VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

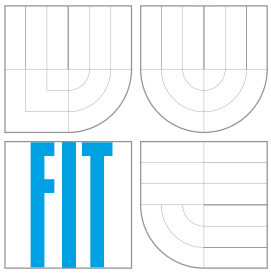
DETEKCE CESTY V OBRAZU Z KAMERY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN ŠEDO
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE CESTY V OBRAZU Z KAMERY

ROAD DETECTION IN THE CAMERA IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN ŠEDO

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2015

Abstrakt

Bakalářská práce se zabývá hledáním cesty v obraze z kamery. Obsahuje přehled v současnosti používaných metod v této oblasti a jejich nejčastější využití, stručný úvod do zpracování obrazu a k aplikačnímu rámci ROS. Dále obsahuje návrh a detaily implementace aplikace, která s využitím knihoven ROS a OpenCV provádí analýzu obrazu a jeho segmentaci za účelem oddělení cesty od okolí pomocí metody globální tresholding. Aplikace je otestována na datech nahraných pro účely soutěže Robotour 2013 a na sérii vlastních dat. Na závěr obsahuje vyhodnocení výsledků a diskuzi na téma možné pokračovaní této práce.

Abstract

Bachelor's thesis deals with road detection in camera image. It contains overview of used methods in this area and their most frequented usage, brief introduction to image processing and to ROS framework. Then it gives proposal and details of implementation of application, which analyzes the image and performs segmentation in order to separate the road and it's surroundings by global thresholding method with use of ROS and OpenCV libraries. Application is tested on the data gathered for purposes of Robotour 2013 contest and on series of own recorded data. Finally it contains evaluation of results and discussion about future work on this project.

Klíčová slova

zpracování obrazu, detekce cesty, segmentace, ROS

Keywords

image processing, road detection, segmentation, ROS

Citace

Jan Šedo: Detekce cesty v obrazu z kamery, bakalářská práce, Brno, FIT VUT v Brně, 2015

Detekce cesty v obrazu z kamery

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D.

.....
Jan Šedo
19. května 2015

Poděkování

Chtěl bych svému vedoucímu panu Rozmanovi za shovívavost za dobré vedení a panu Maternovi za ochotu a cenné rady.

Obsah

Kapitola 1

Úvod

Hlavními vodítky pro lidské vnímání vozovky nebo cesty obecně jsou její ohraničení, textura a barva povrchu. Lze očekávat, že v budoucnu budou částečně nebo úplně samostatná vozidla nebo roboti s člověkem tyto cesty sdílet, a tím pádem budou s velkou pravděpodobností k navigaci po nich spoléhat na stejná vodítka jako člověk. V principu je sice teoreticky možné vybudovat dvojí infrastrukturu tak, aby lidé spoléhali na tradiční značení, zatímco by autonomní systémy spoléhaly na vlastní komunikační systém vozidlo-vozovka, avšak tato myšlenka naráží na velké množství komplikací. Dvojí systém by vyžadoval významnou investici do výzkumu, zbudování a údržby a přinášel by s sebou množství rizik, jako například technické závady na systému, chyby lidského faktoru při implementaci nebo dokonce úmyslnou sabotáž. Vnímání a detekce jízdních pruhů a cest pomocí tradičních vodítek se v tuto chvíli jeví jako nejefektivnější způsob a zůstává nejpravděpodobnější cestou vývoje systémů pro autonomní řízení.

Systémy, které jsou schopny detekovat cestu, mají pro člověka vícero využití. U moderních vozidel se zvětšuje množství bezpečnostních prvků. Detekce jízdních pruhů může řidiče například signálem upozornit, že neúmyslně opouští svůj jízdní pruh a tím zabránit nehodě. V kombinaci s bezpečím lze mluvit i o pohodlí, kdy systém založený na vnímání cesty může umožnit autu po cestě samostatně navigovat a tím usnadnit řidiči práci. Autonomní roboty lze pak nasadit v prostředí, které neumožňuje nasazení lidské posádky k jeho řízení. Jejich využití se nabízí například v armádě pro plnění průzkumných nebo bojových misí.

Cílem této práce je prozkoumat používané algoritmy a způsoby detekce cesty, zvolit vhodný algoritmus a na jeho základě pak implementovat aplikaci, která bude schopna uživateli v reálném čase v obrazu pořízeném kamerou zobrazovat detekovanou cestu. Díky tomu, že bude implementována s pomocí knihovny ROS, bude moci poskytovat relevantní data použitelná pro možnou další navigaci po nalezené cestě. Aplikace bude otestována na datech nahraných pro účely soutěže Robotour 2013. Na základě těchto testů pak bude aplikace vyhodnocena a srovnána se současným detektorem cesty používaným na robotu Toad.

Kapitola 2 obsahuje vysvětlení důležitých pojmů ohledně zpracování obrazu, souhrn současného výzkumu na téma detekce cesty a představení ROS (Robotického operačního systému). V kapitole 3 bude proveden návrh cílové aplikace. Kapitola 4 objasňuje realizaci aplikace a obsahuje konkrétní implementační detaily, výsledky testování na připravených datech a vyhodnocení použitelnosti, vhodnosti a efektivitě řešení, návrhy na vylepšení a případné další směry rozšíření této práce.

Kapitola 2

Teorie

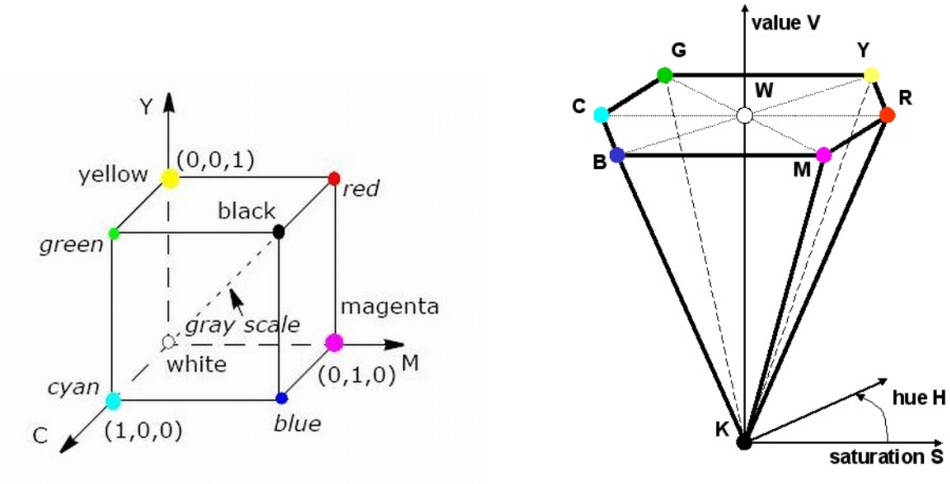
V této kapitole jsou rozebrány základní principy a důležité pojmy, od kterých se bude odvíjet celá práce. Následuje seznámení s procesem zpracování obrazu a přiblížení jednotlivých kroků. Tyto principy poslouží k lepšímu pochopení následujícího souhrnu současného výzkumu a postupů používaných v oblasti detekce cesty. Dále následuje seznámení s Robotickým operačním systémem, jež bude cílová aplikace využívat.

2.1 Zpracování obrazu

Zpracování obrazu je proces, při kterém se se získaným obrazem uloženým v digitální formě provádějí matematické operace za účelem vytvoření sady charakteristik obrazu nebo vytvoření pozměněného obrazu, buď ke zlepšení požitku pozorovatele nebo k pochopení jeho obsahu. Postup zpracování a rozpoznávání obrazu reálného světa lze obvykle rozložit do těchto základních kroků:

- snímání a digitalizace obrazu
- předzpracování obrazu
- segmentace obrazu
- porozumění obsahu obrazu

Snímání obrazu a jeho převod do digitální formy je prvním krokem na nejnižší úrovni zpracování obrazu. Přes objektiv kamery na senzor dopadá světlo, které je konvertováno na analogový signál, který je popsán funkcí f(x,y), kde x a y jsou souřadnice v obraze a funkční hodnota odpovídá například jasu daného bodu. Vstupní signál je vzorkován a kvantován pomocí A/D převodníku. Výsledkem je digitální obraz, který je v počítači reprezentován jako číselná matice. Jednotlivé prvky matice se nazývají **pixely** (z anglického picture element – obrazový element). Pixely mohou být uloženy v matici různými způsoby, které odpovídají barevnému prostoru obrazu. Je-li obraz černo-bílý, hodnota pixelu na bude číslo vyjadřující jasu daného pixelu, typicky v rozsahu 0-255. U barevného obrazu půjde o vektor čísel. Pokud je barevný prostor obrazu RGB, bude hodnota jednoho pixelu odpovídat vektoru třech čísel určujících poměr složek jednotlivých barev. Velmi často používaným modelem je také model HSV, kde hodnota H (Hue) určuje barevný tón, S (Saturation) určuje sytost barvy, a V (Value) jas bodu.



(a) Barevný model RGB.

(b) Barevný model HSV.

Obrázek 2.1: Barevné modely.

Předzpracování obrazu je operace, jejíž cílem může být potlačení zkreslení, odstranění šumu vzniklého při digitalizaci a přenosu obrazu nebo zdůraznění charakteristik pro další zpracování, například pro detekci hran[2]. Velice důležitou operací je filtrace, což je v podstatě úprava jasu jednotlivých bodů na základě hodnot pixelů v jeho okolí. Filtraci lze chápat jako diskrétní konvoluci, kdy konvoluční jádro je lokálním okolím daného pixelu.

V závislosti na hodnotách jádra pak lze obraz vyhlazovat nebo zaostřovat. Vyhlazování obrazu vede k potlačení vyšších frekvencí obrazové funkce a jejím důsledkem je potlačení náhodného šumu, ale zároveň i potlačení ostatních náhlých změn jasové funkce, tedy rozmazání hran. Při gradientních operacích naopak dochází ke zvýraznění hran, ale i náhodného šumu. Příkladem rozostřovacího filtru může být například filtr s Gaussovým rozložením[1] a příkladem detektoru hran Cannyho detektor[2].

Segmentace obrazu je nejdůležitějším krokem zpracování obrazu. Je to proces, při kterém probíhá dělení obrazu na části, které sdílejí stejné nebo podobné vlastnosti. Každému pixelu v obraze je pak přiřazen index vyjadřující konkrétní segment v obraze. Takovéto rozdělení pak umožňuje nalézt v obraze objekty nebo rozhraní mezi nimi a rozdělit tak obraz na smysluplné části připravené k další analýze.

Segmentace obrazu [8] $f(x, y)$ je jeho dělení na podobrazy $R_1, R_2..., R_n$ tak, že podobrazy splňují následující kritéria:

1. $\bigcap_{i=1}^{n} R_i = f(x, y)$
2. $R_i \cap R_j = \emptyset, i \neq j$
3. Každý podobraz splňuje nějaké tvrzení, popř. množinu tvrzení, např:
   - všechny pixely v podobraze mají stejnou úroveň šedi,

[1] http://en.wikipedia.org/wiki/Gaussian_blur
[2] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

– všechny pixely v podobraze se neliší v úrovni šedi více než o předepsanou hodnotu,
– standardní odchylka úrovni šedi všech pixelů poddobrazu je dostatečně malá, apod.

Způsoby segmentace

Segmentačních způsobů existuje celá řada. Základní rozdělení segmentačních metod [8] je následující:

- **metody vycházející z detekce hran** (angl. edge-based) jsou metody, které produkují množinu hran, které pak v obraze oddělují jednotlivé segmenty
- **metody orientované na regiony v obraze** (region-based) jsou metody, které spojováním pixelů podobných vlastností vytvoří rozhraní mezi jednotlivými segmenty
- **statistické metody** se zaměřují na statistickou analýzu obrazových dat, nejčastěji hodnot pixelů. Významným reprezentantem je např. prahování.
- **hybridní metody** nelze zařadit do jedné kategorie, protože obsahují prvky ostatních metod.
- **znalostní metody** (knowledge based) porovnávají nalezené objekty s transformacemi předloh jednotlivých segmentů.

2.2 Existující řešení

Detekce cesty využívající kameru je v současné době jedním z hlavních směrů výzkumu týkajícího se hledání cesty. Obraz a vizuální data jsou klíčová pro člověka, který po cestě naviguje. Značení a hranice cest jsou tedy udělány tak, aby je byl člověk-řidič schopen spatřit pokud možno za každých podmínek. Použití kamery k vyhledání klíčových vodítek sloužících k nalezení správné cesty a získání výpočetního ekvivalentu je tedy logickým krokem.

Velkou motivací pro využití kamery je také cena. Díky dostatečné vyspělosti masivně vyráběných uživatelských kamer jsou v současné době cenově velice efektivním řešením. Nicméně algoritmy na základě vidění stále postrádají klíčovou adaptabilitu v porovnání s chápáním lidského řidiče. Existuje i řada řešení, které se nespoléhají na pouze kameru. Mezi ně patří LIDAR, Stereo Imaging, kombinace GIS, GPS a IMU nebo využití radaru. Příklady jednotlivých senzorů jsou vyobrazeny na obrázku 2.2.

LIDAR (Light Detection And Ranging) je zařízení, které měří vzdálenost k objektu osvětlováním okolí nebo cíle pomocí laseru a následnou analýzou odraženého světla. Díky tomu dokáže konstruovat 3D strukturu okolí vozidla zpracováním získaného mračna bodů. Navíc většina LIDARů dokáže zjistit intenzitu odraženého paprsku a tím úplně nahradit kameru s výhodou, že jsou invariantní k přirozeným světlým podmínkám, díky tomu že jsou aktivním zdrojem světla. Jejich hlavní nevýhodou je cena, která těmto senzorům zabraňuje stát se hlavní modalitou pro detekci cesty v automobilovém průmyslu. Na trhu jsou dostupné verze LIDARů pro komerční použití, ale některé výzkumné instituty používají svoje vlastní.

Stereo imaging tedy použití dvou kamer k získání 3D informace, reprezentuje krok mezi jednou kamerou a 3D LIDARem. Stereo imaging je typicky značně levnější na implementaci než LIDAR a jako způsob praktičtější kvůli velikosti senzorů. Na druhou stranu stereo imaging obecně nemůže dosáhnout takové přesnosti a spolehlivosti jako LIDAR. Na rozdíl od LIDARu, úspěšné měření hloubky obrazu je závislé na textuře povrchu, což představuje velkou výzvu především na velkých jednolitých površích. Přesnost tohoto systému a efektivní vzdálenost na jakou pracuje je dána funkcí, která závisí na vzdálenosti mezi kamerami. Větší vzdálenost poskytne větší přesnost na větší vzdálenosti za cenu nižší spolehlivosti a větší spotřeby výpočetního výkonu. Obecně řečeno je na stereo imaging potřebný větší výpočetní výkon v porovnání se systémem používajícím LIDAR, navíc s větší chybovostí. I přes to může být stereo imaging použit pro stejné základní úkony jako LIDAR například pro detekci překážek, detekci obrubníků, vytváření 3D mapy okolí, odhad svažování cesty apod.

GIS GPS IMU a jejich kombinace se celosvětově používají již dlouhou dobu v komerčních navigačních systémech k navigaci řidičů a robotů. Jde o metody, kdy se využívají předem známá geografická data z map (GIS - Geographic information system), aktuální data o poloze získaná pomocí systému GPS (Global Positioning System) v kombinaci s informacemi z elektronické jednotky IMU (Inertial measurement unit), která pomocí senzorů jako akcelerometr, gyroskop a magnetometr dokáže zjistit aktuální směr, rychlost nebo nadmořskou výšku systému. Současné systémy GPS dosahují přesnosti 5-10m, což lze díky použití IMU vylepšit až na 1m [1]. Digitální mapy s vysokým rozlišením pořízené ze vzduchu jsou dostatečně kvalitní, že je vodorovné značení zřetelné, tím pádem použitelné pro navigaci v jízdních pruzích. Při takovéto nebo lepší přesnosti se dá dokonce ustoupit myšlence, že globální pozicování spolu s přesnou mapou může vést ke globálnímu autonomnímu řízení bez jakéhokoli palubního vnímání cesty a jízdních pruhů (detekce překážek je však pouze pomocí tohoto systému neřešitelná). Problém však nastává se spolehlivostí takovéhoto systému. Úspěšnost GPS spojení závisí na navázaní spojení s dostatkem satelitů a to se nemusí povést z více důvodů. Ztráty GPS spojení můžou být částečně tolerovány díky použití IMU a spolehlivost kombinace těchto dvou technologií je předmětem zkoumání. Další problém je pochybnost, zdali je pořízení velice přesných digitálních map a udržování jejich aktuality reálné. Použití výhradně těchto senzorů tedy není zcela vylučitelné, ale zatím primárně slouží jako doplňkové informace pro komplexnější systémy.

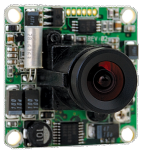
RADAR (Radio Detection And Ranging), tedy metoda zaměření objektů pomocí krátkých elektromagnetických vln, postrádá rozlišovací schopnost pozorování značení cest a nebo drobných 3D struktur. Relevance radarů v této oblasti je následující

- zjistit překážky (ostatní vozidla), které zastiňují značení a hranice cesty
- rozlišit mezi regionem cesty a okolím na základě velké odlišnosti reflektivity

Obě tyto schopnosti jsou ovšem jen limitovanou podmnožinou toho, co umí LIDAR s rozdílnou cenou a technickými parametry.

Kombinace výše zmíněných senzorů má obrovskou výhodu, že jednotlivé systémy mohou suplovat navzájem svoje nedostatky a tím přispět k celkové vyšší spolehlivosti celého systému. Fúze dat může poskytnout odhad jistoty výsledků porovnáváním získaných dat. Další vhodná úvaha je, že některé tyto systémy mohou být namontovány na vozidle nebo

robotu jako součásti jiných systému, takže jejich využití se přímo nabízí. Nevýhodou jsou samozřejmě náklady ať už cena, vyšší režie (spotřeba výpočetního výkonu) nebo velká náročnost realizace systému.



(a) kamera

(b) lidar

(c) stereo kamery

(d) IMU

(e) GPS

(f) onboard radar

Obrázek 2.2: Ukázky senzorů

Využití detekce cesty

V současné době je problém detekce cesty v obrazu kamery předmětem výzkumu především pro pokročilé asistenční systémy pro řidiče automobilů [1]. Tyto systémy buď pouze varují řidiče nebo aktivně přebírají řízení a postupně se stávají standardním vybavením automobilů. Nejvíce se v posledních letech zaměřoval výzkum na LDW (Lane Departure Warning), což je systém signalizace pro řidiče, který varuje v případě předem neohlášeného opouštění jízdního pruhu. Dalšími zkoumanými systémy, na které se práce nejčastěji zaměřují jsou:

- **AAC** (Adaptive Cruise Control) je systém, který umožňuje následovat vozidlo před kamerou a udržovat bezpečnou vzdálenost
- **udržování jízdního pruhu** je systém, který bez signalizované změny udržuje směr v jízdním pruhu
- **asistent změny jízdního pruhu** na vyžádání sám dokáže změnit jízdní pruh

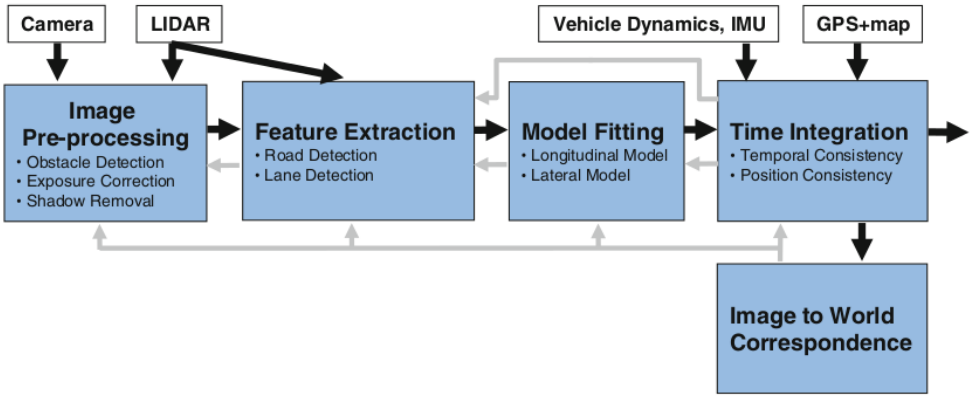
Naproti těmto systémům velkou porci výzkumu zaznamenaly velký pokrok systémy, které řídí vozidla naprosto samostatně. Jsou jimi:

- **plně automatické řízení na zpevněných cestách** tedy autonomní řízení na vybetonovaných nebo asfaltových cestách v osídlených oblastech
- **plně automatické řízení na venkovních cestách** tedy autonomní řízení na nezpevněných kamenitých nebo písčitých cestách ve venkovských oblastech

Dělení je dáno tím, že prostředí, ve kterém se bude autonomní vozidlo pohybovat, hraje velkou roli v návrhu celého systému. Velký pokrok zaznamenal vývoj autonomních systémů pro řízení vozidel v rámci soutěží DARPA Grand Challenge (2005) a Urban Challenge (2007). DARPA (Defence Advanced Research Projects Agency) je agentura amerického ministerstva obrany zodpovědná za vývoj nových technologií. Vypisuje krátkodobější projekty a financuje malé soukromé nebo univerzitní týmy, které svými příspěvky obohacují technologická řešení daných problémů. Výše zmíněné soutěže probíhaly tak, že soutěžícím byla krátce předem dodána trasa, kterou vozidla musí samostatně projet, pouze s daty které dále sama získají a s GPS souřadnicemi jednotlivých záchytných bodů. Po startu nebylo dovoleno navazovat s roboty a vozidly žádný kontakt. Vozidla musely umět operovat i v mlze a se zablokovanou GPS. Ačkoli se může zdát, že jednodušší zkoumané systémy jsou již zahrnuty v systémech komplexnějších (plně autonomní řízení), není tomu tak v případě palubní detekce cesty nebo jízdního pruhu pomocí kamery. Například v soutěžích DARPA totiž vozidla mají předem k dispozici velmi přesnou mapu terénu a předem dané GPS souřadnice. Typické vozidlo v DARPA soutěži navíc mělo více LIDARů, radarů, citlivou IMU (seznam senzorů níže) a velkou výpočetní sílu dosaženou několika počítači, takže se některá řešení v této soutěži na využití vidění pomocí kamery nespoléhaly vůbec a robustnost celého systému dosahovaly kombinací dat ze všech senzorů.

Generický systém detekce cesty

V rámci průzkumu současných řešení autoři článku [1] představují generický model systému detekce cesty sloužící k porovnání stávajících algoritmů. Ačkoliv žádný ze současných systému v literatuře nemá všechny bloky, protože se jejich implementace většinou značně liší, téměř všechny práce mohou být na tento model mapovány s tím, že čím robustnější systém je, tím víc z těchto bloků obsahuje. Model je znázorněn na obrázku 2.3.



Obrázek 2.3: Generický model systému detekce cesty [3]

Image preprocessing (předzpracování obrazu) je prvním blokem celého systému. Ještě před samotným hledáním cesty lze obraz upravit tak, aby v něm bylo nalezení cesty snazší. Cílem tohoto bloku je odstranění šumu, artefaktů a částí obrazu, které jsou pro nalezení cesty irelevantní. Tento blok se dá rozdělit na dvě rodiny úkolů:

[3] Obrázek převzat z [1]

- řešení problémů spojených s osvětlením obrazu a vylepšení kvality obrazu
- odřezání částí obrazu, které lze označit za nepotřebné nebo zavádějící

Osvětlení obrazu je velkým problémem pro detekci cesty, díky tomu, že se kamera musí potýkat s různými světelnými podmínkami - denní světlo, zataženo, šero nebo noční prostředí. Systémy povětšinou nevyvíjejí vlastní kontrolu dynamického rozsahu kamery, ale spoléhají na funkce kamer jako například automatická clona nebo kontrola citlivosti čočky. Řešením tohoto problému taky může být zachycování více obrázků s různými expozicemi, jejich kombinací pak získání snímku s vysokým dynamickým rozsahem, které zachovávají vysokou kvalitu obrazu a dokáží se vyrovnat s náhlými prudkými změnami osvětlení například při průjezdu pod mostem.

Dalším problémem je tzv. "lens flare", což je tříštění světla v čočce kamery například při přímém pohledu do slunce. Tento problém lze například vyřešit pomocí využití geografických dat, které zajistí predikci pozice slunce vůči kameře a dokáží takto odražené světlo ignorovat.

Potíží pro kamery jsou stíny, které můžou vrhat okolní objekty na cesty a produkovat tak v obraze ostré hrany změn intenzity. Tento problém řeší systémy transformací do jiných barevných prostorů jako například HSL, LAB, YCbCr a další. Kombinací kanálů lze získat obrázky invariantní k osvětlení, kde osvětlené a zastíněné oblasti mají stejnou intenzitu. Tyto stíny mohou být též odstraněny až v bloku segmentace, kde jsou brány v úvahu jen hrany, které odpovídají předpokládanému tvaru cesty.

Co se týče odstranění irelevantních částí obrazu, chodce a překážky lze v této fázi detekovat například sledováním směru pohybu jednotlivých bodů ve scéně v porovnání s pohybem kamery. Pokud se body pohybují jinak než podle předpokladu, je možno je na základě tohoto zjištění z procesu vyhledávání cesty vyloučit. Velkou částí je pak selekce tzv. ROI (Region of interest), tedy části obrazu, která nás zajímá. V nejednodušší formě lze například použít předpoklad, že cesta bude ve spodní části obrazu a horní polovinu obrazu zcela ignorovat. ROI lze také odvodit pomocí výpočtu vztahu mezi 2D a 3D prostorem na základě vypočítaných hloubek v obraze.

(a) Efekt *lens flare*

(b) Obrázek v barevném prostoru HSV

Obrázek 2.4: Ukázky efektu lens flare a barevného prostoru HSV

Feature extraction neboli extrakce příznaků je blok, který identifikuje důležité rysy v obraze, které jsou nadále použity pro zasazení do předpokládaného modelu cesty. Toto

typicky zahrnuje statistiky barvy nebo textur umožňující segmentaci cesty v případě hledání cesty nebo nalezení vodorovného značení pro hledání jízdního pruhu. V tomto bloku se hledání cesty a jízdního pruhu velmi liší kvůli fyzické odlišnosti hledaných příznaků.

- **Hranice jízdního pruhu** jsou detekovány na základě vodorovného značení. Vodorovné značení může být detekováno dvěma způsoby – na základě svého tvaru nebo barvy. Nejjednodušším předpokladem je, že se vodorovné značení liší svým vzhledem od silnice. Díky tomu lze například po řádcích hledat prudké změny intenzity v kombinaci s předpokládanou šířkou vodorovného značení. Problém samozřejmě nastává s perspektivou v obraze, kdy jsou bližší značky širší než vzdálenější. Tento problém lze vzít v úvahu již předem a hledat ve vzdálenější části obrazu menší značky nebo perspektivu kompenzovat, což je možné pokud je systému známa translace z 2D do 3D souřadnic.
- **Hranice cesty** je obtížnější obecně klasifikovat, protože se velmi liší. Hranicí cesty může být obrubník, zábrana na dálnici, nebo pokud cesta hranice nemá, muže za hranici být považována pouze změna barvy nebo textury na rozhraní cesty. Kvůli velké rozmanitosti musí být zvoleno vyhledávání množiny příznaků na základě předpokládaného prostředí, ve kterém se bude cesta vyhledávat. Ve velkém množství prací je zvolen předpoklad, že mezi cestou a jejím okolím existuje rozdíl ve vyvýšení, což vede na průzkum 3D struktury cesty pomocí LIDARu nebo stereo kamer. Dalším přístupem je spolehnutí se na vzhled cesty například odlišností barvy umístěním počátečních semínek (seeds) doprostřed dolů do obrazu pro metodu rostoucích regionů. Při předpokladu rovné cesty je možno použitím perspektivy zjistit dominantní směřování textur v obraze k nalezení úběžníku – díky tomu může být cesta segmentována pomocí dvou nejdominantnějších paprsků směřujících do tohoto bodu. Možné je také použít apriori znalost barvy a textury povrchu cesty.

Model fitting znamená zasazení získané hypotézy o cestě do předpokládaného modelu. Cílem tohoto bloku je vytvořit reprezentaci cesty na vysoké úrovni, která bude dále použita pro rozhodovaní. Zašuměná detekce cesty z předchozího bloku je vylepšena odhadem hladkého modelu cesty a přidáním omezení na její šířku a zakřivení. Tato hypotéza je později ještě vylepšena v dalším bloku srovnáním s předchozími snímky. Cesta je modelována buď dvěma křivkami, které reprezentují její hranice nebo střední čarou s jejím okolím do stran. Modely cesty se dají rozdělit na:

- **parametrické**, kdy se jako hranice cesty používají parametrické křivky. především přímky a paraboly. Hlavní výhoda modelování tímto způsobem je jednoduchost a zachování realistických tvarů
- **polo-parametrické**, kdy se k aproximování hranic cesty používají především *spliny*, což má výhodu zejména při korekci, kdy lze snadno měnit křivku pomocí kontrolních bodů
- **neparametrické**, které mají požadavek pouze aby byly linie spojité, ne nutně diferencovatelné

Ve většině případů se musí modely všech typů potýkat se zašuměnými hranicemi ve formě chybějících dat nebo velkého množství bodů ležících mimo danou množinu. Pro vytvoření předpokládaného modelu se velmi často používá algoritmus RANSAC (Random

Sampling Consensus), který má schopnost detekovat body ležící mimo množinu a model tedy zasadit pouze do bodů, které množině náleží.

Time integration (blok časové integrace) slouží k vylepšení přesnosti detekcí a vyloučení špatných detekcí pomocí integrace znalostí získaných v předchozích snímcích. Pokud se například některá z detekcí v krátkém časovém úseku značně liší od těch předchozích, lze ji na základě pravděpodobnostní funkce z výpočtu vyloučit. Častým řešením je sledování modelu cesty v reálných koordinátech, typicky za použití transformace do inverzní perspektivy. Odhad dynamiky vozidla pomocí odometrie auta nebo kombinací dat z GPS a IMU pak slouží k předpovězení pozice dříve získané detekce v aktuálním snímku. Očekávaná pozice bodů je pak kombinována s detekcí bodů v aktuálním snímku za pomoci Kalmanova filtru nebo částicovým filtrem pro dosažení přesnějších výsledků. Slabinou je naivita transformace perspektivy, která předpokládá rovný povrch. Vibrace kamery při změně jízdního pruhu nebo najetí na nerovnosti dělají problémy metodám s předpokladem plynulosti (Kalmanův filtr).

Image to world correspondence je modul, který zajišťuje znalost geometrické transformace mezi 2D a 3D světem. Takováto znalost je užitečná pro všechny bloky celého systému. V bloku předzpracování obrazu umožňuje na základě polohy oříznutí horizontu. Dále tento modul umožňuje transformaci perspektivy vytvořením transformační matice mezi rovinou obrázku a rovinou povrchu země. Schopnost systému odstranit z obrazu perspektivu je výhodná pro bloky extrakce příznaků a zasazování modelu.

2.3 Robotický operační systém

Protože škála robotů je obrovská a každý se může lišit svým hardwarem, je znovupoužitelnost kódu netriviálním problémem. Navíc psaní komplexního softwaru je nad síly programátora jednotlivce, jelikož objem kódu je obrovský, protože musí obsahovat od ovladačů na úrovni hardwaru až po programy s vysokou abstrakcí na úrovni vnímání atd. Pro zjednodušení vznikla v průběhu let řada aplikačních rámců (frameworků), které programátorům značně usnadňují práci a zajišťují velice potřebnou schopnost rapidního prototypování. Jedním z takovýchto aplikačních rámců je Robotický operační systém (dále ROS) [6] sloužící k tvorbě softwaru pro širokou škálu různých robotů nebo jejich součástí. Je kolekcí nástrojů, knihoven a konvencí, které zjednodušují programátorovi tvorbu komplexního softwaru pro jím zvolenou robotickou platformu. Jeho výhodou je to, že je distribuován pod licencí BSD, takže je jeho využití zdarma, což mu přináší širokou podporu komunity a zvyšuje dostupnost existujících řešení.

Historie

Původně byl ROS navržen jako odpověď na specifické požadavky, které vznikly při vývoji servisních robotů jako součást projektu STAIR [5] v roce 2007 na Stanfordské univerzitě pod názvem *switchyard*. Poté technologický inkubátor zabývající se robotickým výzkumem Willow Garage, poskytl prostředky, které pomohly rozšířit a posílit koncept ROSu a vytvořit tak stabilní a otestované prostředí. Už od počátku je jedním ze základních cílů, aby byl ROS zdarma a open-source, proto je distribuován pod licencí BSD. Na vývoji se dodnes podílí řada vývojářů a odborníků z celého světa. V roce 2013 opatrovnictví nad ROS přešlo pod Open Source Robotics foundation. Aktuální verze ROSu se jmenuje Indigo, a byla vydána v roce 2014. V květnu roku 2015 je pak očekáváno vydání nové verze Jade.

V současné době ROS funguje pouze pod Unixovými platformami. Oficiální software pro ROS je testován primárně na Ubuntu a Mac OS. Podpora pro ostatní platformy je však možná díky příspěvkům komunity a to například pro systémy Fedora, Gentoo, Arch Linux a další. Podpora a přenos pod systém Windows, ač teoreticky možná, není v současné době připravena.

Architektura ROSu

ROS není samostatným operačním systémem, pouze poskytuje některé služby standardního operačního systému jako např. hardwarovou abstrakci, ovládání nízkoúrovňových hardwarových zařízení, implementaci často používaných funkcí, předávání zpráv mezi procesy, správu balíků.

ROS také poskytuje uživatelské knihovny, které umožňují programátorovi psát kód stavebních prvků v často používaných programovacích jazycích, přičemž každý z nich má své výhody. Hlavními podporovanými uživatelskými knihovnami jsou roscpp (C++), rospy (Python) a roslisp (Lisp). Existují i další experimentální knihovny, které umožňují pracovat s dalšími programovacími jazyky, dle různých požadavků na cílovou aplikaci (rosjava, roslua, roscs, rosruby a další).

Koncept ROSu [7] má tři úrovně: Úroveň souborového systému, Úroveň výpočetního grafu a Úroveň komunity.

Úroveň souborového systému pokrývá zejména takové zdroje ROSu, se kterými je možno setkat se na disku v uživatelském prostředí. Základní jednotkou pro organizování softwaru v ROSu je tzv. Package (balík). Tyto balíky mohou obsahovat zdrojové kódy, knihovny, konfigurační soubory a další. Balík je to nejatomičtější spravovatelná ROSovská jednotka. Na této úrovni se nacházejí i další věci jako metabalíky, definice zpráv a služeb (viz níže).

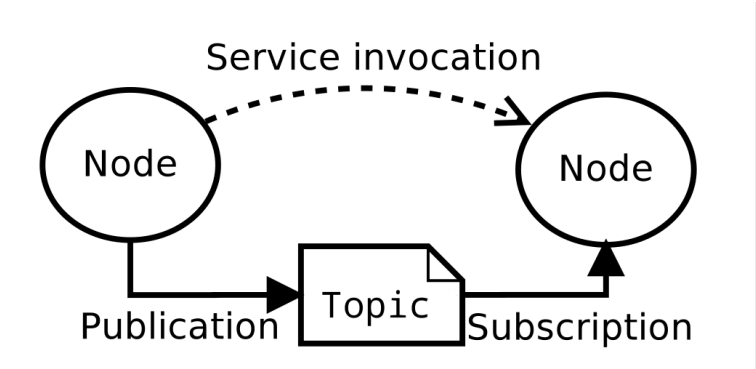
Úroveň komunity je úroveň zdrojů, která umožňuje oddělit jednotlivé komunity k zajištění výměny softwaru a znalostí. Tyto zdroje zahrnují různé distribuce, repositáře, ROS wiki, ROS Q&A (Questions & Answers) a podobně.

Úroveň výpočetního grafu je jádrem funkcionality celého systému postaveného na ROS. Výpočetní graf je peer-to-peer síť ROSovských procesů, které společně zpracovávají data. Základními stavebními prvky této úrovně jsou nodes (uzly), master, messages (zprávy), services (služby), topics (témata) a bags (pytle), které dodávají data výpočetnímu grafu různými způsoby:

- **Nodes (uzly)** jsou výpočetní procesy. Lze si je představit jako jednotlivé zdrojové programy. Uzly jsou napsány v programovacím jazyce za pomoci uživatelských knihoven, které ROS poskytuje. Ovládací systém robota se obvykle skládá z více uzlů. V systému robota je například uzel, který řídí motor, další který zajišťuje lokalizaci, další který zajišťuje plánování trasy a podobně.
- **Master (jádro)** zajišťuje registraci jmen a vyhledávání ve zbytku výpočetního grafu. Bez tohoto prvku by jednotlivé uzly by navzájem nevěděly o svojí existenci, nemohly by si posílat zprávy a volat služby. Jednotlivé uzly komunikují s jádrem, které pro ně zajišťuje informace o ostatních zaregistrovaných uzlech a informuje je o změnách, aby

mohly mezi sebou vytvářet spojení dynamicky pokaždé když je spuštěn nový uzel. Jádro pouze poskytuje informace uzlům, které pak spolu komunikují přímo. Příklad takové komunikace je znázorněn na obrázku 2.5.

- **Messages (zprávy)** jednotlivé uzly spolu komunikují předáváním zpráv buď asynchronně pomocí témat (Topics), nebo synchronně pomocí služeb (Services). Zpráva je jednoduchá datová struktura (boolean, integer, float apod.) nebo také pole jednoduchých datových struktur. Zprávy je možno kombinovat a skládat z jiných zpráv.
- **Topics (témata)** zajišťují asynchronní způsob komunikace mezi jednotlivými uzly pomocí publish/subscribe sémantiky. Uzel posílá zprávy tak, že je publikuje na daném tématu. K identifikaci obsahu zprávy je použito jméno tématu. Tuto zprávu si může kterýkoli jiný uzel přečíst tak, že se zapíše k odběru na dané téma. K odběru zprávy je potřeba pouze mít připravenou datovou strukturu, která se bude shodovat s datovou strukturou odebírané zprávy. Jeden uzel může být zapsán k odběru z více témat a publikovat více témat. Uzly o sobě tedy navzájem nevědí.
- **Services (služby)** zajišťují synchronní komunikaci typu request/reply. Služba je definována dvojicí zpráv žádost a odpověď. Uzel poskytuje službu, která se identifikuje svým názvem. Jiný uzel pak pošle žádost v předem definovaném formátu a očekává odpověď. Tato interakce je běžně implementována tak, že připomíná RPC (remote procedure call).
- **Bags (pytle)** umožňují zaznamenávat a přehrávat jakékoli ROS zprávy. Díky tomuto mechanismu je možno zaznamenat například data ze senzorů v časovém úseku a na těch pak opakovaně testovat upravený algoritmus. Pytle mají obvykle příponu .bag.



Obrázek 2.5: Znázornění komunikace mezi uzly v ROS [7]

Kapitola 3

Návrh řešení

Cílem této kapitoly je připravit návrh aplikace, která zpracováním obrazu dokáže v reálném čase z příchozího videa analyzovat jednotlivé snímky a rozhodnout, kde se v obraze nachází cesta a kde její okolí a výsledek dále publikovat tak, aby na jeho základě bylo možno učinit další rozhodnutí.

3.1 Volba algoritmu

Algoritmus, který jsem zvolil jako vzor svojí práce, byl navržen v práci *A simple and efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision*, kterou v roce 2006 publikovali autoři Arthur de Miranda Neto a Leticia Rittner [3]. Publikovaný algoritmus je součástí navigačního systému, navržený pro vozidlo startující v soutěži DARPA Grand Challenge (2005). Celý systém má za úkol řídit vozidlo tak, aby dokázalo samostatně bez kolize a vychýlení z trasy absolvovat trasu z bodu A do bodu B. K tomu používá celou řadu senzorů, mezi nimi i kameru.

Detekce cesty kamerou ve zmíněném navigačním systému slouží pouze ke korigování směru vozidla, vozidlo se jinak řídí GPS souřadnicemi nebo pomocí ostatních senzorů. Přesto mě výsledky prezentované v této práci zaujaly, protože podle autorů algoritmus přes svojí jednoduchost vykazoval poměrně vysokou spolehlivost a nízkou výpočetní spotřebu. Rozhodl jsem se obdobu tohoto algoritmu implementovat a ověřit, jestli tento algoritmus obstojí i v složitějších podmínkách na složitějších cestách v rozmanitějším okolí, na vlastních datech a na datech nahraných pro účely soutěže Robotour 2013.

3.2 Přijaté předpoklady

Protože rozsah variací cest a nastavení je obrovský, vytvořit jeden univerzální algoritmus, který by jakoukoli cestu detekoval za všech podmínek, není možné. Před návrhem vlastní aplikace bylo tedy zapotřebí přijmout některé důležité předpoklady.

Typ cesty a prostředí

První přijatý předpoklad je, že v obraze, ve kterém se systém pokouší pomocí navržené aplikace cestu detekovat, **se nějaká cesta nachází**. Pokud bude kamera zabírat prostředí, které cestu neobsahuje nebo zobrazuje cokoliv jiného, výstupem detektoru budou nesmyslné hodnoty.

Prostředí, ve kterém byl testován výchozí algoritmus, obsahovalo především kamenitou nebo písčitou pouštní cestu, bez rozmanitých křižovatek nebo okolí. Funkčnost navrženého detektoru bude ověřena v prostředí, které má se jmenovaným společné rysy. Cílové prostředí bude **prostředí parku** s asfaltovými, kamennými nebo hliněnými cestami, zpevněnými nebo nezpevněnými, bez vodorovného značení. Takovéto prostředí odpovídá i prostředí, ve kterém probíhala soutěž Robotour 2013. Ukázky rozdílných prostředí jsou znázorněny na obrázku 3.1.

(a) Beer Bottle Pass, cesta v Americké Nevadě, na které probíhala soutěž Darpa Grand Challenge.

(b) Prostředí parku v polském městě Lodž, vybráno z testovacích dat nahraných pro soutěž Robotour 2013.

Obrázek 3.1: Ukázky prostředí

Poloha a natočení kamery

Kamera vozidla nebo robota by měla zabírat silnici před ním **ve směru jízdy**, kolmo k rovině silnice. Předpokládaná výška je asi 1m nad zemí, toto nastavení je pouze přibližné. Ve spodní části obrazu by se měla nacházet cesta, v horní pak horizont a okolí. Dalším předpokladem je, že kamera bude mít na cestu čistý výhled a v jejím záběru se nebudou nacházet žádné součásti robota nebo jiné předměty. Znázornění polohy kamery se nachází na obrázku 3.2.

Světelné podmínky

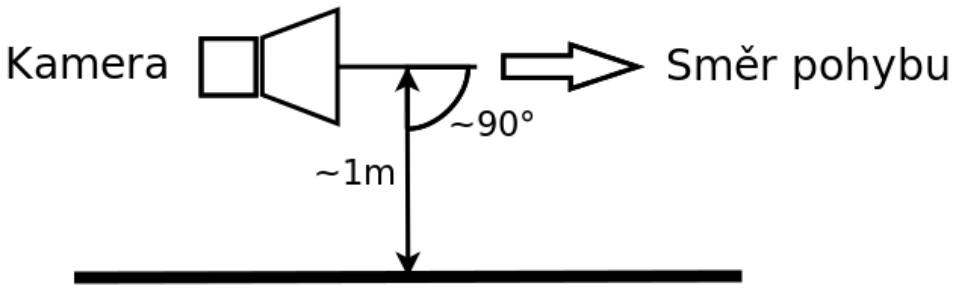
Pro funkci aplikace je důležité, aby obraz byl dostatečně světlý tak, aby bylo možno jeho části jasově odlišit. Předpokladem tedy je, že bude aplikace používána za přítomnosti denního světla. Výrazně snížená viditelnost kvůli mlze také učiní detektor neefektivním.

3.3 Návrh vlastní aplikace

Vlastní aplikace (dále jen detektor) bude uzel v knihovně ROS, který po spuštění bude komunikovat s ostatními uzly v systému.

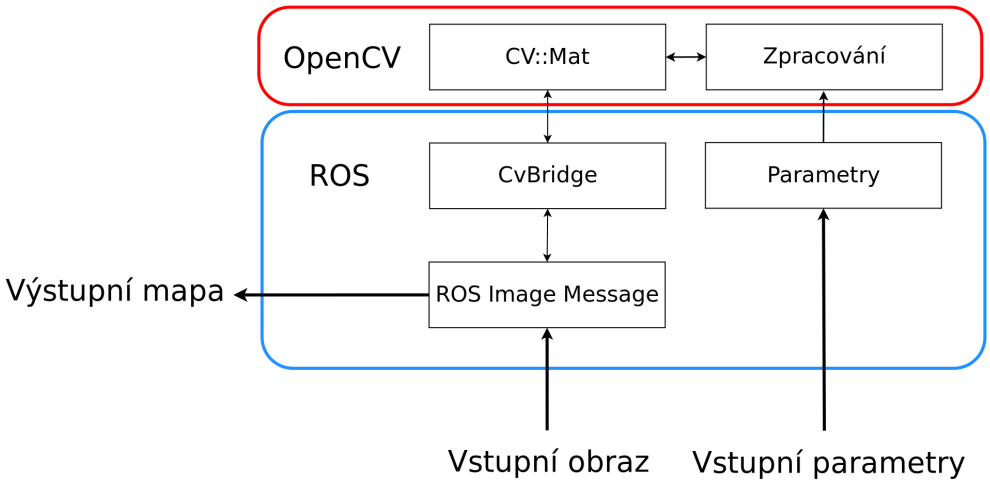
Rozhraní aplikace

Návrh rozhraní detektoru je znázorněn na obrázku 3.3. Předpokládaným **vstupem** do detektoru bude ROSovský topic, který bude obsahovat video stream s obrazem ve formátu



Obrázek 3.2: Poloha kamery vůči rovině povrchu.

knihovny ROS *sensor_msgs::image*, který bude snímat kamera pohybující se po cestě. Předpokládaným formátem je barevný obraz se složkami RGB s hloubkou 8 bitů. Dalším vstupem do detektoru budou parametry, které budou ovlivňovat postup zpracování a případné uživatelské rozhraní. Seznam navržených parametrů bude v sekci 3.3.



Obrázek 3.3: Návrh rozhraní cílové aplikace.

Výstupem detektoru bude obraz, který bude detektor nabízet zpět do systému jako výstupní topic. Tento obraz bude mapou, která pro každý pixel vstupního obrazu bude udávat hodnotu pravděpodobnosti, se kterou detektor tento bod považuje za cestu. Hodnota pravděpodobnosti bude desetinné číslo v rozmezí 0-1 s tím, že např. hodnota 0.6 odpovídá pravděpodobnosti 60%. Konkrétní hodnoty pravděpodobností budou dynamicky nastavitelné za běhu aplikace. Tento přístup je výhodný proto, že pokud by měl být detektor součástí systému, který používá více detektorů dodržujících tuto konvenci, jejich výstupy budou snadno kombinovatelné.

Zpracovaní obrazu bude zajištěno s pomocí funkcí knihovny OpenCV. Protože ma-

tice reprezentující obrázek, se kterými umějí funkce knihovny OpenCV pracovat mají jiný formát než proměnné obsahující obrázky, které používá knihovna ROS, konverzi mezi nimi zajistí třída knihovny ROS *CvBridge*.

Extrakce cesty

Detektor se pokusí klasifikovat objekty v obraze na dva druhy – cestu a okolí. Toho se dosáhne metodou globálního tresholdingu, konkrétně metodou, kterou ve své práci navrhl Nobuyuki Otsu [4]. Tuto metodu ovšem ze své podstaty nelze jednoduše použít na vstupní obraz a její výstup prohlásit za výsledek.

Prvním problémem je citlivost na detaily a šum. Detaily v obraze nás ve výsledku nezajímají a na zpracování touto metodou mají negativní vliv, takže je v rámci předzpracování obrazu vstupní obraz ošetřen filtrem pro jejich odstranění. V základním nastavení půjde o filtrování Gaussovským rozmazáním, tedy konvolucí s maskou složenou z elementů určených 2D Gaussovou funkcí. K dispozici budou pro testování i další filtry.

Další důležitý problém je tzv. "problém horizontu". Protože metoda Otsuho metoda prahování hledá v celém obraze ideální práh, učiní velký rozdíl jasu mezi oblohou a oblastí, ve které se nachází předpokládaná cesta, prahování neúčinným. Tento problém je zobrazen na obrázku 3.4.

(a) Originální obraz.

(b) Obraz po provedení globálního tresholdingu.

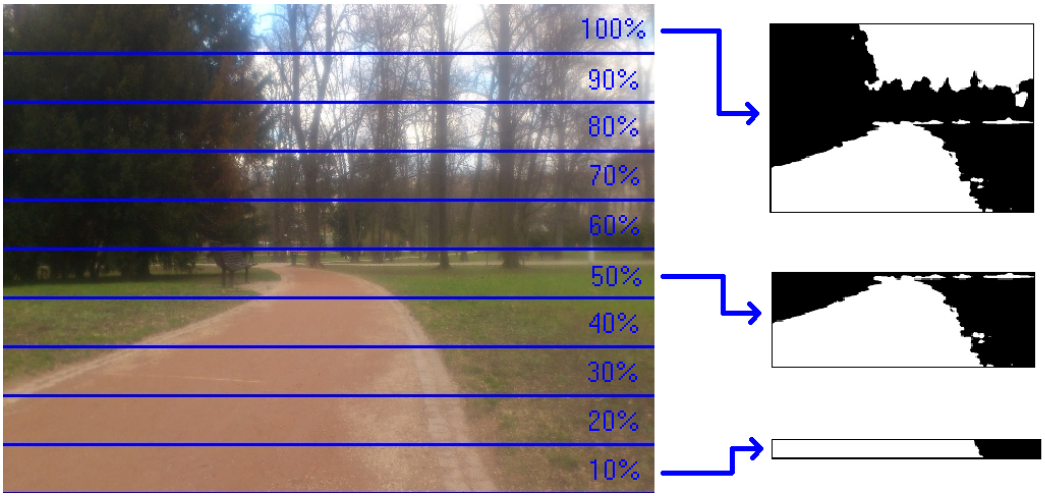
Obrázek 3.4: Ukázka úskalí globálního tresholdingu.

Odstranění tohoto efektu bude docíleno pomocí zúžení části obrazu, ve které se bude tresholding provádět, rozdělením obrazu na dvě části – spodní a horní nebo také oblast, která bude obsahovat cestu a oblast, která bude obsahovat horizont. Tyto dvě oblasti budou komplementární, spojené dohromady budou představovat původní obraz. Jako základní a univerzální předpoklad by se dalo považovat rozdělení obrazu napůl a označení spodní části jako část obsahující cestu a horní za horizont. Toto sice provést lze, nicméně to předpoklad je pro mnoho případů nedostačující. Proto nalezení horizontu probíhá pomocí pro každý snímek zvlášť pomocí algoritmu, který je pojmenován *TH finder*.

Hledání horizontu

K nalezení horizontu je snímek nejprve rozdělen na 10 stejných části, vodorovných pruhů. Algoritmus poté začne analyzovat pruh, který je nejblíž kameře, v obraze tedy nejspodnější

pruh, po první imaginární řez. První podobraz má tedy 10% z celkové výšky snímku. Další podobraz, který bude analyzován, jde od spodního okraje snímku po další imaginární řez, jeho výška tedy odpovídá 20% původního obrazu. Tento postup se opakuje, posledním analyzovaným obrazem je samotný vstupní snímek. Tento postup je znázorněn na obrázku 3.5.



Obrázek 3.5: Proces analýzy obrazu po jednotlivých řezech.

Ve všech získaných podobrazech pak algoritmus vypočítá počet bílých bodů (bílé pixely označují navigační body, tedy cestu), a navrací vektor, který obsahuje procentuální hodnoty pro jednotlivé podobrazy. Tento vektor slouží k zjištění, jak moc zařazení dalšího snímku pozitivně přispívá k zvýšení/snížení počtu navigačních bodů. Získaný vektor je znázorněn v tabulce 3.1.

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
76%	75%	70%	62%	55%	77%	75%	71%	49%	48%

Tabulka 3.1: Získaný procentuální vektor obsahující procentuální hodnoty bílých bodů v jednotlivých podobrazech

Po vytvoření vektoru s procentuálními hodnotami přichází fáze rozhodování, na kterém indexu bude proveden řez obrazu. Toho se dosáhne vypočítáním standardní odchylky vektoru, definované jako:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - x)^2} \tag{3.1}$$

K získání bodu řezu analýzou vektoru se odečte standardní hodnota odchylky (označená jako s) od procentuální hodnoty prvního podobrazu (hodnota obsažená v prvním poli vektoru). Poté je vyhledáváním ve vektoru nalezena poslední větší hodnota, než je vypočítaný rozdíl. Index, na kterém se nalezená hodnota nachází, je poté inkrementován o jedna, tedy je vybrán následující index. Takto získaný index poslouží k finálnímu rozdělení obrazu na spodní (část obrazu odspodu po získaný index) a horní (zbytek obrazu po horní okraj) část. Například pro zkoumaný snímek na obrázku 3.5 je získaná hodnota 5, spodní část bude tedy 50% obrazu a horní část 50% obrazu, obraz bude rozdělen napůl. Výsledek je znázorněn na obrázku 3.6.

Obrázek 3.6: Nalezený řez v obrazu.

Existuje případ, kdy se může stát, že je v prvním řezu cesty nalezeno převážné množství černých pixelů. Algoritmus tedy ověřuje, jestli je bílých pixelů v řezu více, než 30%. Tento případ nastává vždy, když je díky světelným podmínkám silnice tmavší než okolí. Význam pixelů je tedy obrácený a černé pixely reprezentují cestu zatímco bílé pixely reprezentují okolí a překážky. V tomto případě je obraz invertován tak, aby výsledek segmentace odpovídal logice detektoru.

Seznam parametrů

Parametry budou díky parametrovému serveru knihovny ROS měnitelné za běhu pomocí nástroje *rosparam*. Základní nastavení i s významem jednotlivých parametrů bude obsaženo v souboru *.launch*, které bude spouštět celou aplikaci. Seznam vstupních parametrů:

- volba typu předzpracování obrazu
- volba typu hledání horizontu
- zobrazení průběhu segmentace nebo výsledku na obrazovku
- nastavení formátu výstupní masky

Kapitola 4

Realizace

Tato kapitola obsahuje implementační detaily aplikace navržené v kapitole 3, výsledky testování aplikace na datech nahraných pro soutěž Robotour 2013 a na vlastní sadě dat, porovnání s algoritmem pro nalezení cesty na robotu Toad, vyhodnocení a diskuzi o možných vylepšeních a směru dalšího vývoje.

4.1 Konkrétní implementace

Aplikace je implementována v jazyce C++, verze knihovny ROS má název *Hydro*. Verze OpenCV, na které byla aplikace vyvíjena je 2.4.9. Název vytvořené aplikace je *path_detector*.

Popis aplikace

Aplikace se skládá z funkce `main()`, ve které se nachází pouze konstruktor třídy `PathDetector` a funkce `ros::spin()`, která celou aplikaci nechá čekat dokud není uzel ukončen. Při vytvoření nové instance třídy `PathDetector` uzel začne naslouchat na topicu "/camera/image_raw"pro příchozí snímky a publikovat výsledek do topicu "/path_detector/output_video". Názvy těchto topiců je vhodné přemapovat podle potřeby v souboru *.launch*.

Třída `PathDetector` pak obsahuje metodu `imageCallback`, která se provede pokaždé, když uzel přečte příchozí zprávu na zaregistrovaném vstupu. V této metodě jsou pak volány ostatní metody, které zpracovávají příchozí snímek.

Průběh metody imageCallback Po každém zavolání metoda `imageCallback` převede vstupní snímek pomocí metody `toCvCopy` třídy `cv_bridge::CvImagePtr` na obrázek ve tvaru `cv::Mat`. Na ten jsou pak dále aplikovány funkce knihovny OpenCV. Třída `PathDetector` si uchovává dvě instance příchozího snímku – původní snímek `current_frame` a právě zpracovávaný snímek `process_frame`, do kterého se ukládají změny.

Na obrázek uložený v `process_frame` se pak postupně aplikují operace zpracování obrazu. Nejprve je obraz zmenšen na rozlišení nastavené v parametrech metodou `imageResize()`. Pro předzpracování je zavolána metoda `imageSmoothing()`, která na obraz ve výchozím nastavení aplikuje funkci `cv::GaussianBlur()`. Metodu předzpracování je možno měnit pomocí změny parametrů.

Když je obraz připraven k segmentaci, nejprve se analýzou obrazu zjistí, v jaké části obrazu bude segmentace probíhat. Toto zajišťuje volání metody `imageGetRoi()`, která vrací číslo 0-100, které odpovídá velikosti obrazu v procentech odspodu (viz 3.3).

Následuje volání metody `imageProcess(int)`, která jako parametr přijímá číslo nalezené hledáním horizontu. Tato funkce provede segmentaci a vytvoří masku, do bodů označených za cestu přiřadí pravděpodobnost `prob_hit` a do bodů okolí přiřadí pravěpodobnost `prob_miss`, jež jsou nastaveny v parametrech detektoru. Masku pak publisher publikuje na výstup.

Překlad a spuštění aplikace

Detektor je balíčkem vytvořeným pomocí nástroje *catkin*[1] verze 2.8.3. Po nastavení pracovního prostředí *catkin* je balíček přeložen pomocí *catkin_make*. Při vytváření balíčku byly použity komponenty:

- roscpp
- rospy
- std_msgs
- sensor_msgs
- cv_bridge
- image_transport

Po přeložení lze detektor spustit pomocí nástroje *roslaunch*[2] přiloženým souborem `pathdet.launch`. Tento soubor také obsahuje seznam veškerých parametrů a jejich hodnoty a je možno zde případně přemapovat jména vstupních a výstupních topiců. Soubor spustí ROSovský uzel reprezentující detektor a nastaví hodnoty parametrů na parametrovém serveru. Případné hlášení nebo varování budou zobrazeny na standardní výstup. Interakce s detektorem probíhá pomocí nástroje *rosparam*[3], který mění parametry běhu. Aplikace poběží dokud nebude zastavena signálem SIGTERM.

4.2 Testování

Jako testovací vzorek byly zvoleny 3 sady obrázků, přičemž první dvě sady obrázků 4.2 a 4.3 jsou sady snímků pořízené pro testování v rámci soutěže Robotour 2013 v parku vedle VUT FIT a v polském parku ve městě Lodž. Třetí sada obrázků 4.4 je pořízena mnou v parku Lužánky v Brně. V obrázku jsou čtyři sloupce:

(a) originální obrázek s detekcí horizontu

(b) výsledek detektoru *path_detector*

(c) výsledek detektoru *hue*

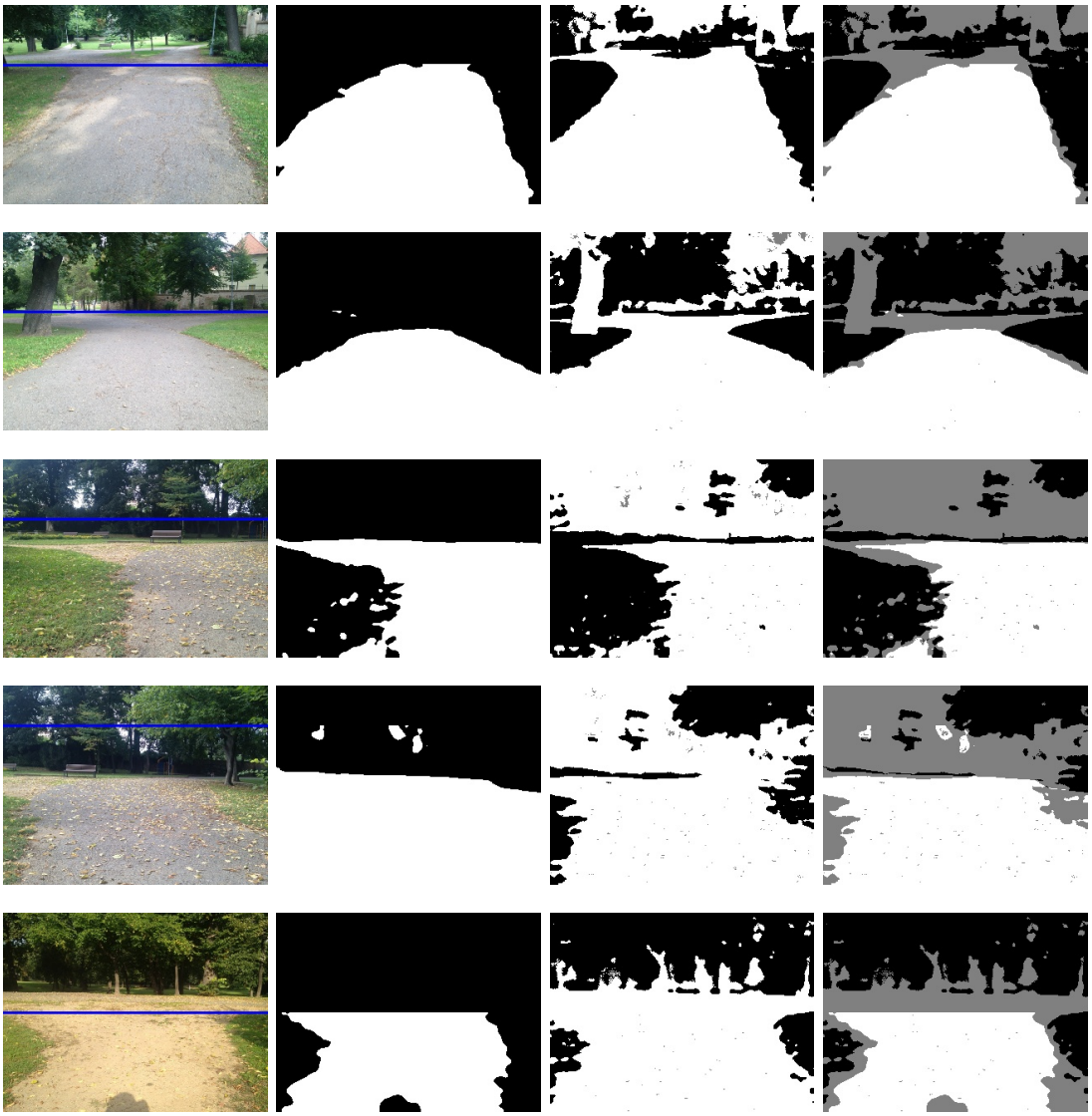
(d) kombinace výsledků obou detektorů

[1] http://wiki.ros.org/catkin
[2] http://wiki.ros.org/roslaunch
[3] http://wiki.ros.org/rosparam

Obrázek 4.1: Robot Toad, který je cílovou platformou pro vyvinutý detektor. Probíhalo na něm nahrávání jednoho z testovacích videosetů.

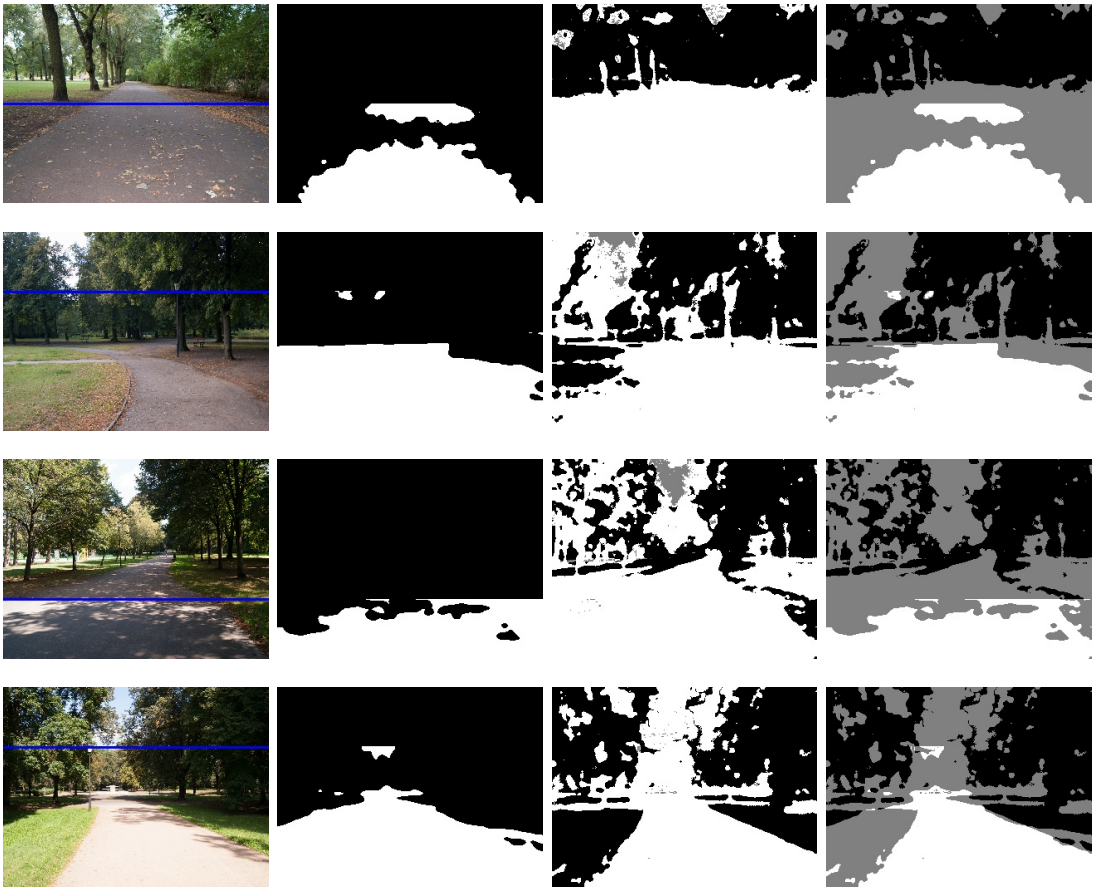
Vstupní obrázky mají rozlišení 320x240 pixelů. Větší rozlišení do obrazu zbytečně přidává velké množství detailů, které pak znehodnocují segmentaci. Pro detekci horizontu v obrázku v prvním sloupci je použit algoritmus *th_finder* navržený v 3. Výsledek detektoru *path_detector* je porovnán s výsledkem detektoru *hue*, který je součástí systému detekce cesty na robotu Toad. Tento detektor cestu v obraze rozlišuje na základě barvy po převodu do barevného prostoru HSV. Použité parametry pro detektor *hue* jsou: `hue_min=30` a `hue_max=70`.

Oba detektory mají nastaveny pravděpodobnost `prob_hit` na 1.0 pro větší názornost, takže bílé body na výsledné mapě reprezentují pixely, které detektor označí jako cestu. Spojení výsledků je uděláno jako průnik, kdy pokud se na jednom pixelu oba detektory shodnou, bod je označen hodnotou 1.0, pokud je pixel označen pouze jedním s detektorů, má na výsledné mapě hodnotu 0.5 a v případě, že pixel ani jeden z detektorů neoznačí jako cestu, je výsledná hodnota v masce 0.

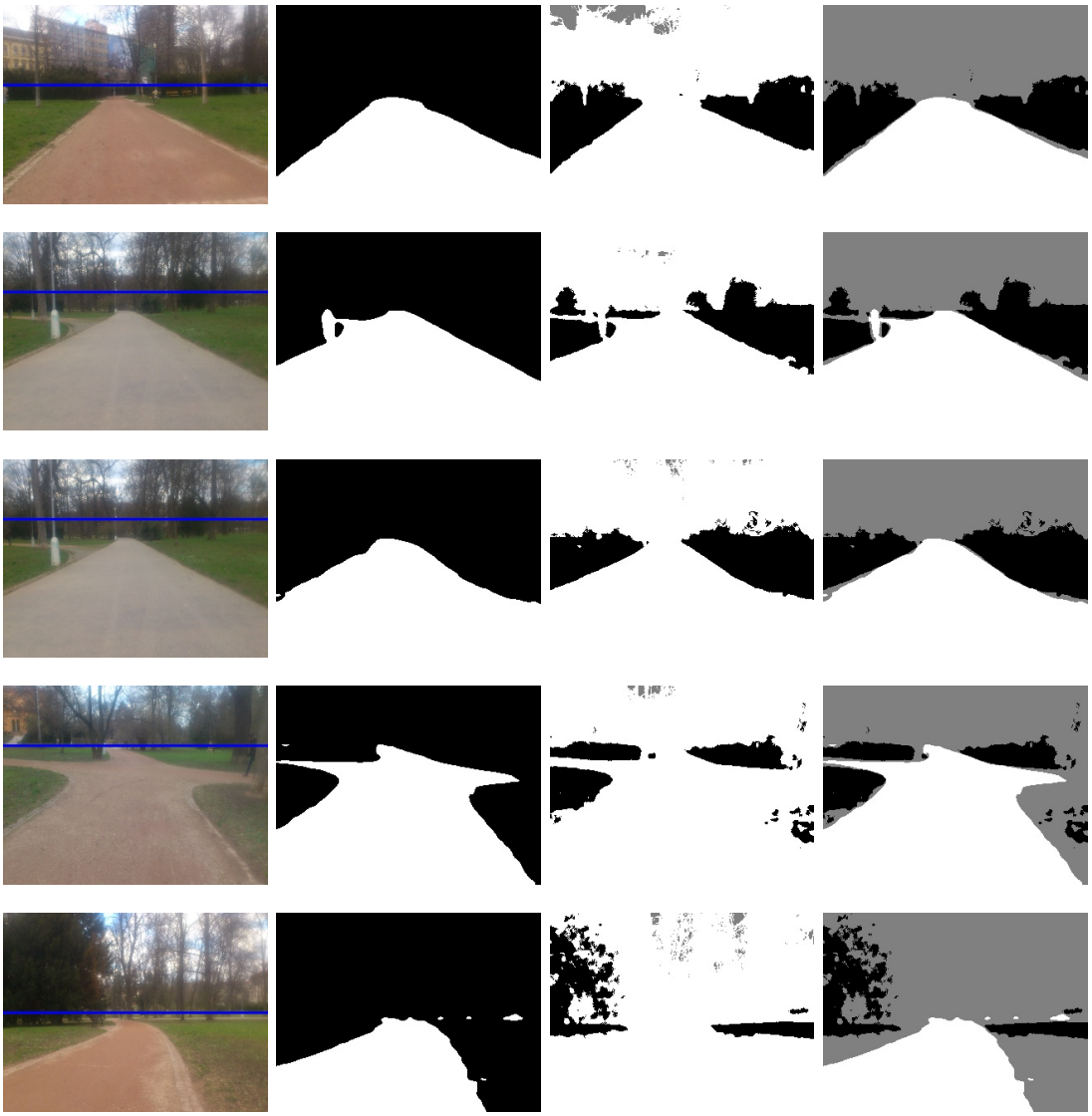


(a) (b) (c) (d)

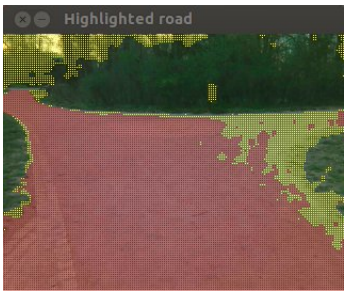
Obrázek 4.2: Sada obrázků připravená pro Robotour 2013 v parku vedle FIT: (a) originální snímek s výsledkem hledání horizontu; (b) výsledek detektoru path_detector; (c) výsledek detektoru hue; (d) kombinace výsledků obou detektorů.



Obrázek 4.3: Sada obrázků připravená pro Robotour 2013 v parku v Lodži: (a) originální snímek s výsledkem hledání horizontu; (b) výsledek detektoru path_detector; (c) výsledek detektoru hue; (d) kombinace výsledků obou detektorů.



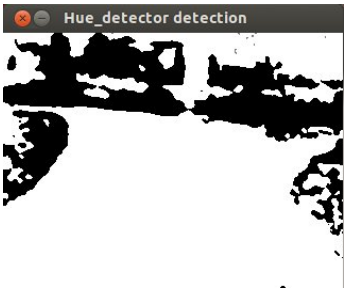
Obrázek 4.4: Vlastní sada obrázků vytvořená v parku Lužánky: (a) originální snímek s výsledkem hledání horizontu; (b) výsledek detektoru path_detector; (c) výsledek detektoru hue; (d) kombinace výsledků obou detektorů.



(a) Zvýrazněný výstup obou detektorů

(b) Výstup detektoru path_detector

(c) Kombinace masek obou detektorů



(d) Výstup detektoru hue_det

Obrázek 4.5: Testovací video set 1, nahraný v parku Lužánky. Na obrázcích je možno vidět výstupy detektorů za běhu aplikace.

Pro druhý test byla použita sada videosekvencí z webkamery spuštěné pod knihovnou ROS balíčkem *GScam*[4], v parku Lužánky nahraných nástrojem *rosbag*[5]. Další video bylo nahráno přímo pomocí kamery umístěné na robotu Toad (obr. 4.1) v parku vedle FIT VUT.

4.3 Vyhodnocení

Vyhodnocení výsledků proběhne vizuálním zhodnocením jednotlivých sad dat navržených v sekci 4.2.

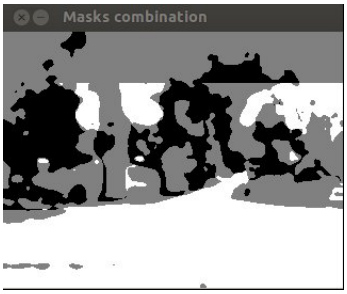
Funkčnost algoritmu

Po začátku testování se ukázalo, že hledání horizontu selhává pro případy, kdy prvních několik řezů obsahuje vizuálně pouze cestu bez okolí, což je v testovacích datech poměrně častý případ. Prahování Otsuho metodou pak na cestě odlišovalo velmi drobné detaily v jasu místo aby označilo celý řez jako cestu, tím pádem byly hodnoty ve vektoru pravděpodobností zkresleny a poloha horizontu vycházela špatně. Tento problém byl vyřešen manipulací s hodnotami vektoru. Na prvních pěti pozicích, kde se počítá s přítomností cesty, je nalezena nejvyšší procentuální hodnota. Touto hodnotou se poté nahradí všechny hodnoty ve vektoru s nižším indexem. Na zbylých pěti pozicích se kontroluje, zdali skok v procentuální hodnotě nepřevyšuje standardní odchylku vypočítanou v první polovině vektoru. Pokud

[4]http://wiki.ros.org/gscam
[5]http://wiki.ros.org/rosbag

(a) Zvýrazněný výstup obou detektorů

(b) Výstup detektoru path_detector



(c) Kombinace masek obou detektorů

(d) Výstup detektoru hue_det

Obrázek 4.6: Testovací video set 2, nahraný v parku Lužánky. Na obrázcích je možno vidět výstupy detektorů za běhu aplikace.

ano, je hodnota nahrazena hodnotou předchozí. Tím se odstraní chybné hodnoty způsobené skokovou změnou prahu. Takto modifikovaný algoritmus při testování podával daleko spolehlivější výsledky a je použit na prezentovaných testovacích datech.

Vyhodnocení jednotlivých datových sad

Na sadě obrázku 4.4 je vidět, že navržený detektor podává dobré výsledky. Algoritmus pro hledání horizontu funguje určuje polohu horizontu přesně. Tento případ je blízko ideálnímu, kdy se v cestě nenacházejí žádné překážky a osvětlení cesty je uniformní, na cestu nejsou vrhány žádné stíny. Naproti tomu algoritmus *hue* detekuje cestu i tam, kde se žádná nenachází, nicméně přesnost výstupu tohoto detektoru by byl znatelně lepší, kdyby také používal detekci horizontu, což je krásně vidět na kombinaci výstupů obou detektorů.

Na sadě obrázků 4.2 již nejsou podmínky tak ideální, přesto je vidět, že se detektor *path_detector* umí vyrovnat i s drobnými ostrůvky slunečních paprsků vrhaných přes listí stromů. Listí stromů ležící na cestě pak není dostatečně výrazné tak, aby narušilo funkci detektoru, takže výsledky jsou rozumné. Detektor *hue* stejně jako v předchozím případě detektuje jako cestu i nebe ve vrchní části obrazu. Kombinace výstupů obou detektorů je však přesvědčivým výsledkem.

Sada obrázků 4.3 zobrazuje výsledky detektorů v obtížných podmínkách. Pokud rozdíl jasu cesty a jejího okolí je minimální, zatímco rozdíly jasu v ostatních částech obrazu jsou větší, *path_detector* není schopen cestu od okolí rozeznat. Také větší množství přímo osvětlených částí cesty skrz listí stromů ruší detektor horizontu. Výsledky sice nejsou úplně nesmyslné, ale nalezená cesta neodpovídá reálnému tvaru cesty. Ani výsledky detektoru *hue*

nejsou přesvědčivé, protože zobrazuje cestu daleko širší, než ve skutečnosti je. Kombinace obou výsledků je o něco lepší, takže se v nouzovém případě jeví jako dostatečná

Výsledky detekce cesty ve videu na obrázcích 4.5 a 4.6 pak potvrzuje stejné závěry, jako testování na předešlých sadách obrázků. Pokud je osvětlení v obraze uniformní, *path_detector* funguje přesně a spolehlivě. Jakmile jsou světelné podmínky v obraze rozmanité, jeho použitelnost klesá. Detector *hue* je stabilnější, nicméně není tak přesný.

Shrnutí

Testování odhalilo silné stránky navrženého detektoru i jeho slabiny. **Výhodou** je především jeho jednoduchost, rychlost a nezávislost na systému na kterém běží. Detektor není potřeba předem trénovat aby mohl okamžitě podávat výsledky. **Nevýhodou** je především malá robustnost. Detektor je velmi spolehlivý v podmínkách, které se přibližují ideálním. Jakmile je vystaven složitějšímu prostředí, jeho spolehlivost klesá. Testy také ověřily, že jednoduchou kombinací s detektorem *hue* lze výsledek rapidně vylepšit.

4.4 Budoucí práce

Po otestování aplikace se nabízí řada směrů dalšího vývoje vedoucího k vylepšení detektoru. Prvním vylepšením by bylo návrh a realizace **vlastního algoritmu prahování**. Detektor používá metodu Otsuho prahování z knihovny OpenCV, která pro některé případy analýzy řezů v obraze není vhodná (viz. 4.3). Způsob určování práhu by mohl být závislý na aktuálně analyzovaném řezu tak, aby se například v případě, kdy celý řez zabírá pouze cesta a rozdíly intenzity v řezu jsou minimální, dal práh uměle snížit.

U testů se osvědčilo **kombinování výstupů detektorů** založených na jiných algoritmech. Navržením a realizováním dalších jednoduchých způsobů detekce cesty a jejich kombinací může být zásadně vylepšen výsledek detekce. Pro kombinování jejich výstupů by pak bylo potřeba zvolit vhodnou pravděpodobnostní funkci.

S kombinováním výsledků souvisí i případný návrh a implementace systému, který by dokázal určit **jistotu výsledku**. Pokud by například analýzou řezů bylo zjištěno, že hodnoty nemohou odpovídat cestě, systém by dynamicky upravil hodnoty na výstupní masce tak, aby měl výstup konkrétního detektoru na celkový výsledek nižší vliv. Špatný výsledek detektoru by například mohl být zjištěn pomocí detekce hran na výstupní masce a vyhodnocení, zdali tato hrana může odpovídat tvaru cesty nebo ne.

Pro předejití výpadku detektoru by také bylo možno uchovávat **předchozí snímky pro porovnání**. Pokud by se podoba detekované cesty rapidně změnila mezi dvěma snímky, bylo by možno použít snímek předchozí nebo jeho transformaci.

Kapitola 5

Závěr

Detekce cesty v obrazu z kamery je velmi komplexním problémem. Cesty představují obrovské množství variací ať už v jejich tvaru, vzhledu, ohraničení nebo prostředí, ve kterém se nachází. Vytvořit univerzální detektor, který by jakoukoli cestu detekoval pouze kamerou za jakýchkoli podmínek je velmi náročný cíl a vyžaduje velice pokročilý a robustní systém. K realizaci aplikace, která by podávala přesvědčivé výsledky, je potřeba co nejvíce zúžit množství předpokladů a přijmout celou řadu podmínek, za kterých má systém fungovat.

Cílem této práce bylo nastudovat existující řešení a vytvořit aplikaci, která detekuje cestu pomocí jedné kamery. Implementovaná aplikace byla otestována na sadě dat vytvořených pro testovací účely soutěže Robotour 2013, na sadě vlastních dat a otestována na robotu Toad.

Na testech se ukázalo, že při specifickém rozložení objektů v obraze je zvolený algoritmus velmi precizní a spolehlivý. Přesnost výsledků však rapidně klesá, je-li prostředí rozmanité a intenzita jednotlivých objektů v obraze se výrazně liší. Také uniformnost osvětlení cesty má na spolehlivost algoritmu velký vliv – velké rozdíly v osvětlení jednotlivých částí cesty značně snižují přesnost segmentace. Testy ovšem také ukázaly, že jednoduchou kombinací výstupů dvou rozdílných detektorů lze tyto vlivy značně omezit a výsledek detekce cesty podstatně vylepšit. Celý detekční systém muže tedy být použit jako dobrý zdroj informací pro navigační systém.

Další možný vývoj této práce může mít dva hlavní směry. Prvním směrem je optimalizace vytvořené aplikace i algoritmu, kterým cestu nachází a testování na konkrétních datech za účelem zjištění optimální konfigurace, která podává nejlepší výsledky pro konkrétní prostředí. Druhým směrem vývoje směřujícího k větší robustnosti je přidání dalších nezávislých algoritmů pro hledání cesty a vytvoření optimální funkce na kombinování jejich výstupů. V úvahu přichází i zpracování a využití dat z ostatních senzorů, která mohou podpořit proces segmentace obrazu.

Literatura

[1] Hillel, A.; Lerner, R.; Levi, D.; aj.: Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, ročník 25, Duben 2014: s. 727–745.

[2] Hlaváč, V.: *Počítačové vidění*. Grada, 1992, iSBN 80-85424-67-3.

[3] Miranda, A.; Rittner, L.: A simple and efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision. 2006.

[4] Otsu, N.: A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, ročník 9, Leden 1979: s. 62–66.

[5] Quingley, M.; Berger, E.; Ng, A.: STAIR: Hardware and Software Architecture. Technická zpráva, Computer Science Department, Stanford University, 2007.

[6] Quingley, M.; Gerkey, B.; Conley, K.; aj.: ROS: an open-source Robot Operating System. Technická zpráva, Willow Garage, Menlo Park, CA, Duben 2010.

[7] WWW stránky: ROS wiki. http://wiki.ros.org/ROS.

[8] Španěl, M.; Beran, V.: Obrazové segmentační techniky [online]. 2006-19-01 [cit. 2015-2-10], http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769322.

Příloha A

Obsah DVD

K bakalářské práci je přiloženo DVD s touto adresářovou strukturou:

- **path_det** - balíček ROS s cílovou aplikací
  - **src** - zdrojové soubory
  - **launch** - spouštěcí soubor
  - **include** - hlavičkové soubory
  - soubory `CMakeLists.txt` a `package.xml` nezbytné pro funkci balíčku
- **thesis**
  - **src** - zdrojové soubory LaTeX
  - **pdf** - soubor s textem bakálářské práce
- **data**
  - **images** - obrázky použité při testování práce
  - **bags** - soubory .bag obsahující videosety nahrané pro testování práce
- **manual** - stručný návod k použití aplikace