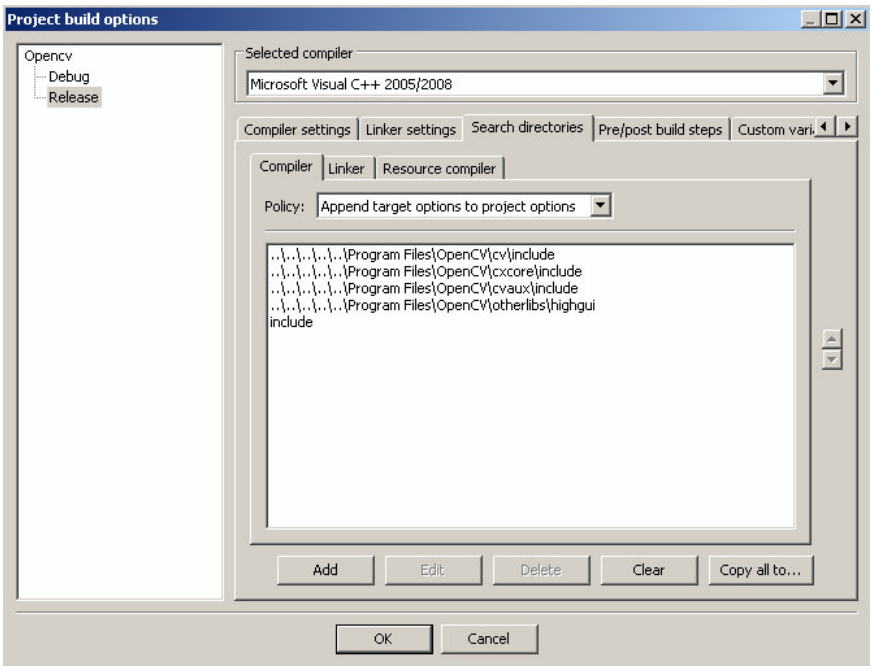
Obrázek 9 : Code::Blocks – nastavení překladače

Jak je vidět, je vybrán překladač Microsoft Visual C++ 2005/2008. Nyní klikneme vlevo na položku Release, kde budeme realizovat nastavení překladače a knihoven pro úspěšný překlad projektu.

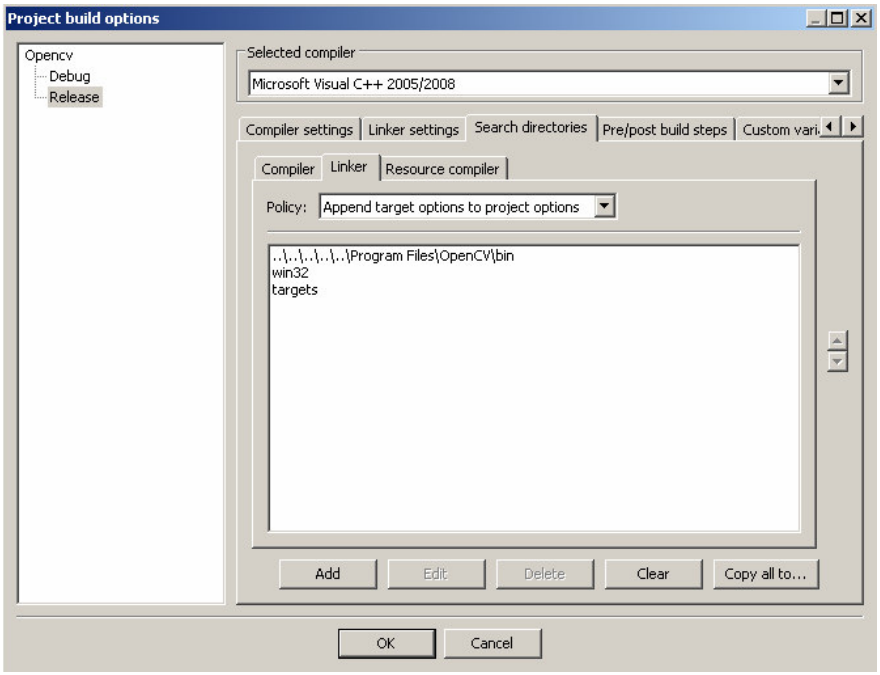


Obrázek 10: Code::Blocks – nastavení linkeru

Do okna Link Libraries vložíme postupně všechny knihovny knihovny OpenCV a knihovny NXT++. Poslední nastavení se týká prohledávaných adresářů pro soubory knihoven a hlavičkových souborů.



Obrázek 11 : Code::Blocks – nastavení adresářů pro překladač



Obrázek 12 : Code::Blocks – nastavení adresářů pro linker

Nyní je projekt připraven na kompilaci a následné spuštění.

6 NÁVRH IMPLEMENTACE

Robot řeší základní úkoly jako je vyhýbání se překážkám a základy počítačového vidění. Projekt je napsán v jazyce C/C++ s využitím vývojového prostředí Code:Blocks. Jsou použity knihovny OpenCV a NXT++. Robot je vzdáleně řízený pomocí operátora a počítače. Řízení robota je uskutečněno diferenciálním řízením. Robot nese řídící jednotku NXT Mindstorms, sonar a webovou kameru Genius Video Cam eye.

6.1 Základní popis programu

Robot má za úkol pomocí operátora vykonávat funkce spojené s počítačovým viděním. Jedná se především o rozpoznávání a následování světla a vyhýbání se překážkám. Operátor obsluhuje 5 módů robota, přičemž každému módu je přidělena určitá funkce. Tyto funkce na sebe navazují, avšak je na operátorovi, jakou funkci bude robot vykonávat. Robot posílá obraz z webové kamery na PC operátora.

Program je rozdělen do pěti souborů :

Main.cpp – obsahuje hlavní smyčku programu

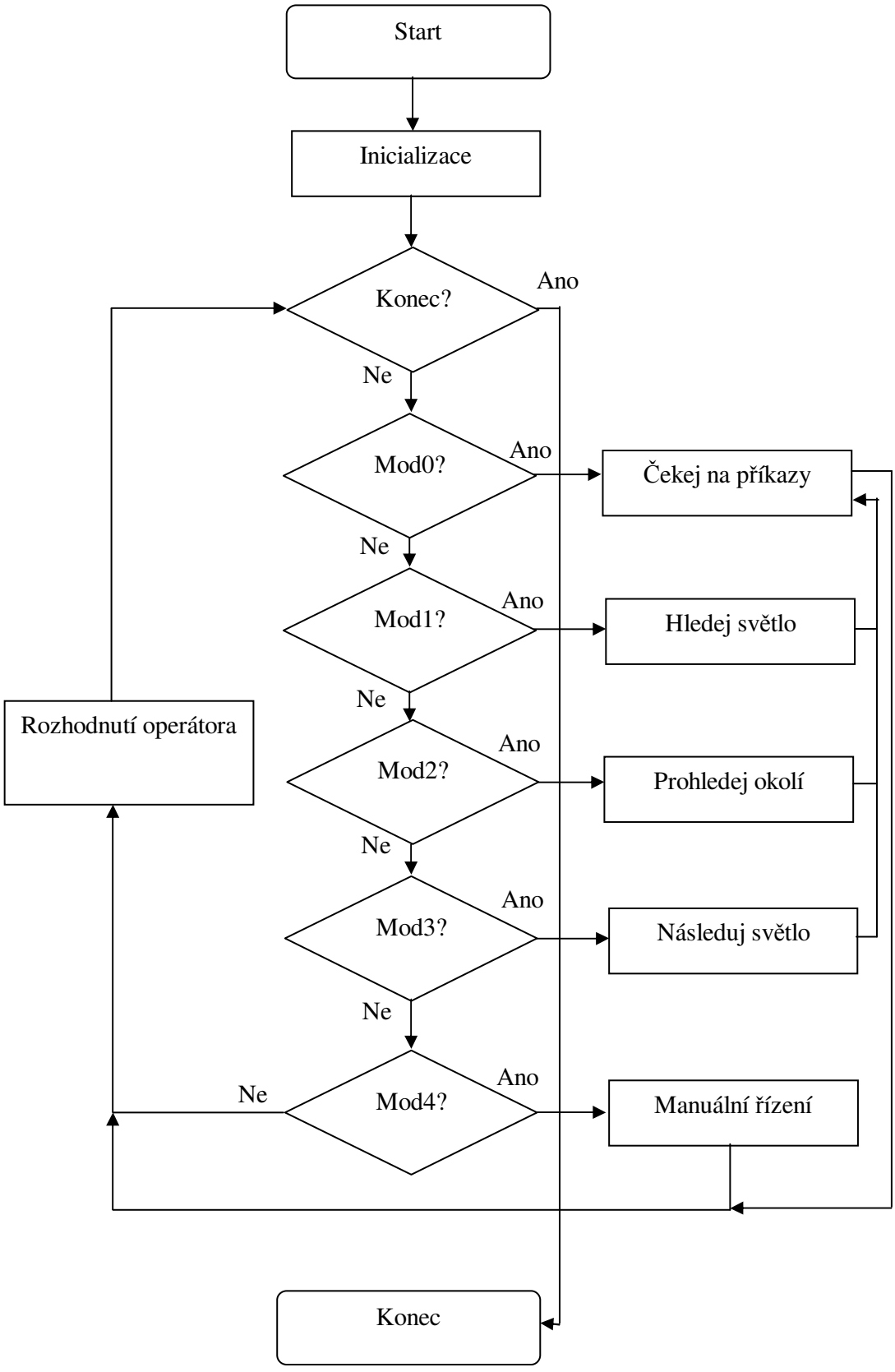
Crobot.cpp – obsahuje implementaci třídy robot pro jednodušší manipulaci s robotem

Function.cpp – obsahuje implementaci pomocných funkcí

Crobot.h – obsahuje prototyp třídy robot

Function.h – obsahuje prototypy pomocných funkcí

Vývojový diagram :



Obrázek 13 : Vývojový diagram aplikace

Jak je zřejmé z vývojového diagramu, program se skládá z pěti základních částí, tedy pěti módů. Tyto módy se přepínají pomocí numerické klávesnice číslicemi 0-4.

6.2 Uživatelské rozhraní

Po spuštění programu se pokusí PC navázat spojení s NXT Mindstorms a webovou kamerou. Jestliže komunikace probíhá správně, otevře se okno OpenCV, kde se aktualizuje obraz z webové kamery. V levém horním rohu tohoto okna je vypsán aktuální mód robota. Po spuštění je nastaven mód 0, tzn. robot stojí a čeká na přepnutí do dalších módů. Na numerické klávesnici lze pomocí číslic 0-4 měnit módy robota.

Módy dle funkce :

- Mód 0 – robot stojí na místě, čeká na přepnutí do dalších módů

- Mód 1 – robot jezdí po místnosti a vyhýbá se překážkám. Jestliže nalezne na obraze příliš světlé místo, přepne se do módu 0 a čeká na zásah uživatele.

- Mód 2 – robot se otočí kolem své osy o cca. 360° a zkoumá intenzitu světla. Po dokončení se natočí na místo, kde je intenzita světla nejvyšší. Jakmile dokončí tuto akci, přepne se do módu 0 a čeká na zásah uživatele.

- Mód 3 – robot následuje nejvyšší intenzitu světla, kterou má aktuálně na obraze. Zastaví se cca. 70 cm před nejbližší překážkou. Jakmile dokončí tuto akci, přepne se do módu 0 a čeká na zásah uživatele

- Mód 4 – robot se nachází v manuálním řízení. Řízení probíhá pomocí klávesnice :

  Písmeno ‘a‘ - robot zatáčí doleva

  Písmeno ‘s‘ - robot zastaví

  Písmeno ‘d‘ - robot zatáčí doprava

  Písmeno ‘w‘ - robot jede rovně vpřed

  Písmeno ‘x‘ - robot jede rovně vzad

Klávesou Esc se program ukončí.

6.3 Popis implementace

Zde popíšu jednotlivé funkce a řešení jednotlivých problémů implementace programu.

6.3.1 Třída robot

Soubor Crobot.cpp obsahuje implementaci třídy robot, která slouží na jednodušší zadávání příkazů robotovi NXT Mindstorms. Protože jsou třídy NXT++ schopny příkazů jako je

NXT::Motor::SetForward(OUT_B,power); , kde parametr OUT_B udává port, na kterém je motor připojen a parametr power udává sílu, se kterou se motor rozjede, je jednoduché vytvořit třídu pro základní diferenciální řízení tak, abychom dostali příkazy :

- jeď rovně
- zahni doleva
- zahni doprava, atd.

Právě na tyto příkazy je vytvořena třída robot, která zjednodušuje výsledné programování. Třída obsahuje metody pro inicializaci robota, řízení pohybu motorů připojených na port B a C, sonaru připojeného na vysokorychlostní port 4, navrácení a resetování počtu otáček motorů a deinicializační metodu.

První metoda je inicializace komunikace PC a NXT Mindstorms.

Jedná se o metodu init (bool USBorBT) , která inicializuje komunikaci na základě parametru USBorBT pomocí rozhraní USB nebo bezdrátovou technologii Bluetooth.

```
int robot::init(bool USBorBT){
    if(USBorBT == true)
      NXT::OpenBT();
    else
      NXT::Open();
    NXT::Sensor::SetSonar(IN_4);
    return(0);
  };
```

Jestliže je parametr USBorBT nastaven na hodnotu pravda, inicializuje se komunikace bezdrátové technologie Bluetooth. Jestliže je hodnota nepravda, inicializuje se komunikace na rozhraní USB. Dále se nastaví sonar na vysokorychlostní port 4. Metoda vrátí 0, jestliže inicializace proběhla v pořádku .

Dále třída obsahuje metody řízení robota.

Jako příklad uvedu metodu straight (int power), která má za úkol rozjet oba motory připojené na porty B a C stejným směrem a o stejné rychlosti dané parametrem power.

```
  void robot::straight(int power){
    NXT::Motor::SetForward(OUT_B,power);
    NXT::Motor::SetForward(OUT_C,power);
  };
```

Je vidět, že metoda je jednoduchá, a že se motory rozjedou stejným směrem a stejnou rychlostí. Jestliže chceme zatočit ať už na jednu nebo druhou stranu, stačí v metodě nastavit 0 na jednom z motorů. Tím získáme pohyb jen jednoho z motorů a robot začne zatáčet. Příklad metody na zatáčení doleva :

```
void robot::left(int power){
    NXT::Motor::SetForward(OUT_B,power);
    NXT::Motor::SetForward(OUT_C,0);
};
```

Dále obsahuje třída metody pro zjištění a resetování hodnot čítačů počtu otáček na jednotlivých motorech.

Jsou to metody motorGetCount(int port) a motorResetCount(int port).

```
int robot::motorGetCount(int port){
    if(port==1) return NXT::Motor::GetRotationCount(OUT_B);
    if(port==0) return NXT::Motor::GetRotationCount(OUT_C);
};

void robot::motorResetCount(int port){
    if(port==1) NXT::Motor::ResetRotationCount(OUT_B,false);
    if(port==0) NXT::Motor::ResetRotationCount(OUT_C,false);
};
```

Parametrem metod je hodnota 1 nebo 0, které navracejí resp. resetují hodnoty čítače otáček motorů.

Ještě je potřeba zmínit metodu sonarValue(), která navrací hodnotu sonaru a tím vzdálenost robota od určitého objektu.

```
int robot::sonarValue(){
    return NXT::Sensor::GetSonarValue(IN_4);
};
```

Poslední metoda slouží pro zrušení komunikace PC a NXT Mindstorms. Jedná se o metodu destroyCom().

```
void robot::destroyCom(){
    NXT::Close();
};
```

6.3.2 Pomocné funkce

Soubor Function.cpp obsahuje pomocné funkce pro potřeby počítačového vidění a funkce určené k jízdě robota, použité v hlavní funkci main(). Funkce pixelCountInArea(IplImage * ,int , int ,int , int) přebírá 5 parametrů a je to ukazatel na strukturu zpracovávaného obrazu. Další 4 parametry vyznačují obdélník, ve kterém se počítá jas pixelů. Funkce slouží pro součet jasu pixelů v zadané obdélníkové oblasti.

```
int pixelCountInArea(IplImage * frame,int area1X, int area1Y,int area2X, int area2Y){
    int value = 0;
    double r,g,b,y;
    int width =frame->width;
    int height =frame->height;
    for(int i=0;i<height;i++)
```

```
      for(int j=0;j<width;j++){
 b=0.114*((uchar *)(frame->imageData + i*frame->widthStep))[j*frame->nChannels + 0];
 g=0.587*((uchar *)(frame->imageData + i*frame->widthStep))[j*frame->nChannels + 1];
 r=0.299*((uchar *)(frame->imageData + i*frame->widthStep))[j*frame->nChannels + 2];
 y = r+b+g;
          if(((i<area2Y) && (i>area1Y)) && ((j>area1X) && (j<area2X))) value+=y;
          }
 return value;
};
```

Funkci je předána struktura obsahující základní data o obrazu, jako je výška a šířka a samozřejmě obrazová data. Data jednoho pixelu jsou rozložena na jednotlivé barvy do proměnných r,g,b a následně pomocí rovnice y=0.299r + 0.587g + 0.114b převedena na pixel v odstínu šedi. Hodnota jasu pixelu se sčítá s ostatními v zadané oblasti. Funkce tedy vrátí součet jasů pixelů v zadané oblasti.

Další funkcí, kterou obsahuje zdrojový soubor Function.cpp je funkce roam(robot *,IplImage * ,int ,int); . Funkce má za úkol řídit robota v prostředí a vyhýbat se překážkám pomocí sonaru. Jakmile funkce zaznamená na obraze určitý součet jasu pixelů, v našem případě hodnotu 9792000, což je polovina součtu jasů u obrazu 320x240, vrátí funkci main() řízení a robot se nastaví do módu 0. Funkci se předávají 4 parametry. První je ukazatel na třídu robot, druhý na strukturu obrazových dat IplImage, další parametr je hodnota sonaru a poslední rychlost robota.

```
int roam(robot *_robot,IplImage * frame,int value, int speed){
   int width =frame->width;
   int height =frame->height;
   int pixelCount=pixelCountInArea(frame,1,1,width,height);
```

```
if(value < 50)
 {
  _robot->stop();
  srand(time(NULL));
  int randomNum = rand() % 10;
  if(randomNum <=5){
     while(value<70){
          value = _robot->sonarValue();
          _robot->left(speed);
           if(value < 20){
             while(value<30){
                  value = _robot->sonarValue();
                  _robot->straight(-speed);
                } } } }
  else {
       while(value<70) {
            value = _robot->sonarValue();
            _robot->right(speed);
             if(value < 20) {
               while(value<30) {
                    value = _robot->sonarValue();
                    _robot->straight(-speed);
                  } } } } }
else {
     _robot->straight(speed);
```

```
  }
if(pixelCount>9792000){
  _robot->stop();
  return 1;
}
else{
   return 0;
}};
```

Vyhýbání se překážkám je řešeno velice efektivně. Základem jsou data ze sonaru, která jsou zpracována v podmínkách "if " následujícím způsobem :

- jakmile je na sonaru hodnota větší než 70 cm, robot jede rovně vpřed.

- jakmile je robot méně jak 70 cm od překážky, program vygeneruje číslo od 0 do 10 a na základě výsledku začne robot zatáčet doprava popř. doleva do té doby, dokud není na sonaru hodnota větší než 70 cm.

- jakmile je robot méně než 20 cm od překážky, začne se pohybovat vzad, dokud není na sonaru hodnota větší než 30 cm. To zapříčiní druhou podmínku, že hodnota sonaru je menší než 70 cm, a to má za následek opětovné generovaní čísla a změnu směru.

- jakmile je na webové kameře zaznamenán součet jasů pixelů větší než 9792000, dojde k zastavení algoritmu a předání řízení módu 0.

Další funkce řeší jízdu robota směrem k nejvyšší intenzitě světla. Jedná se o funkci followLight(robot *,IplImage *,int); . Funkce přebírá 3 parametry : ukazatel na třídu robot, strukturu na obrazová data a rychlost pohybu robota.

```
void followLight(robot *_robot,IplImage * frame,int speed){
      int right=0;
      int left=0;
      int center=0;
      int width =frame->width;
      int height =frame->height;
      right=pixelCountInArea(frame,1,1,width/3,height);
      left=pixelCountInArea(frame,2*(width/3),1,width,height);
      center=pixelCountInArea(frame,width/3,1,2*(width/3),height);
   if((center>left) && (center>right)){
        _robot->straight(speed);
       }
       else{
       if(right<left) _robot->right(speed);
       if(right>left)  _robot->left(speed);
        }
};
```

Funkce rozdělí obraz na 3 stejné části - levou část, středovou část a pravou část. Na každé této části se počítá intenzita jasu pomocí funkce pixelCountInArea. Výsledky v jednotlivých proměnných left, center, right se použijí ve vzájemném porovnání a natočení, resp. jízdě robota k části, která má na obraze největší součet jasu pixelů.

6.3.3 Funkce main()

Obsahuje hlavní smyčku aplikace.

```
int height,width,step,mode=1,scan=1,count=0,value=0,maxValue=0,maxValueCount=0;
bool end=false;
char * text=(char*)malloc(10*sizeof(char));
char ch='0';
uchar * data;
robot * myRobot = new robot();
CvCapture * capture = cvCaptureFromCAM(CV_CAP_ANY);

CvFont font;
double hScale=1.0;
double vScale=1.0;
int    lineWidth=1;
cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX, hScale,vScale,0,lineWidth);
myRobot->init(false);
if(!capture)      {
   fprintf( stderr, "ERROR: Capture is NULL \n" );
   return -1;
  }
cvNamedWindow("Status Window", CV_WINDOW_AUTOSIZE );
```

```
IplImage * frame = cvQueryFrame(capture);
height    = frame->height;
width     = frame->width;
step      = frame->widthStep;
data      = (uchar *)frame->imageData;
```

Zde proběhne inicializace pomocných proměnných, inicializace komunikace robota, inicializace komunikace s webovou kamerou, inicializace hlavního okna knihovny OpenCV a v neposlední řadě inicializace fontu, kterým se budou do okna aplikace vypisovat informace.

```
while(!end){
        IplImage * frame = cvQueryFrame(capture);
        if(!frame){
          fprintf(stderr,"ERROR: frame is null...\n");
          break;
          }
```

Zde začíná smyčka kontrolovaná proměnnou end, která rozhoduje o ukončení celého programu. Do proměnné frame je vložen aktuální obraz kamery, který je následně testován. Jestliže test neproběhne v pořádku, hlavní smyčka se ukončí a program končí.

```
if(mode==0)  {
            myRobot->stop();
            myRobot->motorResetCount(1);
            text="MODE 0\0";
            cvPutText(frame,text,cvPoint(10,210),&font,cvScalar(0,0,0));
```

```
};
```

Testuje se proměnná mode na 0. Jestliže je podmínka splněna, robot se nachází v módu 0. Motory se zastaví, vynuluje se počitadlo otáček u motoru připojeného na port B, což je pravý motor. Následně se do okna aplikace vypíše, že se nacházíme v módu 0.

```
if(mode==1)  {
        if(roam(myRobot,frame,myRobot->sonarValue(),20)==1)  {
          mode=0;
          myRobot->motorResetCount(1);
          myRobot->stop();
         }
        text="MODE 1\0";
        cvPutText(frame,text,cvPoint(10,210),&font,cvScalar(0,0,0));
        myRobot->motorResetCount(1);
      }
```

Jestliže je proměnná mode nastavena na hodnotu 1, pak se vykonává funkce roam popsaná v souboru Function.cpp. Jestliže funkce roam vrátí hodnotu 1, byla dosažena požadovaná intenzita světla, robot se zastaví a přepne se do módu 0, kde čeká na zásah operátora. Jestliže funkce vrátí 0, vypíše se do okna OpenCV, že se nacházíme v módu 1 a robot jezdí a vyhýbá se překážkám.

```
if(mode==2) {
        if(scan==1)  {
          count=myRobot->motorGetCount(1);
          if(count>1500)  {
            scan=0;
```

```
        myRobot->stop();
        myRobot->motorResetCount(1);
       }
     myRobot->left(5);
     text="MODE 2-1\0";
     cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
     value=pixelCountInArea(frame,1, 1,width,height);
     if(maxValue < value)  {
        maxValue=value;
        maxValueCount=count;
       } }
  if(scan==0)   {
     if(((maxValueCount-30)<myRobot->motorGetCount(1))&&
                        ((maxValueCount+30)>myRobot->motorGetCount(1))) {
        myRobot->stop();
        scan=1;
        mode=0;
       }
     else  {
         myRobot->left(5);
         text="MODE 2-2 \0";
         cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
        }   }   }
```

Mód 2 má dvě části. Při první se robot otočí kolem své osy o cca 360° a počítá, kde se nachází největší intenzita světla. Při druhé části využije data z první části a natočí se k největší intenzitě světla.

Tato funkce je řešena pomocí čítače otáček motoru . 360° je dosaženo hodnotou 1500 na čítači otáček motoru. Robot se pomalu otáčí a volá funkci pixelCountInArea, která propočítává hodnoty jasu. Hodnoty se ukládají do proměnné maxValue a zároveň se uloží hodnota natočení pomocí čítače otáček do proměnné maxValueCount. Po dokončení otočení o 360° se robot přepne do druhé části módu 2 a zde využije proměnnou maxValueCount, která udává, ve které části hodnoty čítače byla nalezena maximální intenzita světla. Robot pak jen vykoná určený počet otáček motoru. Je důležité zmínit, že tato hodnotu musí být interval, protože při takové rychlosti robot nedosáhne přesné hodnoty nastavení motoru.

```
if(mode==3)
        {
         if(myRobot->sonarValue()<70) mode=0;
         text="MODE 3 \0";
         cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
         followLight(myRobot,frame,5);
         myRobot->motorResetCount(1);
         cvLine(frame, cvPoint(width/3,height), cvPoint(width/3,0), cvScalar(0,0,0), 1);
         cvLine(frame, cvPoint(2*(width/3),0), cvPoint(2*(width/3),height), cvScalar(0,0,0),
1); }
```

Mód 3 zpracovává funkci následování největší intenzity světla followLight ze souboru Function.cpp . Mód 3 rozdělí okno OpenCV na 3 části. Jedná se jen o vizualizaci dané funkce a na funkci samotnou nemá vliv. Jestliže mód 3 zaznamená překážku vzdálenou 70 cm od robota pomocí sonaru, přepne se do módu 0 a čeká na zásah uživatel.

```
if(mode==4)    {
        if(ch=='w') myRobot->straight(10);
        if(ch=='s') myRobot->stop();
        if(ch=='x') myRobot->straight(-10);
        if(ch=='a') myRobot->left(10);
        if(ch=='d') myRobot->right(10);
        text="MODE 4\0";
        cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
        myRobot->motorResetCount(1);
        ch='0'; }
```

Mód 4 má za úkol řídit robot směrem, který požaduje operátor. Tento mód je řešen pěti if podmínkami s tím, že se vybere požadovaná podmínka na základě stlačení klávesy na klávesnici.

```
cvShowImage("Status Window", frame );
```

Tento příkaz aktualizuje obraz v okně OpenCV.

```
switch(cvWaitKey(10) & 0xFF) {
        case 27: end = true; break;
        case 48: mode=0; break;
        case 49: mode=1; break;
        case 50: mode=2; break;
        case 51: mode=3; break;
        case 52: mode=4; break;
```

```
        case 97:  ch='a'; break;
        case 100: ch='d'; break;
        case 115: ch='s'; break;
        case 119: ch='w'; break;
        case 120: ch='x'; break;
       };
  }
```

Následující podmínka switch s funkcí cvWaitKey zajišťuje odchycení informací z klávesnice. Jak je patrné decimální hodnota 27 je označení pro klávesu Esc. Ta změní proměnnou end na pravdu, a tím ukončí běh programu. Hodnoty 97,100,115,119,120 slouží pro nastavení proměnné ch a tím v módu 4 řídí manuálně robota.

```
cvReleaseCapture( &capture );
cvDestroyWindow("Status Window");
free(text);
myRobot->stop();
myRobot->destroyCom();
return 0;
```

Po opuštění hlavní smyčky programu jsou vykonány příkazy na uvolnění paměti a ukončení komunikace s webovou kamerou a robotem NXT Mindstorms. Funkce main() proběhla v pořádku a funkce vrací hodnotu 0.

ZÁVĚR

Na základě zadání bakalářské práce jsem provedl literární rešerši na téma zpracování obrazu pro řízení mobilního robotického systému. V teoretické části je podán základní popis robotických systémů, jejich rozdělení, řízení, popis senzorů, atd. Dále bylo vypracováno téma zpracování obrazu. Je to obsáhlé téma, proto jsou uvedeny pouze nejdůležitější části teorie zpracování obrazu. V současné době je k dispozici mnoho nástrojů pro řízení robotů a zpracování obrazu. Vybrána byla knihovna OpenCV.

V praktické části jsem navrhl a vypracoval vlastní implementaci řízení mobilního robotického systému Lego NXT Mindstorms. Byly využity knihovny NXT++ pro řízení robota NXT Mindstorms a knihovna OpenCV, která je použita pro potřeby implementace algoritmů počítačového vidění. Program umožňuje řešit základní funkce řízení robotických systémů a funkcí počítačového vidění. Program s robotem NXT Mindstorms může sloužit v tmavých místnostech k vyhledávání světla. Z těchto důvodů se hodí pro vyhledávání ohně, pachatelů trestné činnosti a může pomáhat v zabezpečení objektů nočním hlídačům.

ZÁVĚR V ANGLIČTINĚ

On basis of task for Bachelor thesis I performed literature research on the topic of image processing for the control of a mobile robotic system. In the theoretical part there is a basic description of the robotic systems separation, control, description of sensors etc. The following section deals with the topic of image processing. As it is a comprehensive issue, there are only the most important parts of the image processing theory. Nowadays there are many available tools for robot controlling and image processing. There were selected most common tool OpenCV library.

In the practical part I designed and developed my own implementation of the control in mobile robotic systems Lego NXT Mindstorms. There was used NXT++ library for controlling NXT Mindstorms robot and the OpenCV library for algorithm implementation of computer vision. The program enables us to solve the basic functions of robotic system control and makes some functions of computer vision possible. The program with the NXT Mindstorms robot can be used in dark rooms and it looks for light. For these reasons they are useful for searching fire, criminals and for assisting in security night guards.

SEZNAM POUŽITÉ LITERATURY

[1] DUDEK, Gregory, JENKIN, Michael. *Computational Principles of Mobile Robotics*. [s.l.] : Cambridge University Press, 2000. 280 s. ISBN 0521568765.

[2] *Informace o produktu Lego NXT Mindstorms* [online]. 2009 [cit. 2009-02-14]. Dostupný z WWW: <http://mindstorms.lego.com/eng/New_York_dest/default.aspx>

[3] *Dokumentace knihovny NXT++ na internetové adrese* [online]. 8 May 2009 [cit. 2009-04-28]. Dostupný z WWW: <http://nxtpp.clustur.com/projectdocs/index.html>

[4] DAVID FORSYTH, JEAN PONCE : Computer Vision: A Modern Approach . [s.l.] : Prentice Hall, 2003 ISBN 0130851981

[5] EVERETT, H. R. *Sensors for Mobile Robots: Theory and Application* . [s.l.] : AK Peters, Ltd. , 1995. 528 s. ISBN 1568810482

[6] CHOSET, Howie, et al. *Principles of Robot Motion: Theory, Algorithms, and Implementations* [s.l.] : The MIT Press , 2005. 625 s. ISBN 0262033275.

[7] *Informace o SLAMu* [online]. 2005 [cit. 2009-04-26]. Dostupný z WWW: <http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring-2005/9D8DB59F-24EC-4B75-BA7A-F0916BAB2440/0/1aslam_blas_repo.pdf>.

[8] *Informace o lokalizaci a mapovaní* [online]. 11 April 2009 [cit. 2009-05-20]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Robot_localization>

[9] *Informace o produktu Matlab* [online]. 18 May 2009 [cit. 2009-05-10]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Matlab>

[10] *Informace o knihovně OpenCV na internetové adrese* [online]. 2009 [cit. 2009-05-10]. Dostupný z WWW: <http://www.intel.com/>

[11] *Informace o knihovně OpenCV na internetové adrese* [online]. 19 May 2009 [cit. 2009-05-10]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Opencv>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

GPS Global Positioning System - globální poziční systém

USB Universal Serial Bus – univerzální sériová sběrnice

PC Personal Computer – osobní počítač

LCD Liquid crystal display - displej z tekutých krystalů

SLAM Simultaneous localization and mapping – současné mapování a lokalizace

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÉ KÓDY APLIKACE

```
/**************************
* Začátek souboru main.cpp *
**************************/
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include "crobot.h"
#include "function.h"
int main(){
    int height,width,step,mode=0,scan=1,count=0,value=0,maxValue=0,maxValueCount=0;
    bool end=false;
    char * text=(char*)malloc(10*sizeof(char));
    char ch='0';
    uchar * data;
    robot * myRobot = new robot();
    CvCapture * capture = cvCaptureFromCAM(CV_CAP_ANY);
    CvFont font;
    double hScale=1.0;
    double vScale=1.0;
    int    lineWidth=1;
    cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX, hScale,vScale,0,lineWidth);
    myRobot->init(false);

    if(!capture){
```

```
   fprintf( stderr, "ERROR: Capture is NULL \n" );
   return -1;
  }

cvNamedWindow("Status Window", CV_WINDOW_AUTOSIZE );
IplImage * frame = cvQueryFrame(capture);
height    = frame->height;
width     = frame->width;
step      = frame->widthStep;
data      = (uchar *)frame->imageData;

while(!end){
      IplImage * frame = cvQueryFrame(capture);
      if(!frame){
         fprintf(stderr,"ERROR: frame is null...\n");
         break;
        }

      if(mode==0){
            myRobot->stop();
            myRobot->motorResetCount(1);
            text="MODE 0\0";
            cvPutText(frame,text,cvPoint(10,210),&font,cvScalar(0,0,0));
        };
```

```
 if(mode==1){
    if(roam(myRobot,frame,myRobot->sonarValue(),20)==1){
       mode=0;
       myRobot->motorResetCount(1);
       myRobot->stop();
      }
    text="MODE 1\0";
    cvPutText(frame,text,cvPoint(10,210),&font,cvScalar(0,0,0));
    myRobot->motorResetCount(1);
 }
if(mode==2){
   if(scan==1){
      count=myRobot->motorGetCount(1);
      if(count>1500){
         scan=0;
         myRobot->stop();
         myRobot->motorResetCount(1);
        }
      myRobot->left(5);
      text="MODE 2-1\0";
      cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
      value=pixelCountInArea(frame,1, 1,width,height);
      if(maxValue < value){
         maxValue=value;
         maxValueCount=count;
```

```
            }
          }
        if(scan==0){
          if(((maxValueCount-30)<myRobot->motorGetCount(1))&&
((maxValueCount+30)>myRobot->motorGetCount(1))){
            myRobot->stop();
            scan=1;
            mode=0;
           }
          else{
            myRobot->left(5);
            text="MODE 2-2 \0";
            cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
           }
          }
        }
      if(mode==3){
        if(myRobot->sonarValue()<70) mode=0;
        text="MODE 3 \0";
        cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
        followLight(myRobot,frame,5);
        myRobot->motorResetCount(1);
        cvLine(frame, cvPoint(width/3,height), cvPoint(width/3,0), cvScalar(0,0,0), 1);
        cvLine(frame, cvPoint(2*(width/3),0), cvPoint(2*(width/3),height), cvScalar(0,0,0),
1);
       }
```

```
    if(mode==4){
        if(ch=='w') myRobot->straight(10);
        if(ch=='s') myRobot->stop();
        if(ch=='x') myRobot->straight(-10);
        if(ch=='a') myRobot->left(10);
        if(ch=='d') myRobot->right(10);
        text="MODE 4\0";
        cvPutText(frame,text,cvPoint(10,210), &font, cvScalar(0,0,0));
        myRobot->motorResetCount(1);
        ch='0';
      }
cvShowImage("Status Window", frame );
switch(cvWaitKey(10) & 0xFF){
       case 27: end = true; break;
       case 48: mode=0; break;
       case 49: mode=1; break;
       case 50: mode=2; break;
       case 51: mode=3; break;
       case 52: mode=4; break;
       case 97:  ch='a'; break;
       case 100: ch='d'; break;
       case 115: ch='s'; break;
       case 119: ch='w'; break;
       case 120: ch='x'; break;
      };
```

```
    }
    cvReleaseCapture( &capture );
    cvDestroyWindow("Status Window");
    free(text);
    myRobot->stop();
    myRobot->destroyCom();
    return 0;
}
/***************************
* Konec souboru main.cpp *
***************************/

/***************************
* Začátek souboru function.cpp *
***************************/
#include "cv.h"
#include "highgui.h"
#include <stdlib.h>
#include "crobot.h"
#include "time.h"
#include "function.h"

int pixelCountInArea(IplImage * frame,int area1X, int area1Y,int area2X, int area2Y){
    int value = 0;
    double r,g,b,y;
```

```
    int width =frame->width;
    int height =frame->height;
    for(int i=0;i<height;i++)
        for(int j=0;j<width;j++){
     b=0.114*((uchar*)(frame->imageData+i*frame->widthStep))[j*frame->nChannels + 0];
     g=0.587*((uchar*)(frame->imageData+i*frame->widthStep))[j*frame->nChannels + 1];
     r=0.299*((uchar*)(frame->imageData+i*frame->widthStep))[j*frame->nChannels + 2];
            y = r+b+g;
             if(((i<area2Y) && (i>area1Y)) && ((j>area1X) && (j<area2X))) value+=y;
            }
            return value;
};

int roam(robot *_robot,IplImage * frame,int value, int speed){
    int width =frame->width;
    int height =frame->height;
    int pixelCount=pixelCountInArea(frame,1,1,width,height);
     if(value < 50){
       _robot->stop();
       srand(time(NULL));
       int randomNum = rand() % 10;
       if(randomNum <=5){
          while(value<70){
               value = _robot->sonarValue();
               _robot->left(speed);
```

```
            if(value < 20){
              while(value<30){
                   value = _robot->sonarValue();
                   _robot->straight(-speed);
                  }
              }
           }
     }
   else{
       while(value<70){
            value = _robot->sonarValue();
            _robot->right(speed);
             if(value < 20){
                while(value<30){
                     value = _robot->sonarValue();
                     _robot->straight(-speed);
                    }
                }
            }
       }
   }
else{
    _robot->straight(speed);
   }
```

```
    if(pixelCount>9792000){
      _robot->stop();
      return 1;
    }
    else{
       return 0;
    }
};

void followLight(robot *_robot,IplImage * frame,int speed){
      int right=0;
      int left=0;
      int center=0;
      int width =frame->width;
      int height =frame->height;
      right=pixelCountInArea(frame,1,1,width/3,height);
      left=pixelCountInArea(frame,2*(width/3),1,width,height);
      center=pixelCountInArea(frame,width/3,1,2*(width/3),height);

    if((center>left) && (center>right)){
        _robot->straight(speed);
       }
       else{
       if(right<left) _robot->right(speed);
       if(right>left) _robot->left(speed);
```

```
        }
};
/**************************
* Konec souboru function.cpp *
**************************/

/**************************
* Začátek souboru crobot.cpp *
**************************/
#include "NXT++.h"
#include "crobot.h"

  int robot::init(bool USBorBT){
      if(USBorBT == true)  NXT::OpenBT();
      else NXT::Open();
      NXT::Sensor::SetSonar(IN_4);
      return(0);
   };

   int robot::sonarValue(){
       return NXT::Sensor::GetSonarValue(IN_4);
   };

   void robot::straight(int power){
       NXT::Motor::SetForward(OUT_B,power);
```

```
    NXT::Motor::SetForward(OUT_C,power);
};

void robot::left(int power){
    NXT::Motor::SetForward(OUT_B,power);
    NXT::Motor::SetForward(OUT_C,0);
};

void robot::right(int power){
    NXT::Motor::SetForward(OUT_B,0);
    NXT::Motor::SetForward(OUT_C,power);
};

void robot::stop(){
    NXT::Motor::SetForward(OUT_B,0);
    NXT::Motor::SetForward(OUT_C,0);
};

int robot::motorGetCount(int port){
    if(port==1) return NXT::Motor::GetRotationCount(OUT_B);
    if(port==0) return NXT::Motor::GetRotationCount(OUT_C);
};

void robot::motorResetCount(int port){
    if(port==1) NXT::Motor::ResetRotationCount(OUT_B,false);
```

```
        if(port==0) NXT::Motor::ResetRotationCount(OUT_C,false);
    };

    void robot::destroyCom(){
        NXT::Close();
    };
/***************************
* Konec souboru crobot.cpp *
***************************/



/***************************
* Začátek souboru function.h *
***************************/
int pixelCountInArea(IplImage *,int , int ,int , int );
int roam(robot *,IplImage * ,int ,int);
void followLight(robot *,IplImage *,int );
/***************************
* Konec souboru function.h *
***************************/
```

```
/**************************
* Začátek souboru crobot.h *
**************************/
class robot{
  public:
    int  init(bool);
    int  sonarValue(void);
    void straight(int);
    void left(int);
    void right(int);
    void stop(void);
    int  motorGetCount(int);
    void motorResetCount(int);
    void destroyCom(void);
};
/**************************
* Konec souboru crobot.h *
**************************/
```