UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

AUTONÓNMNY MOBILNÝ ROBOT PRE SÚŤAŽ ROBOTOUR

DIPLOMOVÁ PRÁCA

2015

Michal Moravčík

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

AUTONÓNMNY MOBILNÝ ROBOT PRE SÚŤAŽ ROBOTOUR

DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Vedúci bakalárskej práce:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2015

Michal Moravčík

93449408

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Autonónmny mobilný robot pre súťaž Robotour / *Autonomous mobile robot for Robotour contest*

Cieľ: Cieľom práce je navrhnúť, implementovať a experimentálne overiť nový algoritmus pre riadenie mobilného robota, ktorého úlohou je doručovať náklad vo vonkajšom prostredí za pomoci mapy, senzorov a počítačového videnia. Práca nadväzuje na predchádzajúcu diplomovú prácu a predpokladá využitie rovnakého mobilného robota, ktorý bol pri jej realizácii zostrojený. Úlohou študenta je zefektívniť existujúci riadiaci softvér a navrhnúť nové algoritmy, ktoré pozorovateľne zlepšia úspešnosť robota pri plnení úlohy.

Literatúra: Miroslav Nadhajský: Robotour, diplomová práca, FMFI UK, 2011.
Proceedings of the 5th Robotour Workshop, Robotika.SK, 2010.

Kľúčové slová: mobilný autonómny robot, umelé neurónové siete, počítačové videnie

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 03.12.2013

Dátum schválenia: 10.12.2013

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

.....
študent

.....
vedúci práce

Poďakovanie

Ďakujeme Nadacii Tatrabanky za finančný prispevok z grantového programu Kvalita vzdelávania, projekt AI ROBOTIKA, zmluva 2010vs029, vďaka ktorému bol robot zostrojený. Tiež sa chcem poďakovať môjmu vedúcemu Mgr. Pavlovi Petrovičovi, PhD. za výber témy, jeho pomoc a čas strávený testovaním robota.

Abstrakt

V tejto práci navrhujeme, implementujeme a experimentálne overujeme vylepšenia algoritmu riadenia mobilného robota, ktorého úlohou je doručovať náklad vo vonkajšom prostredí za pomoci mapy, senzorov a počítačového videnia. Práca nadväzuje na predchádzajúcu diplomovú prácu a využíva rovnakého mobilného robota, ktorý bol pri jej realizácii zostrojený.

Pridaním nových algoritmov správania robota a zlepšením pôvodných sme zvýšili spoľahlivosť a úspešnosť robota. Vďaka jednoduchej konverzii vstupu do LAB reprezentácie farby sa výrazne znížila chyba rozpoznávania cesty. Lokalizácia na mape bola vylepšená pridaním výpočtov, ktoré berú do úvahy zakrivenie Zeme. Touto zmenou sa spresnili aj výpočty plánovania, ktoré bolo oproti pôvodnému zlepšené a robot lepšie zvláda komplexnejšie trasy.

Robot sa v rozpracovanom stave zúčastnil súťaže Robotour na ktorú je stavaný a prvýkrát sa umiestnil na prvom mieste.

KĽÚČOVÉ SLOVÁ: mobilný autonómny robot, umelé neurónové siete, počítačové videnie

Abstract

In this thesis we propose, implement and experimentally verify improvements of mobile robot control algorithm, whose task is to deliver cargo outdoors using a map, sensors and computer vision. This thesis continues from previous master's thesis and uses the same mobile robot that was built as its realization.

By adding new behavior algorithms and by improving of original ones we have improved reliability and success rate of the robot. Thanks to simple conversion of input into LAB color representation the error of path recognition has been significantly decreased. Localization on map has been improved by addition of calculations that take Earth curvature into consideration. This addition has improved calculations of planning algorithm, which was further improved and performs better on more complex tracks.

Robot participated in Robotour contest while still in developement and he won for the first time.

KEYWORDS: mobile autonomous robot, artificial neural networks, computer vision

OBSAH

Úvod

Výskum v oblasti autonómnych robotov a vozidiel je v súčasnej dobe relevantný s rastúcim záujmom v rôznorodých odvetviach. Projekty ako Google-Self Driving Car pútajú aj záujem médií a tým sa koncept autonómnych robotov približuje aj k širokej verejnosti. Spôsobov ako riešiť problémy, ktoré sťažujú vývoj týchto robotov, je čoraz viac a ako motivácia pre vytváranie nových a efektívnejších postupov slúžia aj robotické súťaže ako je aj súťaž Robotour, organizovaná českými nadšencami robotiky.
V súťaži "Robotour - robotika.cz outdoor delivery challenge" súperia autonómne roboty v preprave nákladu v rámci parku, pričom smú používať iba chodníky v ňom. Tieto roboty využívajú rôzne senzory, ktoré využijú na to, aby nezišli z cesty a aby neohrozovali ľudí v parku.
V tejto práci vylepšujeme robota Smelého Zajka, ktorý na rozpoznávanie cesty využíva kameru a bol postavený s cieľom uspieť v Robotour súťaži. Medzi doteraz zúčastnenými robotmi je naše riešenie, využívajúce neurónovú sieť na rozpoznávanie cesty, unikátne a tým prispeva k rozmanitosti a zaujímavosti súťaže.
V prvej časti popisujeme použité metódy a problémy, ktoré riešime a v druhej rozoberáme samotnú implementáciu robota, využivajúcu neurónovú sieť na spracovanie obrazu kamery. Na konci zhodnotíme správanie robota, výsledný prínos tejto práce a zhrnieme ďalšie problémy ktorých riešenie by pomohli robotovi k lepším výkonom.

Sumarizácia základných cieľov práce

- revidovať výsledky z experimentov s modelmi umelých neurónových sietí z predchádzajúcej práce
- prepracovať a sprehľadniť plánovací aj navigačný algoritmus, ktoré hoci v základoch funkčné, skrývajú potenciál na výrazné zlepšenie

- vyriešiť dlho pretrvávajúce konceptuálne problémy v riadení motorov na základnej doske
- zamerať sa na jednotlivé technické i konceptuálne detaily celého systému, ktoré sú kľúčové pre úspešnosť robota, vyhľadať a vyriešiť ich
- všetky vykonané úpravy analyzovať a dobre zdokumentovať

Kapitola 1

Základné pojmy

1.1 Umelá inteligencia

Umelá inteligencia ako vedný odbor skúma možosti strojov inteligentne sa chovať, učiť sa a vyvodzovať závery na základe pozorovaní. Takéto inteligentné stroje by boli schopné sa rozhodovať ako človek, uspôsobovať svoje chovanie podľa situácie a vykonávať rôznorodú činnosť.

Inteligencia

Ak chceme skúmať inteligenciu strojov, musíme najprv definovať, čo pod týmto pojmom bežne myslíme. Inteligenciu poznáme najmä od ľudí a k jej prejavom patrí schopnosť rozoznávania vzorov, klasifikácia pozorovaných objektov a javov, plánovanie a riešenie problémov. Keďže je inteligencia tak úzko spätá s ľuďmi, mnoho postupov a algoritmov hľadá inšpiráciu práve v ľudskom správaní.

1.2 Učenie

Dôležitou vlastnosťou ľudí a inteligentných systémov je ich schopnosť sa učiť a tak upravovať svoje správanie podľa predošlých skúseností. S použitím učiacich algoritmov môžeme vytvoriť všeobecnejšie roboty, ktoré sa dokážu prispôsobiť aj niektorým nepredvídaným okolnostiam. Rozlišujeme tri základné formy učenia:

Učenie odmenou a trestom

Agent nedostáva spätnú väzbu za každú svoju akciu ale dostane odmenu/trest za do-

siahnutie nejakého konkrétneho stavu. Prikladom môže byť robot s nohami, ktorého akcie sú pohyby nôh a je odmenený za každý pohyb dopredu. Hýbanie nohami samo o sebe robotovi neprinesie veľkú odmenu, ale správna sekvencia pohybov áno. Takto by sa robot mal skúšaním rôznych pohybov dopracovať k optimálnej sekvencii pohybov, ktorá mu prinesie najviac odmien.

Učenie bez učiteľa

V niektorých prípadoch nemáme správnu odpoveď pre žiaden vstup. V takýchto dátach je ale možné napríklad hľadať podobnosti a vzťahy a podľa toho ich rozdeliť na viacero skupín.

Učenie s učiteľom Ak pri tréningu využívame vstupy, pre ktoré uvádzame aj správny výstup, voláme takýto proces učenie s učiteľom. Cieľom je získať funkciu ktorá vráti pre každý vstup čo najsprávnejšiu odpoveď aj v prípade, ak sa s takým vstupom agent počas trénovania nestretol.

1.3 Robotika

Robotika je odvetvie, v ktorom sa spája elektrotechnika, strojníctvo a materiálové inžinierstvo s cieľom výroby a prevádzkovania robotov. Roboty, ako automatizované zariadenia s určeným cieľom nám zjednodušujú vykonávanie namáhavej práce a prinášajú nové možnosti vo výskume.

Behom posledných desaťročí sa roboty uplatnili najmä vo výrobe, kde je potreba vykonávania fyzicky namáhavej repetitívnej činnosti. Ďalej sa používajú vysoko špecializované roboty vo výskumnej a bádateľskej činnosti v prostredí nevhodnom pre človeka ako je napríklad vesmír alebo hlbokomorské prostredie.

Roboty môžu byť autonómne, teda samočinné, alebo vyžadujú človeka aby ich priamo riadil. Riadené roboty sú menej náročné na vytvorenie, keďže nie je potreba vyvynúť sofistikovaný software určený na ich riadenie. Roboty, ktoré priame riadenie nepotrebujú, je však možné nasadzovať vo veľkom počte a v situáciach, kedy je náročne zabezpečiť kontakt s riadiacou osobou. Aj preto je dopyt po inteligentných robotoch, ktoré sú dostatočne spoľahlivé vo svojej činnosti a dokážu sa adaptovať zmenám prostredia a rôznym podmienkam.

1.4 Neurónové siete

Umelé neurónové siete sú štruktúry inšpirované biologickými neurónovými sieťami zložené z poprepájaných základných výpočtových jednotiek - neurónov. Vďaka ich všeobecnosti a všestrannosti je nimi možné riešiť rôzne zložité úlohy ako napríklad klasifikáciu dát, aproximáciu funkcií alebo predpovedanie časových radov.
Ich nevýhodou je nutnosť správneho výberu dát na trénovanie, forma vstupných dát a výber správneho modelu siete.

1.4.1 Neurón

Inak nazývaný aj perceptrón [2] je základnou súčasťou neurónových sietí. Perceptrón dostáva na vstupe n hodnôt $x_1 ... x_n$, ktoré prenásobí prislúchajúcimi váhami $w_1 ... w_n$, sčíta a výsledok preženie cez jeho aktivačnú funkciu ϕ.
Výsledkom teda bude hodnota

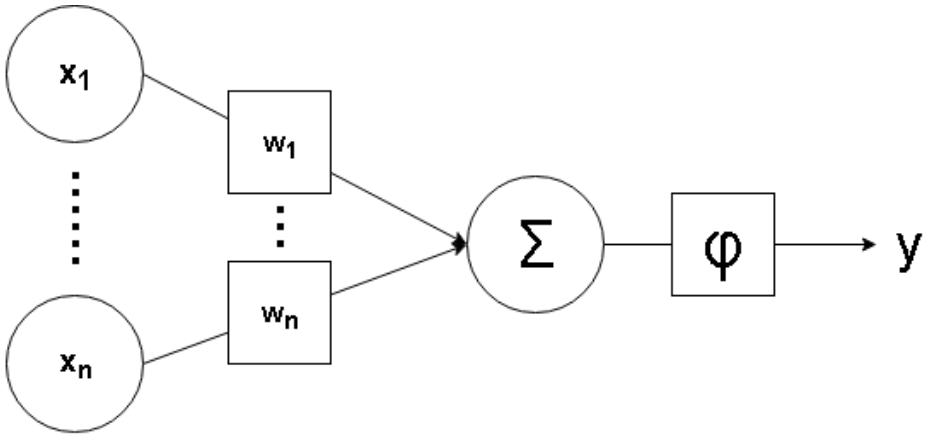
$$v = \sum_{i=1}^{n} (x_i w_i) \tag{1.4.1}$$

$$y = \varphi(v) \tag{1.4.2}$$

Ako jeden zo vstupov je možné dať konštantnú nenulovú hodnotu, ktorú nazývame bias. Bias zlepši a zovšeobecní schopnosť perceptrónu keďže bez neho je výstup silne závislý na aktivačnej funkcií nakoľko ňou perceptrón nemôže "hýbať". Napríklad pre vstup, kde sú všetky hodnoty nulové by perceptrón podal výstup

$$y = \varphi(0) \tag{1.4.3}$$

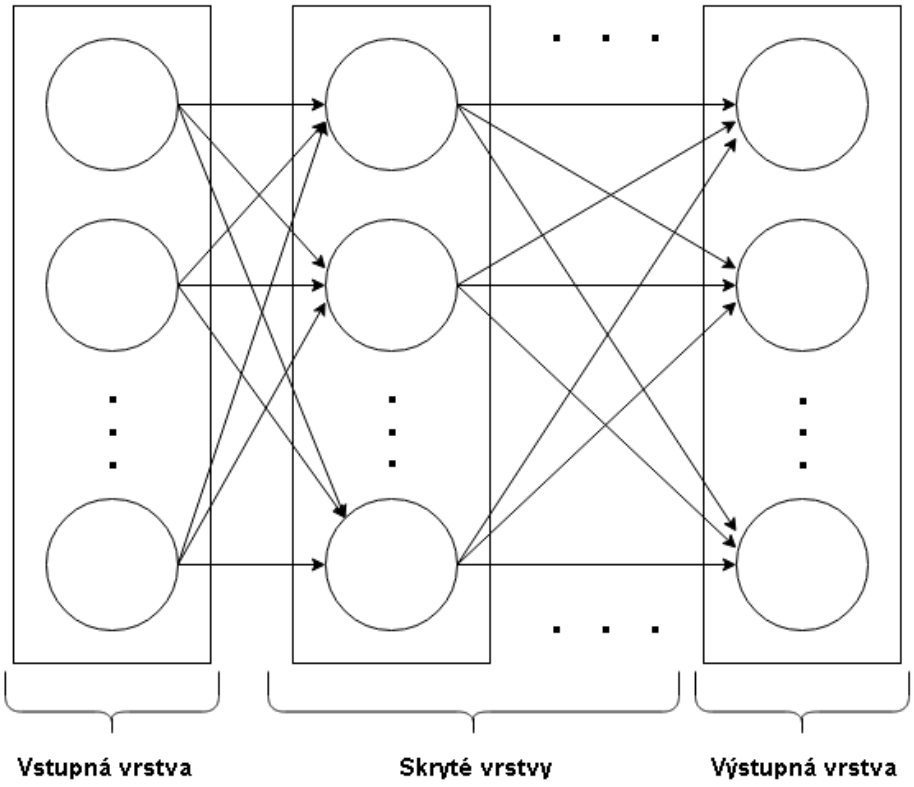
bez ohľadu na váhy, teda by bola jeho generalizačná schopnosť obmedzená pre nenulové vstupy.



Obrázok 1.1: Jednoduchý perceptrón

1.4.2 Viacvrsvový perceptrón

Viacvrstvový perceptrón [2] sa skladá z vrstiev na ktorých ležia jednoduché perceptróny. Jednotlivé vrstvy sú spolu prepojené jedným smerom tak, že vznikne acyklický graf - teda výstupy predošlej vrstvy sú vstupmi ďalšej. Prvú vrstvu nazývame vstupnou, poslednú výstupnou a zvyšné majú pomenovanie skryté vrstvy.



Obrázok 1.2: Viacvrstvový perceptrón

1.4.3 Trénovanie

Na to, aby bola neurónová sieť využiteľná, je potreba správne nastaviť váhy jednotlivých neurónov. Správne nastavenie môžeme dosiahnuť trénovaním s využitím učiteľa. Trénovacie algoritmy často využívajú gradient descent metódu a k chcenému nastaveniu konvergujú minimalizujúc chybu.

Validácia

Keďže chceme, aby natrénované siete dávali správne odpovede aj na vstupy, ktoré neboli obsiahnuté v trénovacom procese, rozdelíme trénovacie dáta na dve podmnožiny - trénovaciu a testovaciu. Ako už názov napovedá, na trénovacej množine model siete trénujeme a pomocou testovacej množiny ohodnotíme úspešnosť siete na neznámych dátach. Takto predchádzame vybratiu modelu, ktorý je preučený na trénovacích dátach.

1.4.4 Backpropagation

Backpropagation [2] je algoritmus, ktorý spätne propaguje chybu naprieč neurónovou sieťou. Vyžaduje správny výstup od učiteľa a od jednotlivých neurónov požaduje, aby ich aktivačné funkcie boli derivovateľné.

Nech $t_i(x)$ je požadovaná hodnota daná učiteľom pre vstup x pre neurón i, $y_i(x)$ je výstup neurónu i pre vstup x, potom

$$e_i(x) = t_i(x) - y_i(x) \tag{1.4.4}$$

je chyba neurónu i pre vstup x. Kvadratickú chybu siete definujeme ako

$$E(x) = \frac{1}{2}\sum_i (e_i(x))^2 \tag{1.4.5}$$

Sumu násobíme zlomkom $\frac{1}{2}$ pre jednoduchšie derivovanie. Vahy neurónov upravíme o hodnotu

$$\Delta w_{ij}(x) = \alpha \frac{\partial E(x)}{\partial w_{ij}(x)} \tag{1.4.6}$$

čím minimalizujeme strednú kvadratickú chybu $MSE = \frac{1}{N}\sum_{i=1}^{N} E(i)$ pre všetkých N vstupov. Keďže požadovaný vstup t je dostupný len pre výstupnú vrstvu vypočítame pre ne lokálny gradient neurónu i

$$\delta_i(x) = \varphi_i'(v_i(x))\sum_k (e_k(x)\varphi_k(v_k(x))w_{ki}(x)) \tag{1.4.7}$$

a váhy upravíme o hodnotu

$$\Delta w_{ij}(x) = \alpha \delta_i(x) y_j(x) \tag{1.4.8}$$

Váhy môžeme upravovať po každom vstupe (online metóda) alebo až po prejdení celej trénovacej množiny (batch metóda) použitím priemernej chyby. Výber metódy závisí od úlohy ktorú riešime a aj od veľkosti trénovacej množiny, lebo pri použití online prístupu majú vstupy, ktoré sa nachádzajú skôr v množine väčší vplyv na trénovanie. Hodnota α vyjadruje rýchlosť učenia, ktorú treba vhodne nastaviť. Vysoká rýchlosť učenia zabezpečí vyššiu rýchlosť konvergencie, ale zvyšuje pravdepodobnosť, že skonvergujeme k lokálnemu, nie globálnemu minimu.

1.4.5 Rprop

Rprop algoritmus [3] (z Resilient backpropagation) je odvodený od tradičného backpropagation. V Rprop veľkosť zmeny váhy nezávisí od hodnoty derivácie chyby ale sa adaptívne mení podľa výsledku predošlej iterácie(epochy) učenia. Z derivácie chyby berie iba znamienko a to udáva smer zmeny váhy. Keďže váhy mení aj na základe predošlého stavu učenia je tento algoritmus použiteľny len ako batch metóda nakoľko zmeny v krokoch online prístupu vypovedajú o zmenách len na určitom príklade, nie o celej trénovacej množine. Algoritmu nastavíme dva parametre η^+ a η^- $0 < \eta^- < 1 < \eta^+$ ktoré budú zväčšovať resp. zmenšovať rýchlosť zmeny váh podľa derivácie nasledovne

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \Delta_{ij}^{(t-1)}, & \text{ak } \frac{\partial E}{\partial w_{ij}}^{(t)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t-1)} > 0 \\ \eta^- \Delta_{ij}^{(t-1)}, & \text{ak } \frac{\partial E}{\partial w_{ij}}^{(t)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t-1)} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{inak} \end{cases} \tag{1.4.9}$$

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{ak } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)}, & \text{ak } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0, & \text{inak} \end{cases} \tag{1.4.10}$$

Vďaka tomuto prístupu netreba nastavovať rýchlosť učenia α a rýchlosť konvergencie sa v priemernom prípade zvýši oproti batch metóde spätnej propagácie.

1.4.6 iRprop-

iRprop- [4] je jednoduché vylepšenie Rprop algoritmu v ktorom sa v prípade zmeny znamienka derivácie vynuluje zapamätaná hodnota derivácie, teda

$$\text{ak } \frac{\partial E}{\partial w_{ij}}^{(t)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t-1)} < 0 \text{ potom } \frac{\partial E}{\partial w_{ij}}^{(t)} := 0 \tag{1.4.11}$$

Idea tohoto vylepšenia je pomôcť algoritmu v prípade, že sa pri gradientovom zostupe preskočilo cez lokálne minimum. Takýto trik zapríčiní, že sa váhy po zmene znamienka derivácie neaktualizujú a menia sa len ak je smer zostupu po gradiente jednoznačný, no veľkosť kroku sa zmenší.

1.5 Robotour

Robotour [5] je súťaž autonómnych robotov pohybujúcich sa po cestičkách v parku. Koná sa od roku 2006 každoročne v rôznych parkoch štátov strednej Európy.

1.5.1 Zadanie

Robot má za úlohu dostať sa zo štartovacej pozície do cieľu a následne sa vrátiť. Pozícia cieľa je zúčastneným oznámená až tesne pred štartom. Súťaž sa skladá zo štyroch štartov a sčítajú sa z nich vzdušné vzdialenosti od štartu resp. aj cieľa ak sa už robot vracia. Roboty nesúce 5 litrový súdok piva získavajú bodové zvýhodnenie. Pokiaľ sa robot ocitne všetkými kolieskami mimo cesty, je zastavený a do hodnotenia sa berie vzdialenosť od toho bodu do cieľa. Ak súťažný robot zavadzia inému, musí sa počas každej minúty pohnúť aspoň o jeden meter. Navyše musia všetky roboty prejsť pred súťažou homologizáciou, v ktorej musia splniť nasledovné úlohy:

- Zareagovať na prekážku buď jej obídením, alebo zastať a pokračovať v pohybe po jej odstránení
- Prejsť aspoň 10 metrov po rovnej ceste bez vyjdenia mimo nej

- Zastaviť po stlačení povinného, dostupného núdzového vypínača

Roboty sa nesmú pohybovať rýchlosťou vyššou ako $2,5ms^{-1}$ a musia odštartovať do desiatich minút od zadaného štartovacieho času bez zásahu človeka. Štart je hromadný a roboty sú pred štartovacou čiarou zoradené podľa ich bodového ohodnotenia v predošlom kole tak, že robot s najvyšším skóre je vpredu a teda má výhodu oproti ostatným.

1.5.2 Riešené problémy

Pre úspešné absolvovanie súťaže je potrebné vyriešiť viacero problémov. Prvým je zostrojenie robota ktorý je dostatočne silný na to, aby preniesol pivný súdok po nerovných cestách parku. Ďalej je treba vyriešiť navigáciu v rámci parku s pomocou poskytnutej mapy. Homologizácia vyžaduje snímanie prostredia pre detekciu prekážok, správne vysporiadanie sa s ich výskytom a zabezpečiť, aby robot nezišiel z cesty. V tejto práci sa zaoberáme riešením týchto problémov aplikovaním a upravovaním známych postupov do jedného celku.

Kapitola 2

Hardware

Využívame robota zostrojeného pre prácu Robotour[5], na ktorú nadväzujeme. Robotova podoba zostala z veľkej časti nezmenená, len niektoré jeho časti boli vylepšené alebo vymenené.

2.1 Komponenty

2.1.1 Senzory

Robot má viacero senzorov, ktoré sú umiestnené na jeho vrchnej časti pre lepší rozhľad a menšie rušenie spôsobené magnetmi motorov.

Ultrazvukové

Na detekciu objektov využivame päť ultrazvukových senzorov Devantech SRF08 s rozsahom $3cm - 6m$.

GPS

Zistenie polohy zo signálu GPS zabezpečuje modul Navilock NL-302U (chipset SIRF III) s USB rozhraním.

Kompas

Určenie orientácie zabezpečuje jednotka SparkFun HMC6343, ktorá obsahuje 3-osý kompas s kompenzáciou náklonu a akcelerometer.

Kamera

Obraz pre rozpoznanie cesty poskytuje klasická webkamera. K dispozícii má robot aj lepší camcorder avšak v čase písania práce je jeho spojenie s robotom nefunkčné kvôli chýbajúcemu ovládaču frame grabbera pre kernel Linuxu v aktuálnej distribúcii.

2.1.2 Riadenie

Mikrokontroler

Na komunikáciu s motormi a zber výstupov z ultrazvukových senzorov používame mikrokontroler ATMega128, ktorý je súčasťou riadiacej dosky pre robota SBot[6]. Mikrokontroler tiež zabezpečuje vykonávanie niektorých bezpečnostných subrutín ako núdzové zastavenie, alebo zastavenie pred prekážkou.

Notebook

Rozhodovanie má na starosti notebook Asus UL30Vt (Intel SU7300 Core 2 Duo, 4GB RAM, NVidia G210M) ktorého pevný disk bol nahradený SSD diskom kvôli otrasom ktoré vznikajú počas jazdy. Notebook posiela príkazy mikrokontroleru a spracúva obraz a výstupy GPS a kompasu.

2.1.3 Pohon

Motory

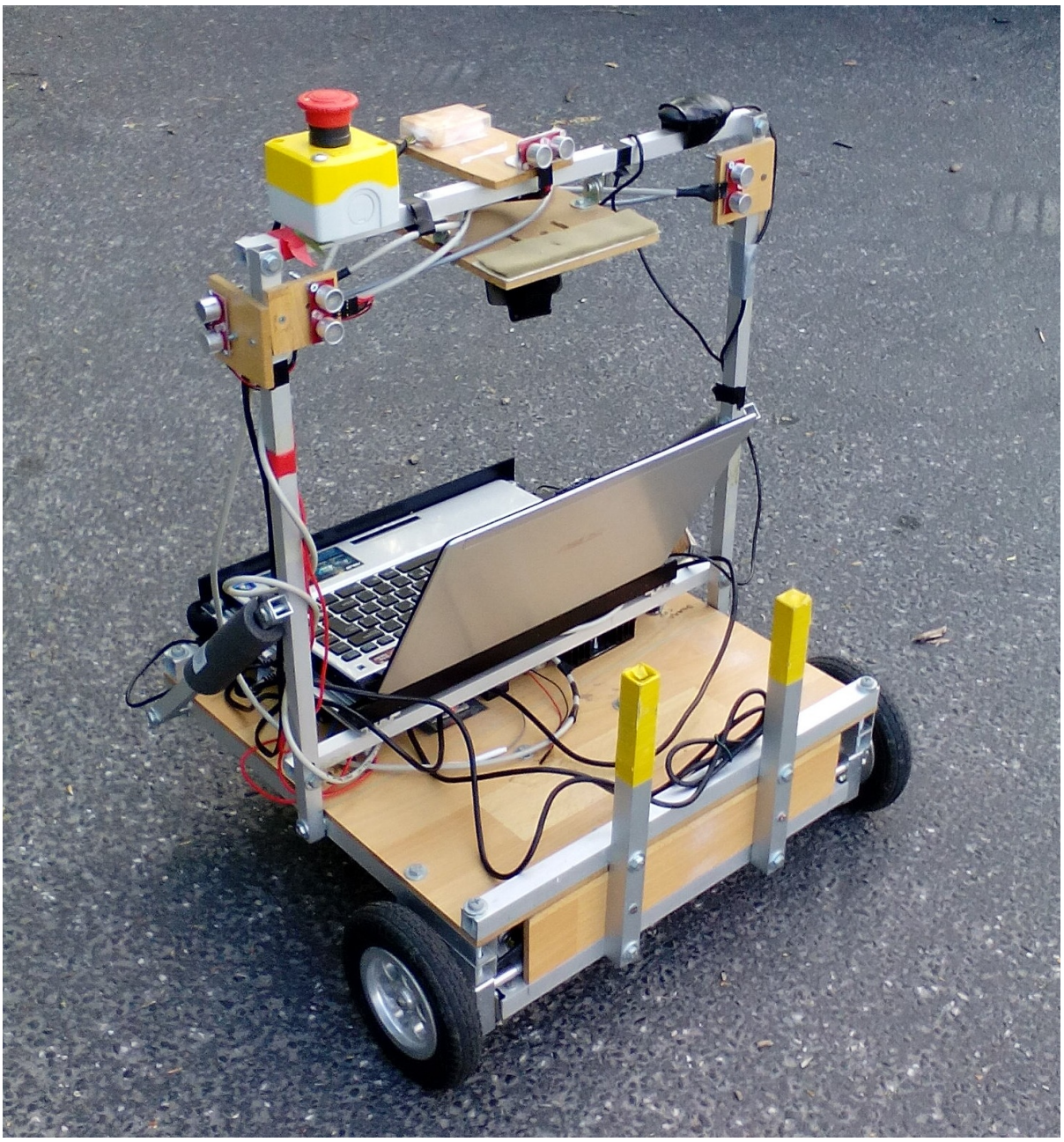
Na pohon robot využiva Parallax Motor Mount and Wheel Kit. Motory z tejto sady poskytujú dostatočný ťah na to, aby robot vyšiel do kopca, aký sa môže na chodníkoch vyskytovať aj s nákladom a zároveň sú dostatočne rýchle ($1.2 m/s$) pre súťaž Robotour.

Napájanie

Napájanie motorov a mikrokontrolera zabezpečuje akumulátor HAZE HZS 12V-9Ah. Plne nabitý postačuje na niekoľko hodín prevádzky. Notebook má vlastné napájanie.

2.2 Kostra

Kostra robota je tvorená hliníkovými profilmi so štvorcovou podstavou, vďaka čomu je vhodnou platformou pre uchytávanie zvyšného hardwareu nakoľko sa do nej dá vŕtať ľahšie ako keby boli tyče okrúhle. Riadiaca doska leží na drevenej doske, ktorá tvorí podstavu robota. Z hlinníku a plastových líšt je postavená platforma pre notebook, ktorá má na sebe aj popruh pre uchytenie notebooku.



Obrázok 2.1: Robot

Kapitola 3

Rozpoznávanie cesty pomocou počítačového videnia

3.1 Prehľad prác zaoberajúcich sa rozpoznávaním cesty

Uvádzame vybraté práce, z ktorých sme brali inšpiráciu pre naše riešenie alebo považujeme za zaujímavé a prínosné pre ďalšiu prácu.

3.1.1 Segmentácia prahovaním

Článok *"A simple and efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision"*[11] sa zaoberá rozpoznávaním cesty rozsegmentovaním kamerového vstupu na dve časti - cestu a prekážky. Prístup, ktorý si zvolili bol prahovanie, teda určenie prahu, ktorý rozdelí jednotlivé pixely obrázka do dvoch kategorií. Motiváciou tejto práce bola súťaž Grand Challange, ktorá je veľmi podobná súťaži Robotour.
Určenie jedného prahu je z hľadiska úspešnosti vo vonkajšom priestore nedostačujúce nakoľko sa farba cesty môže meniť rovnako ako osvetlenie a okolie cesty.
Tento problém riešili rozdelením obrázka na menšie tak, že brali od spodnéj časti obrázka postupne väčšie časti celku. Pixely týchto obrázkov potom ohodnotili prahovaním, pričom prah vypočítali pomocou histogramu. Potom sledovali koľko pozitívnych pixelov má zachované ohodnotenie aj vo väčšom obrázku. Zobratím štandardnéj odchylky pomerov zachovania medzi obrázkami potom určili kde je najvýraznejši skok. Na tomto mieste bude s vysokou pravdepodobnosťou horizont a tak získali časť obrázka bez horizontu ktorý ohodnotili podľa jeho histogramu.

3.1.2 Segmentácia K-means kompresiou

Práca *"A Hierarchical Segmentation Approach towards Roads and Slopes for Collapse Recognition"*[8] sa zaoberá aj problémom rozlišovania cesty. Cieľom tejto práce bolo rozsegmentovať pixely obrázku do štyroch tried - cesta, vegetácia, obloha a zvyšok obsahujúci malé objekty, značenie a podobne. Motiváciou tejto práce bola automatická detekcia zosuvov pôdy kamerou.

Vegetáciu určujú prahovaním podľa červenej zložky farby R a zelenej G. Takto nesprávne označia aj časti cesty, lebo RGB model je na takéto prahovanie nespoľahlivý. Pixely, ktoré sú nesprávne klasifikované za vegetáciu odfiltrujú konverziou obrázka do HSI farebného modelu a ďalším prahovaním podľa zložky S(saturácia) ich z triedy vegetácie vyhodia.

Ďalej spracuvajú obrázok bez vegetácie, prevedú ho tento raz do LAB farebného modelu vďaka ktorému sa zbavia veľkej časti tieňov odignorovaním L zložky. Zo zvyšných zložiek A a B pomocou Gabor filtru vyextrahujú pre každý pixel feature vektor.

Tento obrázok potom podľa feature vektorov rozsegmentujú K-means algoritmom. Feature vektory majú iný smer na hranách objektov a preto zo segmentácie v spojitých clustroch budú zlúčené iba jednoliate objekty ako je cesta. Vybraním najväčšieho spojitého komponentu získajú cestu a prípadne oblohu. Menšie roztrúsené komponenty budú značky, šum a pododbne. Nakoniec klasifikujú oblohu tak, že ak sa cluster dotýka hornej časti obrázku a je pomerne veľký bude považovaný za oblohu.

3.1.3 Odhad smerovania cesty vanishing point algoritmom

Práca *"Mobile Robot Navigation System in Outdoor Pedestrian Environment Using Vision-Based Road Recognition"*[7] popisuje implementáciu rozpoznania cesty pre potreby robota Beobot 2.0. Táto implementácia sa špecializuje na mestské prostredie so širokými cestami, kde nie je jednoduché určiť okraje cesty, lebo cesta zaberá väčšinu okolitého priestoru. Na zistenie orientácie takýchto ciest využíva Beobot vanishing point algoritmus.

Vanishing point je bod, do ktorého sa zbiehajú v diaľke priamky a úsečky a preto jeho nájdením na obraze možno dobre odhadnúť smer cesty aj jej stred. V práci najskôr z obrazu vyextrahovali hrany s využitím OpenCV[14] knižnice. Tie potom použili na výpočet vanishing pointu, čím si určili smerovanie cesty.

Na udržanie robota v strede cesty použili odhad okrajov cesty daný výpočtom vanishing pointu a smerovanie robota upravujú tak, aby sa robot nachádzal rovnako vzdialený od oboch

okrajov ako na počiatku. Týmto prístupom si vlastne pomyselne zúžia cestu, čo vedie k priamejšiemu chodu robota.

Práca porovnáva prínos jednotlivých súčastí na testovacej ceste (75 metrový úsek so šírkou 6 metrov), zaznamenávajúc vybočenie zo stredu cesty. Vo výsledku na tom bol najhoršie prístup zobratia iba odometrie kolies, čo bolo očakávané a robotovi sa podarilo po ceste prejsť menej ako 35 metrov. Použitie údajov IMU spolu s odometriou kolies dosiahlo lepší výsledok a prešlo celou cestou, no postupne sa robot presunul na okraj cesty. S použitím vanishing pointu pre optimalizáciu smeru sa táto chyba zmiernila. Najúspešnejším sa ukázalo pridanie udržiavania sa v strede pomocou okrajov cesty, kedy sa po celej dĺžke robot od stredu nevzdialil ani o jeden meter (priemer 0.2 metra, maximum 0.7 metra).

3.1.4 Rozpoznávanie cesty pomocou neurónových sietí

V článku *"Fast visual road recognition and horizon detection using multiple artificial neural networks"*[9] je na rozpoznanie cesty využitá kombinácia viacerých neurónových sietí. Vstupom pre tieto siete je oblasť obrázku, ktorá je pre každú sieť inak predspracovaná. Ich výsledok je potom skombinovaný do jedného ohodnotenia.

Predspracovanie vstupu je jednoduché - obrázok sa prekonvertuje do rôznych farebných reprezentácii (RGB, HSV, YUV) z ktorých sú potom ďalej vypočítané ďalšie informácie ako variancia, entropia a energia alebo sa vypočíta normalizovaná hodnota podľa celého obrazku.

Siete potom dostanú rôzne kombinácie týchto vstupov a tvoria jednotný celok v práci nazvaný identifikátor. Jeden takýto identifikátor je natrénovaný na rozpoznanie cesty a jeden sa používa na detekciu horizontu pre odstránenie časti obrázku na ktorom sa cesta nemôže nachádzať.

Autori využívajú implementáciu neurónovej siete FANN[13] a OpenCV[14] pre prácu s obrázkami. Tento článok nadväzuje na predošlú prácu [10], ktorá slúžila aj ako inšpirácia pre prácu M. Nadhajského[1] na ktorú nadväzujeme v tejto práci.

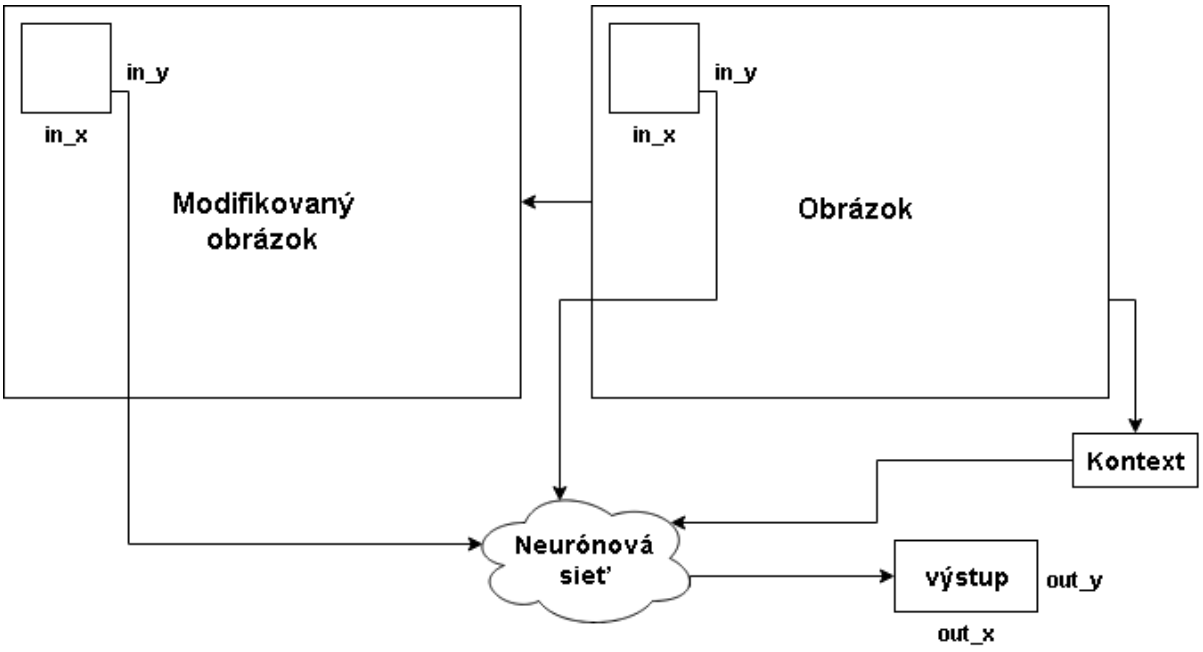
3.2 Využitie neurónových sietí na rozpoznanie cesty

Vhodnou metódou pre rozpoznávanie cesty v prostredí parkov sa ukázalo použitie neurónovej siete v článkoch [9][10] aj v pôvodnej práci Robotour[1]. Cesty parkov bývajú aj v rámci jedného parku rôznorodé a narozdiel od veľkých ciest pre autá neposkytujú pomocné značenie, ktoré by mohlo rozpoznávanie zjednodušiť. Uvádzame postupy použité na vytvorenie neurónovej siete, ktorá má za úlohu rozpoznať cestu z obrázka.

3.2.1 Model

Vstup a výstup

Neurónová sieť dostane na vstup časť obrázka ako hodnoty farebných zložiek vybraných pixelov a prídavné vstupy dvoch typov.
Jedným je kontext, ktorý sa vypočíta z celého obrázka a má popisovať jeho celkovú charakteristiku, nezávisiac na tom, ktorá časť obrázka vstupuje do siete.
Druhým typom prídavného vstupu je prislúchajúca časť modifikovaného obrázka, ktorý bol vopred vytvorený z pôvodného obrázka.
Výstupom siete je ohodnotenie časti obrazu, ktorý bol vybráný ako vstup.



Obrázok 3.1: Vstup a výstup siete

Ohodnotenie celého obrazu získame postupným posúvaním vstupného okna po obrázku

pričom výstupy patrične poskladáme podľa toho, ako sme ním hýbali. Takto vznikne ohodnotenie celku, ktoré môžme využiť ďalej pre výber správnej akcie.

3.2.2 Predspracovanie vstupu

Vstup siete môžeme predspracovať a tým uľahčiť neurónovej sieti klasifikáciu. Vybraním správnych metód sa dá zvýšiť úspešnosť a urýchliť čas potrebný na trénovanie siete. Takéto predspracovanie je treba navrhnúť konkrétne pre špecifický problém.

LAB colorspace

LAB model bol navrhnutý aby lepšie simuloval ľudské vnímanie oproti RGB modelu, navrhnutému pre zobrazovacie zariadenia. Jednou z jeho troch zložiek je aj jas (L - lightness) a vďaka tomu sú rovnaké farby s iným jasom ľahko rozoznateľné. Tento fakt dokážeme využiť na klasifikáciu cesty ktorá je nerovnomerne osvetlená alebo sú na ňu vrhané tiene. Ostatné dve zložky tiež dobre charakterizujú farbu tým, že blízke farby sú pre človeka podobné a farby v modeli vzdialené sú výrazne odlišné. Konverzia z RGB vyzerá nasledovne[12]

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.2.1}$$

$$X_2 = \frac{X}{0.950456} \tag{3.2.2}$$

$$Z_2 = \frac{Z}{1.088754} \tag{3.2.3}$$

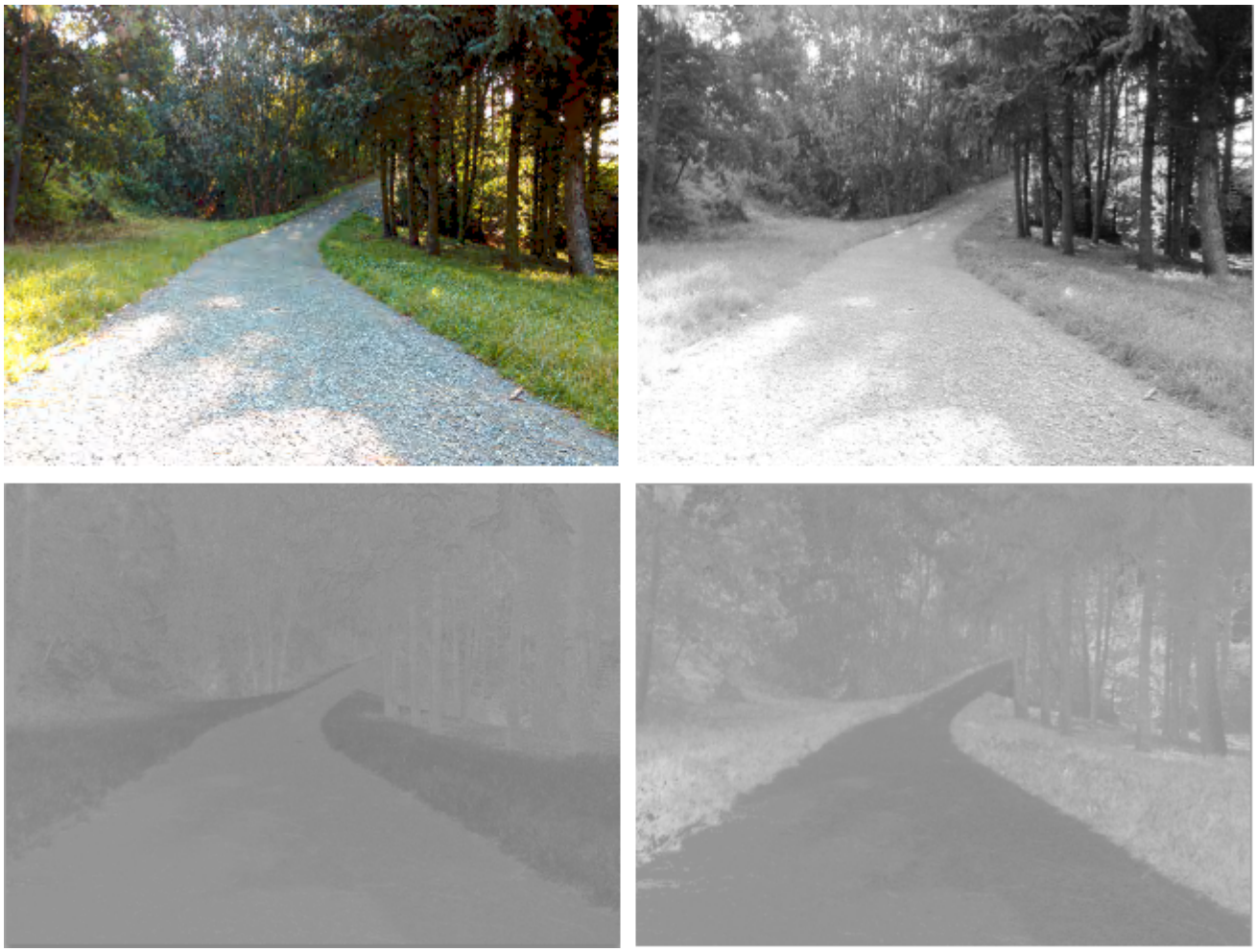
$$L = \begin{cases} 116 * Y^{1/3} - 16, & \text{pre } Y > 0.008856 \\ 903.3 * Y, & \text{pre } Y \leq 0.008856 \end{cases} \tag{3.2.4}$$

$$a = 500(f(X_2) - f(Y)) + 128 \tag{3.2.5}$$

$$b = 200(f(Y) - f(Z_2)) + 128 \tag{3.2.6}$$

kde

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & \text{pre } t > 0.008856 \\ 7.787t + \frac{16}{116}, & \text{pre } t \leq 0.008856 \end{cases} \tag{3.2.7}$$



Obrázok 3.2: Obrázok cesty a jeho L,A,B zložky zobrazené čiernobielo

Na obrázku vyššie si všimneme, že v zložkách A a B sú tieňe a odrazy slnka potlačené a teda popisujú farbu nezávisle na osvetlení. Tento fakt by mal algoritmu spracujúcemu obraz pomôcť lepšie rozlišovať predmety, v našom prípade cestu, podľa farby, nakoľko svetelné podmienky netvoria výrazné hrany. V ľavom dolnom rohu je stále možné rozoznať tiene, avšak oproti rozdielu medzi trávou a cestou je tento tieň nevýrazný.

3.2.3 Prídavné vstupy

Časť s vysokou pravdepodobnosťou výskytu cesty

Tento kontext sa priamo týka aplikácie pre autonómneho robota. Keďže robot kamerou nehýbe a stále by sa mal hýbať po ceste, je istá časť obrazu, ktorá zobrazuje cestu počas väčšiny prevádzky. Podobné zmýšľanie využíva aj práca o segmentácii[8] na detekovanie oblohy a článok o využití prahovania na detekciu cesty[11]. Takýto prídavný vstup tiež zlepšil úspešnosť siete v pôvodnej implementácii[1].

V našom prípade je to okolie stredu dolnej časti obrazu a teda z tejto oblasti zoberieme priemernú hodnotu každej zložky a posunieme ju neurónovej sieti. Vďaka tomuto kontextu by malo byť pre sieť jednoduchšie klasifikovať cesty rozličných typov(napr ak ide po červenej ceste). Ako nadstavbu môžeme vybrať takýchto oblastí viacero a zobrať medián ich hodnôt. Medián berieme z usporiadania podľa zložiek A a B lebo svetlosť chodníka závisí od pozície a nie je tak dobrou charakteristikou ako jeho farba. Pokiaľ sa na väčšine z oblastí nachádza cesta, získame jej hodnoty.

Obrázok 3.3: Zvýraznená časť má vysokú pravdepodobnosť cesty

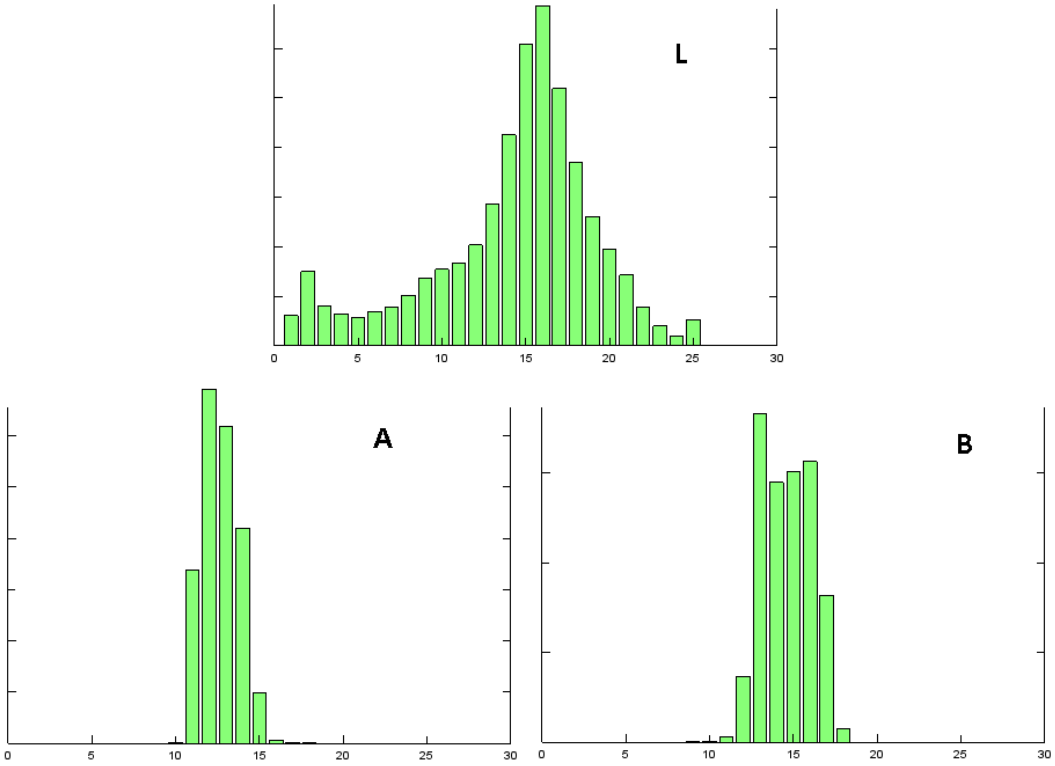
Nevýhodou tohoto kontextu môže byť náchylnosť na to, že sa cesta, ktorá náhle mení svoju charakteristiku (z červených kachličiek prejde hneď na čierne) nemusí byť správne klasifikovaná a prípad, keď robot stojí pred trávnatou oblasťou, môže chybne považovať všetku trávu ako cestu.
Aby sa čo najlepšie predišlo týmto nevýhodám, je nutné v trénovacích dátach poskytnúť aj obrázky bez cesty v danej časti obrazu.

Histogram

V pôvodnej práci[1] sa ako najúspešnejší prídavný vstup ukázal histogram farieb, preto ho prinášame aj do novej implementácie, aby sme zsistili, či je jeho prínos pri zvolení iného farebného modelu stále pozitívny.
Histogram obrázka je tvorený pre každú z troch zložiek farby obrázka teda je to trojica vektorov Hl, Ha, Hb s počtom výskytov hodnôt danej zložky, ktoré sú následne normalizované tj.

$$\sum Hl = \sum Ha = \sum Hb = 1 \tag{3.2.8}$$

Histogram dobre reprezentuje prostredie z obrázka a celkové osvetlenie. Vďaka tomuto by mal pomôcť hlavne v okrajových prípadoch(poľná cesta v lese je chodník a v zástavbe nie) a s neštandardnými farbami - outliermi (napríklad výrazne červený štvorec na zemi môže byť kachlička, ale ak na zbytku obrázka nie je veľa červenej bude to skôr prekážka).
Keďže pracujeme s 8bitovou farbou, zložky nadobúdajú hodnoty 0-255 teda veľkosť histogramu by bola 3*256 preto pridávame parameter p a hodnoty zgrupujeme do $\frac{256}{p}$ bucketov tak, že hodnota h je priradená bucketu $\left\lfloor \frac{h}{p} + \frac{1}{2} \right\rfloor$ čím sa síce zníži rozlíšenie histogramu ale jeho veľkosť sa zredukuje na $\frac{3*256}{p}$.



Obrázok 3.4: Histogram zložiek L,A,B, rozdelené krokom $p = 10$

Na histograme obrázku vyššie je možno si všimnúť, že hodnoty A a B zložiek nenadobúdajú nízke a vysoké hodnoty. Toto je zapríčinené konverziou obrázka z RGB. Histogram preto môžeme ešte orezať o tieto nevyskytujúce sa hodnoty.

Segmentácia pomocou K-means kompresie

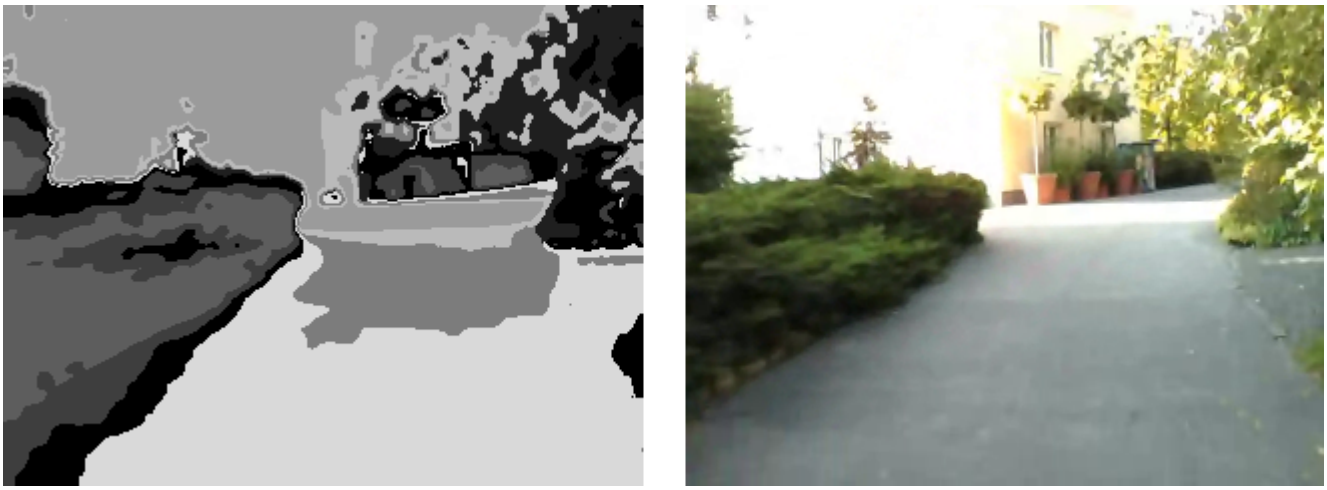
V práci o detekcii zosuvov pôdy [8] autori úspešne použili k-means algoritmus pre rozdelenie obrázku na časť s cestou a bez nej. Skúsime, aký prínos bude mať použitie takto modifiko-

vaného obrázku pre neurónovú sieť.

K-means kompresia je regresívna metóda, ktorá vyberie k centier $\mu_1..\mu_k$ daných zložkami farieb ku ktorým pričlení jednotlivé pixely obrázka $(p_1, p_2 \dots p_n)$ tak, aby sa minimalizovala chyba

$$\sum_{i=1}^{k} \sum_{p \in S_{\mu_i}} ||p - \mu_i||^2 \tag{3.2.9}$$

Obrázok sa najprv prefiltruje pomocou blur algoritmu, čím sa potlači šum a výsledok bude hladký a keďže je cesta spojitá, mala by byť obsiahnutá v segmentoch v ktorých nie je tráva. Pridaním rozsegmentovaného obrazu k vstupu by sa mohla zvýšiť úspešnosť klasifikácie a tréningu na okrajoch cesty.



Obrázok 3.5: K-means kompresia(k=8) a pôvodný obrázok

Kapitola 4

Lokalizácia

Pre správne zistenie polohy na mape a výpočet optimálneho smeru potrebujeme počítať vzdialenosti a smery na zemeguli. Vysokou presnosťou výpočtov chceme minimalizovať chyby v algoritmoch plánovania.

4.1 Sférická geometria

Sférická geometria sa zaoberá dvojrozmernými objektami na povrchu gule. Je to špeciálny prípad eliptickej geometrie. Uvádzame definície a vzťahy, ktoré využivame v algoritmoch plánovania a lokalizácie.

4.1.1 Bod

Bod P na guli je daný jeho súradnicovou šírkou (ozn. $P.lat$) a dĺžkou (ozn. $P.lon$). Šírka a dĺžka vyjadrujú uhlovú vzdialenosť od dvoch na seba kolmých rovín pretínajúcich stred gule, ktoré určujú súradnicovú sústavu. V prípade zemegule sú dané rovníkom a poludníkom.

4.1.2 Vzdialenosť dvoch bodov

Nech $P1$ a $P2$ sú body na guli s polomerom R. Ich vzdialenosť $d(P1,P2)$ a radiálnu vzdialenosť $rd(P1,P2)$ vypočítame nasledovne:

$$\Delta lat = P2.lat - P1.lat \tag{4.1.1}$$

$$\Delta lon = P2.lon - P1.lon \tag{4.1.2}$$

$$rd(P1,P2) = 2 \cdot sin^{-1}(\sqrt{sin(\frac{\Delta lat}{2})^2 + cos(P1.lat) \cdot cos(P2.lat) \cdot sin(\frac{\Delta lon}{2})^2}) \quad (4.1.3)$$

$$d(P1,P2) = rd(P1,P2) \cdot R \quad (4.1.4)$$

4.1.3 Výpočet počiatočného azimutu

Počiatočný azimut(uhol od severného smeru) pre smer z bodu A do bodu B (ozn $brng_{A,B}$) vypočítame nasledovne:

$$brng_{A,B} = atan2(sin(B.lon - A.lon) \cdot cos(B.lat), \quad (4.1.5)$$

$$cos(A.lat) \cdot sin(B.lat) - sin(A.lat) \cdot cos(B.lat) \cdot (B.lon - A.lon)) \quad (4.1.6)$$

4.1.4 Prienik dvoch priamok

Majme priamku $Q1$ danú bodom $P1$ a azimutom $brng_1$ a priamku danú bodom $P2$ a azimutom $brng_2$ na guli s polomerom R. Potom ich prienik je bod $P3$ ktorý získame pomocou vzťahov:

$$\theta_a = cos^{-1}(sin(P2.lat) - sin(P1.lat) \cdot \frac{cos(rd(P1,P2))}{sin(rd(P1,P2))} \cdot cos(P1.lat)) \quad (4.1.7)$$

$$\theta_b = cos^{-1}(sin(P1.lat) - sin(P2.lat) \cdot \frac{cos(rd(P1,P2))}{sin(rd(P1,P2))} \cdot cos(P2.lat)) \quad (4.1.8)$$

$$\theta_{12} = \begin{cases} \theta_a & \text{ak} \quad sin(P2.lon - P1.lon) > 0 \\ 2\pi - \theta_a & \text{inak} \end{cases} \quad (4.1.9)$$

$$\theta_{21} = \begin{cases} 2\pi - \theta_b & \text{ak} \quad sin(P2.lon - P1.lon) > 0 \\ \theta_b & \text{inak} \end{cases} \quad (4.1.10)$$

$$\alpha_1 = (brng_1 - \theta_{12} + \pi)\%2\pi - \pi \quad (4.1.11)$$

$$\alpha_2 = (brng_2 - \theta_{23} + \pi)\%2\pi - \pi \quad (4.1.12)$$

$$\alpha_3 = cos^{-1}(-cos(\alpha_1) \cdot cos(\alpha_2) + sin(\alpha_1) \cdot sin(\alpha_2) \cdot cos(rd(P1,P2))) \quad (4.1.13)$$

$$\delta_{13} = atan2(sin(rd(P1,P2)) \cdot sin(\alpha_1) \cdot sin(\alpha_2), cos(\alpha_2) + cos(\alpha_1) \cdot cos(\alpha_3)) \quad (4.1.14)$$

$$P3.lat = sin^{-1}(sin(P1.lat) \cdot cos(\delta_{13}) + cos(P1.lat) \cdot sin(\delta_{13}) \cdot cos(\theta_{13})) \quad (4.1.15)$$

$$\Delta lon = atan2(sin(\theta_{13}) \cdot sin(\delta_{13}) \cdot cos(P1.lat), cos(\delta_{13}) - sin(P1.lat) \cdot sin(P3.lat)) \quad (4.1.16)$$

$$P3.lon = (P1.lon + \Delta lon + \pi)\%2\pi - \pi \quad (4.1.17)$$

4.1.5 Vzdialenosť bodu od priamky

Vzdialenosť bodu P od priamky Q určenej bodom S a azimutom $brng_Q$ na guli aproximujeme tak, že zoberieme vzdialenosť bodu P od prieniku priamky Q a priamky K, ktorá je na ňu kolmá teda:

$$brng_K = \begin{cases} brng_Q + 90 & \text{ak} \quad (2\pi + brng_{S,P} - brng_Q)\%2\pi > \pi \\ brng_Q - 90 & \text{inak} \end{cases} \tag{4.1.18}$$

a prechádza bodom P. Pre malé vzdialenosti s ktorými sa v tejto práci zaoberáme (rádovo desiatky metrov) je chyba spôsobená zakryvením smeru od kolmého dostatočne malá, aby nespôsobovala výrazné výchýlenie.

4.1.6 Prienik priamky a kružnice

Na kružnicu k s polomerom r a stredom P použijeme flattening tak, aby sme mohli prieniky vypočítať ako v euklidovskom priestore. Elipsa, ktorú takto získame je daná rovnicou

$$\frac{x^2}{a} + \frac{y^2}{b} = 1 \tag{4.1.19}$$

kde

$$f_{lat} = d(P, P1 = (P.lat + \varepsilon, P.lon)) \tag{4.1.20}$$

$$a = (\frac{r}{f_{lat}} \cdot \varepsilon) \tag{4.1.21}$$

$$f_{lon} = d(P, P2 = (P.lat, P.lon + \varepsilon)) \tag{4.1.22}$$

$$b = (\frac{r}{f_{lon}} \cdot \varepsilon) \tag{4.1.23}$$

Chyba použitia vzťahu pre výpočet prienikov elipsy a priamky v euklidovskom priestore je pri kružniciach s malým polomerom relatívne k polomeru gule a priamkach, ktoré sú dané bodmi blízko k P zanedbateľná.

Prienik elipsy a priamky v euklidovskom priestore

Majme elipsu e so stredom v počiatku danú rovnicou

$$\frac{x^2}{a} + \frac{y^2}{b} = 1 \tag{4.1.24}$$

a priamku p danú rovnicou

$$y = m \cdot x + c \tag{4.1.25}$$

ich potencionálne prieniky $i1$ a $i2$ vypočítame nasledovne:

nech

$$\alpha = \frac{2 \cdot a^2 \cdot c \cdot m}{2 \cdot (b^2 + a^2 \cdot m^2)} \tag{4.1.26}$$

$$\theta = \frac{b^2 - c^2}{b^2 + a^2 * m^2} + \frac{a^2 * c^2 * m^2}{(b_2 + a_2 * m_2)^2} \tag{4.1.27}$$

$$\beta = \begin{cases} a \cdot \sqrt{\theta} & \text{ak} \quad \theta > 0 \\ 0 & \text{inak} \end{cases} \tag{4.1.28}$$

potom

$$i1.x = -\alpha + \beta \tag{4.1.29}$$

$$i1.y = m \cdot i1.x + c \tag{4.1.30}$$

$$i2.x = -\alpha - \beta \tag{4.1.31}$$

$$i2.y = m \cdot i2.x + c \tag{4.1.32}$$

Ak $i1 \neq i2$ potom $i1, i2$ sú prieniky elipsy e a priamky p inak je ich prienik nulový, alebo je $i1$ dotyčnicou.

4.2 Určenie polohy na mape

4.2.1 Mapa

Majme množinu M dvojíc bodov (P_a, P_b), táto množina tvorí mapu M a dvojice v nej obsiahnuté nazývame cesty.

4.2.2 Trasa

Trasa T dĺžky d na mape M je usporiadaná postupnosť ciest $(P_{a_1}, P_{b_1}) \ldots (P_{a_d}, P_{b_d})$ taká, že

$$\forall i \in N : (P_{a_i}, P_{b_i}) \in M \tag{4.2.1}$$

$$\forall i \in N,\ 0 < i < d : P_{b_i} = P_{a_{i+1}} \tag{4.2.2}$$

4.2.3 Najbližší bod na ceste

Majme cestu (P_a, P_b). Bodom na ceste (P_a, P_b) nazývame každý bod K ležiaci na priamke danej bodmi P_a, P_b pričom platí

$$d(K, P_a) \leq d(P_a, P_b) \tag{4.2.3}$$

$$d(K, P_b) \leq d(P_a, P_b) \tag{4.2.4}$$

Najbližším bodom na ceste (P_a, P_b) k bodu R je potom bod na ceste (P_a, P_b) Q taký, že

$$A \text{ je bod na ceste } (P_a, P_b) \Rightarrow d(Q, R) \leq d(A, R) \tag{4.2.5}$$

Kapitola 5

Plánovanie

V tejto časti popíšeme algoritmy plánovania. Delíme ho na dlhodobé ktoré sa vykonáva len na začiatku behu alebo pri špeciálnych údalostiach (neočakávaná zmena polohy/chyba hardwaru a pod.) a krátkodobé, ktoré rozhoduje podľa aktuálnej situácie raz za cyklus.

5.1 Dlhodobé

5.1.1 Mapa

Robotour povoľuje iba použitie mapy z projektu OpenStreetMap. Mapa obsahuje súradnice ciest a ich typ a teda postačuje ako základ navigácie v danom prostredí. Keďže je OpenStreetMap editovateľná užívateľmi, môžu ju pred súťažou editovať aj účastníci súťaže a opraviť tak nepresnosti, ktoré zistili na mieste. Cesty ktoré niesu zmapované na OpenStreetMap nemusia rozhodcovia uznať za cestu v rámci pravidiel súťaže.

5.1.2 Výber trasy

Keď už robot pozná svoju polohu, mapu svojh okolia a cieľ, potrebuje naplánovať trasu tak, aby sa k cieľu dostal použitím ciest. Pokiaľ je robot mimo svojej trasy, alebo si ešte žiadnu neurčil, využije mapu a údaje o svojej pozícii pre vytvorenie novej trasy.

Najkratšia

Ako trasu si volí postupnosť $s_1, s_2, ...s_n$ segmentov mapy začínajúc od svojej polohy na segmente s_1 do cieľa na segmente s_n. Túto postupnosť volí tak, aby sa minimalizovala očaká-

vaná prejdená vzdialenosť. Vybraním kratšej trasy sa znižuje šanca na chybu.

5.2 Krátkodobé

5.2.1 Prekážky

Po detekovaní prekážky sa robot presunie do subrutiny pre riešenie prekážok. Najprv čaká, či prekážka nezmizne (teda fyzicky odíde alebo bola detekcia falošná). Pokiaľ ani po krátkej dobe nemá robot voľnú cestu zapamätá si robot na ktorej strane je vidieť viac cesty, zacúva späť, natočí sa zapamätaným smerom a pokračuje ďalej.

5.2.2 Nerozoznaný chodník

Ak na výstupe kamery nedokážeme rozoznať cestu, nezastavujeme, lebo sa na nej aj tak môžeme nachádzať. V súťaži Robotour je vyjdenie z cesty a dlhodobé státie na mieste trestané rovnako (označí sa posledná poloha a jazda sa pre robota skončí) a teda chceme aspoň skúsiť cestu nájsť. Ísť iba podľa výstupu lokalizácie je priveľmi riskantné a preto robot zníži rýchlosť a budeme ním opatrne obracať zo strany na stranu smerujúc k medzicieľu. Takýmto obracaním sa robotovi môže podariť objaviť cestu a vydať sa k nej keďže strata cesty mohla byť zazpríčinená napríklad odrazom svetla a oslnením kamery z jedného smeru. Ak to tak nebolo a iba nevieme cestu rozpoznať od okolia stále pôjde robot podľa mapy.

5.2.3 Zídenie z trasy

Pokiaľ sa robot vyberie zlým smerom, napríklad na križovatke, bude sa uhol medzi jeho smerovaním a medzicieľom zvyšovať. Ak tento uhol prekoná hranicu 150°, robot zastane a na mieste sa otočí tak, aby smeroval na medzicieľ. Prirodzeným spôsobom by sa robotovi na ceste nepodarílo otočiť, lebo podľa medzicieľu iba koriguje svoj smer a to mu nedovoľuje zatáčať prudko. Ak robot zišiel z cesty tak, že nie je ani v okolí svojej trasy (napríklad ak sa vybral cestičkou, ktorá je skoro rovnobežná s naplánovanou, no postupne sa vzďaľuje) jednoducho prepočíta novú trasu od pozície na ktorej sa nachádza.

5.2.4 Výber medzicieľa

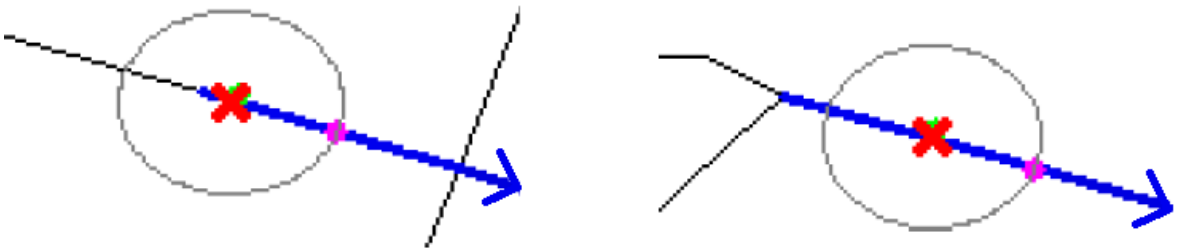
Medzicieľ je bod na trase, ktorým je nutné prejsť, vybraný tak, aby smer od polohy robota k nemu zodpovedal čo najlepšie optimálnej trase. Jeho správny výber je kritický pre správne prechádzanie v zákrutách.

Výber v okolí

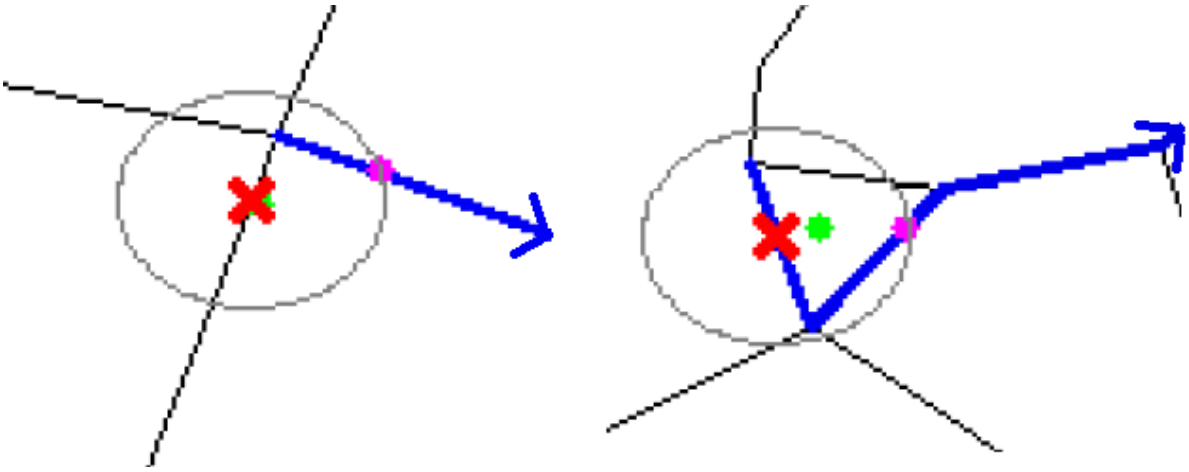
Ako medzicieľ vyberieme bod trasy ktorý je vzdialený n metrov od terajšej polohy. Tento bod vypočítame ako prienik kružnice s polomerom n metrov a orientovaných úsečiek tvoriacich trasu, pričom prieniky vzniknuté vojdením do kružnice ignorujeme. Pokiaľ je prienikov viac, zoberieme prienik, ktorý je po trase bližšie k cieľu ako poloha robota a zároveň je od cieľa zo všetkých takých prienikov najvzdialenejší.
Takto zvolený medzicieľ núti robota zahýbať do správneho smeru ešte pred križovatkou a nezávisí od toho na ktorom segmente sa robot nachádza.

Prípady

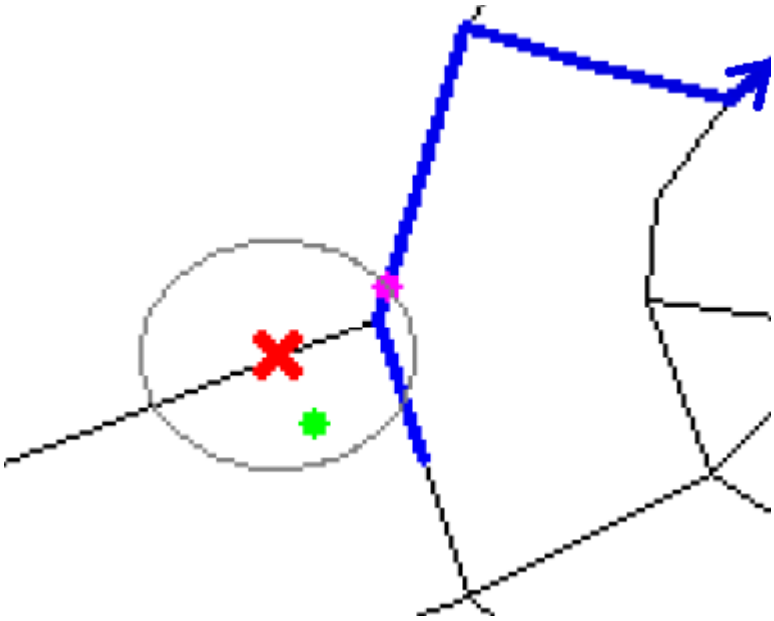
Uvádzame názorné ukážky rôznych prípadov polohy robota na mape s naplánovanou trasou a vypočítaný medzicieľ.



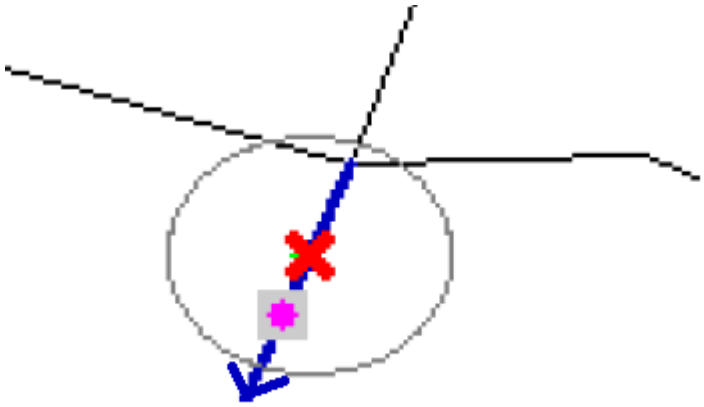
Obrázok 5.1: Medzicieľ (ružová bodka) je braný ako prienik okolia a trasy. Berie sa iba vychádzajúci prienik.



Obrázok 5.2: Na križovatkách je robot vedený strihať zákruty, čo zabezpečí, že ich neminie. V ostrých zákrutách(vpravo) to však môže predstavovať problém, lebo je robot vedený až kolmo na cestu a tým sa jeho postup spomaľuje.



Obrázok 5.3: GPS signál (zelená bodka) môže byť nepresný čo može spôsobiť "preskočenie"na inú cestičku. Ak sa robot stále nachádza na modrej čiare vpravo je zbytočne vedený vpravo no smeruje aj hore a teda bude ďalej postupovať(vpravo sa hýbať nemôže lebo ho zastaví rozpoznávanie cesty).



Obrázok 5.4: Cieľ (šedý štvorec) je uprednostnený pred prienikom pokiaľ je v okolí.

5.2.5 Výpočet smeru

Smer závisí od polohy medzicieľa, robota a od ohodnotenia terénu pred robotom poskytnutého neurónovou sieťou. Najskôr z výstupu siete určíme, ktorými smermi sa môžeme vydať bez vybočenia mimo cestu. Toto dosiahneme spriemerovaním hodnôt v trojuholníkoch indikujúcich smer, čím získame ohodnotenia smerov

$$h(smer) = \frac{1}{N}\sum_{k}^{N} pixel_{k,smer} \tag{5.2.1}$$

kde *smer* je index použitého trojuholníka a trojuholníky sú zoradené podľa x-ovej súradnice ich najvrchnejšieho bodu. Od týchto ohodnotení ešte odčítame hodnotu $threshold = 0.4$ čím zahodíme smery s malou pravdepodobnosťou pre ďalšie kroky

$$h(smer) = max(h(smer) - 0.4\ ,\ 0) \tag{5.2.2}$$

Pomenujme smer ktorý ukazuje najbližšie k medzicieľu Δ, počet rôznych smerov g a vytvorme koeficient $coef_{smer}$ pre každý smer podľa

$$coef_{smer} = \frac{g - |\Delta - smer|}{g * Q} \tag{5.2.3}$$

$$coef_{smer} = coef_{smer} + 1 \tag{5.2.4}$$

týmto koeficientom potom prenásobíme prislúchajúce ohodnotenia smerov a ako najlepší smer S vyberieme smer s maximálnym ohodnotením.

$$S = \arg\max_{smer}(coef_{smer} h(smer)) \tag{5.2.5}$$

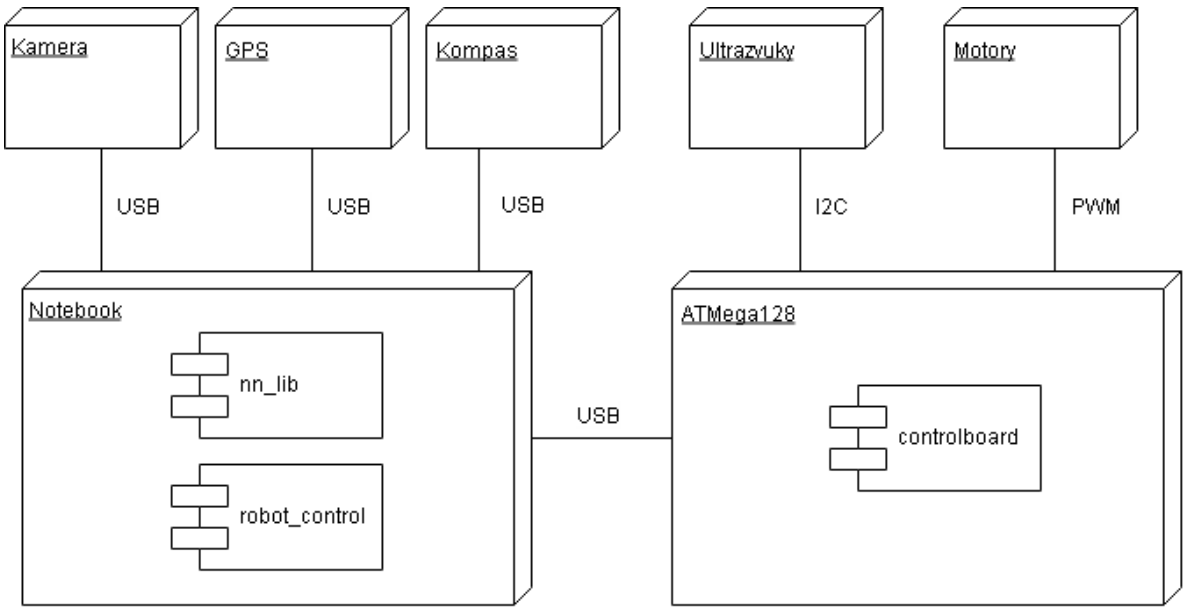
Smer s nedostatočným ohodnotením teda nebude vybraný ak je dostupný aspoň jeden s ohodnotením dostatočným a uprednostnené budú smery ktoré sa málo líšia od smeru k medzicieľu. Konštantou Q môžme nastavovať prioritu smeru k medzicieľu oproti priorite pohybu po lepšie ohodnotenej ceste.

Kapitola 6

Implementácia

V tejto kapitole popíšeme implementáciu riešenia, architektúru softwareu a hardwareu. Detailnejšie uvádzame aj vylepšenia a zmeny oproti starému riešeniu. Implementovaný software je voľne dostupný na adrese http://dai.fmph.uniba.sk/projects/smelyzajko/ pod open-source licenciou.

6.1 Architektúra riešenia



Obrázok 6.1: Deployment diagram riešenia

Základom robota je základná doska Sbot s ATMega128 mikrokontrolerom. Zabezpečuje pohyb robota a rutinu využívajúcu ultrazvuky pre detekciu prekážok. Zložitejšiu časť má na

starosti notebook, ktorý s doskou komunikuje pomocou USB rozhrania simulujúceho sériový port. Ostatné zariadenia sú prepojené s notebookom.

6.2 Controlboard

Controlboard je program na Sbot doske bežiaci na mikrokontroleri ATMega128. Implementácia v jazyku C je rozdelená do viacerých súborov podľa obsahu. Hlavný súbor *main* zlučuje ostatok do funkčného celku.

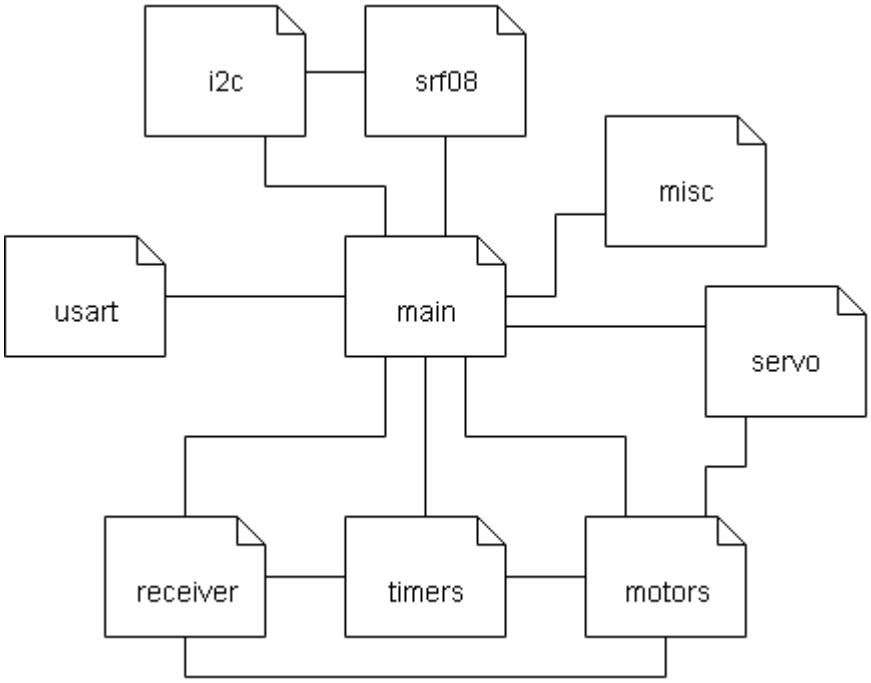
Jednou z úloh tohto programu je spravovať výstup ultrazvukov. Komunikáciu s nimi zabezpečuje I^2C protokol v súbore *i2c*. O zvyšok sa starajú metódy v *srf08* a poskytujú výstup ultrazvukov ďalej.

Ďalšou úlohou je správa motorov, na ktorú slúžia súbory *motors* a *servo*. *Motors* spracúva požiadavky smeru a rýchlosti a *servo* už iba vypočítané rýchlosti prevádza na PWM signál pre motory. S motormi sa ešte spája súbor *timers*, ktorý obsahuje interrupty pre získavanie údajov odometrie motorov, stav núdzového tlačidla a sleduje aj príkazy od diaľkového ovládania, ktoré je pre túto prácu relevantné iba na testovanie.

Keďže na aktualizovanie rýchlosti motorov používame interrupty timera a táto rutina je závislá na spätnej väzbe z enkóderov, ktorú počas čakania v hlavnom cykle nezískavame, je v súbore *misc* implementovaná špeciálna verzia wait funkcie, ktorá pri dlhom čakaní aktualizuje hodnoty z enkóderov.

Všetky výstupy a metódy sú prístupné hlavnej časti programu main na detekciu prekážok pričom program poskytuje možnosť komunikácie cez protokol implementovaný v *usart* a podľa neho potom mení smer a rýchlosť.

Architektúra tohoto programu sa od pôvodnej implementácie nemenila, no implementácia jednotlivých častí bola doplnená, zmenená alebo opravená podľa potrieb robota a výsledkov testovaní.



Obrázok 6.2: Súbory controlboard programu

6.3 Robot_controll

Hlavná trieda *main* zlučuje ostatné triedy do programu robot_controll, ktorý beží na notebooku robota a vykonáva rozhodovaciu funkciu s úlohou doraziť do cieľa.

Program má potrebuje viac vlákien nakoľko komunikuje s rôzne rýchlymi zariadeniami. Jedným z týchto zariadení je GPS modul, ktorý je spravovaný triedou *GpsThread*. Táto trieda sa pripája na GPS modul a v cykle updatuje polohu ním určenú, ktorú poskytuje ďalej programu. Podobne vyzerá aj *ImuThread* spravujúci výstup kompasu.

Komunikáciu s Sbot doskou zabezpečuje *SbotThread*. Narozdiel od predošlých vlákien je ale komunikácia obojsmerná a Sbot prijíma príkazy ohľadom smeru, rýchlosti a či má Sbot ignorovať prekážky detekované ultrazvukovými senzormi. Tieto príkazy poskytuje v špeciálnych príapadoch hlavná časť programu (riešenie prekážok, robot sa nachádza v cieli a pod.) ale bežné riadiace signály posiela trieda *Coordinate*.

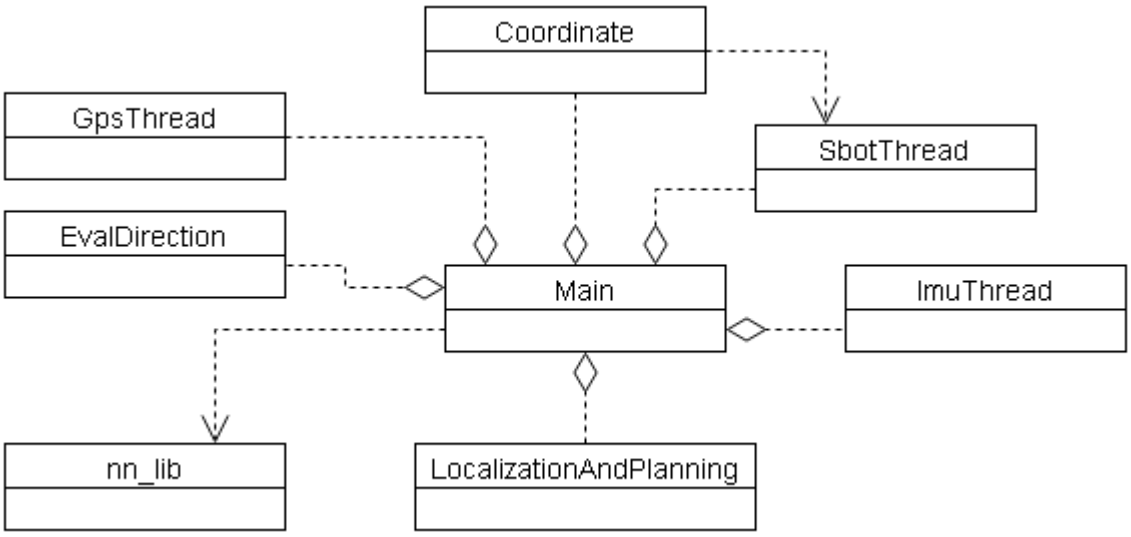
Coordinate preberá smer určený neurónovou sieťou vypočítaný triedou *EvalDirection* z dát, ktoré boli spracované knižnicou nn_lib a tento smer spája s výsledkom lokalizácie a plánovania(*LocalizationAndPlanning*) a natočením ktoré udáva kompas. Po výpočte požadovaného smeru ho oznámi programu počúvajúcemu na Sbot doske, ktorý sa už postará o ostatok.

Oproti pôvodnej implementácii nastala najväčšia zmena v prípade lokalizácie a plánovania,

kde boli prerobené takmer všetky časti. S narastajúcou šancou dorazenia do cieľa boli pridané aj rutiny chovania v *main*. Zvyšné časti boli ladené v menšom rozsahu.

6.3.1 OpenCV

Robot_controll ďalej okrajovo využíva knižnicu OpenCV [14]. OpenCV je rozsiahla knižnica počítačového videnia no využívame ju iba na jednoduché vybranie výstupu kamery, prevod z RGB do LAB a prípadnú K-means kompresiu. Všetky tieto akcie sú pomerne jednoduché a preto tvorí OpenCV zbytočný overhead a požiadavku pre program a mal by byť neskôr nahradený inou, menej náročnou implementáciou.



Obrázok 6.3: Class diagram programu robot_controll.

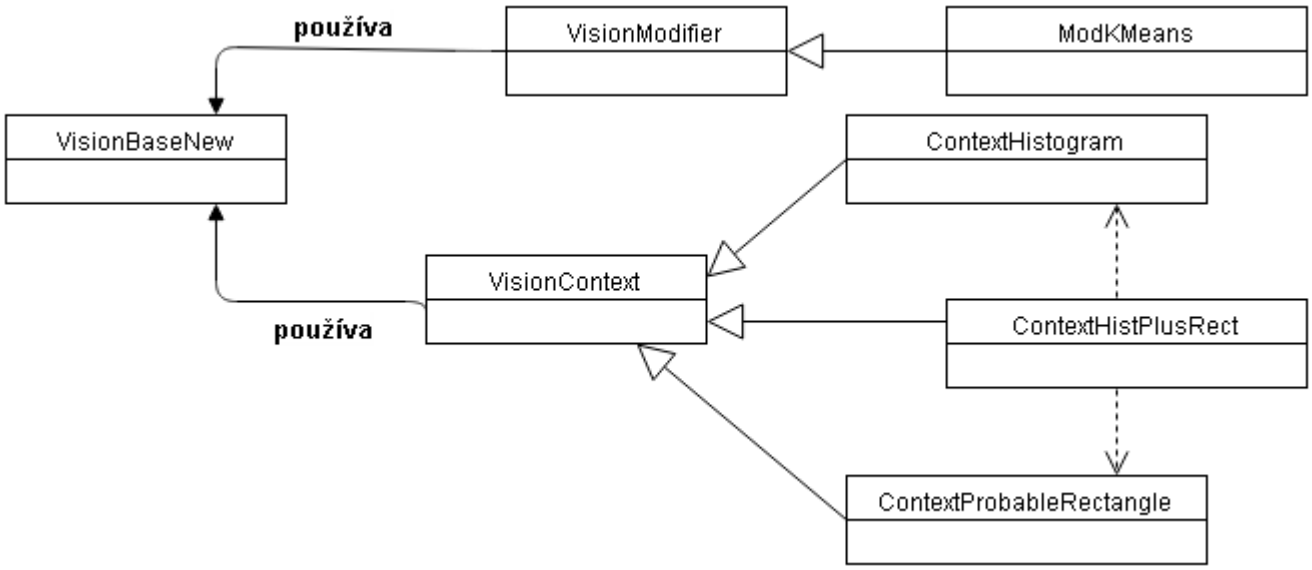
6.4 Rozpoznávanie cesty

6.4.1 FANN

Na rozpoznávanie používame implementáciu neurónovej siete FANN[13] ktorá na učenie využíva iRprop algoritmus. Táto knižnica je pre náš problém vhodná, nakoľko máme veľké množstvo trénovacích vstupov, ktoré sa rýchlo spracujú iRprop algoritmom. FANN sa stará aj o pridanie biasov do každej vrstvy siete vďaka čomu je používanie jednoduchšie.

6.4.2 Knižnica nn_lib

Neurónovú sieť spravuje trieda VisionBaseNew, ktorá poskytuje metódy pre tvorbu trénovacích vstupov z pripravených súborov, predikciu neurónovou sieťou a samotné trénovanie siete. Táto trieda je podobná pôvodnej implementácii [1] no poskytuje viac možností nastavenia. Ďalej boli vytvorené abstraktné triedy VisionContext a VisionModifier pre jednoduchú implementáciu rôznych prídavných vstupov. VisionContext pre vstupy, ktoré sú pre jeden obrázok nezávislé od sledovanej oblasti obrázku (histogram, oblasť s vysokou pravdepodobnosťou cesty) a VisionModifier pre prídavné vstupy, ktoré sa menia podľa pozorovanej oblasti (segmentovanie K-means metódou).



Obrázok 6.4: Class diagram knižnice nn_lib.

6.4.3 Trénovanie

Na trénovanie siete sa použili fotky z parkov, ktoré boli získané počas testovania a počas minulých ročníkov Robotour. Ku každej fotke bolo ručne vytvorené ohodnotenie človekom a preto obsahujú chyby najmä na okrajoch cesty, v prípadoch keď je nejasné, kde je cesta kvôli napadanému lístiu a podobne. Tieto chyby avšak tvoria iba malú časť príkladov a dobre vystihujú želaný výstup.

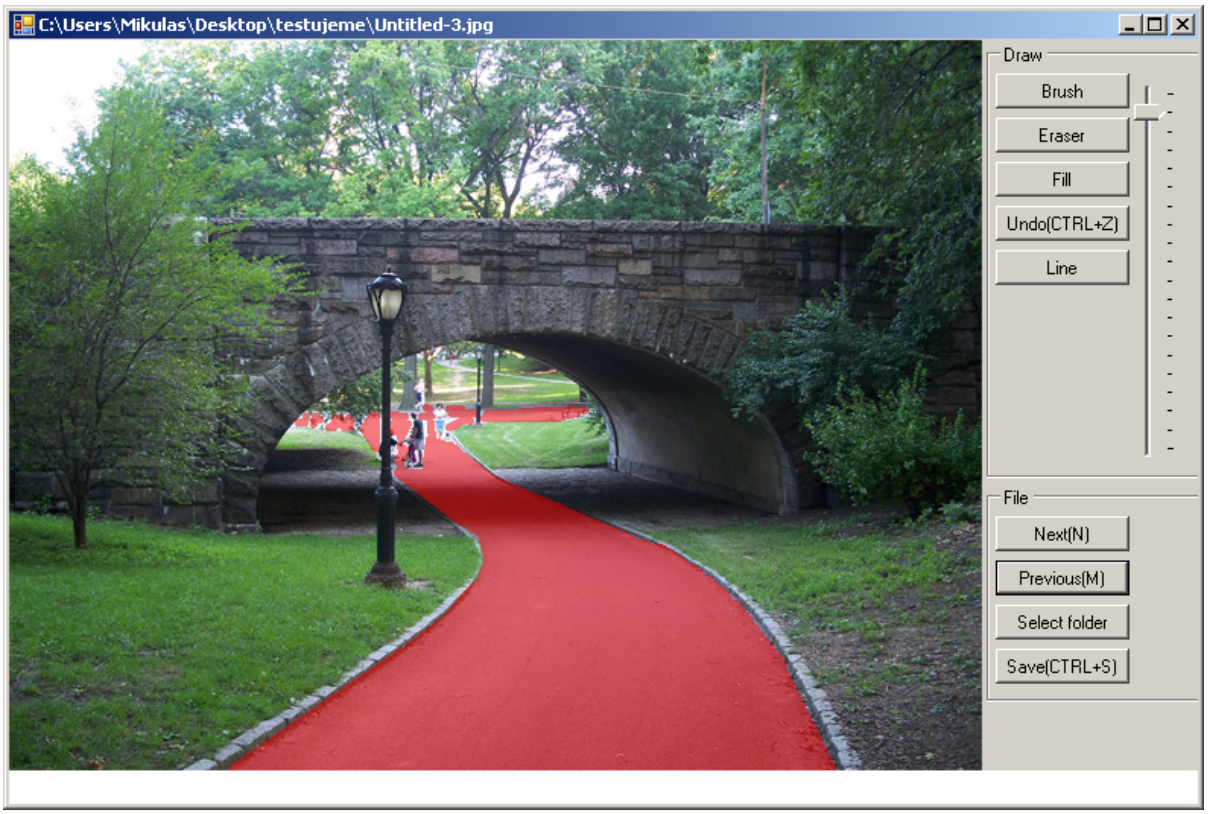
Program pre tvorbu trénovacích dát

Pôvodný program pre tvorbu trénovacích vstupov bol nedostačujúci možnosťami a neintuitívnym ovládaním, čo zbytočne znepríjemňovalo proces tvorby výstupov. Preto bol vy-

tvorený nový program, ktorým je možné tvoriť ohodnotenia ako v jednoduchom grafickom editore. Možnosti takéhoto maľovania pre používateľa sú:

- kreslenie "perom" pomocou myši
- nastavenie hrúbky pera
- tvorba úsečiek o hrúbke pera
- vyplnenie oblasti
- mazanie vyššie uvedenými možnosťami
- undo

Po uložení program pre obrázok vytvorí prislúchajúci výstup v ďalšom súbore s názvom obrázka a koncoukou .trn . V rámci zložky s obrázkami je možno sa medzi obrázkami ľahko presúvat. Program podporuje jpg, bmp a png formáty.



Obrázok 6.5: Tvorba trénovacieho výstupu. Cesta je prekrytá červene.

6.5 Lokalizácia

6.5.1 Predošlý stav

Na výpočet polohy na mape sa používal triviálny prístup, ktorý používal vzťahy a algoritmy pre euklidovský priestor a nezohľadňoval tak rozdiely medzi stupňami zemepisnej šírky a dĺžky a zakrivenie Zeme. V strednej európe predstavuje rozdiel jedného stupňu šírky približne 111.2km a dĺžky približne $73,6$km. Chyba pri výpočtoch trasy a polohy mohla dosahovať až 35% ($\frac{73}{111} \cong 0.65$) ak sa porovnávali vzdialenost na seba kolmé a rovnobežné so šírkou a dĺžkou .

6.5.2 Nové riešenie

Poloha robota je vypočítaná z výstupu GPS vzťahmi, ktoré uvádzame v kapitole 4 ako najbližší bod na mape. Tieto vzťahy sú obsiahnuté v triede LocalizationAndPlanning, kde sa zdieľajú aj pre potreby plánovania.

6.5.3 Testovanie

Testovanie lokalizácie robota chodením po parku je časovo aj fyzicky náročné, preto sme implementovali program locplantest, ktorý umožňuje testovať schopnosti lokalizácie aj bez fyzického pohybu robota. Program je nadstavbou nad robotovým GUI a umožnuje užívateľovi zadať hodnotu GPS výstupu klikaním, alebo spojitým pohybom kurzorom myši priamo v robotovej mape. Takto môžeme v rámci lokalizácie simulovať zmenu GPS signálu a sledovať výsledky plánovania a lokalizácie. Tiež je možné skúmať reakciu na stavy, keď je GPS signál nepresný alebo konštantne posunutý jedným smerom na rôznych mapách.

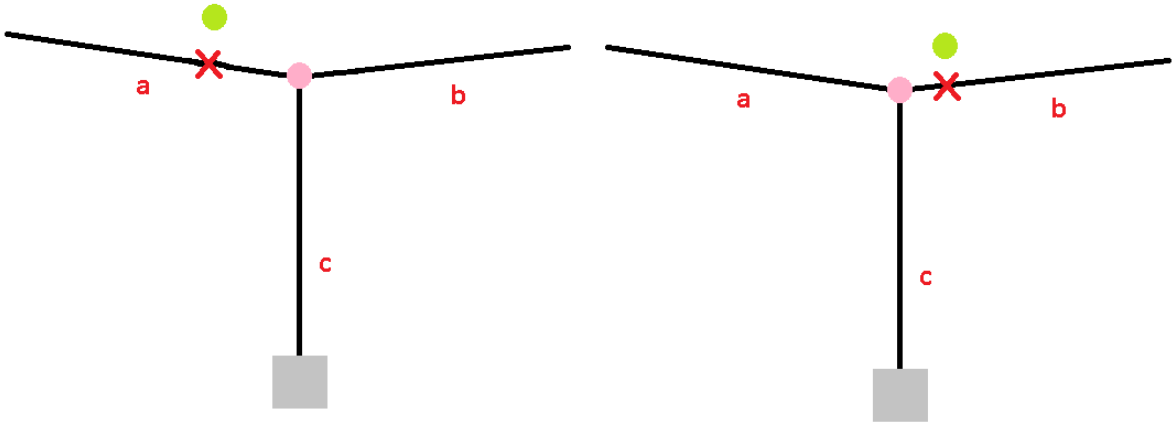
6.6 Plánovanie

6.6.1 Predošlý stav

Pôvodne sa medzicieľ volil ako koncový bod segmentu trasy, na ktorom sa robot nachádza. Tento prístup a nedostatočne presná lokalizácia spôsobovali ťažkosti pri prechode zákrutami trasy v križovatkách. Jednoduché zákruty robot prešiel bezproblémovo, no najmä pri nepresnom GPS signále mal robot problém zaregistrovať zmenu segmentu, čo viedlo k zlému

výberu odbočky na križovatke alebo aj k opakovanému prechádzaniu križovatky hore-dole. GPS údaj je nespojitý a preto je vysoko pravdepodobné, že sa preskočí jediný spoločný bod segmentov.

Tento problém bol riešený tak, že sa segment menil aj v prípade priblíženia k začiatku segmentu no nastavenie tejto vzdialenosti bolo nespoľahlivé a nekonzistentné.



Obrázok 6.6: Príklad s T križovatkou. Chyba GPS alebo mapy spôsobí, že je meranie mimo cesty. Lokalizácia sa na cestu *c* vedúcu k cieľu (šedý štvorec) nikdy nepremietne, lebo vždy je jedna z ciest *a* alebo *b* bližšie a robot bude chodiť zľava doprava s medzicieľom v začiatku cesty *c* (ružový kruh). Takáto situácia nastáva v praxi, keď kvôli počasiu alebo stromom je výstup GPS v jednej oblasti konštantne posunutý jedným smerom.

6.6.2 Popis nového algoritmu

Pre zistenú polohu na mape sa vypočíta najlepšia trasa k cieľu, potom vypočítame medzicieľ na trase v okolí. Takto vybraný medzicieľ už priamo nezávisí na segmente, na ktorom bola určená poloha. Pokiaľ je robot pred zákrutou, je naviac vedený k zatáčaniu smerom k odbočke, čo znižuje šancu minutia zákruty. Trieda LocalizationAndPlanning spája výstup kompasu s vypočítanou polohou na určenie smeru k medzicieľu.

6.6.3 Problémy

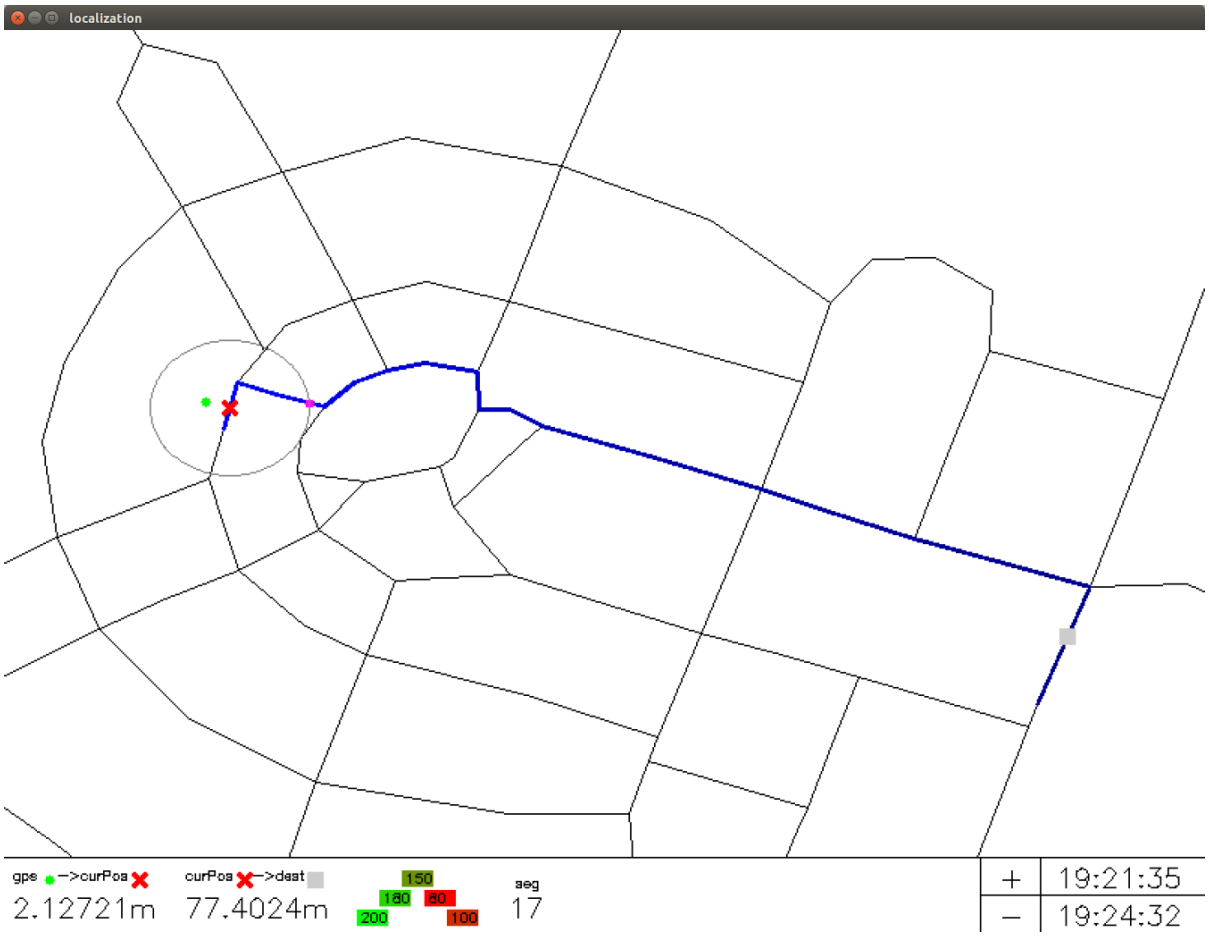
Komplexnejšie riešenie plánovania prináša potrebu správneho nastavenia, konkrétne vybratie okruhu správnej veľkosti. Ďalším problémom je, že skoršie zatáčanie dostáva robota do situácií, keď smeruje von z cesty a teda sa musí spoliehať na to, že nezlyhá detekcia cesty. V tomto prípade je však rozpoznávanie náročnejšie, keďže tráva pri zákrute býva často vy-

šliapaná ľudmi a robot sa môže jednoduchšie pomýliť. Prínos správnejšieho zatáčania však prevažuje zvýšenie rizika, nakoľko je nami implementované rozpoznávanie schopné vysporiadať sa s týmito prípadmi.

6.7 GUI

GUI programu bolo tvorené jednoduchou mapou, oknom s výstupom kamery a prislúchajúcim ohodnotením z neurónovej siete. Štartovací čas bolo treba zadávať ručne cez konzolu, čo bolo nepohodlné.

Preto sme GUI obohatili o tlačidlá pre nastavenie štartovacieho času a v okne mapy sa zobrazujú informácie o stave lokalizácie a plánovania. Mapa je ďalej obohatená výraznejšími a čitateľnejšími ikonami, ktoré uľahčujú proces testovania.



Obrázok 6.7: Nové GUI.

Vypočítaná poloha robota je na mape ukázaná červeným krížikom, výstup GPS označuje zelený krúžok. Cieľ v pravej časti obrázka je šedý štvorec. Modro vyznačená cesta

je plánovaná najkratšia trasa. Farba tejto trasy je naviac gradientovaná od svetlomodrej po tmavomodrú vyjadrujúc smer. Elipsa okolo terajšej pozície je okolie, na ktorom sa vyberá medzicieľ označený ako ružový kruh.

Ďalej je v spodnej lište umiestnená informácia o odhadovanej presnosti GPS signálu (vzdialenosť výpočítaneho bodu mapy k surovému GPS výstupu) a vzdušná vzdialenosť k cieľu. Obe vzdialenosti obsahujú aj popisok, aby bol po krátkom skúmaní jasný význam ikoniek. Na lepšie debugovanie je pri týchto vzdialenostiach pyramída vyjadrujúca výstupy ultrazvukov, pričom sa pozadie jednotlivých hodnôt sfarbujú dočervena s približujúcou sa prekážkou. Nasleduje informácia o počte zostávajúcich križovatiek a aktálny a štartovací čas s možnosťou nastavenia.

Kapitola 7

Výsledky

7.1 Porovnanie modelov neurónovej siete

7.1.1 Vstupné dáta

Pre trénovanie sietí bolo vybraných 100 rôznych obrázkov z parkov. Vstupom siete sú 5x5 štvorce pixelov získané rozstrihaním vstupných obrázkov a prípadne prídavné vstupy popísané v kapitole 3. Porovnávali sme úspešnosti sietí s rôznymi prídavnými vstupmi a rôznym počtom neurónov a skrytých vrstiev.

Po rozdelení obrázkov na vstupy získame približne 385 000 vstupov z čoho je 40 % ohodnotených ako cesta, teda triviálnym prístupom by sme vedeli získať úspešnosť 60 % klasifikovaním každého vstupu negatívne.

Dáta obsahujú dobrú vzorku situácii, v ktorých sa robot môže ocitnúť, teda aj prípady bez cesty alebo s cestou mimo stredu záberu.

7.1.2 Trénovanie modelov

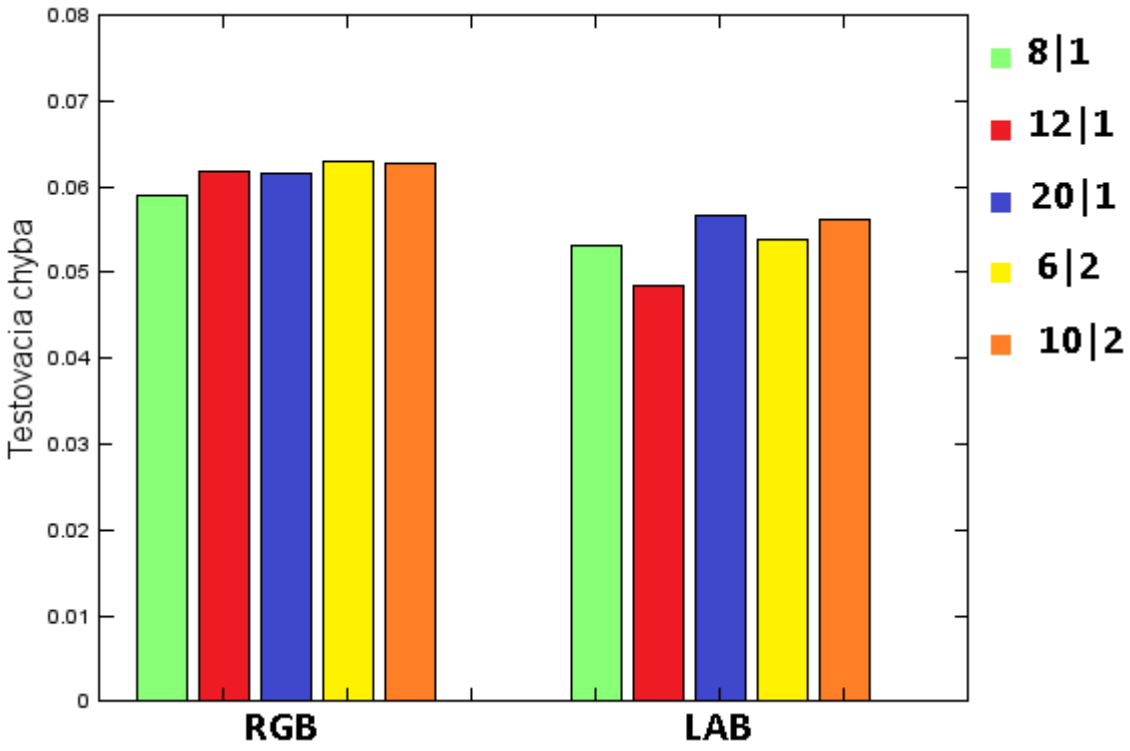
Vstupné dáta boli pri každom trénovaní náhodne rozdelené na trénovaciu a testovaciu množinu v pomere 90:10. Pre každý model sme natrénovali tri jeho inštancie a ako výsledky uvádzame priemer MSE každého modelu na testovacej množine.

7.1.3 Porovnanie RGB a LAB

Prekonvertovaním obrázku do LAB reprezentácie farby sa nám podarilo konzistentne znížiť chybu siete vo všetkých testovaných konfiguráciach, ktoré sme vybrali na základe predošlých

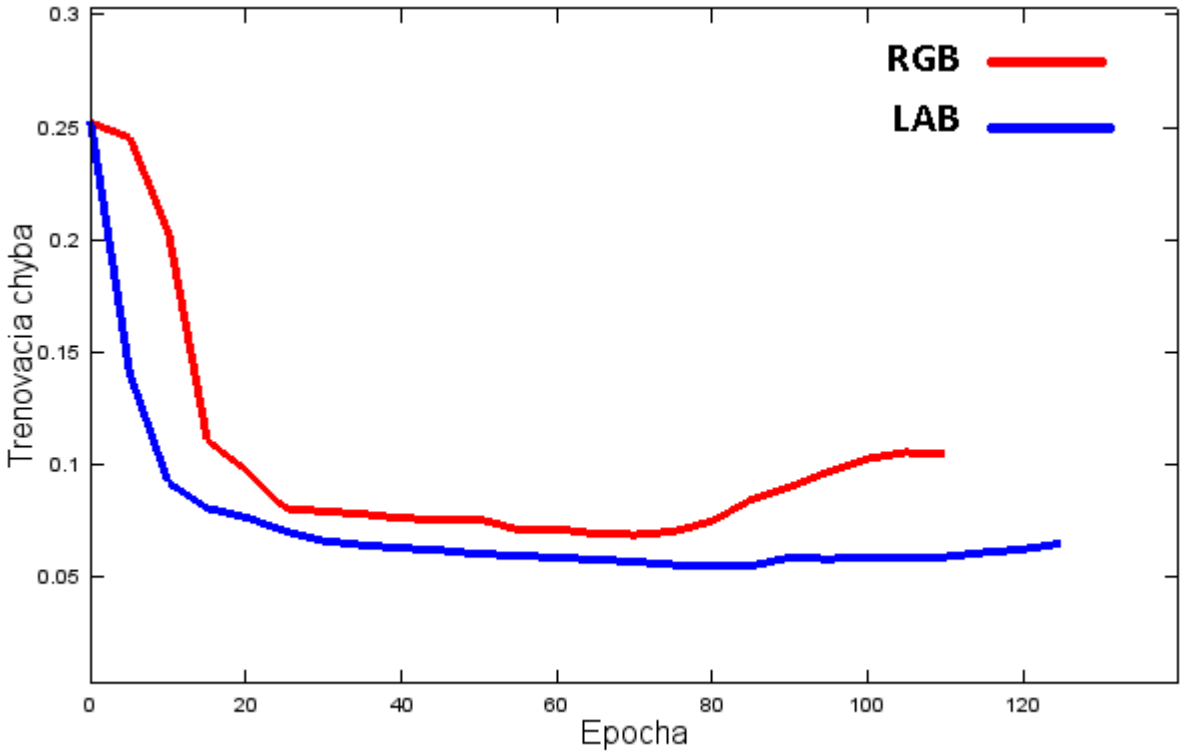
skúseností a výsledkov prác [9][10]. Vybraté konfigurácie sú:

- 1 skrytá vrstva s 8 neurónmi (zelená na obrázkoch)
- 1 skrytá vrstva s 12 neurónmi (červená na obrázkoch)
- 1 skrytá vrstva s 20 neurónmi (modrá na obrázkoch)
- 2 skryté vrstvy s 6 neurónmi teda spolu 12 skrytých neurónov (žltá na obrázkoch)
- 2 skryté vrstvy s 10 neurónmi teda spolu 20 skrytých neurónov (oranžová na obrázkoch)



Obrázok 7.1: Porovnanie priemerných testovacích chýb sietí s RGB a LAB reprezentáciou farby vstupu.

Chyba na testovacích dátach s RGB reprezentáciou sa pohybovala medzi 0.0590524 a 0.0629318. Pre LAB bola chyba medzi 0.0485086 a 0.056628 a teda aj najhorší výsledok s LAB prekonal najlepší model s RGB.
Keď porovnáme najlepšie výsledky 0.0485086 pre LAB a 0.0590524 pre RGB, získavame zníženie chyby o vyše 17 %(17,855).

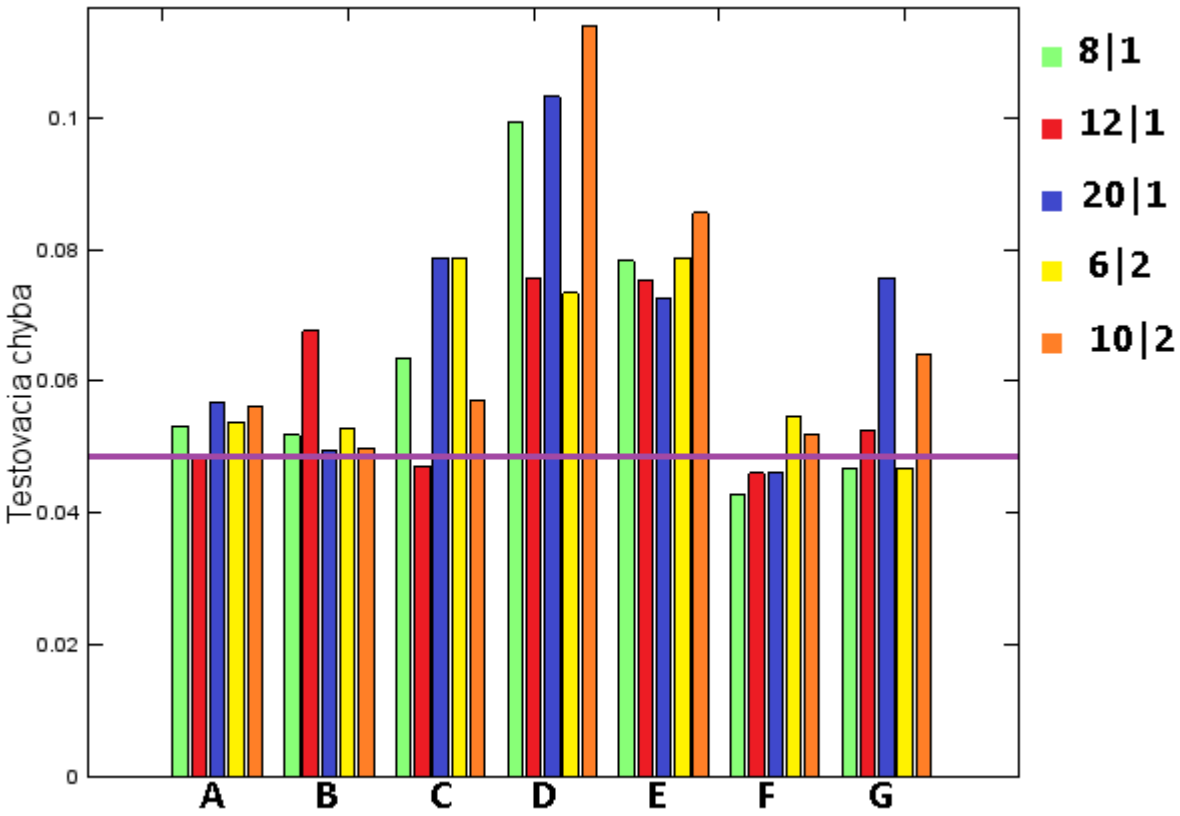


Obrázok 7.2: Vývoj trénovacej chyby počas trénovania najúspešnejšieho modelu s RGB a LAB vstupom.

Ako možno vidieť z grafu vyššie, konverzia do LAB neovplyvnila rýchlosť učenia siete negatívne, naopak algoritmus konverguje rýchlejšie. Rovnaký trend vykazovali aj ostatné modely a teda zmena formátu farby napomohla aj v učiacom procese.
Konverzia z RGB do LAB sa ukázala užitočná v každom smere a v ďalšom porovnaní už RGB testovať nebudeme.

7.1.4 Porovnanie prídavných vstupov

Porovnali sme 6 rôznych konfigurácii prídavných vstupov so sieťou bez nich. Očakávaním bolo zníženie chyby alebo aspoň zachovanie úspešnosti keďže sieť môže prídavný vstup odignorovať.



Obrázok 7.3: Porovnanie priemerných testovacích chýb sietí s rôznymi prídavnými vstupmi. Fialová čiara označuje najnižšiu priemernú chybu dosiahnutú bez prídavného vstupu.

Testované konfigurácie prídavných vstupov:

- **A-** bez prídavného vstupu
- **B-** oblasť s vysokou pravdepodobnosťou cesty - jedna oblasť
- **C-** oblasť s vysokou pravdepodobnosťou cesty - medián ôsmich oblastí
- **D-** K-means kompresia
- **E-** K-means kompresia v kombinácii s oblasťou s vysokou pravdepodobnosťou cesty - medián ôsmich oblastí
- **F-** histogram
- **G-** histogram v kombinácii s oblasťou s vysokou pravdepodobnosťou cesty - medián ôsmich oblastí

Ukázalo sa, že použitie K-means kompresie ako prídavného vstupu neurónovej siete (konfigurácie D, E) nepomohlo v správnom rozpoznaní cesty. Aj v najlepšej konfigurácii prevyšovala chyba s K-means kompresiou (0.0724527) najmenej úspešnú sieť bez prídavného vstupu (0.056628).

Pridanie oblasti s vysokou pravdepodobnosťou cesty ako prídavného vstupu (konfigurácie B,C) nevykazuje výrazné zlepšenie a chyby sa pohybovali v okolí chýb referenčných modelov bez prídavného vstupu. Použitie mediánu z ôsmich oblastí dosiahlo nižšiu chybu (0.0470711) ako naša najlepšia referenčná konfigurácia (0.0485086). Tento rozdiel je však malý a v ostatných prípadoch je chyba vyššia.

Konfigurácie F a G s histogramom obrazu boli úspešnejšie ako referenčné konfigurácie. V prípade G je zlepšenie nevýrazné čo je pravdepodobne spôsobené prídavnou oblasťou s vysokou pravdepodobnosťou cesty ako v prípadoch B a C kde chyba poskočila vyššie. V prípade F s pridaním iba histogramu je viditeľne nižšia chyba pri troch konfiguráciach (0.0427045, 0.0459659, 0.046113) oproti najlepšiemu referenčnému výsledku (0485086) a najmenej úspešná konfigurácia dosahovala menšiu chybu (0.0547351) ako najmenej úspešná referenčná konfigurácia (0.056628).

Analýza výsledkov

Zvýšenie chyby po pridaní prídavného vstupu môže byť spôsobený tým, že sieť môže ľahšie skonvergovať k lokálnemu minimu vďaka tomu, že prídavný vstup je vo vzťahu s riešením. Sieť sa potom až príliš spolieha na prídavný vstup, ktorý ale nie je dostatočný na určenie správneho ohodnotenia.

Úspech histogramu je v súznení s výsledkom práce Robotour[1], kde bol použítý k RGB vstupu.

Oblasť s vysokou pravdepodobnosťou narozdiel od výsledku z práce Robotour nepriniesol výrazné zlepšenie. Po prezretí vstupných dát bolo ale možné si všimnúť malú varianciu typov chodníka a teda sa tento prídavný vstup nevyužil na maximum.

7.2 Správanie

Robot bol viackrát testovaný v Botanickej záhrade UK v Bratislave. Robot dokáže v terajšom stave prekonávať jednoduchšie zákruty a ísť po rovnej ceste s iba drobnými problémami,

ktoré by nespôsobili vylúčenie (ide nie najkratšou cestou alebo občas jeným kolieskom vyjde na trávu). Implementáciou otočenia sa po zídení z trasy sa zamedzilo občasnému odbočovaniu mimo trasu a pokračovaním keď robot šiel v protismere.
Ťažšie zákruty zvláda lepšie ako na začiatku tejto práce keď často ignoroval odbočenie alebo do otočky vošiel iba vďaka náhode, no stáva sa mu, že kvôli prílišne agresívnemu zatáčaniu vyjde mimo cestu ale často nie všetkými kolieskami, čo je ešte postačujúce pre súťaž.
Za priaznivých svetelných podmienok je schopnosť rozpoznať cestu dobrá, no pri zlom svetle je robot už menej úspešný.
Kvôli chybe GPS signálu (napríklad pri zatiahnutej oblohe alebo medzi stromami) sa poloha môže posunúť na iný segment a robot potom chce ísť podľa toho iným smerom. Ak je ale nepresnosť dočasná, opraví si kurz späť.

7.3 Analýza nedostatkov

Pozorovaním výstupu kamery sme zistili, že je obraz značne skreslený pri zlom osvetlení. Tento fakt podozrievame ako hlavného strojcu chýb rozpoznania cesty. Táto nedokonalosť je najnebezpečnejšia vzhľadom na úspech v súťaži nakoľko pri chybách plánovania a lokalizácie robot nie je vylúčený a má šancu sa zotaviť. Napraviť by to mohlo použitie lepšej kamery alebo pridanie laserového senzoru.
Správanie na križovatkách a výber medzicieľa stále potrebuje doladiť, lebo sa prílišne spolieha na rozpoznávanie cesty, ktoré mu nedovolí z cesty vypadnúť. Vytvara sa tak zbytočne veľa príležitostí pre chybu.
Na užších cestách, ktoré majú na okrajoch prekážku, ako napríklad kríky, má robot tendenciu sa zdržať na pomerne dlhý čas. Zapríčiňuje to detekcia prekážok keď robot nejde pekne stredom cesty. Ako objaví prekážku zastaví, chvíľu počká cúvne a pokračuje. Toto sa opakuje mnoho krát až kým sa mu nepodarí z obkolesenej časti cesty odísť. Na vyriešenie tohoto proble´mu bude treba navrhnúť sofistikovanejší prístup k prekážkam.
Stavba robota je pevná a robustná no robot nie je odolný voči vode, čo si žiada spoľahlivé dostavanie, ktoré neobmedzí jeho senzory lebo Robotour súťaž sa koná aj počas dažďa.

7.4 Robotour 2014

Robot sa zúčastnil súťaže Robotour 2014, ktorá sa konala v Borskom parku v Plzni. Podmienky v parku boli náročné, nakoľko už začalo opadávať lístie zo stromov a sťažovalo rozpoznávanie cesty. Súťaž sa skladala zo štyroch súťažných kôl a počas druhého a tretieho kola pršalo.

7.4.1 Stav na Robotour 2014

Počas súťaže bola implementovaná neurónová sieť s LAB predspracovaním farieb a skoršie verzie subrutín pre prípady, keď robot nevie nájsť cestu, stojí pred prekážkou alebo sa vydal zlým smerom.

Chyby

Počas súťaze robot robil chyby, ktoré sme sa v tejto práci riešili, konkrétne mal problém v krížovej križovatke, kde naplánovaná trasa bola pokrytá lístím a vybral si bočnú cestu čo malo za následok opakované otáčanie a volenie druhej bočnej cesty.
Ďalej sme neboli dostatočne pripravení na dážď a robota sme pred ním museli chrániť provizórnym pripevnením dáždnikov, ktoré blokovali všetky ultrazvukové senzory okrem predného a museli byť odpojené. Týmto neutrpela schopnosť robota zastať pred prekážkou, ale mohol sa zachytiť bokom o nízke predmety napríklad lavičku.
Počas prvého kola bolo slnečné počasie a kamera robota sa so zmenami jasu pri prechode z/do tieňa nevysporiadavala veľmi dobre a klasifikácia siete, natrénovanej z fotiek iného zariadenia, ktoré mali správne vyvážený jas, bola neuspokojivá. Robot mal preto tendenciu váhať a obchádzať výrazne svetlé miesta.

7.4.2 Výsledok

Robot získal $46+40+64+80=230$ bodov vďaka čomu sa umiestnil na prvom mieste. Za ním zdieľali druhé miesto tímy Radioklub Písek $0+0+2+179=181$ a AmBot $0+16+127+37=180$. Za zmienku stojí, že sa v žiadnom kole žiadnemu zo zúčastnených tímov nepodarilo úspešne doraziť do cieľa.

Analýza

Robot sa ukázal robustným a spoľahlivým avšak jeho maximálny výsledok (80) bodov bol výrazne horší oproti celkovo najlepšiemu kolu tímu Radioklub Písek (179). Výhru nad tímom Radioklub Písek prisudzujeme k ich technickým problémom s pohonom počas prvých troch kôl, nakoľko mali lepšie vybavenie obsahujúce laserový senzor a v poslednom kole ukázali funkčnosť ich pohybovej logiky. Preto bolo a je dôležité zvýšiť výkonnosť robota a zachovať jeho spoľahlivosť.

Záver

Cieľom tejto práce bolo zvýšiť úspešnosť robota Smelý Zajko v rámci súťaže Robotour. Pozorovaním robota sme identifikovali slabé miesta pôvodnej implementácie a následne sme sa ich pokúsili zlepšiť.

Existujúci systém rozpoznávania ciest pomocou neurónovej siete sme vylepšili a vyskúšali sme efektivitu pôvodných prídavných vstupov, ktoré sme aj rozšírili. Implementácia teraz poskytuje jednoduchý spôsob tvorby prídavných vstupov, ktorý je možno použiť v ďalšej práci.

Lokalizácia robota bola výrazne spresnená použitím vhodnejšieho prístupu a plánovanie sme zmenili na základe skúseností z testovacích behov. Vďaka tomuto je robot úspešnejší v navigácii v parkoch pomocou mapy, kompasu a GPS signálu, no naďalej robí značné množstvo chýb, čo vytvára priestor ďalšieho zlepšovania.

Spoľahlivosť bola zvýšená opravami a zlepšením základného riadiaceho systému, čím sa odstránilo nekontrolované správanie a chod robota sa stal plynulejším.

Námetom na ďalšiu prácu môže byť navrhnutie komplexnejšieho plánovania v oblasti križovatiek, ktoré by počítalo s nepresnosťou GPS signálu a bolo schopné robota udržať bližšie stredu cesty. Ďalším prínosom by bolo vytvorenie modulu na obchádzanie prekážok, ktoré by bolo dobre využiteľne aj v aplikáciach mimo Robotour súťaže. Keďže je robot stavaný na prácu v otvorených priestoroch robotovi ešte chýba ochrana pred dažďom, ktorá by ale nebránila vo výhľade senzorom robota a výrazne ho neobmedzovala.

Literatúra

[1] Miroslav Nadhajský, Robotour, diplomová práca, FMFI UK, 2011.

[2] Haykin, S., Neural Networks: A Comprehensive Foundation. Prentice Hall. 1999. ISBN 0-13-273350-1

[3] Martin Riedmiller and Heinrich Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. 1993.

[4] Christian Igel and Michael Hüsken, Improving the Rprop Learning Algorithm. 2000.

[5] Matin Dlouhý, Zbyněk Winkler, 2014. online: http://robotika.cz/competitions/robotour/2013/Robotour-pravidla.pdf

[6] David Gustafík, Aplikácie mobilného robota Sbot2. Institute of Robotics and Cybernetics (FEEIT). 2011.

[7] C. Siagian and C.K. Chang and L. Itti, Mobile Robot Navigation System in Outdoor Pedestrian Environment Using Vision-Based Road Recognition, 2013.

[8] Yuanyuan Jia, Zhongshi He, Huazheng Zhu, A Hierarchical Segmentation Approach towards Roads and Slopes for Collapse Recognition, International Journal of Signal Processing, Image Processing and Pattern Recognition, 2013.

[9] Patrick Yuri Shinzato, V Grassi, Fernando Santos Osório, Denis F Wolf, Fast visual road recognition and horizon detection using multiple artificial neural networks, 2012.

[10] Patrick Y. Shinzato, Leandro C. Fernandes, Fernando S. Osorio, Denis F. Wolf, Path Recognition for Outdoor Navigation Using Artificial Neural Networks: Case Study. 2010.

[11] A. Miranda Neto, Leticia Rittner, A simple and efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision, Robotics Symposium, 2006.

[12] János Schanda, Colorimetry p. 61. Wiley-Interscience. 2007. ISBN 978-0-470-04904-4

[13] S. Nissen, Implementation of a Fast Artificial Neural Network Library (fann). Department of Computer Science University of Copenhagen (DIKU). 2003.

[14] Bradski, G., The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2000.

Príloha

K textu je priložené CD obsahujúce implementáciu softwareu robota (sz.zip), program pre trénovanie siete (train.zip) aj s ukážkovými trénovacími vstupmi a program pre tvorbu trénovacích vstupov (traincrt.zip).