Mendelova univerzita v Brně
Provozně ekonomická fakulta

Analýza obrazu pro potřeby řízení mobilního robota

Diplomová práce

Vedoucí práce:

Dr. Ing. Radovan Kukla **Bc. Michal Hammerschmiedt**

Brno 2013

Na této straně bude zadání práce

Tímto bych rád poděkoval vedoucímu mé práce, panu Dr. Ing. Radovanu Kuklovi a Ing. Janu Kolomazníkovi, za cenné rady, vstřícný přístup a odborné vedení při vypracování této práce.

Prohlašuji, že jsem bakalářkou práci na téma Analýza obrazu pro potřeby řízení mobilního robota vypracoval samostatně s použitím odborné literatury a pramenů, uvedených na seznamu, který tvoří přílohu této práce.

V Brně dne 21. května 2013 _____

Abstract

Hammerschmiedt, M. Image Analysis for the needs of mobile robot. Diploma thesis. Brno: Mendelu unvezity in Brno, 2013.

The thesis contains analysis of individual libraries and methods for picture recognition, which serves autonomous robot for picture recognition and designatoin of its position on its path. The solution itself is done in Java programming language with aid of artificial neuron network. The goal of this thesis is to create and implement software module for the needs of the autonomous robot. The outcome of the thesis is a module, which is capable of determining position and movement direction of the robot by processing the scanned picture.

Keywords

Java, road detection, autonomous robot, neural network .

Abstrakt

Hammerschmiedt, M. Analýza obrazu pro potřeby mobilního robota. Diplomová práce. Brno: Mendelova univerzita v Brně, 2013.

Práce obsahuje analýzy jednotlivých knihoven a metod pro rozpoznání obrazu, které slouží k rozpoznání obrazu pro účely autonomního robota a určení jeho polohy na cestě. Samotné řešení je provedeno v programovacím jazyce Java za pomocí neuronové sítě. Cílem této práce je vytvořit a implementovat softwarový modul pro potřeby autonomního robota. Výsledkem práce je modul, který je schopen za pomoci snímaného obrazu určit polohu a směr pohybu robota.

Klíčová slova

Java, detekce cesty, autonomní robot, neuronová síť.

Obsah

Seznam zkratek

CMOS	(Complementary Metal-Oxide-Semiconductor) - technologie pou žívaná na převážnou většinu integrovaných obvodů. Používá se na výrobu čipů.
CCD	(Charge-Coupled Device) - elektronická součástka používaná pro snímání obrazové informace
PPS	(Passive-pixel sensors) - pasivní senzor pro CMOS
APS	(Active-pixel sensors) - aktivní senzor pro CMOS
OpenCV	Open Source Computer Vision
RTG -	Rentgenové záření
RGB	(red-green-blue) - barevný model červená-zelená-modrá viz. text
CMY(K)	(cyan-magenta-yellow-(key-balack)) - barevný model viz. text
HSV	(Hue-Saturation-Value) - barevný model viz. text
HSL	(Hue-Saturation-Light) - barevný model viz. text
BSD	(Berkeley Software Distribution, též Berkeley Unix) je odvozeno od Unixu distribuovaná Kalifornskou univerzitou v Berkeley, mající počátky v 70. letech 20. století.
C	je programovací jazyk, který vznikl počátkem 70. let 20. století
C ++	je objektově orientovaný programovací jazyk, který vyvinul Bjarne Stroustrup a kolektiv v Bellových labořích AT&T rozšířením jazyka C
Mac	zkratka používaná pro společnost Macintosh
NIH Image	společnost podporující vývoj knihovny Jimage
Mac OS X	operační systém od společnosti Macintosh, kde X značí verzi jejich systému
TIFF	(Tag Image File Format) - souborový formát pro ukládání rastrové počítačové grafiky
GIF	(Graphics Interchange Format) - souborový formát pro ukládání rastrové počítačové grafiky
JPEG	(Joint Photographic Expert Group) - je standardní metoda ztrátové komprese používané pro ukládání počítačových obrázků ve fotorealistické kvalitě
BMP	(BitMaP) - souborový formát pro ukládání rastrové počítačové grafiky
DICOM	(Digital Imaging and Communications in Medicine) - je standard pro zobrazování, distribuci, skladování a tisk medicínských dat pořízených snímacími metodami jako jsou CT, MRI či ultrazvuk
WWW	(World Wide Web) - celosvětová síť(pavučina) distribuovaný soubor dokumentů v Internetu, lze jej hypertextově prohledávat.
EPL	(Eclipse Public License) - všechny programy, které vydává Elipse, jsou pod touto licencí

JVM	(java virtual machine) - je sada počítačových programů a datových struktur, která využívá modul virtuálního stroje ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java
JAR	slučuje více souborů do sebe viz. text
OSGi	je specifikace dynamického modulárního systému pro programovací jazyk Java
HTTP	(HyperText Transfer Protocol) - je internetový protokol určený pro výměnu hypertextových dokumentů

Orientace v textu

`Příklad kódu` - takto jsou v práci značeny ukázky kódu
`Procedura` - takto jsou v práci značeny názvy procedur a funkcí
`Proměnná` - takto jsou v práci značeny názvy proměnných

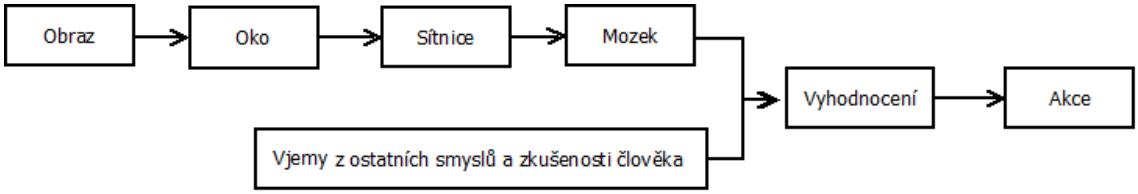
Seznam obrázků

Seznam tabulek

1 Úvod a cíl práce

1.1 Úvod

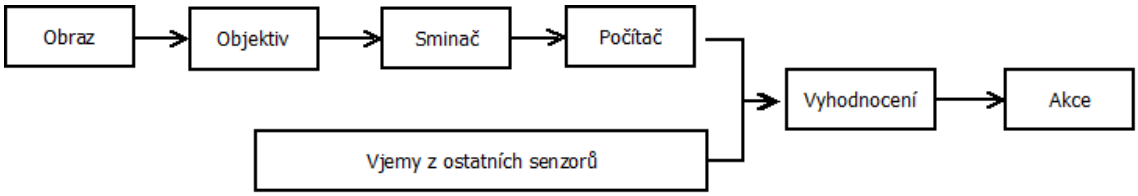
Jedním ze základních lidských smyslů je zrak, využíváme jej k rozpoznávání lidí, zvířat a věcí, přičemž využíváme nejen oka, jako vjemového orgánu, ale i svých zkušeností a znalostí. Na obrázku je velice zjednodušeně znázorněna posloupnost akcí.



Obr. 1 Posloupnost akcí lidského vnímání

Jasová informace z obrazu je zpracována za pomocí oka a sítnice a odeslána do mozku, kde je zpracována na obrazovou informaci. Za pomocí obrazové informace, vjemů z ostatních smyslů a zkušeností dané osoby, se vyhodnotí a podle něj se provede akce.

Při pokusech v technice napodobit vidění člověka vznikl obor počítačové vidění, který si dává za úkol, co nejlépe nahradit posloupnost akcí z Obr. 1. Na obrázku Obr. 2 je vidět, že mezi lidským vnímáním a počítačovým viděním je určitá analogie. Na rozdíl od lidského oka u kamery jsme schopni snímače měnit a můžeme tedy z obrazu získávat různé informace.



Obr. 2 Posloupnost akcí počítačového vidění

Například snímání tepla (termovize), magnetického pole (magnetická rezonance), rentgenového záření (RTG), ultrazvuk (sonary) a v neposlední řadě také jas a barva obrazu (fotoaparáty). Využíváme toho nejen ve výrobě či pro robotické účely, ale hlavně v lékařství, kde počítačové vidění pomáhá odhalovat nádory a další onemocnění a tím dopomáhá k jejich včasnému odhalení a tím zachraňuje lidské životy.

1.2 Cíl práce

Na základě analýzy jednotlivých knihoven pro rozpoznání obrazu vyberu vhodné prostředky, které jsou určené k rozpoznání obrazu pro účely autonomního robota a určení jeho polohy na cestě v přírodním amfiteátru Mendelovy univerzity v Brně. Cílem této práce je vytvořit a implementovat softwarový modul pro tohoto autonomního robota. Tento modul bude schopen určit polohu a směr pohybu robota pro splnění vstupních požadavků.

2 Zpracování obrazu

Zpracování obrazu by se dalo popsat jako převod "analogového" světa kolem nás do elektronické (digitální) podoby pro další zpracování. Tento postup by se dal popsat v šesti krocích. Tyto kroky jsou však obecné a pro rozdílné softwary mohou být různé.

Kroky pro zpracování obrazu:

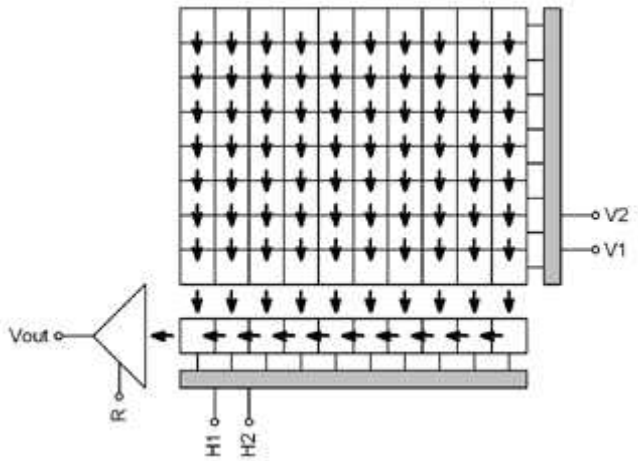
- Snímání
- Digitalizace
- Předzpracování
- Segmentace
- Popis objektů
- Klasifikace

2.1 Snímání

Snímání by se dalo popsat jako převod optické veličiny na elektrický signál, přičemž tento signál bude spojitý jak v čase, tak v úrovni. Proces snímání lze také chápat jako radiometrické měření. Na výsledný obraz má samozřejmě vliv mnoho nejrůznějších faktorů, například jaké je osvětlení snímané scény či objektu. Následně nás zajímají odrazivé vlastnosti snímaného objektu. Pokud ale předem známe některé veličiny, které jsou pro nás nepříznivé a snižují kvalitu snímaného obrazu, můžeme tento stav ovlivnit například zvolením vhodného osvětlovače. Vstupní informací pro snímání nemusí být vždy jen jasová informace pro kameru či scanner, ale také i jiné veličiny jako jsou například intenzita rentgenového záření, ultrazvuk či tepelné záření.

2.1.1 Snímače CCD

Princip CCD je poměrně jednoduchý. Pokud světlo přechází přes polovodič, tak v něm vytváří elektrický náboj. Na čipu jsou vytvořeny negativní potenciálové valy, proto se elektrony nemohou volně po čipu pohybovat. Systém vodorovných elektrod, rovně s negativním nábojem, vytváří na čipu mřížku potenciálových studní, z nichž elektrony nemohou uniknout. Každá studna reprezentuje jeden obrazový bod. CCD má oproti lidskému oku tu výhodu, že dokáže nahromadit světlo i z opravdu slabých zdrojů. (BÍLÝ, 2013)



Obr. 3 Schéma FF čipu (počet vertikálních a horizontálních elektrod se liší v závislosti na architektuře čipu) (BÍLÝ, 2013)

2.1.2 Snímače CMOS

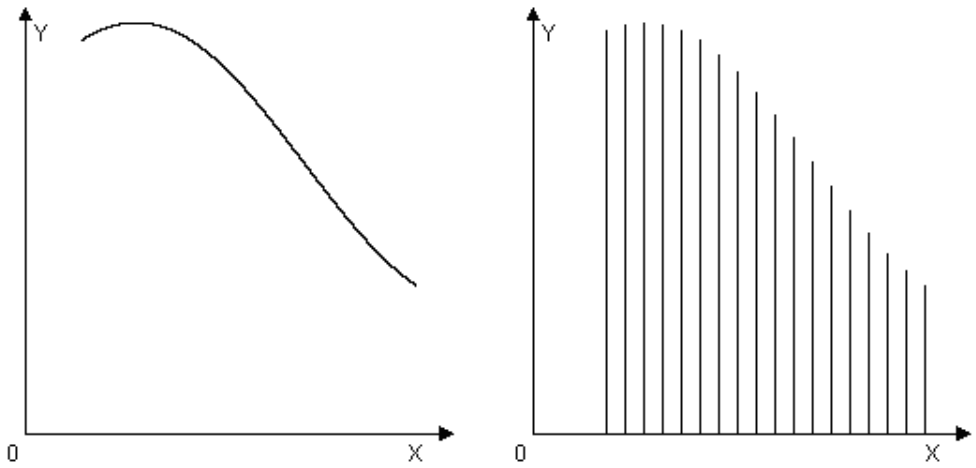
Snímače CMOS se vyrábějí podobnými postupy jako procesory, proto je jejich cena výrazně nižší (asi o třetinu) než u technologie CCD. CMOS snímače navazují na PPS, které generují elektrický náboj, který je úměrný energii dopadajících paprsků. Náboj jde přes zesilovač do analog-digitálního konvertoru jako u běžného CCD. V praxi tyto pasivní CMOS dávají špatný obraz. Proto se v dnešní době využívají lepší aktivní snímače APS. Narozdíl od pasivní technologie je v APS každá světlo citlivá buňka doplněna o analytický obvod, který vyhodnocuje šum a aktivně jej eliminuje. Moderní CMOS mají stejnou kvalitu obrazu jako levnější CCD. CMOS snímače je možné doplnit o integraci specializovaných čipů, například pro kompresi a stabilizaci obrazu. (digineff.cz, 2013)

2.2 Digitalizace

Snímače, které jsou učeny pro vstup obrazové informace, jsou většinou zdrojem spojitého signálu. Abychom tuto informaci mohli zpracovat v počítači, je nutné provést digitalizaci. Obraz je dvojrozměrný. Digitalizace spočívá ve vzorkování obrazu do matice bodů v kvantování spojité jasové úrovně každého vzorku do intervalů. Z tohoto důvodu se jasová informace v digitalizovaných obrazech převádí do celočíselných hodnot. Čím jemnější vzorkování - čím více vzorků uděláme, tím lépe je aproximován původní spojitý obraz. Pokud chceme docílit dokonalého vzorkování spojité funkce, je potřeba rozhodnout dvě věci, určení intervalu vzorkování a výběr vzorkovací mřížky. (home.zcu.cz, 2013)

1. Určení intervalu vzorkování

Interval vzorkování je roven vzdálenosti mezi jednotlivými body ve spojitém signálu. Je otázka jakou vzdálenost vzorků (plošnou vzorkovací frekvenci) využít. Tímto problémem se zabývá Shannonova věta o vzorkování. Vzorkovací frekvence se většinou v praxi volí dvakrát větší než výsledná vzorkovací frekvence vypočtená Shannonovou větou. Interval vzorkování se musí volit tak, aby byl menší nebo rovný polovině rozměru nejmenších detailů v obraze. Při zpracování obrazů je rozumné vzorkovat alespoň 5 krát jemněji, než je teoretická mez daná vzorkovací větou. (home.zcu.cz, 2013)

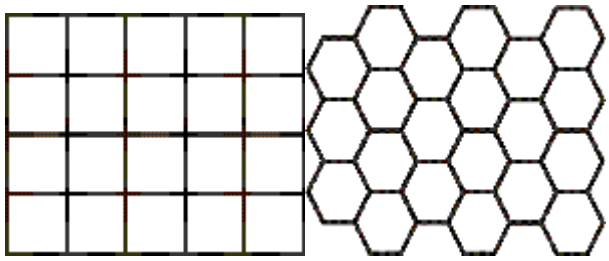


Obr. 4 Ukázka vzorkování

2. Výběr vzorkovací mřížky

Jedná se výběr vhodného uspořádání bodů při vzorkování. Ve většině případů se využívá pravidelná mřížka. Existují pouze tří pravidelné mnohoúhelníky, jejichž složením jsme schopni dokonale pokrýt rovinu a to rovnostranné trojúhelníky,

čtverce a pravidelné šestiúhelníky. V praxi se nejčastěji využívá čtvercová mřížka, i když je příčinou problémů se spojitostí oblastí. (home.zcu.cz, 2013)



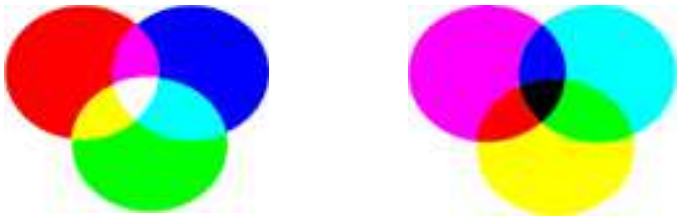
Obr. 5 Příklady vzorkovacích mřížek: (a) čtvercová, (b) hexagonální. (home.zcu.cz, 2013)

2.2.1 Reprezentace barev

Barevný model je model, který popisuje základní barvy a určuje, jakým způsobem, smísením těchto základních barev, lze dosáhnout barvy požadované.
V přírodě kolem nás je barva dána směsí světla různých vlnových délek. Barevné modely se snaží co nejvěrněji tyto barvy napodobit. V praxi se setkáváme s volbou mezi dvěma limitujícími parametry a to mezi přesností a podáním barevného dojmu a složitostí modelu. (dmp.spsei.cz, 2013)

Základní dělení barev je dvou kategorií:

- Adaptivní míchaní barev – smícháním všech barev vznikne barva bílá (pracuje se světlem) využívají jej například monitory nebo projektory.
- Subadaptivní míchaní barev – smícháním všech barev vznikne barva černá (pracuje s odrazem světla) využívá jej například tiskárna.



Obr. 6 Adaptivní míchání barev (vlevo), Subadaptivní míchání barev (vpravo) (dmp.spsei.cz, 2013)

RGB

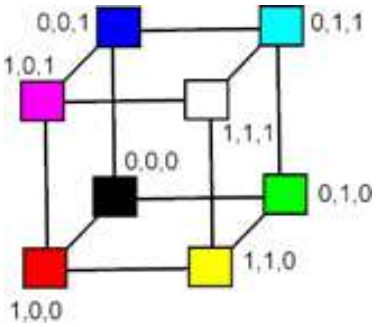
Reprezentace modelu RGB vychází z principu, na kterém je založeno lidské oko, lidské oko se skládá ze tří základních buněk, které jsou citlivé na barevné podněty. Buňky, které dokáží vnímat barvy, jsou citlivé na elektromagnetické záření

určité vlnové délky. Například vlnová délka, která přibližně odpovídá červenému světlu má vlnovou délku 630 nm, červené 530 nm a modré 450. Pokud budeme tyto barvy kombinovat, jsme schopni získat skoro téměř všechny barvy z barevného spektra. (dmp.spsei.cz, 2013)

Nejčastější reprezentace tohoto modelu bývá za pomoci jednotkové krychle. Pokud je na místě barvy, je zastoupena v maximální intenzitě. Častější vyjádření bývá v rozmezí 1 – 255, kde 255 symbolizuje 1 bit. Bity je možné převést na hexadecimální číslo, např. FF0000 je červená barva. Jedná se o adaptivní model.

Základní barvy modelu RGB:

- červená (red)
- zelená (Green)
- modrá (Blue)



Obr. 7 Reprezentace modelu RGB za pomoci jednotkové krychle (dmp.spsei.cz, 2013)

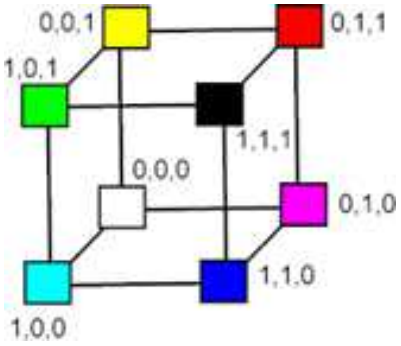
CMY(K)

Reprezentace barevného modelu CMY vychází z lidské zkušenosti míchat barvy. Postup zobrazení barev do modelu CMY je odvozen od míchání malířských či tiskařských barev. Tento model je subadaptivní, proto se využívá především v tiskařské technice. Problém vzniká při nedokonalém krytí jednotlivých barevných složek. Po smíchání všech tří základních barev nevznikne čistě černá barva, ale ve skutečnosti směs hnědé a černé. Proto se tento model doplňuje o další barvu a to černou, která zároveň slouží pro ztmavení odstínů. Takto doplněný model se nazývá CMYK a využívá se v litografii. (dmp.spsei.cz, 2013)

Reprezentace modelu probíhá stejně jako u modelu RGB za pomoci jednotkové krychle.

Základní barvy:

- azurová (Cyan)
- purpurová (Magenta)
- žlutá (Yellow)
- černá (Black), označuje se jako Key



Obr. 8 Reprezentace modelu CMY(K) za pomoci jednotkové krychle (dmp.spsei.cz, 2013)

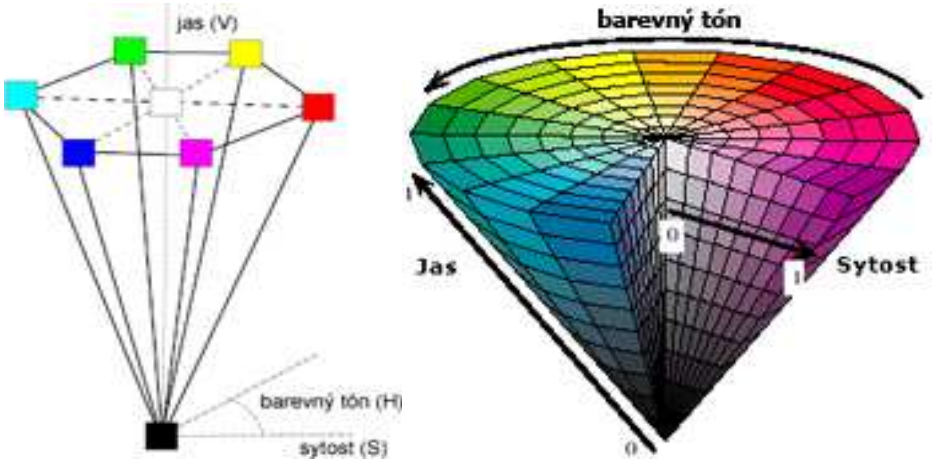
HSV

Předchozí dva modely vychází z technické praxe, kdežto model HSV je věrnější vůči popisu jednotlivých barev, pro člověka je tedy mnohem intuitivnější.

Model HSV pracuje se 3 základními parametry:

- barevný tón (Hue)
- sytost (Saturatin)
- jas (Value)

Barevný tón určuje převládající spektrální barvu, sytost příměsí jiných spektrálních barev a jas příměsí bílé barvy (bílého světla). V některých případech se model HSV označuje jako HSB.
Pro popis modelu HSB a zobrazení barev se na rozdíl od předchozích modelů nevyužívá jednotková krychle, ale šestiboký jehlan umístěný do souřadnicového systému tak, že jeho vrchol je v počátku a osa jehlanu splývá se svislou osou, tato osa zároveň určuje změny jasu. Jas i sytost, které jsou umístěny na vodorovné ose, se mění v intervalu <0,1> – na obvodu podstavy se nachází čisté barvy. Barevný tón je definován jako velikost úhlu, která se měří od osy S proti směru hodinových ručiček – barevný tón může nabývat hodnot 0-360°. HSV model má ovšem bohužel nedostatky: přechod mezi černou a bílou barvou není plynulý a pohyb barevného tónu se neodehrává po kružnici, ale po šestiúhelníku (změna barevného tónu není plynulá). (dmp.spsei.cz, 2013)



Obr. 9 Znázornění modelu HSV za pomoci šestibokého jehlanu (dmp.spsei.cz, 2013)

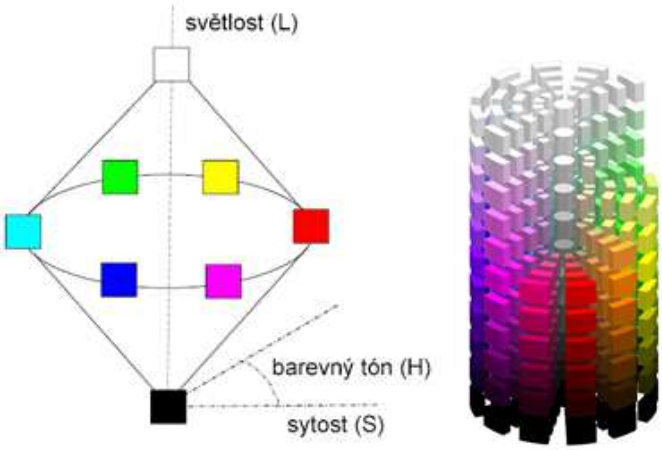
HSL

Barevný model HSL vytvořila společnost Tektronix. Tento model odstraňuje nedostatky modelu HSV, HLS je velice podobné modelu HSV.

Model HSL pracuje se 3 základními parametry:

- barevný tón (Hue) – vodorovná osa
- sytost (Saturatin) – svislá osa
- barevný tón (Lightness) – úhel

Tvar modelu odpovídá více skutečnosti - schopnost rozlišovat barevný odstín klesá ke ztmavování a zesvětlování základní čisté barvy, zvyšování a snižování světlosti spočívá v přidávání světlého nebo tmavého pigmentu. Modely HSV a HLS bývají také nazývány modely psychologickými a psychofyzikálními. Model HLS i HSV, na rozdíl od RGB a CMY, umožňují měnit i jen jeden parametr barvy, zatímco ostatní dva zůstanou zachovány - to je důležité pro počítačové grafiky, tiskaře i kartografy. (dmp.spsei.cz, 2013)



Obr. 10 Znázornění modelu HSL (dmp.spsei.cz, 2013)

2.3 Předzpracování

Předzpracování si klade za cíl odstranění nežádoucích vlivů, které mohou vzniknout při pořizování obrazu.

Cíle, kterých můžeme dosáhnout:

- *Potlačit zkreslení (např. korekce geometrického zkreslení díky zakřivenosti Země u družicového snímku).*
- *Odstranění šumu.*
- *Zvýšení kontrastu (jen pro prohlížení obrazu člověkem).*
- *Zdůraznění charakteristik obrazu pro další zpracování, např. nalézání hran*

(HLAVÁČ, 2013)

Metody, kterými toho můžeme dosáhnout:

- Jasová transformace
- Geometrická transformace
- Filtrace šumu a ostření obrazu

Tab. 1 Metody setříděné podle velikosti zpracovávaného pole (HLAVÁČ, 2013)

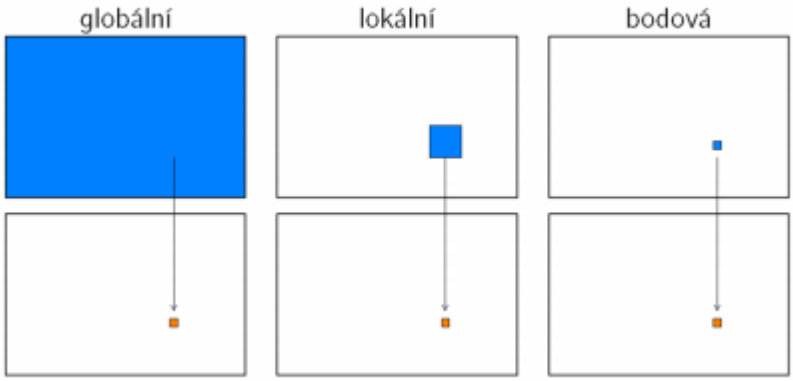
Operace	Zpracovávané okolí
Transformace jasové stupnice	Stejná pro všechny pixely
Jasové korekce	Jen okamžitý pixel
Geometrická transformace	Teoreticky pixel, prakticky malé okolí
Lokální filtrace	Malé okolí
Obnovení (restaurace) obrazu	Celý obraz

2.3.1 Jasová transformace

Jasová transformace slouží ke zvýraznění informací na daném snímku. Tato transformace se provede za pomocí pravidla T, které provede potřebné změny. Jasová transformace však žádným způsobem nemění rozlišení či bitovou hloubku obrazu.

Jsou 3 typy jasové transformace, rozdělují se podle velikosti pole, které je převedeno do daného bodu viz obr. 11:

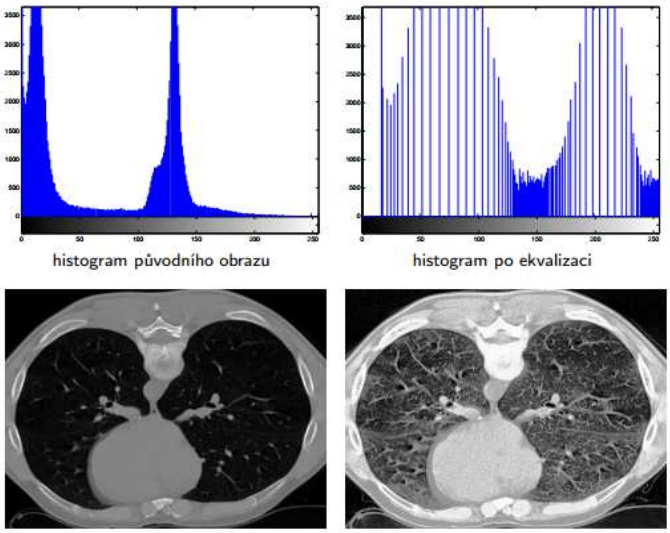
- Globální – vypočítává se z celého obrázku
- Lokální – vypočítává se z ohraničeného prostoru
- Bodové – bod se nahrazuje za bod



Obr. 11 Příklady jasové transformace (HORÁK, 2013)

Ekvalizace histogramu

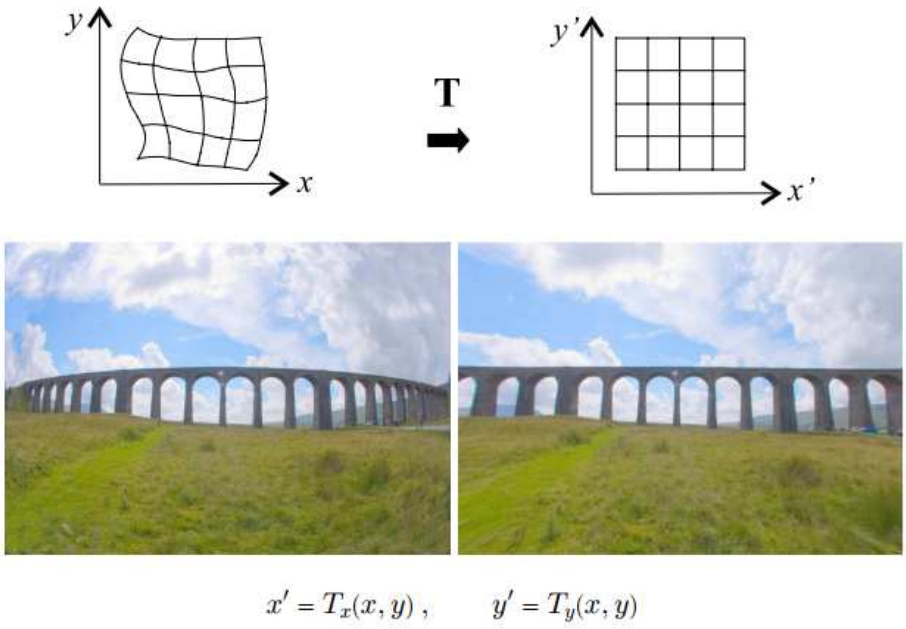
Úkolem této metody je ekvalizovat histogram tak, aby se zvýšil kontrast celého obrazu v rámci dostupné jasové stupnice spektra. Na příklad následují obrázek z praxe, kde vlevo je obrázek před transformací a vpravo po transformaci, obrazová informace je podstatně kvalitnější.



Obr. 12 Ekvalizace histogramu

2.3.2 Geometrická transformace

Tato technika se využívá pro zvětšování, posouvání, pootáčení a zkosení 2D obrázků. Velice podobné techniky se využívají v teoretické mechanice, robotice a počítačové grafice. Transformace se provádí tak, že nad danou souřadnicí x, y se provede vektorová funkce T a ta převede předchozí souřadnice na souřadnice nové a to x´ a y´. Určení funkce T může být při složitějších případech zkreslení složité určit. Ke správné transformaci nám mohou dopomoci znalosti snímané scény. (HLAVÁČ, 2013)



Obr. 13 Geometrická transformace

Pro geometrické transformace je nutné provést dva kroky:

Transformace souřadnic bodů – při tomto kroku se počítá poloha každého bodu v souřadnicovém systému, aby se zjistilo, jestli výsledný bod není mimo rastr. Pro tento účel se souřadnice zapisují homogenním souřadnicovým systémem. Jedná se o základní myšlenku, jak reprezentovat bod ve vektorovém prostoru o jednu větším. (HLAVÁČ, 2013)

Transformační vztah je vyjádřen maticí ve tvaru:

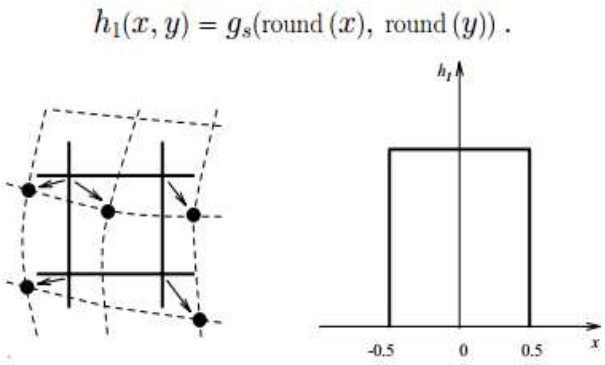
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Obr. 14 Vyjádření homogenní matice (HLAVÁČ, 2013)

Aproximace jasové funkce – Vyjadřuje jasovou hodnotu bodu na souřadnicích x´, y´. Tyto transformované souřadnice velice často leží mimo souřadnicovou mřížku. Z toho důvodu je nutné jasové hodnoty z původního obrazu nějakým způsobem aproximovat. Nejčastěji se používá aproximace polynomem. (HLAVÁČ, 2013)

Další tipy aproximace:

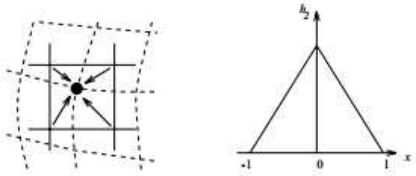
Aproximace nejbližších sousedů – přiřadí bodu (x, y) hodnotu jasu nejbližšího bodu v diskrétní mřížce. (HLAVÁČ, 2013)



Obr. 15 Aproximace nejbližších sousedů (HLAVÁČ, 2013)

Lineární interpolace – z okolí využijeme 4 body a podle předpokladů je lineárně zkombinujeme. Vliv každého ze čtveřice bodů v lineární kombinaci je přímo úměrný jeho ke zpracovávanému bodu. (HLAVÁČ, 2013)

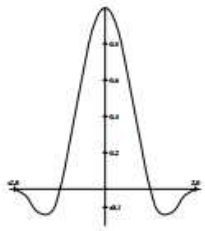
$$
\begin{aligned}
f_2(x,y) = \; & (1-a)(1-b)\, g_s(l,k) + a(1-b)\, g_s(l+1,k) \\
& + b(1-a)\, g_s(l,k+1) + ab\, g_s(l+1,k+1)\,, \\
\text{kde} \quad & l = \text{round}\,(x)\,, \quad a = x - l\,, \\
& k = \text{round}\,(y)\,, \quad b = y - k\,.
\end{aligned}
$$



Obr. 16 Lineární interpolace (HLAVÁČ, 2013)

Bikubická interpolace - lokálně interpolujeme obrazovou informaci bikubickým polynomem z 16 bodů v okolí. (HLAVÁČ, 2013)

$$
h_3 = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{pro } 0 \le |x| < 1\,, \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{pro } 1 \le |x| < 2\,, \\ 0 & \text{jinde.} \end{cases}
$$



Obr. 17 Bikubická interpolace (HLAVÁČ, 2013)

2.3.3 Filtrace šumu a ostření obrazu

Filtrace šumu a ostření obrazu probíhá za pomocí statických metod a klade si za cíl, co nejlépe odstranit šum a rozostření obrazu. Pokud chceme z obrazu odstranit šum, nezbývá, než se spolehnout na nadbytečnost údajů v obraze. Sousední pixely mají převážně stejnou nebo velice podobnou hodnotu jasu. Hodnotu jasu můžeme opravit na základě sousedících bodů, použijeme typického reprezentanta okolí nebo kombinaci několika hodnot. (HLAVÁČ, 2013)

2.4 Segmentace

Obecná definice týkající se segmentace říká, že se jedná o proces, v němž se dělí obraz do částí, které korespondují s určitými objekty v obraze, popsáno jinými slovy, každému obrazovému pixelu, je přiřazen index segmentu vyjadřují určitý objekt v obraze. Pro zjištění o rozdělení obrazu do jednotlivých segmentů využívají vyšší algoritmy určené pro zpracování obrazu. Snaží se porozumět obsahu

obrazu. Jedním z konkrétních úkolů může být detekce přítomnosti příslušného objektu nebo nalezení a klasifikace objektů v obraze. Segmentace je jeden z nejdůležitějších kroků analýzy obrazu. (HLAVÁČ, 2013)
V praxi se využívá několik algoritmů, dělíme je podle přístupu na základní:

2.4.1 Detekce hran

Detekce hran je jednou z několika nejdůležitějších oblastí v nižší úrovni zpracování obrazu, přesto že v reálném čili vnějším prostředí se jedná o velice složitý a dosud nevyřešený problém. Hranami se rozumí ty body v obrazu, u kterých se hodnota jasu prudce změní. Hranu lze chápat jako vlastnost obrazového bodu započteného jako funkci obrazu v okolí tohoto bodu. Hrana není tedy jen reprezentována velikostí, ale i směrem.

Změny či přerušení v obrazu patří k nejzákladnějším charakteristikám obrazu, snaží se naznačit fyzické rozmístění jednotlivých objektů na dané scéně. Pokud se lokálně změní stávající úroveň jasové hodnoty na jinou, jedná se tzv. jasové hrany a jedná-li se o změnu globální, pak je nazýváme jasové hraniční segmenty.

Ideální je, pokud se hrana podobá skokové funkci, ale v reálných obrazech není změna jasu skoková, ale postupná, je tedy vhodnější použít šikmou funkci. Pokud se obě definované funkce objeví v obraze vedle sebe, vznikají další dva typy hran a to čára a střecha. (ŠPANĚL, 2013)

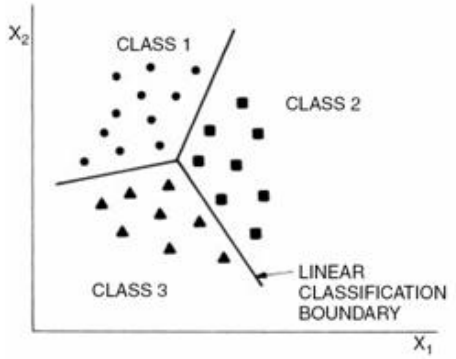
Obr. 18 Popis hran z leva skoková, šikmá, čára a střecha funkce (ŠPANĚL, 2013)

2.4.2 Region-based techniky

Tyto fungují tam, kde by hranové metody selhaly, například pokud je obraz velice zašuměn. Hlavním segmentačním kritériem, pro detekce sektorů (oblastí) v obraze je celistvost těchto celků. Mezi kritéria, celistvosti mohou být: úroveň šedi, barva, textura, tvar, model, apod. Mezi tyto techniky patří Region growing, Split and merge. (ŠPANĚL, 2013)

2.4.3 Statistické metody

Tyto metody pracují na základě statických metod. Jednou z netypičtějších metod je shlukovací analýza. Tato metoda pracuje na základě rozdělní bodů do jednotlivých skupin a tím mezi nimi určí jednoznačně jejich hranice. Další je tzv. prahování, kde se z histogramu určí maximální hodnoty pixelů do T a tím jejich roztřídění do skupin. Další metody jsou Fuzzy Connectedness, Mharkov Random Fields.



Obr. 19 Shlukovací analýza (ŠPANĚL, 2013)

2.4.4 Hybridní metody

Tyto algoritmy vznikli spojením již dříve zmíněných metod. Například metody:

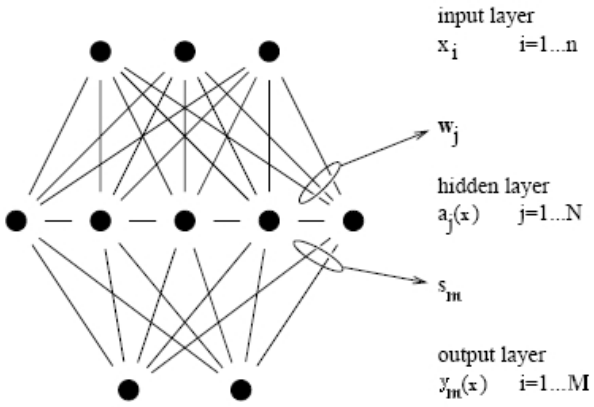
Watershed transform

Tato metoda by se dala zařadit mezi region-based metody. Jedná se o morfologickou metodu segmentace, která je postavena na myšlence pocházející z geografie. Obraz chápeme jako terén nebo topografický reliéf, který je zaplavován vodou. (ŠPANĚL, 2013)

Neuronové sítě

V dnešní době většina metod segmentace obrazu je založena na znalostech a zkušenostech. Opakem je segmentace obrazu za pomocí neonových sítí. Neuronové sítě nejsou založeny na podobných meta-pravidlech. Trénování neuronové sítě (dále jen NN — Neural Network) orientované na data probíhá podle principu "učení příklady". Na druhé straně leží algoritmy analyzující data vzhledem k zadané množině pravidel a příznaků.

Existují dva druhy NN a to s učitelem a bez učitele. První druh NN se snaží klasifikovat data do tříd bez jakékoliv další interpretace. Druhý druh vyžaduje ručně segmentovaná trénovací data. Na vstupu jsou dána data a NN se snaží o klasifikaci. Na konci se podívá, jestli její výsledek byl správný a na tomto základě se upraví váhy. (ŠPANĚL, 2013)

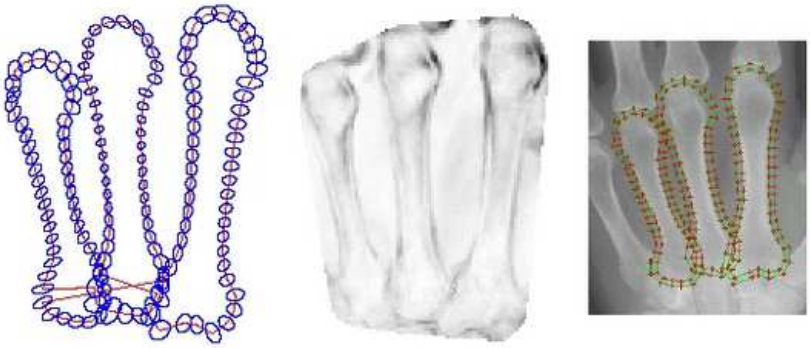


Obr. 20 Schéma neuronové sítě (ŠPANĚL, 2013)

2.4.5 Znalostní metody

Znalostní metody jsou založeny na nějaké předešlé získané znalosti objektů, které se vyskytují v obraze. Tyto jsme schopni reprezentovat buď šablonou objektů, anebo modely objektů. Tato šablona či model se porovnávají s novým obrazem a pokoušíme se najít případné shody. (ŠPANĚL, 2013)

Hlavní nevýhodou této metody je, že pokud porovnáváme složitější objekty, může vzniknout určitá nepřesnost, proto je nutné vhodně zvolit práh, který určuje odlišnosti objektů. Pokud je ale struktura jednoduchá a podobná šabloně, jsme schopni velice dobře jednotlivé předměty rozeznat.

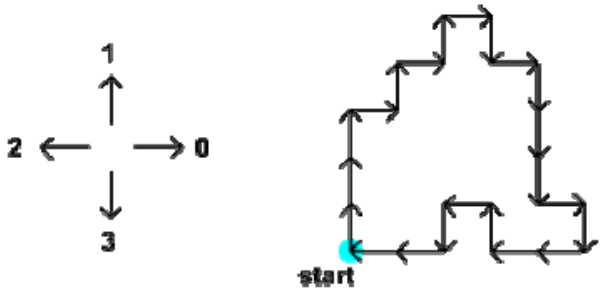


Obr. 21 Znalostní metoda aplikovaná na nalezení článků kostí ruky (ŠPANĚL, 2013)

2.5 Popis objektů

Pokud chceme obraz zpracovávat dále, musíme se pokusit co nejvěrněji popsat objekty, které chceme rozpoznat. K charakterizaci těchto objektů lze použít velkou řadu primitivních metod, které mohou tyto objekty popsat, jsou to například metody popisující jejich tvar, velikost, texturu atd. (Šonka M., Hlaváč V., Boyle R., 2013)

Jedna z těchto metod se nazývá řetězové kódy taky nazývané jako Freemanovy kódy. Tato metoda pracuje na základě skládání objektů z úseček jednotkové délky. Tyto úsečky jsou reprezentovány nejen svou velikostí, ale i směrem. Při reprezentaci objektů pomocí řetězového kódu vyjdeme z předem určeného bodu a snažíme se pohybovat po hranici objektu. Takto zaznamenaný pohyb ukládáme do výstupní věty, kterou lze porovnat s objektem, který hledáme. Příklad na obrázku, kde výstupní slovo W = 11101010303330322123 22. (MIKŠÍK. 2013)



Obr. 22 Matice hran a testovací obraz (MIKŠÍK. 2013)

2.6 Klasifikace

Klasifikace je proces, jehož snahou je zařadit objekty podle určitých pravidel do většinou známých tříd. Objekty, jež jsou obsažené v rámci jedné třídy, jsou si navzájem mnohem podobnější mezi sebou, než objekty v jednotlivých třídách navzájem.

Klasifikátory dělíme do dvou základních skupin:

- **Příznakové** – v této skupině rozpoznáváme za pomocí příznaků, což jsou skupiny charakteristických čísel objektu, jimiž je rozpoznávaný objekt popsán. Často se pro tyto účely využívá shluková analýza.

- **Strukturální** – toto rozpoznávání je založeno na kvalitativním popisu vlastností rozpoznávaného objektu. Na tyto vlastnosti se snažíme aplikovat algoritmy, které souvisí s analýzou slova. Tato slova se snažíme porovnávat s předefinovanou gramatikou. (MIKŠÍK, 2013)

2.7 Používané knihovny

V současné době je dostupné velké množství knihoven, které se zabývají rozpoznáváním obrazu. V této části bude přiblíženo několik z těchto knihoven.

2.7.1 OpenCV

OpenCV je knihovna obsahující funkce pro programování počítačového vidění v reálném čase. OpenCV je uvolněn pod liberální licencí BSD, a proto je zdarma pro akademické i komerční využití. Tuto knihovnu jsme schopni programovat v C ++, C, Python a podporuje operační systémy Windows, Linux, Android a Mac OS. Knihovna má více než 2500 optimalizovaných algoritmů. Byla přijata po celém světě, OpenCV má komunitu čítající více než 47.000 uživatelů a odhadovaný počet stažení přesahující 6.000.000. Použití OpenCV se pohybuje od interaktivního umění, inspekce dolů, spojování map na webu, až po vyžití v pokročilé robotice. (code.opencv.org, 2013)

2.7.2 BoofCV

BoofCV je open source Java knihovna pro real-time počítačové vidění a robotické aplikace. Vyznačuje se snadným používáním a vysokým výkonem. Často předčí i nativní knihovny. Funkčnost zahrnuje optimalizované nízkoúrovňové postupy pro zpracování obrazu, funkce sledování a geometrické počítačové vidění. BoofCV byl spuštěn pod licencí Apache pro akademické i komerční využití. BoofCV je rozdělena do několika balíčků: zpracování obrazu, geometrické vidění, kalibrace, vizualizace a IO. Zpracování obrazu obsahuje běžně používané funkce pro zpracování obrazu, které působí přímo na pixely. Funkce obsahuje algoritmy extrakce příznaků pro použití ve vyšší úrovni operací. Kalibrace má postupy pro určení vnitřních a vnějších parametrů kamery či fotoaparátu. Geometrická představa se skládá z postupů pro zpracování získané z obrazových funkcí pomocí 2D a 3D geometrie. Vizualizace má postupy pro renderování a zobrazování rozbalené funkce. IO obsahuje běžné postupy pro načtení obrázků z různých vstupních zdrojů. (boofcv.org, 2013)

2.7.3 ImageJ

ImageJ je knihovna a program napsaný v Javě, určený pro zpracování obrazu. Program je inspirovaný NIH Image pro Macintoshe. Běží, buď jako on-line applet nebo jako aplikace ke stažení, na libovolném počítači s Javou 1.4 nebo novějším virtuálním stroji. Ke stažení jsou distribuce pro Windows, Mac OS, Mac OS X a Linux.

Je možné zobrazovat, upravovat, analyzovat, zpracovávat, ukládat a tisknout 8-bit, 16-bit a 32-bitové obrázky. Je schopen číst mnoho obrazových formátů včetně TIFF, GIF, JPEG, BMP, DICOM. Podporuje "komíny", série snímků, které sdílejí v jednom okně. Je multithreaded, takže časově náročné operace, jako je čtení obrazových souborů, se provádí souběžně s jinými operacemi.

Je možné v zadané oblasti vypočítat statistiku hodnot pixelů v uživatelsky definovaného výběru. Je možné měření vzdáleností a úhlů. To může vytvořit histogram hustoty. Podporuje standardní funkce zpracování obrázků jako je úprava kontrastu, ostrosti, měkkosti, detekovat okraje a také filtrovat mediány. (rsb.info.nih.gov, 2013)

2.8 Programovací jazyky pro knihovny

V této části budou popsány vlastnosti nejčastěji využívaných programovacích jazyků, které využívají předchozí knihovny.

2.8.1 Java

"Java je objektově orientovaný programovací jazyk, navržený jako přenositelný mezi platformami jako je Windows, Linux, Solaris, atd. Často bývá považován za jazyk určený pouze pro Internet. V Javě lze samozřejmě programovat pro Internet, ale také v ní lze vytvářet aplikace, které pracují s databázemi nebo soubory a s WWW nemají nic společného.
V Javě můžeme vytvářet tři základní druhy programů:

- *aplety – programy běžící v rámci WWW stránky na stanici klienta v prohlížeči*
- *servlety – programy běžící na WWW serveru*
- *Aplikace – programy, které se spouštějí na stanici"*

(DARWIN, 2013)

2.8.2 C++

"C++ je objektově orientovaný programovací jazyk, který vyvinul Bjarne Stroustrup a další v Bellových laboratořích AT&T rozšířením jazyka C. C++ podporuje několik programovacích stylů (paradigmat), jako je procedurální programování, objektově orientované programování a generické programování, není tedy jazykem čistě objektovým. V současné době patří C++ mezi nejrozšířenější programovací jazyky." (pefka.mendelu.cz, 2013)

3 Aktuální stav

V této kapitole popíši aktuální stav řešení pro autonomní vozidla a nalezení cesty.

3.1 Bezpilotní autonomní bojové vozidlo XUV

Experimentální bezpilotní XUV byl vyvinut pro Demo III, zahrnující schopnost řídit autonomně při rychlostech až 60 kilometrů za hodinu (km / h) na silnici, 35 km / h off-road za denního světla, a 15 km / h off-road v noci nebo za špatných povětrnostních podmínek. Kontrolní systém pro vozidlo je navržen v souladu s 4D-Real-time Control System (RCS). Architektura, která rozděluje systém do vnímání světa, modelování a subsystémů řídících chování. XUV má dvě hlavní sady senzorů pro jízdu, jak je znázorněno na obr. 23. Na levé straně zařízení bílé barvy je Ladar systém, který produkuje snímky v rozsahu cca 20 Hz. Výše nad LADAREM je barevný fotoaparát, který produkuje snímky v rozsahu až 30 Hz. Na pravé straně je umístěn pár stereo barevných kamer a sada stereo FLIR kamer. (IEEE, 2013)

Obr. 23 Vozidlo XUV (IEEE, 2013)

3.2 Lindar

LIDAR je optická dálková technologie určená pro průzkum, která může měřit vzdálenost, nebo jiné vlastnosti cíle. Cíl se ozáří laserovým světlem a poté se analyzuje zpět odražené světlo. LIDAR technologie má uplatnění v oblasti geomatiky, archeologie, geografie, geologie, geomorfologie, seismologie, lesnictví, dálkový průzkum Země, atmosférické fyziky, letecké laserové řádkovací mapování (ALSM), laserové měření výšek a obrysy mapování. Autonomní vozidla používají LIDAR pro detekci a vyhýbání se překážkám. Hlavně pak pro bezpečné manévrování přes prostředí.

3.3 Závěr kapitoly

Nalezená řešení se týkají poměrně velkých projektů, které jsou velmi finančně náročné. Většinou jsou investovány z armádních či státních financí. Výsledná řešení, jak po stránce softwarové, tak hardwarové, odpovídají vynaloženým prostředkům. Snad nejvýhodnějším řešením se zdá být projekt americké armády XUV, kde vozidlo je schopné za dobré viditelnosti jet rychlostí až 60 km/h.

4 Metodika

Pro tvorbu softwarového modulu bude použit programovací jazyk Java a jako vývojové prostředí velice oblíbený nástroj Eclipse. K analýze a předzpracování obrazu budou využity knihovny zmíněné v teoretické části.

4.1 Eclipse

Důležitým aspektem Eclipse je zaměření na možnost využití open source technologie v komerčních softwarových produktech a službách. Veřejně podporuje a povzbuzuje dodavatele softwaru, aby používali Eclipse technologie pro tvorbu svých obchodních softwarových produktů a služeb. To je umožněno tím, že všechny projekty Eclipse jsou licencovány pod licencí Eclipse Public License (EPL), komerční OSI schválených licencí.

Eclipse Foundation rovněž podniká řadu kroků, aby se pokusili zajistit původ jednotlivých modulů obsažených v projektu Eclipse. Prvním krok v procesu due diligence se snaží, aby bylo zajištěno, že všechny moduly byly vytvořeny právoplatným držitelem autorských práv pod licencí Eclipse Public (EPL). Všichni vývojáři, jsou povinni podepsat dohodu, která stanoví, že vývojář všechny moduly jsou jeho původní prácí a jsou příspěvkem do EPL.

Druhým krokem je, že zdrojové kódy týkající se všech modulů, které jsou vyvíjeny mimo vývoj Eclipse jsou do procesu zpracována prostřednictvím IP Eclipse Foundation schvalovacího procesu. Tento proces zahrnuje analýzu vybraných částí kódu daného modulu, aby se Elipse pokusilo zjistit, odkud pochází kód a zajistit kompatibilitu s licencí EPL. Moduly, které obsahují kód licencovaný na základě licencí, které nejsou kompatibilní s EPL mají být touto cestou schvalovacího procesu nalezeny a nebudou přidány do projektu Eclipse. Konečným výsledkem je úroveň důvěry, že Eclipse open source projekt uvolní pouze technologie, které mohou být bezpečně distribuovány v komerčních produktech. (eclipse.org, 2013)

4.1.1 Vznik Eclipse

Projekt Eclipse (Eclipse 1.0) vznikl uvolněním kódu IBM pod EPL licencí. Hodnota tohoto příspěvku open source se odhaduje na 40 miliónu dolarů.[3] Pro účely tohoto projektu byl vyvinut grafický framework SWT. Výhodou SWT je nativní vzhled aplikací na každé platformě, kde je SWT portován (SWT využívá nativního kódu operačního systému). Naproti tomu konkurenční framework Swing využívá pouze služby JVM, což umožňuje lepší portovatelnost (omezenou pouze dostupností Javy pro danou platformu).

Eclipse ve verzi 3.0 adaptoval na široce podporovaný standard OSGi R4 (Eclipse project Equinox), čímž získal na atraktivnosti jako vývojová platforma. Technologie balíčků (OSGI bundle ~ Eclipse plugin) umožňuje snadnou rozšiřitelnost produktů. Výhodou této architektury je dynamické nahrávání pluginů až

v okamžiku potřeby, čímž se minimalizují systémové nároky i čas potřebný pro start aplikace. (eclipse.org, 2013)

4.2 JAR

JAR (Java Archive) je platformě nezávislý formát souboru, který agreguje mnoho souborů do jednoho. Java applety a jejich potřebné komponenty (Class soubory, obrázky a zvuky), mohou být dodávány v souboru JAR a následně stáhnuty do prohlížeče v jedné transakci přes HTTP a mohou tím výrazně zlepšit rychlost stahování. Formát JAR také podporuje kompresi, což snižuje velikost souboru. Jedním z dalších zlepšení je čas stahování souboru. Kromě toho může autor appletu digitálně podepisovat tento soubor a tak u jednotlivých položek v souboru JAR je možné ověřit jejich původ. (oracle.com, 2013)

4.3 Použité snímací zařízení

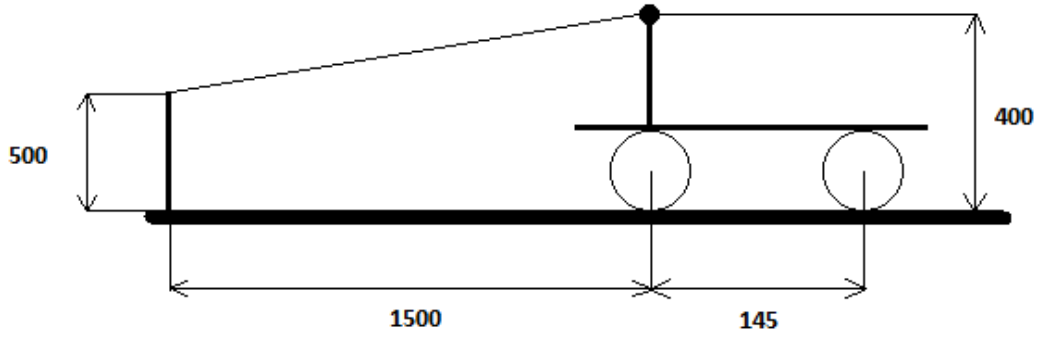
Jako snímací zařízení bude využita webová kamera od společnosti Microsoft LifeCam Studion.

Tato webová kamera LifeCam Studio je osazena snímačem s vysokým rozlišením 1080p, který přináší špičková videa ve vysokém rozlišení. Skvělé video však nezávisí jen na vysokém rozlišení. Velmi podstatnou součástí je i ostření. Tato kamera má automatické ostření, které zajistí ostrý obraz od 10 cm do nekonečna. Precizní skleněný objektiv kamery LifeCam umožňuje natáčet širokoúhlá videa s větší přesností, než bylo do teď možné. A aby bylo vše ještě jednodušší, kamera LifeCam se může pochlubit technologií TrueColor, která přináší jasná a barevná videa téměř za všech světelných podmínek. Dále je kamera doplněna o technologii ClearFrame, která zajišťuje plynulý a detailní obraz. (microsoft.com, 2013)

Obr. 24 Kamera LifeCam Studio (microsoft.com, 2013)

4.4 Umístění kamer

Pozice kamer je dána konstrukcí viz obr 25. Ve schématu jsou uvedeny vzdálenosti v mm. Kamery jsou umístěny na konstrukci ze stavebnice Tetrix, tato stavebnice obsahuje lehké hliníkové profily a přitom je velice modulární. Jedinou nevýhodou této stavebnice je, že je vyráběna v USA, tudíž instalační matriál je v palcových velikostech. Schéma navazuje na reálné fotky umístění kamer, více v příloze A.



Obr. 25 Schéma umístění kamery

4.5 Konstrukce a vzhled robota

Konstrukce robota je provedena na podvozku RC terénního auta, nadstavba na podvozku je vyrobena z již výše popsané stavebnice Tetrix. Vnitřní součástky robota jsou chráněny lepenkovým krytem v barvách naší univerzity. Fotky robota se nachází v příloze B.

4.6 Závěr kapitoly

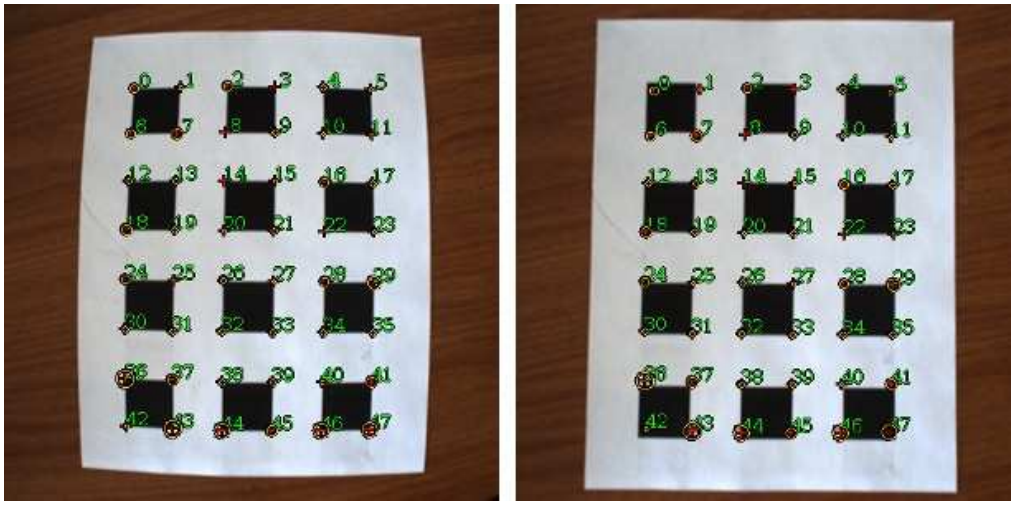
V této kapitole jsou popsány vývojové a technické prostředky, které budou potřebné pro praktickou část diplomové práce.

5 Vlastní práce

Tato kapitola se bude zabývat metodami a postupy, se kterými se budeme v průběhu řešení daného problému setkávat.

5.1 Kalibrace obrazu z kamery

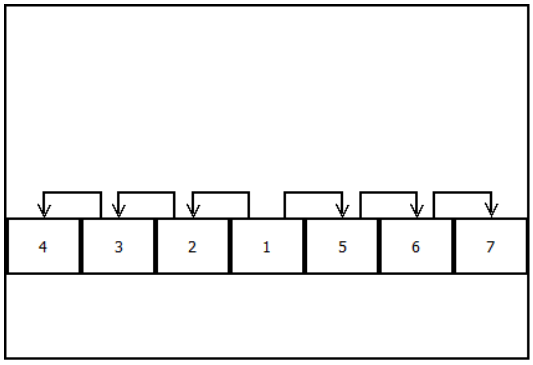
Kvůli takzvanému efektu rybího oka je nutné provést geometrickou transformaci snímaného obrazu. Tuto funkci podporuje výborně knihovna BoofCV. Na obrázku 26 vlevo je vidět, že efekt rybího oka znatelně zkresluje snímaný obraz. Proto se provede úprava, kde se obraz převede podle čtverců na papíře do reálného stavu bez tohoto efektu. Tato úprava je vidět na obrázku vpravo.



Obr. 26 Kalibrace kamery a odstranění efektu rybího oka

5.2 První řešení

První řešení bylo založeno na metodě postupného porovnávání vzorků ze snímaného obrazu. Předpoklad pro toto řešení je ten, že se histogram vzorku cesty bude dostatečně lišit od histogramu okolí. Na obrázku 27 je vidět jak, se jednotlivé vzorky budou porovnávat. Vezme se vzorek číslo 1, který se nachází uprostřed obrazu a vytvoří se z něj histogram. Poté se vytvoří histogram ze vzorku číslo 2 a ty se mezi sebou porovnají. Pokud se vyhodnotí jako podobné, ohodnotí se jako cesta. Dále budeme pokračovat s porovnáváním 2. vzorku se vzorkem 3. atd. tak dlouho dokud nenarazí porovnávání na okolí cesty nebo konec obrazu. Vše se pak opakuje od středu doprava 1. vzorek s 5. vzorkem, 5. vzorek se 6. vzorkem atd. opět dokud nenarazí porovnávání na okolí cesty nebo konec obrazu.



Obr. 27 Posloupnost porovnávání vzorků

5.2.1 Získávání vzorků

Získávání vzorků probíhá za pomocí funkce getSubimage, kterou je možné zavolat nad objektem typu BufferedImage, do kterého se ukládá analyzovaný snímek. Jako parametry této funkce se dávají souřadnice x, y, šířka a výška požadovaného vzorku. Po otestování bylo zjištěno, že nejlepší velikost snímaného vzorku je 60x60 pixelů.

5.2.2 Histogram

V prvním řešení byla aplikována porovnávací funkce, která porovnává jednotlivé čtverce či obdélníky mezi sebou. Pro tuto funkci byla využita implementace vlastního histogramu.

Programovací jazyk Java umožňuje velice jednoduchým způsobem získat hodnotu jednotlivých pixelů v daném vzorku. Následující funkce vytvoří ze zdrojového vzorku (obrázku) histogram pro jednotlivé barvy, určí barvu a její intenzitu.

```
private int[][] histogram(BufferedImage image)
int height = image.getHeight();//zjistíme výšku a šířku vz.
int width = image.getWidth();
Raster raster = image.getRaster();//převede vzorek na
rastrový
int[][] bins = new int[3][256];//vytvoří se pole pro vý-
sledný histogram
      for (int i = 0; i < width; i++)
          for (int j = 0; j < height; j++) {
                 bins[0][raster.getSample(i, j, 0)]++;
                 bins[1][raster.getSample(i, j, 1)]++;
                 bins[2][raster.getSample(i, j, 2)]++;
                 }
return bins;
}
```

Nejprve je nutné daný vzorek, který je reprezentován obrázkem, převést na rastrový. Po tomto převodu je možné nad vzorkem provést funkci `getSample` . Tato metoda vrátí hodnotu 0 až 256, jež reprezentuje intenzitu barvy. Tato hodnota se uloží do pole.

5.2.3 Porovnání vzorků

Porovnání dvou vzorků mezi sebou byla zvolena metoda SSD. Tato metoda je určena k vzájemnému porovnávání jednotlivých histogramů.

Let h and g be two histograms.

$$\|h-g\| = \sum_{i=1}^{N} (h(i)-g(i))^2$$

Obr. 28 Metoda SSD porovnání histogramů

Výsledkem tohoto vzorečku je hodnota rozdílu dvou histogramů, pokud je hodnota rovna 0, tak jsou histogramy totožné, ale čím je výsledek větší, tím jsou histogramy rozdílnější.

5.2.4 Obarvování výsledků

Obarvování vyhovujících vzorků probíhá za pomocí metody `vypln`. Tato metoda v daném prostoru obarví pixely na modrou barvu. Tato metoda má použití informativní a výsledném modulu nebude figurovat.

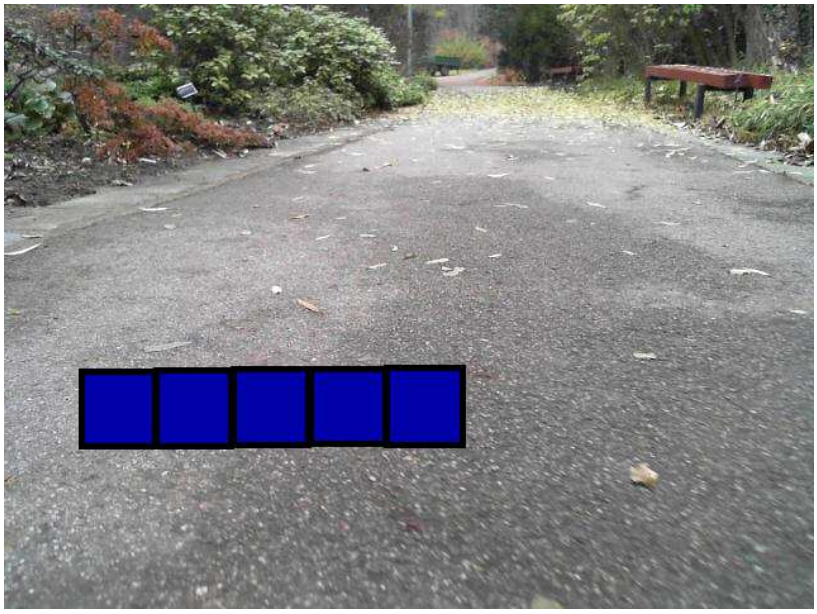
```
BufferedImage vypln(int x,int y,int a,int b,BufferedImage
image){
      for (int j = y; j != y+a; j++) {
          for (int i = x; i != x+b; i++) {
                    image.setRGB(j, i, 425);
           }

             }
return image;
}
```

5.2.5 Výsledky řešení

V této části budou předvedeny výsledky výše popsaného řešení. Na obrázcích jsou modře znázorněna místa, která odpovídají danému algoritmu.

Na následujícím obrázku je zobrazena cesta bez listí a je na ní aplikován předchozí algoritmus. Na výsledku je vidět, že výsledky, které jsou znázorněny modrou barvou, jsou neuspokojivé. Pokud se barva cesty znatelně změní, cesta je například mokrá, tak už algoritmus selhává.



Obr. 29 Výsledky prvního řešení (na čisté cestě)

První obrázku 30 je nasnímána cesta s částečně napadaným listím na cestě a byl na ni aplikován předchozí algoritmus. Jak je zde vidět tato metoda podává poměrně neuspokojivé výsledky. Algoritmus není schopen si poradit s větším či menším množstvím listí ve vzorku.

Obr. 30 Výsledky prvního řešení (na cestě je částečně napadané listí)

První obrázku 31 je nasnímána cesta s větším množstvím listí na cestě a byl na ni aplikován předchozí algoritmus. Jak je vidět zde, tato metoda podává absolutně neuspokojivé výsledky. Algoritmus není schopen si poradit s větším množstvím listí na cestě.

Obr. 31 Výsledky prvního řešení (na cestě je napadáno větší množství listí)

5.2.6 Závěr pro první metodu

Pokud je cesta jednobarevná a bez zbarvení, jako jsou například mokré skvrny či jinak obarvená místa, algoritmus vrací v celku uspokojivé výsledky. Však pokud je cesta zbarvena nebo na ní leží ať už větší či menší množství listí, algoritmus naprosto selhává.

5.3 Druhé řešení

Druhé řešení bylo založeno na rozpoznávání cesty za pomocí neuronové sítě s učitelem. Pokud se pokusíme uvažovat nad řešením, první co se nabízí, je hledání okolí cesty, tak že se budou rozpoznávat stromy, keře atd. Tento přístup je ale velice složitý a příliš náročný na implementaci. Nezbylo tedy, než se zaměřit na jednodušší příznaky. Budou se tedy rozpoznávat jednotlivé vzorky v obraze a proběhne jejich klasifikace, ke kterým částem obrazu patří.

5.3.1 Získávání a obarvování vzorků

Získávání a obarvování bude probíhat jako u předchozího řešení a to za pomocí funkce `getSubimage` pro získání vzorku a metody `vyplň` pro vyplnění výsledných čtverců podle kriterií určených danou metodou.

5.3.2 Histogram

Pro výpočet histogramu slouží níže uvedená funkce `getHistogram` . Tato funkce je obsažena v samostatné třídě Histogram. Tato funkce vytvoří histogram, který je rozdělen na 64 barev. Převod na 64 barev nijak neuškodí kvalitě obrazu, jak je vidět na obrázku 32, který je do 64 barev převeden. Opět je nutné převést obrázek na rastrový, aby bylo možné za pomocí funkce `getPixel` získat informace o jednotlivých pixelech. V proměnné **a** je uložena procentuální hodnota jednoho pixelu z daného vzorku. Poté proběhne výpočet pro každou jednotlivou složku barvy (červenou, zelenou, modrou). Z těchto složek se vypočítá příslušná barva a k té se přičte hodnota proměnné **a**, tedy procentuální podíl pixelu.

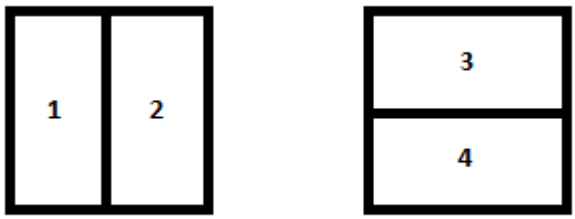
Obr. 32 Paleta 64 barev [20]

```
public static double[] getHistogram(BufferedImage img){
double[] histogram = new double[64];
Raster image = img.getData();
int[] color = new int[3];
int r,g,b;
double a = 1.0/(img.getWidth()*img.getHeight());
     for(int i=0;i<img.getHeight();i++){
          for(int j=0;j<img.getWidth();j++){
          image.getPixel(j, i, color);
          r = (int)Math.floor((double)color[0]/63.75);
          if(r > 3) r = 3;
          g = (int)Math.floor((double)color[1]/63.75);
          if(g > 3) g = 3;
          b = (int)Math.floor((double)color[2]/63.75);
          if(b > 3) b = 3;
          histogram[r*16+g*4+b]+= a;
          }
     }
return histogram;
```

Obr. 33 Převedený obrázek na 64 barev

5.3.3 Velikost a rozdělní vzorků

Pro rozlišení obrazu 640 x 480 byla velikost určovaného vzorku 16 x 16 pixelů, z důvodů kompozice jednotlivých částí ve vzorku. Vybraný vzorek je rozdělen do 4 částí, jak je vidět na obrázku 34. Toto rozdělní vychází z kompozice vzorku. Například pokud je na cestě listí, je nutné určit, kde je list. Může se nacházet v jakékoliv části, na které je vzorek rozdělen. Pro každé z rozdělení se vypočítá histogram, čímž vzniknou čtyři histogramy, každý o 64 barvách. Tímto nám vznikne informace o velikosti 256 (lze ji reprezentovat jednorozměrným polem o velikosti 256 polí), ve které je zakódována informace o daném vzorku. Tímto způsobem je možné jednoznačně identifikovat jednotlivé vzorky.



Obr. 34 Rozdělní vzorků do sektorů

5.3.4 Vstupy a výstupy neuronové sítě

Jak bylo již popsáno v předchozí části, vstupem bude 256 příznaků jednoznačně identifikující daný vzorek. Jako výstup budou předpokládány čtyři kategorie.

První kategorie bude čistá cesta, druhá cesta s listím, třetí okolní zeleň a čtvrtá ostatní okolí.

5.3.5 Implementace neuronové sítě

Pro implementaci sítě využijeme Feed-forward síť, toto schéma neuronové sítě je jednou z nejpoužívanějších. Toto schéma je založeno na tom, že jednotlivé vrstvy jsou složeny z neuronů a ty jsou propojeny mezi sebou navzájem ve schématu každý s každým, viz vizualizace na obrázku 20. Výhoda tohoto řešení je ta, že lze přenos mezi neurony realizovat za pomoci násobení matic, což výpočet velice zjednodušuje a také zrychluje.

Implementace neuronové sítě je provedena ve třídě `NeuralNet`. Při vytváření této třídy je dán konstruktor `NeuralNet(int vrstva1pocet, int vrstva2pocet, int pocetVstup, int pocetVystup)` podle parametrů je jasně patrné, že je možné zadat počet neuronů v obou vrstvách, počet vstupů a výstupů. Jak bylo již dříve zmíněno, váhy budeme reprezentovat za pomocí dvourozměrných polí, jedná se o **`vahy1`**, **`vahy2`** a **`vahy3`**. Matice **`vahy1`** realizuje přenos mezi vstupní vrstvou a první skrytou vrstvou, **`vahy2`** realizují přenos mezi první a druhou skrytou vrstvou, **`vahy3`** realizují přenos mezi druhou a výstupní vrstvou. Dále jsou implementovány matice **`zmenyVah1`**, **`zmenyVah2`** a **`zmenyVah3`** tyto matice představují změnu vah jednotlivých neuronů v daných vrstvách v případě aktualizace vah.

Matice **`vahy1`** a **`zmenyVah1`** mají **`pocetVstup`** + 1 řádek a **`vrstva1pocet`** sloupců. Matice **`vahy2`** a **`zmenyVah2`** mají **`pocetVstup`** + 1 řádek a **`vrstva2pocet`** sloupců. Matice **`vahy3`** a **`zmenyVah3`** mají **`pocetVstup`** + 1 řádek a **`vrstva3pocet`** sloupců.

Dále jsou ve třídě NeuralNet umístěny funkce a procedury:

- `secti` – tato funkce sčítá matice
- `generujVahy` – tato funkce generuje váhy pro matice při prvním průchodu neuronové sítě
- `klasifikace` – tato metoda je určená ke klasifikaci v rámci neuronové sítě
- `uceni` – metoda, která obsluhuje učení neuronové sítě za po moci metody Backpropagation

Celkové řešení neuronové sítě je implementováno ve třídě `Neural`, tato třída vytvoří instanci třídy `NeuralNet neural = new NeuralNet(75, 30, 256, 4)`. V tomto konstruktoru jsou definovány počty neuronů pro jednotlivé vrstvy a počet vstupů a výstupů.

Tato třída dále obsahuje funkci `natrénuj`, ta se stará o načtení trénovacích dat a jejich převedení na histogramy, poté se provede za mocí funkcí obsažených ve třídě `NeuralNet` natrénování neuronové sítě.

Funkce `vysledek` dostane jako vstupní parametr daný vzorek a vrátí hodnotu od 0 až 3, ve které se skrývá klasifikace daného vzorku.
Aby se oddělilo načítání a trénování neuronové sítě od výběrů jednotlivých vzorků a jejich porovnávání, vznikla třída `Rozpoznání`. Tato třída vytvoří instanci třídy `Neural` a bezprostředně poté natrénuje neuronovou síť. Pak se zjistí velikost zkoumaného snímku a uloží se do proměnných. Podle velikosti snímku se vytvoří stejně velké pole `hodnoty` indexované podle jednotlivých pixelů. Pak se provedou dva for cykly, které projdou celý sejmutý obraz a klasifikují jednotlivé vzorky a výsledky se ukládají do připraveného pole.

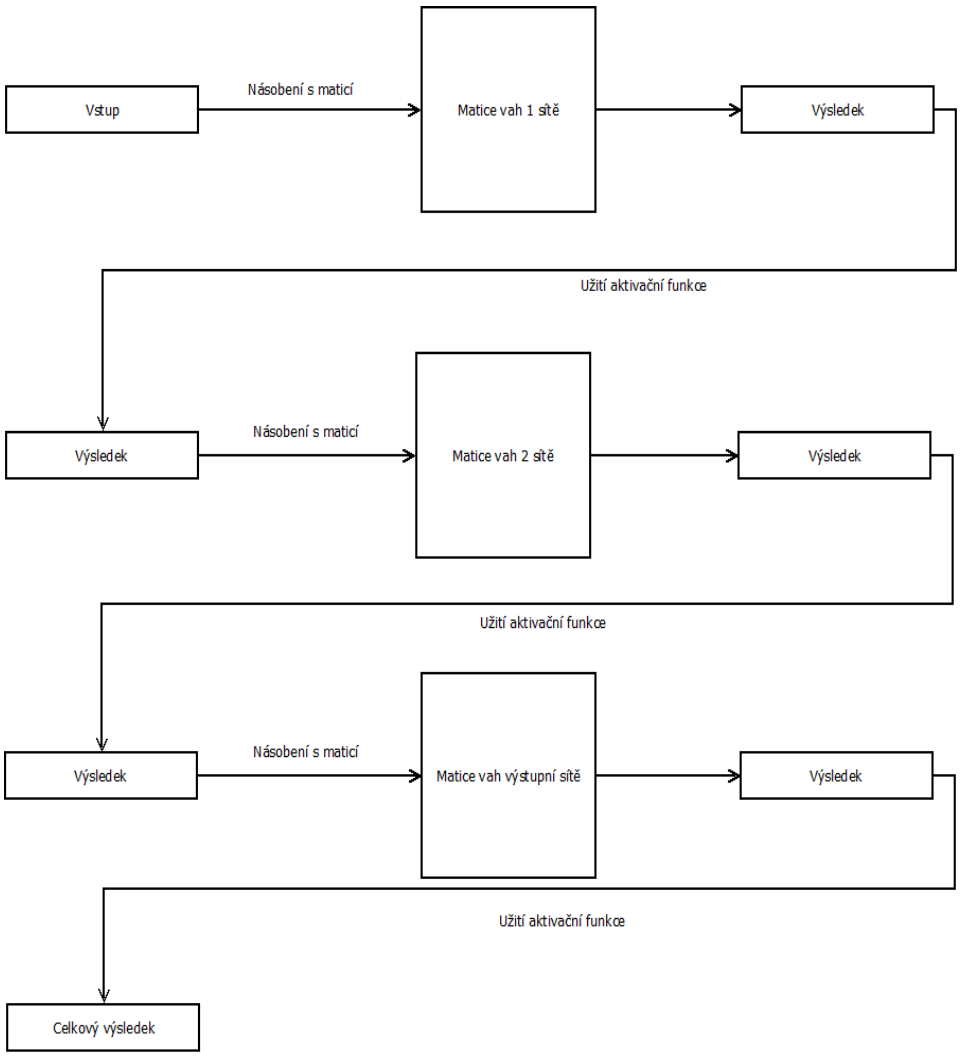
Ukázka metody:

```
for( int y = 0; y < vyska; y = y + VELIKOST ){ // VELIKOST
- konstanta velikosti vzorku
     for( int x = 0; x < sirka; x = x + VELIKOST ){
            BufferedImage sub = null;
            sub = getSubimage(x ,y, VELIKOST, VELIKOST);
     int vysledek = neu.vysledek(sub);//neu - instance tří-
dy neural
                    if(vysledek == 2 || vysledek == 3 ) {
                hodnoty[x][y] = 1  ;
          }
     }
}
```

Podle získaných výsledků uložených v poli `hodnoty` se jednotlivé vzorky obarví metodou `obarvi`.

5.3.6 Klasifikace vzorků

Klasifikace probíhá tím způsobem, že se třikrát vynásobí matice vah a po každém z násobení se aplikuje aktivační funkce Sigmoida, v prvním kroku se do matice nagenerují náhodné hodnoty, poté se vstupní vektor vynásobí touto maticí a pak se provede aktivační funkce a opět se vynásobí další maticí vah. Použije se aktivační funkce, poté se vynásobí s maticí výstupů a nakonec se použije aktivační funkce a už je znám výsledek. Názorná ukázka je na obrázku 35.



Obr. 35 Postup výpočtu neuronové sítě

5.3.7 Metoda Backpropagation

V algoritmu je implementována metoda Backpropagation s momentem. Tento moment zaručuje, že algoritmus neuvízne v průběhu učení na lokálním extrému, což zaručí, že se neuronová síť naučí kvalitně klasifikovat vzorky a zároveň velice urychluje učení neuronové sítě. Pokud se několikrát po sobě aktualizují váhy neuronů stejným směrem, je možné aktualizovat váhy výrazněji a tím vznikne zmíněné zrychlení. Informace o těchto změnách se ukládají do proměnných **`zmenyVah1`**, **`zmenyVah2`** a **`zmenyVah3.`**

5.3.8 Trénovací data pro neuronovou síť

Celá trénovací skupina dat má velikost 40 vzorků a je rozdělena do 4 kategorií podle výstupů viz kapitola 5.3.4. Prvních 10 vzorků je čistá cesta, dalších 10 vzorků je cesta s množstvím popadaného listí. Vzorky 21 až 30 jsou vzorky okolí cesty a posledních 10 vzorků je okolní zeleň. Trénovací data jsou rovnoměrně rozvržena po snímané scéně a jsou schopna obsáhnout barvy ve snímané scéně obsažené.

Obr. 36 Ukázka trénovacích dat

5.3.9 Výsledky řešení neuronovou sítí

V této kapitole budou popsány a předvedeny výsledky metody řešení samotné neuronové sítě na nasnímaných datech v arboretu u Mendelovy univerzity v Brně.

Na následujícím obrázku 37 je nasnímána cesta bez listů. Z výsledků je vidět, že algoritmus velice dobře klasifikuje vzorky, ale některé vzorky, které se nacházejí mimo cestu, jsou chybně klasifikovány jako cesta.

Obr. 37 Výsledek neuronové sítě na nasnímané cestě bez listí

Na obrázku 38 je nasnímána cesta s listím, což je pro robota velké ztížení dané úlohy. Podle výsledku je jasné, že neuronová síť i tentokrát chybuje v některých vzorcích, které jsou odebrány z okolí. Ovšem chybovost, co se týká rozpoznávání cesty pokryté listím, se velice zhoršila a je nepřesná.

Obr. 38 Výsledek neuronové sítě na nasnímané cestě s listím

5.3.10 Závěr pro druhé řešení

Výsledky získané neuronovou sítí jsou uspokojivé, ale stále je ve výsledku velké množství chybně určených vzorků. Pokud cesta není pokryta listím, je dobře možné ze získaného výsledku získat směr nalezené cesty. Co se týká cesty pokryté listím, tak tam je velké množství špatně určených vzorků.

5.3.11 Post-processing

V druhém řešení dává neuronová síť poměrně dobré výsledky, však špatně určené vzorky je nutné odstranit. Po klasifikaci všech vzorků v obraze byl dalším krokem post-processing. Byla zvolena metoda KNN, tato metoda se také nazývá metodou klasifikace nejbližších sousedů (dále už jen KNN). Na následující tabulce je ukázka toho, jaký má metoda princip. Už z velikosti tabulky je patrné, že se bude jednat o KNN2. Dvojka u názvu znamená, že budeme porovnávat okolní sousedy ve vzdálenosti dva vzorky od posuzovaného vzorku.

V tabulce je vzorek, který se pokoušíme klasifikovat, označen velkým písmenem P a vzorky, které byly neuronovou sítí ohodnoceny jako pozitivní hodnou 1 a ostatní hodnotou 0. V tomto případě by byl vzorek P metodou KNN2 posouzen jako 0 a to protože počet hodnot 0 převládá nad hodnotami 1.

Tab. 2 Ukázka funkce KNN2 v post-processingu

1	1	0	1	0
0	0	0	0	0
1	0	P	0	1
0	0	0	0	0
1	0	1	0	1

Pro tyto účely vznikla třída Klasifikace, která implementuje metodu KNN2. Tato třída obsahuje funkci `KNN(int x ,int y,int hodnoty[][])`. Metoda má jako vstupní parametry souřadnice zkoumaného vzorku a matici hodnot, ve které jsou uloženy specifikace jednotlivých vzorů. Metoda je schopna určit, kde se posuzovaný bod nachází a jestli je možné provést metodu KNN2 v plném rozsahu. Pokud je vzorek umístěn například v rohu, je možné provést jeho klasifikaci pouze s možnými hodnotami ve vzdálenosti 2 vzorky. Po klasifikaci daného vzorku metoda porovná výsledek se stávajícím stavem a vrátí hodnotu true nebo false podle toho, jestli je vzorek správně ohodnocen a podle vrácené hodnoty se přizpůsobí matice hodnot.

Ukázka klasifikace a rozhodování pro metodu KNN2:

```
if (pocetModrych > (pocetCelkem-pocetModrych)&& jeModry ==
true){
vysledek = true;
     }else  if(  (pocetCelkem-pocetModrych)  >  pocetModrych
&& jeModry ==  false)
     {
     vysledek = true;
     }
return vysledek;
```

5.3.12 Výsledky řešení neuronovou sítí a metodou KNN

V následující kapitole budou předvedeny a popsány výsledky neuronové sítě v kombinaci s klasifikační metodou KNN2.

Na obrázku 39, kde je cesta bez listí, je jednoznačně viditelné, že metoda KNN2 odstranila většinu špatně určených vzorků, až na několik vzorků v horní části obrazu. Pro nás to znamená, že tato metoda má velice dobrou účinnost. Takto se chová na velké části testovacím dat nasnímané cesty za normálních světelných podmínek.

Obr. 39 Výsledek řešení neuronovou sítí a metodou KNN2 na cestě bez listí

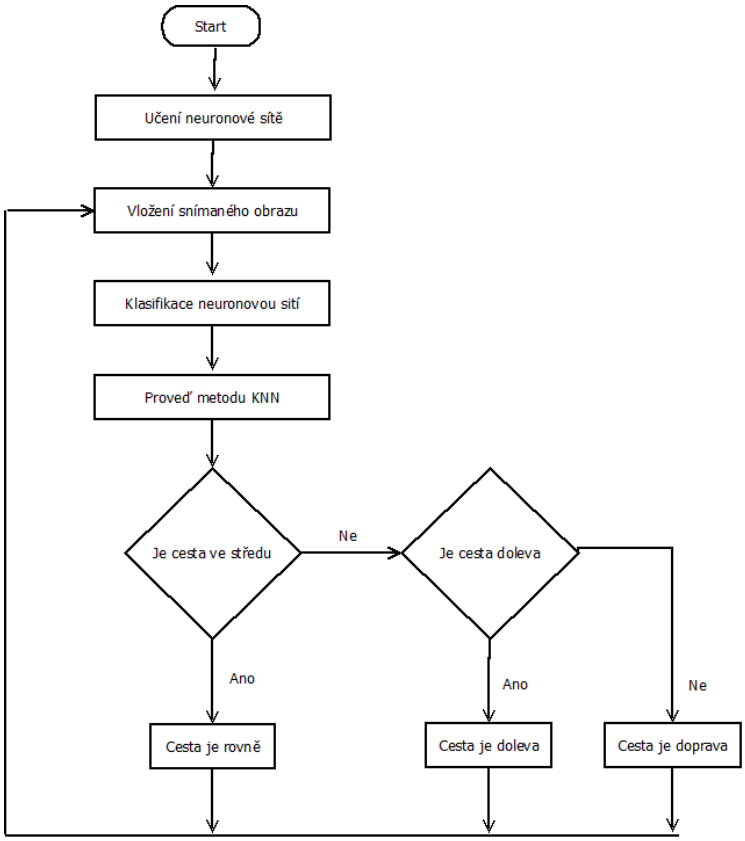
Na obrázku 40 je nasnímaná cesta s listím, což je pro robota velké ztížení dané úlohy. Podle výsledku je jasné, že neuronová síť v kombinaci s metodou KNN2 odstranila veškeré chybně určené vzorky v rámci cesty s listím. Cesta se jeví lépe ohraničena a jedná se o výrazné zlepšení. Na horizontu snímaného obrazu je viditelný vozík a za vozíkem jsou stromy bez listí rozmazané a šedé, tedy se neurčují jako okolí cesty.

Obr. 40 Výsledek řešení neuronovou sítí a metodou KNN2 na cestě s listím

5.3.13 Implementace řešení směru cesty

Implementace, která řeší samotné řešení určení směru cesty, se nachází ve třídě `Rozpoznani`. Tato třída si v konstruktoru vytvoří instance všech potřebných tříd a provede natrénování neuronové sítě. Dále obsahuje pouze funkci `int smerCesty(BufferedImage image)`, tato funkce dostane jako parametr snímaný obraz a po vyhodnocení vrátí hodnotu v rozmezí 0-2. Hodnota 0 znamená, že cesta je rovná, hodnota 1 znamená, že cesta vede doleva a hodnota 2, že cesta směřuje doprava. Na následujícím diagramu (obrázek 41) je znázorněna funkce modulu.

Časová náročnost u této knihovny není velká, časově nejnáročnější je než se natrénuje neuronová síť. Délka trénování při 40 trénovacích vzorcích a daném množství neuronové sítě trvá 1,275 s, samotná klasifikace jednoho snímku o velikosti 640 x 480 při velikosti jednoho vzorku 16 x 16 pixelů trvá 476,338 ms. Post-processing je nejméně náročný proces. Je to dáno tím, že pracuje pouze s dvourozměrným polem, doba jeho trvání při určeném množství vzorků je 0,226 ms.



Obr. 41 Diagram funkce modulu rozpoznávajícího cestu

5.3.14 Výsledky při nepříznivých podmínkách

Robot byl určen pro jízdu v mírně nepříznivých podmínkách, mezi ně se počítá i cesta s listím, která byla uvedena v průběhu práce, ale také pokud bylo v okolí malé množství sněhu, ale cesta byla odhrnuta. O nepříznivé podmínky se jedná, i pokud byl velice slunečný den, když se robot pohyboval po cestě a změny mezi osvětlenými a tmavými místy byly rozdílné. Co se týká počasí, kdy sněží nebo prší, nebudeme vůbec uvažovat, protože robot nebyl konstruován pro tyto podmínky.

Na následujícím obrázku 42 je nasnímaná cesta, kde se v okolí nacházel sníh. Z výsledků je jasné, že vytvořený algoritmus funguje se slušnými výsledky. Rozpoznání správných vzorků probíhá pouze v místech, kde se nachází zelené okolí cesty, nikoliv v okolí, které je zapadané sněhem. I za takto zhoršených podmínek byl algoritmus schopen určit směr cesty.

Obr. 42 Výsledek řešení neuronovou sítí a metodou KNN2 pokud je v okolí sníh

Na obrázku 43 se algoritmus testoval za dalších nepříznivých podmínek a to pokud svítí velice ostré jarní slunce. Bylo zřejmé, že pokud bylo okolní rostlinstvo nasvíceno tímto velice ostrým sluncem, tak se barvy staly skoro nezřetelné a tudíž si s nimi algoritmus nevěděl rady a výsledky nebyly nijak uspokojivé. Doporučení pro zlepšení výsledků bude navrženo v závěrečné části jako možné vylepšení.

Obr. 43 Výsledek řešení neuronovou sítí a metodou KNN2 při ostrém slunci

5.4 Závěr kapitoly

V této kapitole bylo popsáno praktické řešení celého modulu. Jednalo se o základní postup řešení daného problému, kromě této holé kostry bylo nutné optimalizovat jednotlivé části tak, aby byly co nejvýkonnější. Hlavní částí úlohy bylo rozpoznání cesty na snímaném obrazu kamerou LifeCam, které je řešeno v programovacím jazyce Java a vývojovém prostředí Eclipse. Tyto programy (knihovny) jsou dostupné v rámci nekomerčních licencí a jsou dostupné na internetových stránkách.

Tvorba celého modulu probíhala postupným vývojem a rozebíráním každého kroku, než byl vykonán. Programovací jazyk Java byl zvolen, protože nás jej vyučují na univerzitě, ale i protože se jedná o velice rozšířený a podporovaný nástroj.

Závěr

5.5 Shrnutí práce

V této práci bylo řešeno vytvoření modulu pro rozpoznání cesty v Arboretu u Mendelovy univerzity v Brně za pomocí programovacího jazyku Java a vývojového prostředí Eclipse. Obraz je snímán kamerou LifeCam a pro upevnění kamery na robota byla využita stavebnice Tetrix. Díky této práci jsem se s vývojovým prostředím Eclipse a programovacím jazykem Java hlouběji seznámil. Dospěl jsem k závěru, že programovací jazyk Java je opravdu silný nástroj, který dokáže řešit i velice složité problémy s velkou jednoduchostí. Vývojové prostředí Eclipse bylo nainstalováno pod operačním systémem Windows 7 64x s doinstalovanou JDK a JRE sadou podporující programování v jazyce Java. Díky multiplatformitě jazyku Java nebyl žádný problém spustit modul pod systémem 86x.

V teoretické části této práce jsou popsány nejpodstatnější postupy a řešení jednotlivých problému týkajících se strojového vidění. Po celkovém prozkoumání metod určených pro strojové vidění jsem zjistil, že má využití nejen v průmyslu, ale také se začíná docela výrazně prosazoval v lékařství pro analýzu snímků získaných z magnetické rezonance a RTG. Další z možných analýz je analýza a zvýraznění objektů snímaného obrazu z mikroskopu, pro tuto problematiku je primárně vytvořena knihovna boofCV, která je open source a to urychluje její vývoj.

Část vlastní práce objasnila způsob, jakým probíhal vývoj a tvorba výsledného modulu. V této části jsou vybrané části kódu, které jsou důležité a obrázky jednotlivých řešení. První řešení se ukázalo jako naprosto nevhodné, proto jsem od něj upustil a navrhnul další mnohem účinnější a sofistikovanější řešení za pomocí neuronové sítě. Toto řešení bylo nutné doplnit o post-processing. Využil jsem metodu nejbližších sousedů KNN2, což způsobilo výrazné zlepšení v nalezení cesty.

5.6 Zhodnocení výsledků

Pokud se zaměříme na zhodnocení výsledků, musíme vzít v úvahu předem určené cíle v začátcích práce. Výsledný modul měl za úkol nalezení cesty v arboretu u Mendelovy univerzity v Brně a určení jejího směru. Do daného modulu stačí pouze vložit snímanou obrazovou informaci a modul vyhodnotí obraz a formou návratové hodnoty vrací směr cesty. Řešení neuronovou sítí má dobré výsledky na stejných trénovacích datech i v nepříznivých podmínkách. Celkový modul je navržen tak, aby se byl schopen zařadit se k ostatním modulům do hlavní aplikace pro řízení robota.

Co se týká analýzy snímané scény v nestálých podmínkách, tak se jedná o velice složitý problém, ideální by bylo prostředí stálé – světelné podmínky, čistá stejnobarevná cesta atd. Vzhledem k těmto okolnostem jsou získané vý-

sledky velice dobré. Robot se nebude spoléhat pouze na snímání cesty, ale také na data získaná z GPS a dalších senzorů.

5.7 Možná vylepšení

Pokud se zaměříme na možná vylepšení daného modulu, zjistíme, že by bylo ideální, co se týká nepříznivých podmínek, robota doplnit o čidla, která by byla schopna získat základní vlastnosti okolního prostředí. Ty by se předaly modulu, který rozpoznává cestu a ten by z těchto vlastností (například: světlost či teplota atd.) určil, jakou trénovací sadu bude nejlepší využít. Možné vylepšení by se tedy týkalo rozšíření modulu o různá trénovací data.

6 Literatura

BÍLÝ, Radek. MORAVIA INSTRUMENT. *Dokumentace Control Web*. 2010.

Barevný model: Výukový modul do EPO - Digitální fotografie. *SPŠei, Ostrava* [online]. 2007 [cit. 2013-04-07].

Dostupné z: http://www.dmp.spsei.cz/digi/model.php

Co je to CMOS. *Digineff* [online]. 2008 [cit. 2013-04-07]. Dostupné z: http://www.digineff.cz/cojeto/cmos/cmos.html

Digitalizace obrazu. *Www.e-learning.tul.cz* [online]. 2000 [cit. 2013-04-13]. Dostupné z: Digitalizace obrazu. [online]. [cit. 2013-04-13]. Dostupné z: http://home.zcu.cz/~alenapos/digitalizace.html

Zpracování obrazu. *Zpracování obrazu* [online]. 2002 [cit. 2013-04-14]. Dostupné z: http://195.178.89.121/mm/k_4_2.htm

HLAVÁČ, Václav. PŘEDZPRACOVÁNÍ: v prostoru obrazů. *CVUT* [online]. 2013 [cit. 2013-04-16]. Dostupné z: http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/PredzpracObr.pdf

HORÁK, Karel. Jasová transformace. In: *VUTBR* [online]. 2010 [cit. 2013-04-16]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ZVS/lectures-pdf/04_Jasove_transformace.pdf

ŠPANĚL, Ing. Michal. Obrazové segmentační techniky. *VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ* [online]. 2005 [cit. 2013-04-18]. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769325

DARWIN, Ian F. *Java: kuchařka programátora*. Vyd. 1. Brno: Computer Press, 2006, 798 s. ISBN 80-251-0944-5.

Šonka M., Hlaváč V., Boyle R.: *Image Processing, Analysis, and Machine Vision*, Chapman&Hall, 1993

Praktické využití metod digitálního zpracování obrazu. In: MIKŠÍK, Ondřej. *Soc.nidm.cz* [online]. 2007 [cit. 2013-05-07]. Dostupné z: http://soc.nidm.cz/data/2007/01-2.pdf

OpenCV DevZone. OPENCV. *OpenCV DevZone* [online]. 2003 [cit. 2013-04-23]. Dostupné z: http://code.opencv.org/projects/opencv/wiki

BoofCV. *BoofCV* [online]. 2013 [cit. 2013-04-23]. Dostupné z: http://boofcv.org/index.php?title=Main_Page

Introduction. *ImageJ* [online]. 2001 [cit. 2013-04-23]. Dostupné z: http://rsb.info.nih.gov/ij/docs/intro.html

Objektově orientované programování v jazyku C++. *Pef mendelu* [online]. 2011 [cit. 2013-04-23]. Dostupné z: http://ui.pefka.mendelu.cz/files/ZOO_opora_2.pdf

IEEE transactions on intelligent transportation systems: a publication of the IEEE Intelligent Transportation Systems Council [online]. 2012 [cit. 2013-05-19]. ISBN 1524-9050. Dostupné z: www.google.cz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CDYQFjAA&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpl%2FarticleDetails.jsp%3Farnumber%3D6182716&ei=9PJ3UZH_Kq6Q4AT790CQAQ&usg=AFQjCNHTIjEa_j7GoiEfxJ7TbGQlXdzLJw&sig2=xEHaYyJXQm2uBhsRmhSSig&bvm=bv.45580626,d.ZGU

Microsoft. *Microsoft* [online]. 2012 [cit. 2013-05-05]. Dostupné z: http://www.microsoft.com/hardware/cs-cz/p/lifecam-studio#overview

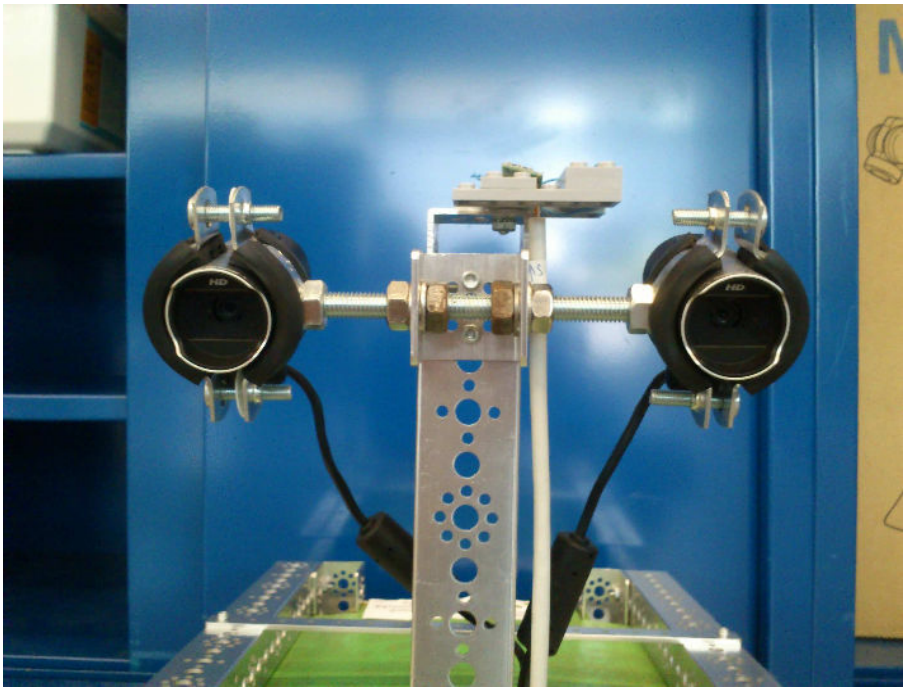
Eclipse. *Eclipse* [online]. 2013 [cit. 2013-05-07]. Dostupné z: http://www.eclipse.org/org/

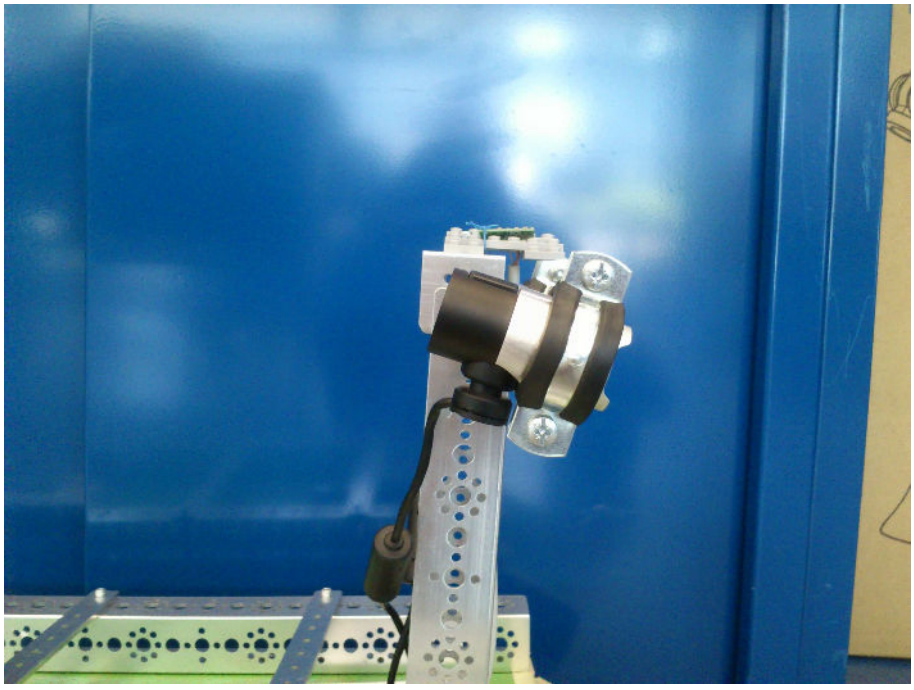
Oracle Java. *Oracle* [online]. 2011 [cit. 2013-05-07]. Dostupné z: http://docs.oracle.com/javase/6/docs/technotes/guides/jar/index.html

Input/Choose Color Usage Guideline. *Oracle* [online]. 1999 [cit. 2013-05-08]. Dostupné z: http://www.oracle.com/webfolder/ux/middleware/richclient/index.html?/webfolder/ux/middleware/richclient/guidelines5/inputColor.html

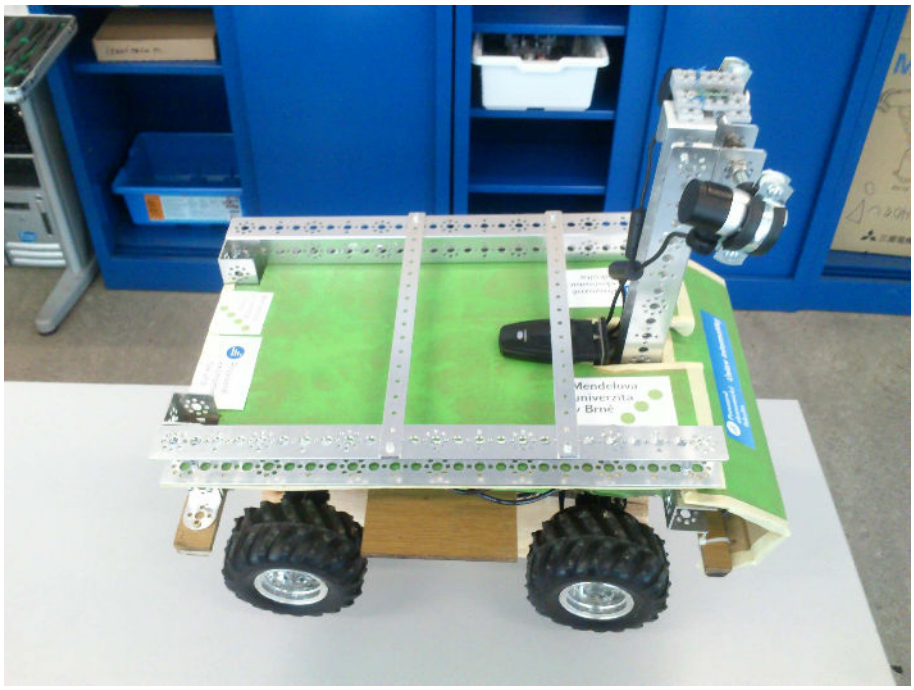
Přílohy

A Umístění kamery na robotovi





B Robot



C Obsah CD

Přiložené cd osahuje:

- zdrojové kódy aplikací
- vytvořený jar z aplikace
- elektronické vydání této práce
- ukázky zpracovaných fotek