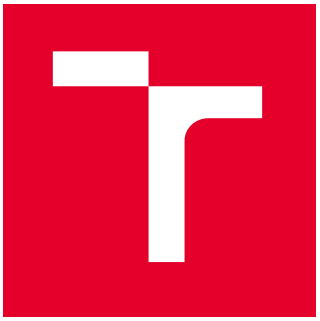


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SEGMENTACE SILNIC A CEST V OBRAZOVÝCH DATECH PRO ÚČELY AUTONOMNÍ JÍZDY

ROAD AND PATH SEGMENTATION IN IMAGES FOR AUTONOMOUS DRIVING SCENARIO

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Ondřej Janíček**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Stanislav Svědiroh**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Ondřej Janíček ***ID:*** 240358

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Segmentace silnic a cest v obrazových datech pro účely autonomní jízdy

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce a úkolem studenta je implementovat algoritmus schopný segmentovat cestu pro autonomní jízdu robotické platformy z obrazových dat. Důraz je kladen zejména na schopnost detekce parkových cest různých povrchů. Předpokládá se umístění kamery ve směru jízdy robotické platformy. Důležitým parametrem výsledného řešení je jeho výpočetní náročnost.

1) Prozkoumejte možnosti detekce a zejména segmentace silnic a cest v obrazových datech. Předpokládá se umístění kamery ve směru jízdy.

2) Diskutujte výhody a nevýhody již existujících řešení s ohledem na výpočetní náročnost, latenci, robustnost a schopnost segmentovat i parkové cesty s různými povrchy.

3) Po konzultaci s vedoucím vytvořte nebo jinak získejte dataset videí zachycujících zpevněné i nezpevněné parkové cesty.

4) Zprovozněte vybraná existující řešení, nebo implementujte algoritmus či neuronovou síť, jehož výstupem bude maska ohraničující detekovanou cestu a ověřte jeho spolehlivost na vytvořeném datasetu.

5) Diskutujte výsledky Vaší práce s ohledem na výpočetní náročnost (potřebná operační paměť CPU/GPU, latence) a robustnost (světelné podmínky, detekce více typů cest najednou).

DOPORUČENÁ LITERATURA:

Deng, F.; Zhu, X.; He, C. Vision-Based Real-Time Traversable Region Detection for Mobile Robot in the Outdoors. Sensors 2017, 17, 2101. https://doi.org/10.3390/s17092101

https://www.cvlibs.net/datasets/kitti/eval_road.php

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 22.5.2024

Vedoucí práce: Ing. Stanislav Svědiroh

Ing. Miroslav Jirgl, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zabývá tématem segmentace silnic a cest pro účely autonomního řízení. V teoretické části se zabývá počítačovým viděním, jednoduchými metodami segmentace a praktickými řešeními problému pomocí konvolučních neuronových sítí a klasických metod. V praktické části se práce zabývá sběrem testovacích dat, výběrem vhodného programovacího jazyka a výběrem vhodných knihoven. Následně se představí postup při programování vlastního řešení. Zde se začíná předzpracováním pro převod obrazu do šedotónového obrazu a filtrace šumu, poté nalezení hran v obraze pomocí Cannyho hranového detektoru, následuje definice oblasti zájmu s navazující Houghovou transformací pro detekci přímek v obraze a v poslední fázi filtrace horizontálních čar a průměrování zbylých čar. Na konci práce jsou porovnány výsledky představeného řešení s ohledem na robustnost a výpočetní náročnost.

KLÍČOVÁ SLOVA

Autonomní řízení, Segmentace, Počítačové vidění, Konvoluční neuronové sítě, Detekce cesty

ABSTRACT

This bachelor's thesis deals with the topic of segmentation of roads and paths for the purposes of autonomous driving. In the theoretical part, it deals with computer vision, simple segmentation methods, and practical solutions to the problem using convolutional neural networks and classical methods. In the practical part, the work deals with the collection of test data, the selection of a suitable programming language, and the selection of suitable libraries. Subsequently, the procedure for programming our own solution will be presented. Here it starts with pre-processing to convert the image into a grayscale image and filtering the noise, then finding the edges in the image using the Canny edge detector, followed by the definition of the region of interest, with the subsequent Hough transform to detect the straight lines in the image, and in the last stage, filtering the horizontal lines and averaging the remaining lines. At the end of the thesis, the results of the presented solution are compared with respect to robustness and computational complexity.

KEYWORDS

Self-driving car, Segmentation, Computer vision, Convolutional neural network, Road detection

Vysázeno pomocí balíčku `thesis` verze 4.09; `https://latex.fekt.vut.cz/`

JANÍČEK, Ondřej. *Segmentace silnic a cest v obrazových datech pro účely autonomní jízdy*. Semestrální práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2024. Vedoucí práce: Ing. Stanislav Svědiroh

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Ondřej Janíček
VUT ID autora:	240358
Typ práce:	Semestrální práce
Akademický rok:	2023/24
Téma závěrečné práce:	Segmentace silnic a cest v obrazových datech pro účely autonomní jízdy

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Stanislavu Svědirohovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

S autonomním řízením se čím dál častěji setkáváme v naší společnosti, převážně ve formě autonomních aut. Autonomní řízení ale není omezeno pouze na autonomní auta, ale také zahrnuje aplikace s mobilními roboty, které umožňují lidem prozkoumávat nebezpečné nebo malé prostory, vykonávat pomocné práce, jako například robotické vysavače, anebo se přímo snaží simulovat lidské chování ve formě humanoidních robotů.

Systémy jsou nazývány autonomními, protože při své funkci nevyžadují pomoc člověka. Proto je potřeba nahradit jednotlivé funkce pomocí výpočetní techniky. Protože se robot nebo vozidlo pohybuje ve volném prostoru, je nezbytné pracovat s obrazem. Za tímto účelem se v první podkapitole teoretické části seznámíme s počítačovým viděním a zpracováním obrazu, kde si představíme obecný postup a podrobnější informace o jednotlivých fázích zpracování obrazu. Na to navážeme podkapitolou hlouběji se zabývající segmentací obrazu a různými metodami rozdělení obrazu na segmenty. Dále si definujeme základní pojmy a požadavky na reálné aplikace a podíváme se na předchozí praktické návrhy implementace algoritmů pro detekci cest a silnic. Praktické metody detekce cest lze rozdělit do dvou kategorií. První kategorie zahrnuje metody založené na konvolučních neuronových sítích, které umožňují detekovat a klasifikovat objekty na obraze. Druhá kategorie jsou klasické metody, které využívají operace na obraze za pomoci příznaků, podle kterých se detekuje cesta.

V praktické části následně vytvoříme vlastní řešení problému detekce cesty. Podíváme se na základní rozhodnutí před začátkem programování, kde vybereme vhodný dataset, vhodný programovací jazyk a vhodné knihovny. Následně vytvoříme algoritmus, který bude pracovat pomocí operací na obraze. Na začátku se podíváme na předzpracování, kde se obraz převede na šedotónový a na něj je aplikován Gaussův filtr pro filtrace šumu. Poté využijeme Cannyho hranový detektor pro nalezení hran v obraze. Na hranový obraz následně aplikujeme masku oblasti zájmu pro filtraci nadbytečných hran. Na zbylé hrany využijeme Houghovu transformaci pro detekci přímek na obraze. Dále vyfiltrujeme horizontální čáry a v poslední fázi zprůměrujeme čáry pravé a levé strany pro identifikaci okrajů silnice a vykreslíme finální obraz. Výsledky programu v poslední podkapitole otestujeme s ohledem na robustnost a výpočetní náročnost.

1 Teoretická část práce

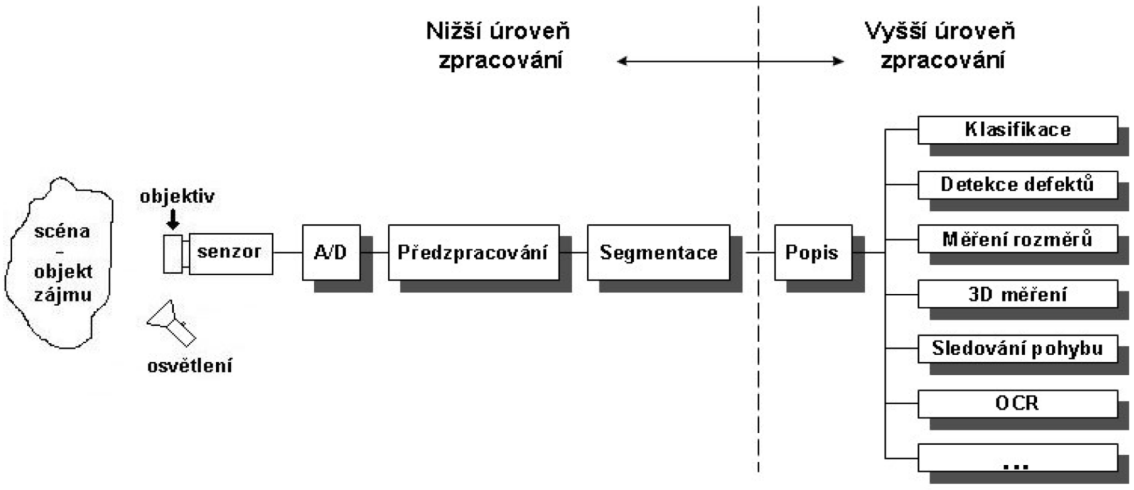
Tato kapitola se zabývá teoretickou částí bakalářské práce. Hlavním tématem této kapitoly jsou principy počítačového vidění, segmentace obrazu a představení nejpoužívanějších metod detekce silnic a cest a jejich srovnání.

1.1 Počítačové vidění

Počítačové vidění zahrnuje široké spektrum aplikací, které umožňují strojům interpretovat, porozumět a rozhodovat se na základě předložených dat ze skutečného světa. Počítačové vidění se obecně snaží pomocí výpočetní techniky a softwaru napodobit lidský zrak s určitou mírou inteligence při analýze obrazu.

S počítačovým viděním je spjato velké množství příbuzných oborů, jejichž kombinací jsme schopni realizovat praktické aplikace počítačového vidění. Patří sem například optika, snímání, umělá inteligence, teorie signálů, robotika, matematika, teorie řízení, strojové učení, biologie a další [1].

Tato podkapitola se bude hlavně zabývat způsobem zpracování obrazu od počátečního snímání, přes digitalizaci, předzpracování a následnou segmentaci. Tato část zpracování obrazu je podobná pro všechny aplikace. V poslední řadě následuje popis a klasifikace objektů do patřičných kategorií. Popis a klasifikace jsou různé pro různé druhy aplikací [2].



Obr. 1.1: Obecný postup zpracování obrazu [1]

1.1.1 Snímání obrazu

První část zpracování obrazu je snímání. Snímání obrazu je proces, při kterém dochází k získávání dat z vizuální scény. Obraz bývá snímán pomocí kamer, nejčastěji využívajících dvou typů senzorů CCD nebo CMOS.

CCD je elektronická součástka využívající fotoelektrický jev. Fotoelektrický jev spočívá v tom, že částice světla při nárazu do atomu předává energii elektronům, které se mohou dostat do excitovaného stavu. V závislosti na množství dopadajících fotonů se zvyšuje nebo snižuje počet uvolněných elektronů. Tyto uvolněné elektrony se mohou u polovodičů podílet na elektrické vodivosti přiložením elektrody. U CCD jednotlivé pixely obsahují kladně nabitou elektrodu, která přitahuje uvolněné elektrony. Elektrony jsou odvedeny změnou fáze elektrod, hlavní a i předtím nezapnutých vedlejších elektrod, do výstupního zesilovače, který zesílí malé proudy [3].

CMOS je elektronická součástka pracující na principu fotodiody. Využívá také fotoelektrického jevu, s tím rozdílem, že každý pixel obsahuje zesilovač, což umožňuje čtení náboje z každého pixelu oproti posuvnému čtení u CCD [4].

Velmi důležitým parametrem při snímání obrazu je kvalita snímání, která závisí na několika faktorech, jako je rozlišení, kvalita, ostrost, ale také na parametrech obrazu, jako je expozice nebo barevný kontrast snímaného objektu.

Při každém snímání dochází ke ztrátě informace, kterou se snažíme zamezit kvůli obtížnostem při následném zpracování, protože ztracená data nelze zpětně rekonstruovat. Nejčastějšími ztrátami informace, na které musíme dávat pozor, jsou:

- Při převodu 3D obrazu na 2D ztrácíme informace kvůli perspektivní projekci.
- Při rušení, které je přítomné u každého měření, nejčastěji ve formě tepelného šumu a elektromagnetického rušení.
- Při špatně zvolené expozici se ztrácí detaily přesvětlením nebo podsvětlením obrazu.
- Špatné zaostření vede ke ztrátě informací mimo zaostřenou oblast.

1.1.2 Digitalizace obrazu

Další fází procesu je digitalizace obrazu. Digitalizace je proces převodu analogového signálu na digitální signál. Po digitalizaci jsme schopni pracovat s digitálními daty pomocí výpočetní techniky.

S obrazem se pracuje v tří barevném spektru RGB, neboli s červenou, zelenou a modrou, protože jejich kombinací můžeme vytvořit dostatečně pestré barevné spektrum. Informace o barvě získáváme pomocí Bayerova filtru, který je součástí většiny moderních kamer. Bayerův filtr obsahuje tři filtry, kde každý filtruje ostatní dvě složky RGB.

Při digitalizaci obrazu dochází k vzorkování analogového signálu do matice s řádky a sloupci. Signál je dále kvantizován, což znamená, že se ukládá informace o velikosti složky jednotlivých pixelů. Čím je větší matice a počet kvantizačních hladin, tím lepší je aproximace obrazu. U CCD a CMOS snímačů je použit A/D převodník, který převádí proud na napěťové úrovně, jež jsou následně převedeny na digitální hodnotu.

1.1.3 Předzpracování obrazu

Samotné předzpracování nezvětšuje množství informace v obraze, ale pomáhá zvýraznit chtěné a potlačit nechtěné rysy, proto se často hodí pro specifické aplikace. Pro zvýraznění chtěných rysů můžeme použít jasové transformace nebo geometrické transformace pro úpravu vlastností obrazu.

Jako jedna z nejpoužívanějších operací předzpracování se používá filtrace šumu. Pro potlačení nevýznamných dat lze dosáhnout pomocí časové, prostorové nebo frekvenční filtrace.

1.1.4 Segmentace obrazu

Segmentace obrazu je metoda zpracování digitálního obrazu, která slouží k rozčlenění obrazu do více oblastí, nazývaných segmenty, které obsahují společné vlastnosti. To umožňuje rozlišovat různé prvky a struktury na obraze. Výsledkem segmentace jsou nepřekrývající se oblasti, které se využívají pro další zpracování.

Segmentace se může snažit o úplnou segmentaci, která hledá jednoznačné objekty, nebo o částečnou segmentaci, která může hledat plochy, které nemusí přímo korespondovat s objekty.

Hlavním problémem segmentace je dvojznačnost dat, protože rušení a rozmazané hranice textur a barev mohou narušit přesnost segmentace. Proto je velmi důležitý správný výběr metody segmentace, která dokáže částečně minimalizovat nepřesnosti. Bližší popis metod segmentace obrazu je v podkapitole 1.2. Často také vznikají problémy se stíny na obraze, které mění jas povrchu a zhoršují podmínky pro rozpoznávání kontinuálních ploch.

1.1.5 Popis objektu

Popis objektu slouží k získání příznaků ze segmentovaných dat, které jsou typické pro daný objekt. Příznaky zahrnují tvar, barvu, texturu, rozměry, orientaci a nebo jejich vzájemnou kombinaci. Tyto příznaky jsou nezbytné pro následnou klasifikaci. Naším hlavním požadavkem jsou vhodně zvolené příznaky pro odlišení různých objektů a pro shodu příznaků u stejných objektů.

1.1.6 Klasifikace objektu

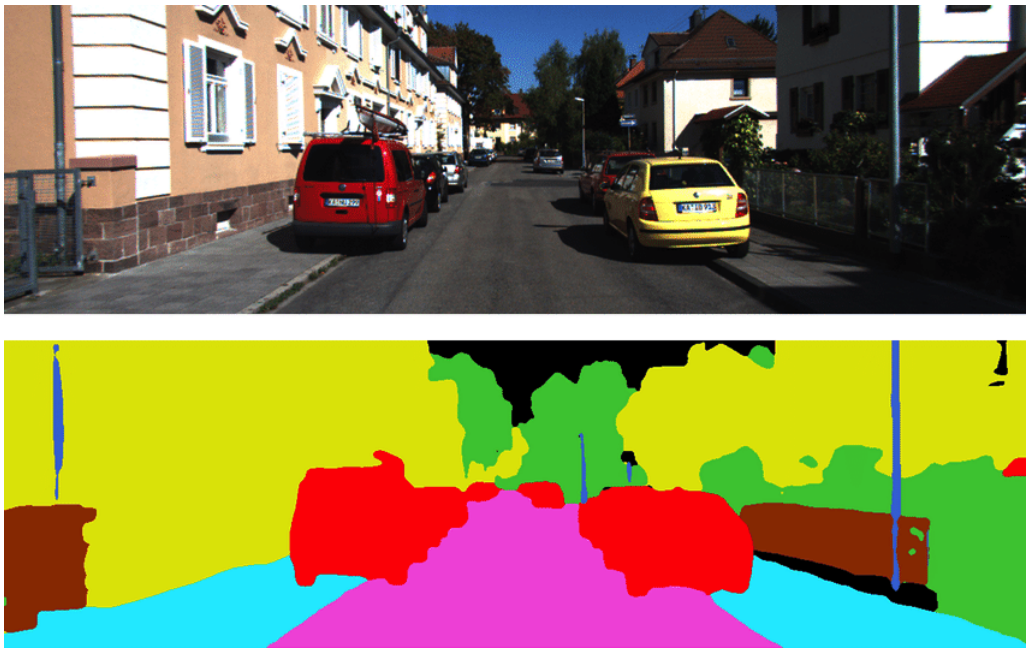
Klasifikace objektů je založena na třídění jednotlivých objektů do náležitých tříd, podle společných charakteristik. Cílem je naučit model automaticky rozpoznávat a klasifikovat objekty na základě vlastností, které jsme získali z předchozích kroků. Model, který zajišťuje klasifikaci, se nazývá klasifikátor.

Při klasifikaci objektů jsou velmi důležitá tréninková data. Klasifikace objektů není možná bez předchozí znalosti podobných objektů, proto je nutné naučit klasifikátor na různé varianty objektu a je nutné při tréninku kontrolovat správnost zařazení objektů. Následně je nutné upravit klasifikátor na základě znalostí předchozích chyb. Možné úpravy spočívají ve změnách nebo rozšíření tréninkových dat, nebo například ve změně klasifikačního modelu.

1.2 Základní metody segmentace obrazu

Jak již bylo zmíněno v předchozí podkapitole, segmentace obrazu je metoda zpracování digitálního obrazu, která se snaží rozčlenit obraz na segmenty se společnými vlastnostmi, což umožňuje rozlišovat různé prvky a struktury na obraze. Výsledkem segmentace je více nepřekrývajících se oblastí. Samotná segmentace však není schopna vytvořit perfektní výsledky, a proto je nutné upravit výstup segmentace. Zpravidla se stává, že segmenty jsou příliš malé, proto je potřeba segmenty spojit. Naopak, pokud se obraz rozdělí na malé množství oblastí, kde se některé oblasti objektů překrývají, je potřeba oddělit objekty.

Cílem této podkapitoly je rozebrat různé metody segmentace, jejich vlastnosti a použití. Nic takového jako dokonalá segmentace neexistuje, proto jsme odkázáni na metody, které se snaží dosáhnout maximální homogenity oblastí pro konkrétní obraz. Správný výběr metody segmentace je velmi důležitý pro správné rozdělení oblastí, které umožňují získat potřebné informace o objektech nacházejících se na obraze. Jelikož každá metoda má své výhody a nevýhody, je nutné brát ohled na potřeby konkrétní aplikace [5].



Obr. 1.2: Příklad segmentace obrazu silnice s objekty [6]

1.2.1 Prahování

Prahování obrazu je jedna z nejčastějších a nejjednodušších používaných metod segmentace obrazu. V anglické literatuře je používan název "Thresholding".

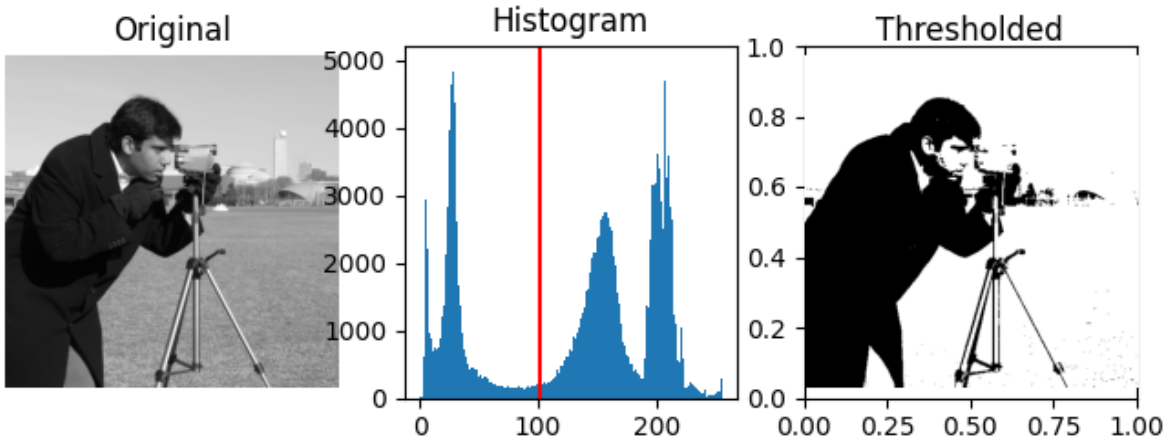
Cílem metody prahování je rozdělení obrazu na dvě nebo více homogenních oblastí pomocí jasových hodnot pixelů. Nejprve se definují prahy, které rozdělí spektrum pixelů do intervalů. Každý pixel je porovnán s prahovou hodnotou. Pokud má pixel nižší hodnotu jasu než je hodnota prahu, je přiřazen do první třídy. Pokud má pixel vyšší hodnotu jasu, je přiřazen do druhé třídy. Vytvoří se různé oblasti s přesně definovanými intervaly. Díky jednoduchosti přiřazení hodnot do patřičných tříd je metoda velmi rychlá na realizaci.

Matematicky se transformace vstupního obrazu g(i,j) převádí na binární obraz f(i,j), kde je důležité dbát na volbu úrovně prahu T.

$$f(i,j) = \begin{cases} 1 & pro \quad g(i,j) \geq T \\ 0 & pro \quad g(i,j) < T \end{cases} \tag{1.1}$$

Způsobů prahování v praxi je několik. Nejčastější je globální prahování, kde je pro celý obraz zvolena jedna nebo více stálých hodnot. Hlavním problémem globálního prahování je nerovnoměrné osvětlení objektů, které lze částečně odstranit vhodným předzpracováním. Další možnou metodou je poloprahování, kde se při vyšší hodnotě než je hodnota prahu ponechává hodnota jasu, a pro nižší hodnoty je přiřazena nula. Adaptivní prahování je metoda, při které se obraz rozdělí na několik

stejných obdélníků, a pro každou část se určuje práh zvlášť. Další metody jsou například hysterezní nebo vícestupňová. Práh může být hledán automaticky [5].



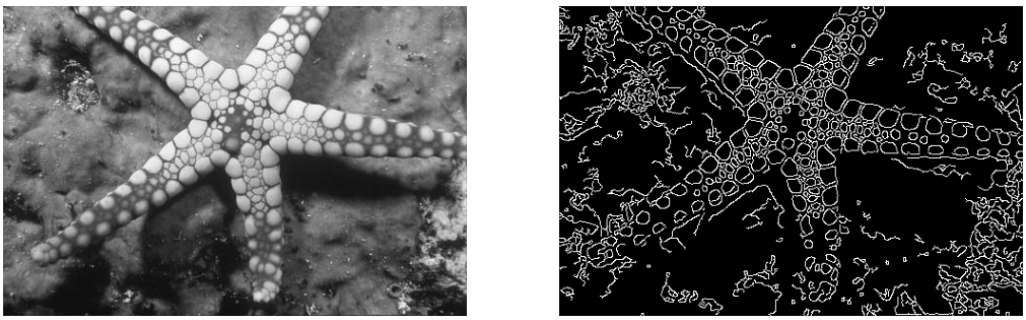
Obr. 1.3: Prahování s jedním prahem přibližně v polovině šedotónového spektra [7]

1.2.2 Detekce hran

Segmentace založená na detekci hran je metoda, která využívá hran a kontur k rozdělení obrazu do segmentů. V anglické literatuře je používán název "Edge-based segmentation".

Pro tuto segmentaci je zásadní výběr hrany. Hrany v obraze jsou hledány pomocí skokové změny intenzity jasu na obraze, což naznačuje rozhraní mezi objekty. Tvary hran ale mohou být různých typů, což ztěžuje přesné určení hrany. Problém spočívá v určení, jak velká změna intenzity jasu musí nastat, aby byla považována za hranu. Dalším problémem je šum, který je přítomný v každé aplikaci. U obrazu se šumem je obtížné určit, zda se u hrany jedná o chybu způsobenou šumem nebo o validní hranu. Tento problém se snažíme eliminovat pomocí předzpracování obrazu.

Pro detekci hran používáme různé metody. Každá metoda vychází z jiných principů a má rozdílné výsledky segmentace, ale vždy vytváří nový obraz, kterému říkáme mapa hran a je použita pro výpočet konečné segmentace. Metoda založená na první derivaci vychází z myšlenky, že v místech hrany dochází k největší změně intenzity jasu, zatímco v ostatních oblastech je první derivace nulová. Metoda první derivace se též nazývá metoda gradientní. Pokud nás nezajímá směr a velikost hrany je možné použít metodu druhé derivace. U druhé derivace v místech, kde se nachází hrana, neboli kde je největší změna intenzity, bývá hodnota druhé derivace nulová. Další možnou metodou je srovnání se vzorem. Základem je maska, která se přikládá na kontrolované místo. Pokud je odezva dostatečná, detekujeme hranu.

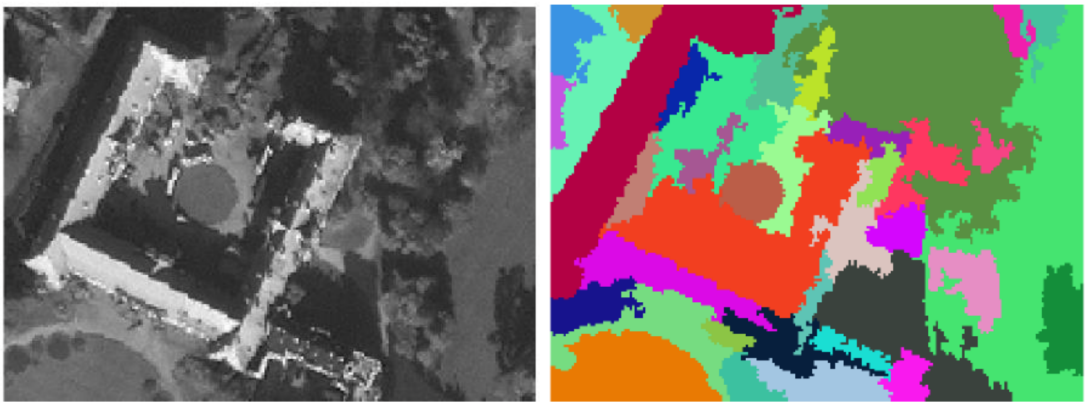


Obr. 1.4: Detekce hran [8]

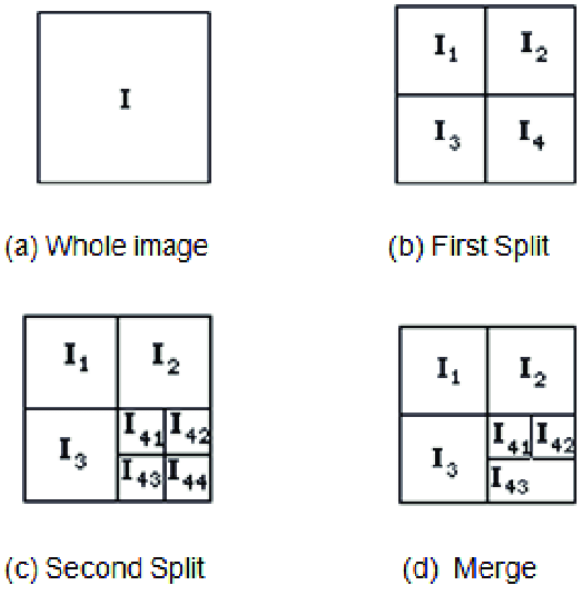
1.2.3 Hledání oblastí

Hlavní myšlenkou této segmentace je rozdělit obraz na maximálně souvislé oblasti, aby byly dané oblasti homogenní. V anglické literatuře je používán název "Region-based segmentation".

Segmentace hledání oblastí je založena na principu spojování a dělení oblastí zvané "region merging"a "region splitting". Nejprve je nutné si definovat strukturu algoritmu. Nejčastěji se používá stromová struktura, ve které se oblast, pokud není homogenní, vždy při přechodu na nižší úroveň rozdělí na čtyři části. Pokud jsou oblasti homogenní, spojují se na vyšší úrovni dohromady. Pro spojení je nutné splnit kritéria homogenity. Kritérium homogenity definujeme podle vlastností obrazu. Nejčastěji bývá určeno podle jasových vlastností pixelů.



Obr. 1.5: Hledání oblastí [9]



Obr. 1.6: Dělení a spojování oblastí [10]

1.3 Metody detekce silnic a cest

V této podkapitole se podíváme na různé přístupy k detekci silnic a cest. Na začátek si představíme používané pojmy a poté budou následovat metody detekce cest založené na konvolučních neuronových sítích a na klasických metodách detekce cest.

1.3.1 Definice pojmů

Předtím, než se začneme bavit o samotných metodách detekce cest je důležité se seznámit s často používanými pojmy spojenými s detekcí cest.

Výpočetní náročnost

Výpočetní náročnost určuje množství výpočetní techniky potřebné pro efektivní vykonání algoritmu. Tato informace je velmi důležitá, protože ovlivňuje možnou rychlost vykonávání celého programu. Obvykle se snažíme dosáhnout nejrychlejšího programového řešení, s co největší přesností segmentace a co nejnižší cenou hardwaru. Jelikož si dané požadavky odporují, je nutné najít kompromis, který je závislý na specifických požadavcích aplikace.

Latence

Definuje časovou prodlevu, která nastává mezi dvěma událostmi. V našem kontextu ji lze definovat jako prodlevu mezi snímáním a rychlostí odezvy systému.

Robustnost

Robustnost programu určuje schopnost algoritmu odolat vlivům, které by mohly poškodit kvalitu výsledné detekce. Správně navržený robustní algoritmus je schopný poskytnout přesné a konzistentní výsledky i při stížených podmínkách. Cílem robustního algoritmu je minimalizovat působení rušivých faktorů a dosáhnout vysoké přesnosti. Pro vylepšení robustnosti algoritmu detekce cest je nutné trénovat nebo postupně vylepšovat model, aby byl schopný správně identifikovat silnice a cesty za stížených podmínek.

KITTI Benchmark

Benchmark je proces porovnávání produktů, metod nebo výsledků za účelem srovnání jejich výkonu s podobnými výrobky.

KITTI Benchmark je společný projekt od Karlsruhe Institute of Technology a Toyota Institute at Chicago, který slouží k porovnávání výkonu programů počítačového vidění na praktických aplikacích. Samotný projekt zahrnuje více úloh počítačového vidění, ale pro nás je důležitá část, která se zabývá rozpoznáváním silnic. Benchmark rozpoznávání silnic obsahuje 289 tréninkových a 290 testovacích obrazů, na kterých se testuje schopnost detekce označených a neoznačených silnic. Benchmark obsahuje informace jako je rychlost algoritmu, přesnost, F-skóre a další. KITTI Benchmark nám umožňuje rychlé porovnání dostupných řešení detekce silnic s odkazy na jednotlivé vědecké práce [11].

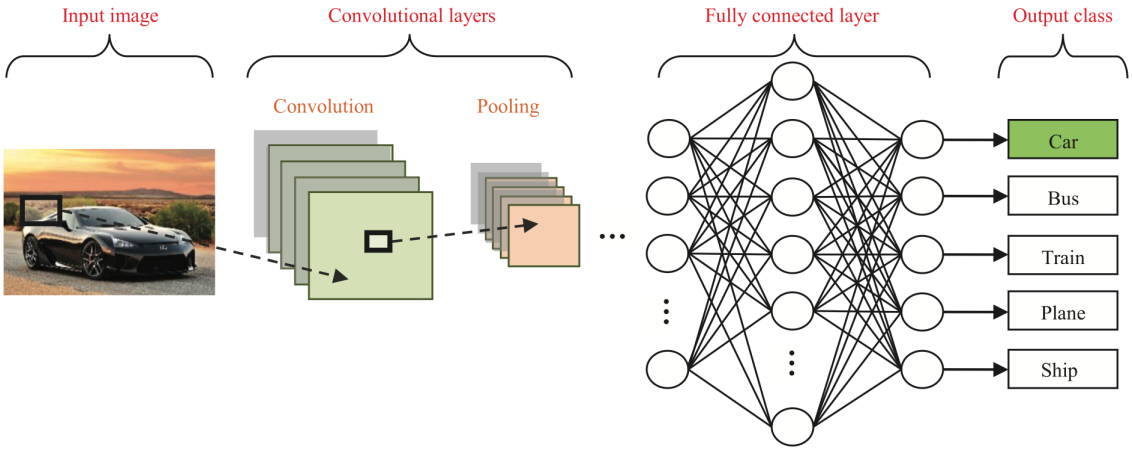
1.3.2 Metody založené na konvolučních neuronových sítích

Konvoluční neuronová síť, také anglicky známé jako Convolutional Neural Network nebo zkráceně CNN, je speciální typ neuronové sítě navržený speciálně pro řešení úloh počítačového vidění. Informační tok v konvolučních neuronových sítích proudí jedním směrem od vstupů k výstupům, známé jako feed-forward.

Jejich architektura je koncepčně inspirována funkcí lidského mozku, která tvoří složitou síť vzájemně propojených nervových buněk neboli neuronů. Každý neuron má vstupy a každý vstup je ohodnocen synaptickou váhou, která určuje propustnost neuronu. Sumace násobků vstupů a vah nám dává vnitřní potenciál neuronu. Vstupní potenciál vstupuje do aktivační přenosové funkce, kde určuje hodnotu výstupu. Jako aktivační přenosová funkce se nejčastěji používá hyperbolický tangens nebo sigmoida. Vzájemné propojení velkého množství neuronů jako vstupy do dalších neuronů následně tvoří architekturu neuronové sítě.

V dnešní době je tato architektura úspěšně používána pro aplikace segmentace obrazu, detekci objektů, klasifikaci objektů a mnoho dalších úloh počítačového vi-

dění. Každá aplikace konvoluční neuronové sítě má vlastní částečně rozdílnou architekturu, ale obecně obsahují tři vrstvy: konvoluční vrstvy, pooling vrstvy a plně propojenou vrstvu [12] [13].



Obr. 1.7: CNN pro klasifikaci objektů [12]

Konvoluční vrstva využívá matematických konvolučních operací k detekci vzorů v obrazových datech. Každý pixel obsahuje informaci o intenzitě jasu tří základních barev. Základem konvoluce je konvoluční maska, která slouží k výpočtu hodnot konvoluce. Maska je podle předpisu aplikována na jednotlivé pixely, kde každý pixel je vynásoben hodnotou masky. Všechny vypočtené hodnoty pro jeden pixel a jeho okolí se sečtou, čímž dostáváme hodnotu prostředního pixelu. Poté se přesouváme na další pixel, dokud neprojdeme všechny pixely obrazu. U neuronových sítí je každý neuron připojen k oblasti vedlejších neuronů, které tvoří recepční pole neboli reception field. Konvoluční maska je tvořena synaptickými váhami spojů. Výstupem konvoluční vrstvy je mapa příznaků, která nám pomáhá získat jednotlivé obrazové příznaky. Proto, abychom mohli získat více příznaků, je nutné aplikovat filtry.

Hlavním účelem poolingové vrstvy je snížení velikosti výstupních dat konvoluční vrstvy a tím dochází ke zvýšení prostorové invariantnosti vůči posunu. Nejčastější používané typy poolingu jsou max pooling a average pooling. Max pooling vybírá jeden neuron ze vstupní vrstvy podle největší výstupní hodnoty, a jeho výstupní hodnotu použije jako svoji výstupní. Average pooling se liší od max poolingu pouze výběrem neuronu podle průměrné hodnoty.

Plně propojená vrstva spojuje každý neuron z jedné vrstvy s předchozí vrstvou. Tato vrstva transformuje příznakovou mapu a plní funkci vyhodnocování příznaků na vyšší úrovni.

Pro učení neuronových sítí se obecně nejčastěji používá algoritmus backpropagation. Backpropagation je založen na zpětném šíření chyby pomocí metody gradi-

entního sestupu. Metoda je založena na úpravě synaptických vah na základě rozdílu mezi výstupem sítě a chtěným výstupem, který nazýváme chyba sítě. Pomocí úpravy vah snižujeme chybu sítě za využití gradientních metod.

Vývoj

Nyní se podíváme na počáteční průlom v detekci silnic a cest. Začátek používání konvolučních sítí pro řešení segmentace začal v roce 2015 díky plně konvoluční síti zkráceně FCN [14], kde byla poprvé navržena end-to-end konvoluční síť pro sémantickou segmentaci obrazu. Ihned poté se počet akademických prací na toto téma výrazně zvýšil.

Na základě FCN následoval U-Net [15], který se skládá ze smršťující a rozšiřující cesty. Přidává možnost přeskočení spojů, skip connection, mezi smršťující a rozšiřující cestou pro lepší obnovení plného prostorového rozlišení.

Na rozdíl od FCN poprvé SegNet [16] použil architekturu enkodér-dekodér, která je nyní nejpoužívanější architekturou. Mimo enkodér a dekodér obsahuje ještě pixelovou klasifikační vrstvu.

Nejlepší metody

V dnešní době již existuje nespočet možných řešení pro detekce silnic, která staví na výše zmíněných řešeních anebo vymýšlí nová inovativní řešení s výhodami a nevýhodami. Vzhledem k velkému množství prací se nyní zaměříme pouze na ty nejlepší v KITTI benchmarku. Pro hlubší zájem je možné najít více informací a všechny zveřejněné akademické práce zde: [11]. V KITTI benchmarku se především zaměříme na detekci neoznačených silnic, ale rozdíly v pořadí metod mezi označenými a neoznačenými nejsou příliš velké. Dále se budeme zabývat pouze aplikacemi s veřejnými akademickými pracemi, protože o ostatních aplikacích nemáme žádné bližší informace.

Prvně se zaměříme na robustnost, která se určuje pomocí F-skóre. Při zaměření na robustnost je nutné počítat s vyšší výpočetní náročností. Nejrobustnější řešení je SNE-RoadSeg+ [17], které používá data-fusion DCNN. Uvnitř je použit surface normal estimator (SNE) a lightweight module pro efektivní end-to-end převod mezi hloubkou obrazu do povrchových normálových inferenčních map. Tohle řešení staví na předchozí verzi SNE-RoadSeg, kde snižuje výpočetní náročnost.

Dalším robustním řešením je RoadFormer [18]. Využívá také data-fusion DCNN a využívá enkodér-dekodér architekturu pro extrakci různorodých vlastností z obrazu RGB a z povrchových normálových informací.

Dále se zaměříme na metody s nejnižší výpočetní náročností. První je RoadNet-RT [19], který zvládl jeden obraz na KITTI benchmarku na neoznačených silnicích

za 8 ms. RoadNet-RT je založen na architektuře enkodér-dekodér za použití pyramidových enkódovacích bloků pro enkódování kontextových prvků s hloubkově dilatovanými konvolucemi ve všech fázích architektury.

Další rychlou metodou je LoDNN [20], který zvládl jeden obraz na KITTI benchmarku za 18 ms. Používá pouze Lidarová data. V první fázi kóduje základní rysy obrazu na který je poté aplikována plně konvoluční síť, která je navržena s cílem vysoké rychlosti zpracování.

V dnešní době je již velké množství praktických úloh řešeno pomocí konvolučních neuronových sítí. Konvoluční sítě umožňují velmi robustní řešení, které si dokáže poradit s širokou variací možných situací detekce silnic. Při pohledu na KITTI benchmark je převážná většina aplikací založena na konvolučních neuronových sítích, pouze s rozdíly v implementaci dané architektury. Hlavním problémem konvolučních neuronových sítí je vysoká výpočetní náročnost, díky nutnosti zpracování velkého množství obrazových dat, která zamezuje aplikovatelnosti jinak funkčních návrhů. Nicméně v dnešní době již existují dostatečně rychlé algoritmy pro praktické využití. Přesto, pro účely, kdy je nutná velmi malá výpočetní náročnost, lze použít klasické metody detekce cest.

1.3.3 Klasické metody detekce cest

Klasické metody detekce cest se snaží pomocí geometrických modelů v prostoru najít nejlepší příznaky pro určení cesty. Modely mohou být založeny na základních metodách segmentace obrazu, jak je napsáno v podkapitole 1.2, nebo využívají matematických operací na obraze.

Zaměříme se hlavně na detekci neoznačených cest. Prvním úspěšným pokusem o detekci neoznačených cest byl algortimus "General road detection from single image"[21]. Navržený algoritmus se skládá ze dvou kroků. Nejprve se provede odhad bodu úběžníku, neboli bodu ve kterém se setkávají horizont a body okrajů cesty způsobené perspektivou. Následuje segmentace oblasti silnice založená na bodu úběžníku. Hlavním limitem této metody je vysoká výpočetní náročnost pro výpočet bodu úběžníku. Proto bylo na KITTI benchmarku dosaženo detekce jednoho snímku neoznačené cesty za velmi pomalých 63,56 s [22].

Dalším návrhem byla "Fast vanishing-point detection"[23], kde je navržena optimální lokální dominantní orientace používající čtyři Gaborovy filtry a adaptivní vzdálenostní hlasování pro odhad úběžníku.

Dalším návrhem byl "Rapid vanishing-point estimation"[25], kde je využito rozšíření Gaborových vlnek do lineární kombinace podobné Harrově funkci pro rychlé filtrování a použití superpixelů v systému hlasování pro zrychlení procesu.

Obr. 1.8: Ukázka detekce silnice pomocí bodu úběžníku [21]

Některé modely se úplně vyhýbají použití odhadu úběžníku a přímo z obrazu rozpoznávají pixely cesty přímo z pozadí za použití modelů výskytu (appearance models) [26].

Další model je založen na vícebarevném histogramu pro zachycení variability povrchu vozovky a jednobarevném histogramu pro model pozadí v prostoru RGB [27].

Další metoda [28] je založena na Gaussově smíšeném model (GMM), který využívá UV záření, normalizované červené a zelené světlo, a osvětlení pozadí. Tenhle barevný model byl trénován pouze offline, takže měl problémy s reálnými cestami.

Model neměnnosti osvětlení [29] využívá normalizovaný histogram okolního prostředí pro sadu zrnek pixelů (seed pixel) na spodní části obrazu za použití neměnného osvětlení prostoru.

Podobně jako předchozí metoda vybírá [30] spodní část obrazu pro modelování silnice a pozadí kombinující doplňkové barevné prvky, jako textura RGB, Lab a SILTP.

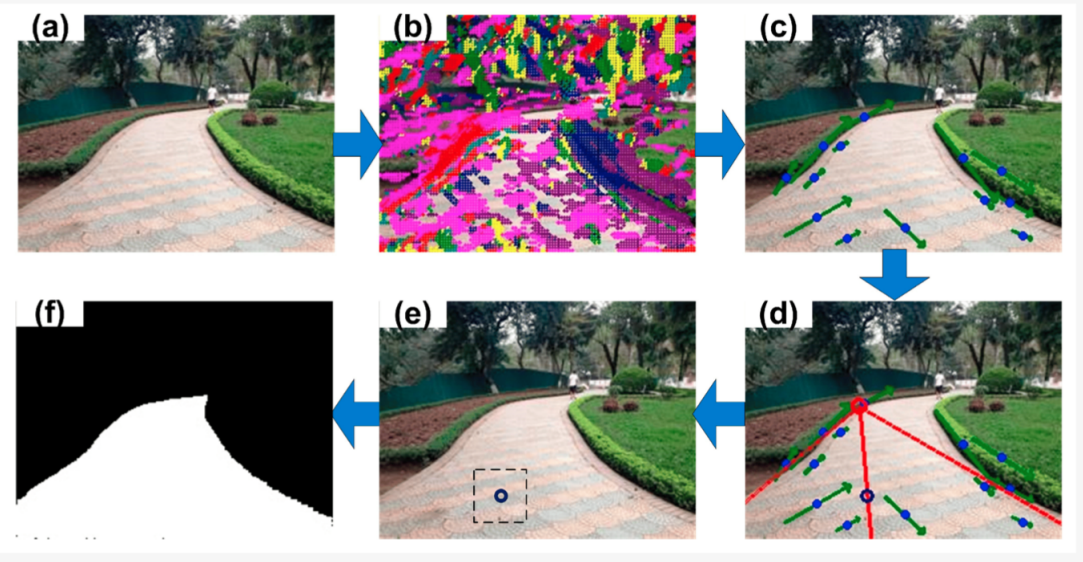
Další metoda [31] využívá lichoběžník v dolní částí obrazu a ve středu obrazu jako počáteční oblast a poté se upřesňuje pomocí úběžníku. Na KITTI benchmarku tato metoda rychlostně absolvovala jeden snímek neoznačené cesty za 2,93 s [22].

Pro zlepšení přesnosti a robustnosti se začaly objevovat hybridní metody. V [26] byl poprvé nalezen úběžník podle geometrie a barvy cesty. Poté byl využit model kombinující hrany, tvar a barvy pro nalezení cesty. Dále [32] uvedl geografické informace kontextové nápovědy jako horizontální čára, úběžník, geometrie cesty a dopravní značení pro detekci cest v rozdílných podmínkách.

Další metoda [33] se snažila snížit výpočetní náročnost. Pro odhad cesty v komplexním a neznámém prostředí byla vyvinuta samokontrolovaná online samoučící se architektura založená na inkrementálním neparametrickém bayesovském shluko-

vání. Tato metoda má tři fáze: superpixelové značení s inerciální měřící jednotkou a kolovým enkodérem dat, inkrementální clusterové učení a klasifikace průchodných oblastí s k-nejbližším okolím. Problém všech předchozích metod je stále vysoká výpočetní náročnost.

Další metoda [22] pro detekci průchodné oblasti cesty z jednoho obrazu v reálném čase bez závislosti na hardwaru je rychlý a robustní gaussovský model kombinující RGB, osvětlení neměnného prostoru a lokální binární vzor. Tento model je získán pomocí vzorkování regionu vycházejícího z limitů stanovených úběžníkem a hranicemi cesty. Rychlost této metody byla otestována na KITTI benchmarku, kde dokázala detekovat neoznačené silnice za 28 ms, což je nejrychlejší výsledek mezi všemi klasickými metodami. Tato metoda je schopna pracovat rychleji než 30 fps na standardním CPU.



Obr. 1.9: Postup při detekci prostupné oblasti: (a) vstupní obraz (b) odhad orientace textury pomocí Gaborova filtru (c) spojení dominantních pixelů se stejnou orientací (d) odhad úběžníku pomocí hlasování pro úsečky a zvolení pixelu jádra (modrý pixel) (e) vybrání vzorkovacího plochy v okolí pixelu jádra a vytvoření modelu výskytu (f) klasifikace obrazu do propustné a nepropustné oblasti [22]

Hlavní výhodou geometrických modelů je jejich nízká výpočetní náročnost, proto jsou především vhodné pro aplikace, které vyžadují omezené výpočetní zdroje. Problémem však je u klasických metod převážně jejich robustnost, protože je nutné ručně navrhnout matematické modely, matematické funkce nebo příznaky.

2 Praktická část práce

Praktická část práce začne výběrem vhodných testovacích dat. Poté bude následovat výběr programovacího jazyka a vhodných knihoven. Dále se bude zabývat zprovozněním vlastního algoritmu pro detekci cest a otestováním a porovnáním výsledků oproti již existujícím volně přístupným řešením.

2.1 Testovací data

Testovací data slouží k ověření správné funkčnosti naprogramovaného algoritmu. Správná volba testovacích dat je velmi důležitá pro ověření správnosti detekce cest, neboli robustnosti algoritmu, v různorodých situacích, které mohou nastat při reálném použití algoritmu. Proto se nyní podíváme na některé prvky, které by měl správný dataset obsahovat.

Rozmanitost prostředí - Je důležité zajistit velkou variabilitu prostředí a podmínek. Dataset často obsahuje městské silnice, venkovské cesty nebo i polní cesty. Dále se také testují různé povětrnostní podmínky, jako jsou suché, mokré silnice nebo silnice se sněhem.

Světelné podmínky - Je nutné vyzkoušet při různých světelných podmínkách v průběhu dne, jako je denní světlo, šero a noční podmínky. Důležité je otestovat vliv náhodných stínů z okolí, které kazí jednoduchou detekci, díky změně jasu silnice při snímání. Při noční viditelnosti může docházet k horšímu barevnému kontrastu okolí a silnice.

Různé typy cest - Je nutné zajistit různé typy cest, jako jsou jednosměrné a dvousměrné cesty, křižovatky nebo kruhové oblouky.

Různé typy objektů - Je důležité připravit algoritmus pro setkání s různými objekty, které by mohly narušit detekci cesty. Nejčastěji sem patří automobily, motocykly, nákladní auta, cyklisté, ale také objekty mimo cestu, jako například chodci.

Různé scénáře - Pokud již chceme mít pokročilejší algoritmus, je nutné zahrnout různé dopravní situace, jako jsou změny pruhů, odbočování, zastavení, řazení do fronty a další běžné situace v provozu.

Kontrola správnosti detekce - Zpětně je nutné kontrolovat, jak správně si algoritmus poradil při detekci cesty.

Pro náš dataset bylo použito několik krátkých videí, délky do dvou minut. Dataset obsahuje videa z jízdy autem při různých světelných podmínkách za denních a nočních podmínek, různé typy cest, obsahuje také nespočet objektů v okolí a různé scénáře. Dále dataset obsahuje parkové cesty, kde se testují různé povrchy cest, jako jsou kostky, štěrk a lesní cesty. Parkové cesty také obsahují různé povětrnostní podmínky, kde jedna cesta obsahuje kaluže. Některá videa jízdy autem byla převzata

z Brno Urban Dataset [34].

Obr. 2.1: Ukázka z datasetu videí. První řádek zleva: (1) Denní jízda autem se stíny na vozovce (2) Noční jízda autem (3) Denní jízda autem v zimě. Druhý řádek zleva: (4) Parková cesta s kalužemi (5) Parková cesta s kostkami (6) Lesní cesta

2.2 Výběr programovacího jazyka

Tato podkapitola se bude zabývat výběrem programovacího jazyka pro detekci cest. Při výběru programovacího jazyka hrají roli mnoho parametrů.

Výkon - Asi nejčastěji vyžadovaným parametrem, jelikož nám ovlivňuje, kolik snímků jsme schopni zpracovat za sekundu, což také závisí na použitém hardwaru.

Dostupnost knihoven - Protože většinou nevyvíjíme aplikaci úplně od začátku je velmi vhodné vybrat programovací jazyk, který má dostupné knihovny pro ulehčení naší práce.

Jednoduchost programování - V případě, že upřednostňujeme rychlý vývoj nebo jednoduchý programovací jazyk je ideální zvolit jazyk s jednoduchou syntaxí.

Komunita - Pokud vybereme programovací jazyk s aktivní komunitou, je jednodušší řešit potenciální problémy, které již někdo před námi vyřešil nebo nám případně může někdo poradit při nevyřešených problémech.

Nyní se podíváme na nejčastěji používané programovací jazyky pro detekci cest.

2.2.1 Python

Python je vysokoúrovňový programovací jazyk, který v roce 1991 navrhl Guido van Rossum. Podporuje široké spektrum programovacích paradigmat, včetně objektově orientovaného, funkcionálního a imperativního. Python je vyvíjen jako open source

projekt nabízející podporu pro všechny běžné platformy. V současné době je jeden z nejuniverzálnějších a nejpoužívanějších jazyků na trhu [35].

Jednou z největších výhod Pythonu je jeho velmi jednoduchá syntaxe, která umožňuje rychlé psaní, dobrou čitelnost a snadnou údržbu kódu, a proto je vhodný i pro začátečníky.

Další velkou výhodou je velmi široká podpora a aktivní komunita, díky velkému počtu uživatelů. V oblasti počítačového vidění a obecně v oblasti umělé inteligence existuje velké množství různých knihoven, frameworků a nástrojů, které jsou k dispozici a usnadňují práci vývojářům. Nejčastěji používané knihovny v oblasti umělé inteligence a strojového učení jsou TensorFlow a PyTorch. Pro počítačové vidění je možné využít knihovnu OpenCV.

Hlavní nevýhodou Pythonu je jeho výkon, protože Python je interpretovaný jazyk, což obecně znamená, že je méně efektivní než kompilované jazyky. Pro aplikace, které vyžadují vysoký výkon v reálném čase, nemusí být vhodný.

I přes nevýhodu ve výkonu převažují kladné vlastnosti Pythonu pro práci v aplikacích počítačového vidění, proto je v dnešní době nejrozšířenějším jazykem pro detekci cest a v KITTI benchmarku převažují řešení používající Python. Díky tomu byl vybrán jako jazyk pro návrh algoritmu detekce cest.

2.2.2 C++

C++ je vysokoúrovňový programovací jazyk, který vyvinul Bjarne Stroustrup jako rozšíření jazyka C. Podporuje několik paradigmat, včetně objektově orientovaného, generického a procedurálního. V současné době je jedním z nejrozšířenějších programovacích jazyků [36].

Hlavní výhodou používání C++ je, že se jedná o kompilovaný jazyk s nízkou úrovní abstrakce, což umožňuje dosáhnout vysokého výkonu. Je tedy vhodný pro aplikace, které vyžadují rychlé zpracování obrazu v reálném čase.

Dále umožňuje kontrolu nad pamětí, což je důležité pro aplikace s omezenými paměťovými prostředky, kde je zapotřebí přesně kontrolovat využití paměti.

Podobně jako u Pythonu má C++ vysoké množství knihoven pro počítačové vidění, jako například OpenCV. C++ je také velmi používaný jazyk, a proto má velmi dobrou podporu.

C++ je velmi často používán pro programování vestavných systému. Pokud tedy specifická aplikace vyžaduje vestavěné systémy, je C++ vhodným programovacím jazykem.

Nejčastějším problémem je složitá syntaxe, která vyžaduje větší programátorské dovednosti a zpomaluje vývoj programu.

2.3 Použité knihovny

Knihovny v programování se používají jako soubory předem napsaného kódu, které mohou programátoři znovu použít pro svůj vlastní program. Knihovny obsahují funkce, třídy nebo datové struktury, které usnadňují vývoj nového softwaru, protože umožňují využití již hotových a funkčních částí specifického kódu, aby nemusel programátor programovat celý software od začátku.

Programovací jazyk obsahuje standardní knihovny pro základní funkce jazyka. Dále se využívají knihovny třetích stran, které jsou vytvořeny nezávislými vývojáři poskytující rozšíření jazyka a specifické funkce pro konkrétní použití. Také se používají interní knihovny uvnitř organizací, které nejsou přístupné veřejnosti a využívají se pro interní projekty.

2.3.1 OpenCV

OpenCV, neboli Open Source Computer Vision library, je open-source knihovna určená pro počítačové vidění a zpracování obrazu v reálném čase. Byla vyvinuta společností Intel a poprvé vyšla pro veřejnost v roce 2000. Od té doby se stala klíčovým nástrojem pro výzkum a vývoj v oblasti počítačového vidění [37].

Knihovna obsahuje velké množství algoritmů zaměřených na práci s obrazy a videi. Knihovna se hlavně zaměřuje na úkoly související s počítačovým viděním, jako je rozpoznávání obličejů, sledování pohybu, detekce objektů, segmentace obrazu a mnoho dalších. To umožňuje hojné využití v aplikacích praktického využití.

OpenCV je napsána v jazyce C++, což jí umožňuje rychlé vykonávání kódu. Knihovna je však multiplatformní, takže umožňuje využití i pro programovací jazyky mimo C++, jako je Python, Java a Matlab, což usnadňuje použití pro řadu projektů.

OpenCV také podporuje velké množství operačních systémů jako je Windows, Linux, macOS, iOS, Android a mnoho dalších. Dále také podporuje techniky strojového učení a hlubokého učení.

Míra používání knihovny OpenCV je dále způsobena jednoduchým použitím v programu, dobře zdokumentovaným rozhraním a širokým množstvím učebních prostředků pro naučení se práce s knihovnou.

2.3.2 NumPy

NumPy je knihovna pro programovací jazyk Python, která se specializuje na práci s multidimenzionálními poli a maticemi. Dále také obsahuje širokou sbírku matematických funkcí vyšší úrovně pro práci s poli a maticemi [38]. Použití matic je velmi důležité při práci s obrazem, protože velikost intenzity pixelů obrazu lze reprezentovat pomocí matice.

2.4 Programování vlastního řešení

V této podkapitole se budeme zabývat návrhem a programováním vlastního řešení pro rozpoznání cesty. Jak již bylo zmíněno v předchozích podkapitolách, program je napsán v programovacím jazyce Python a využívá knihoven OpenCV a NumPy.

Nyní se podíváme na jednotlivé kroky nutné k rozpoznání cesty. Nejprve převádíme obraz pomocí Cannyho hranového detektoru na obraz s hranami. Na tento obraz s hranami aplikujeme masku, která slouží k filtraci prostoru, který nenese žádné důležité informace. Poté pomocí Houghovy transformace určujeme kontinuální čáry v obraze a zbavujeme se čar hran, které nejsou dostatečně dlouhé. Poté filtrujeme horizontální čáry, protože víme, že potřebujeme čáry pouze s určitým sklonem. Všechny zbylé čáry rozdělíme na čáry pravé a levé strany, a z nich vytváříme průměrnou čáru pro levou a pravou stranu, které označují okraj silnice. Výsledná čára a maska jsou zobrazeny přes původní obraz a dostáváme výsledek.

Jelikož předchozí odstavec je pouze jednoduchým shrnutím funkčnosti algoritmu, podíváme se nyní na podrobnější popis všech kroků a mezivýsledků obrazu.

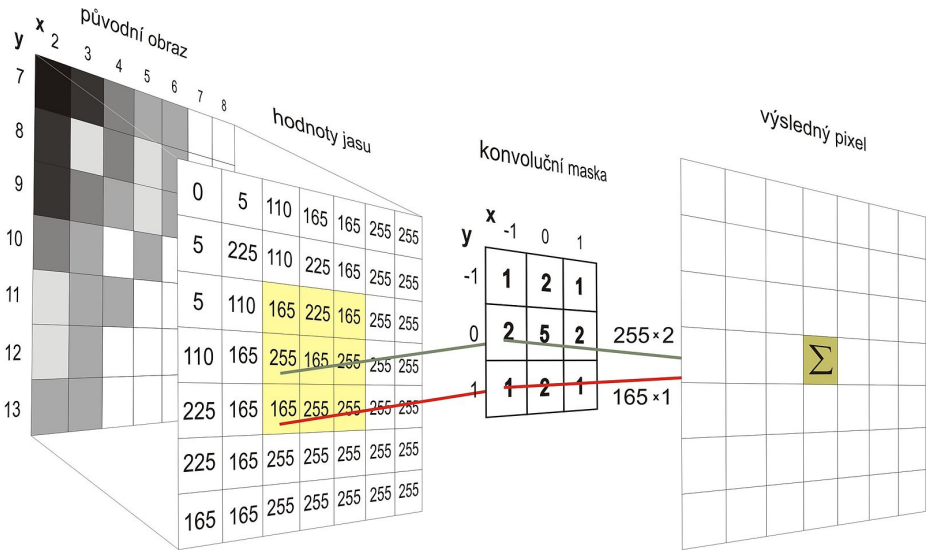
Pro lepší ilustraci funkčnosti jednotlivých kroků využijeme testovacího obrazu, na kterém budeme provádět všechny operace.

Obr. 2.2: Testovací obraz

2.4.1 Předzpracování obrazu

Na začátku programu převádíme obraz z tříbarevného modelu RGB do šedotónového spektra, protože nepotřebujeme pracovat s třemi složkami barvy. Převodem obrazu do šedotónového spektra redukujeme každý pixel obrazu na jedinou hodnotu intenzity šedé barvy, oproti třem potřebným hodnotám intenzity jasu v RGB modelu. Díky tomu zrychlíme celý algoritmus zpracování obrazu, protože máme třetinovou matici oproti původnímu obrazu při každé další operaci s obrazem. Díky této redukci velikosti matice tedy snižujeme množství zpracovaných dat o třetinu.

Po převodu na šedotónový obraz je obraz částečně rozostřen pomocí Gaussovského filtru. Operaci rozostření používáme, abychom se částečně zbavili nahodilého šumu na obraze, který by mohl poškodit náš obraz a další operace s ním. Pro rozostření používáme princip konvoluce. Konvoluce je jedna ze základních operací na obraze. Princip konvoluce spočívá v tom, že máme definované jádro, možné nazývat maskou, které na každý pixel v obraze aplikuje matematickou operaci. Jádro je nutné před použitím převrátit, ale u symetrického jádra dostáváme stejný výsledek i bez převrácení. Při výsledné konvoluci pixelu má vliv na výslednou hodnotu intenzity jasu pixelu okolní jasové hodnoty a intenzita jasu pixelu samotného, a to v závislosti na velikosti jádra a jeho vnitřních hodnotách.



Obr. 2.3: Princip konvoluce (maska je již převrácena)[39]

V našem provedení používáme Gaussův filtr, neboli jádro, které má hodnoty rozloženy podle Gaussovy funkce. V programu bylo použito Gaussovo jádro o velikosti 5x5 [40].

Maska Gaussova filtru 5x5:

$$h = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \tag{2.1}$$

Obr. 2.4: Obraz po předzpracování

Při rozostření obrazu je nutné počítat s tím, že dochází k částečnému zhoršení kvality obrazu a částečnému vyhlazení ostrých hran. Z výsledného obrazu lze ale vidět, že obraz je stále dostatečné kvality pro následnou detekci hran.

2.4.2 Cannyho hranový detektor

Cannyho hranový detektor je vícekrokový algoritmus pro detekci hran v obraze. Byl vyvinut v roce 1986 Johnem F. Cannym a dodnes je hojně používaný algoritmus, díky vysoké přesnosti detekce a schopnosti eliminovat falešné detekce [41].

Princip algoritmu lze rozdělit na několik kroků:

1. Potlačení šumu
2. Výpočet gradientu intenzity
3. Potlačení nemaximálních hodnot
4. Prahování s hysterezí

V první fázi dochází k převedení původního obrazu na obraz šedotónový a aplikuje se Gaussovský filtr pro redukci šumu. Tento krok již byl proveden v 2.4.1.

Ve druhé fázi dochází k výpočtu gradientu intenzity. Gradient obrazu je směrová změna intenzity nebo barvy v obraze. Jednoduše řečeno, gradient měří velikost změny intenzity pixelu v daném směru. Odhaduje směr a velikost změny intenzity. Díky tomu se gradient používá pro detekci hran v obraze. Prakticky se pro výpočet gradientu používá konvoluce pomocí Sobelova jádra ve směru horizontálním a vertikálním. Tím získáváme první derivaci v obou směrech.

Sobelova maska 3x3:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.2}$$

Pomocí G_x a G_y jsme schopni díky základní znalosti trigonometrie vypočítat velikost gradientu a jeho směr, protože G_x a G_y tvoří odvěsny pravoúhlého trojúhelníku s přeponou G.

$$G = \sqrt{G_x^2 + G_y^2}, \varphi = tan^{-1}(\frac{G_y}{G_x}) \tag{2.3}$$

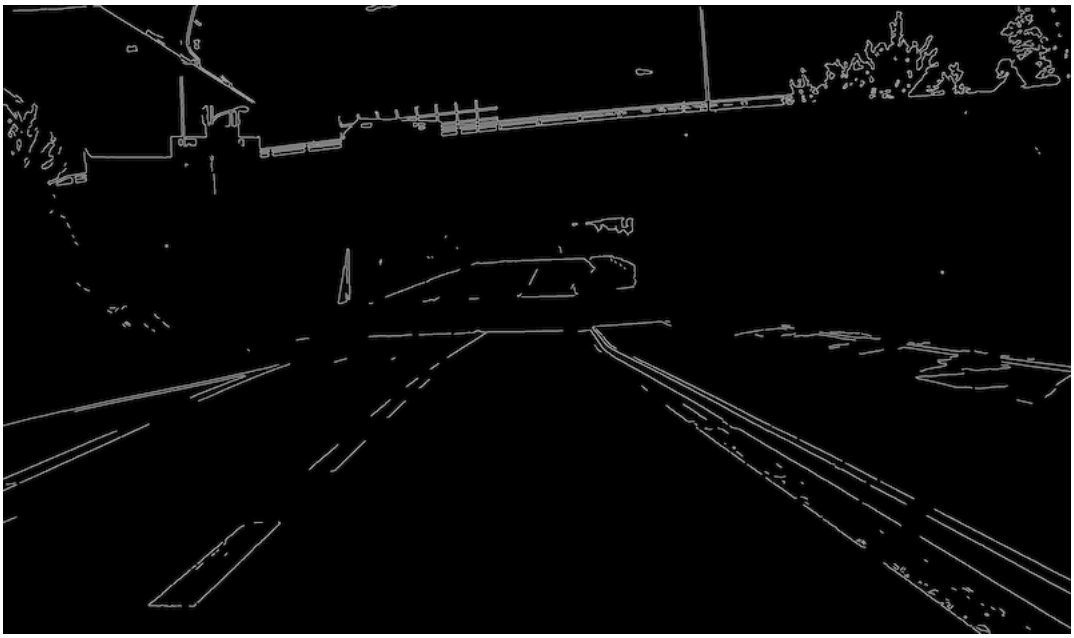
Třetí fází je potlačení nemaximálních hodnot. V této fázi se prochází celý obraz a potlačují se nepodstatné hrany, aby v obraze zůstaly pouze ostré a jasné hrany, které jsou jasně oddělené od ostatních částí okolí. Každý pixel v obraze je kontrolován, zda je pixel lokálním maximem v okolí pixelu ve směru gradientu. Obraz se tedy prochází ve směru gradientu a porovnává se hodnota aktuálního pixelu s okolím. Pokud je intenzita aktuálního pixelu vyšší než okolí, je ponechána jako hrana, jinak je potlačena snížením na nulu.

Poslední fází je prahování s hysterezí, kde se rozhoduje, které hrany budou považovány za skutečné hrany a které ne. Proto jsou potřeba dvě hranice prahu, minimální a maximální. Všechny hrany s intenzitou gradientu vyšší než je maximální hodnota jsou považovány za hrany. Dále všechny hrany s hodnotou nižší než je minimum nejsou považovány za hrany. Hodnoty mezi minimem a maximem jsou považovány za hrany v případech, kdy jsou propojeny se silnou hranou.

Po vykonání všech kroků Cannyho hranového detektoru získáváme obraz s jasně definovanými hranami.

2.4.3 Maska oblasti zájmu

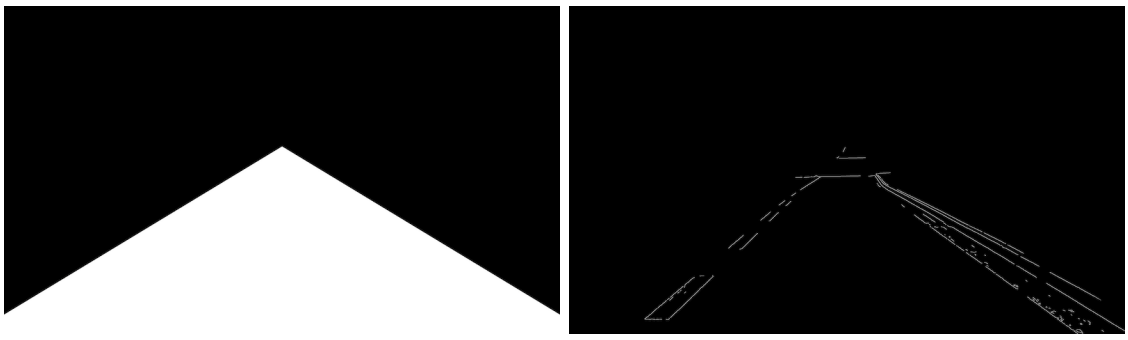
V další fázi je nutné si definovat oblast zájmu. Oblast zájmu je část obrazu, kde očekáváme, že se bude nacházet námi hledaná cesta. Tento krok je nutný, jelikož



Obr. 2.5: Obraz s hranami

v obrazu s hranami dostáváme velké množství hran, které nejsou nijak použitelné, protože nenesou informaci o hranici cesty, a proto je odstraníme z obrazu.

Obraz, který dostáváme po hranovém detektoru, je datového typu uint8 a obsahuje hodnoty 255 pro hrany a 0 pro zbytek obrazu. Tuto vlastnost využíváme při návrhu masky. Masku si definujeme jako nepravidelný pětiúhelník začínající u kamery vozidla s vrcholem přibližně ve středu obrazu. Vnitřek pětiúhelníku vyplníme hodnotami 255 a mimo pětiúhelník necháme hodnoty na 0. Nyní provedeme operaci bitového ANDu na dvou získaných obrazech pro získání obrazu pouze s oblastí zájmu.



Obr. 2.6: (1) Pětiúhelníková maska (2) Obraz po aplikaci masky

2.4.4 Houghova transformace

Houghova transformace je technika extrakce příznaků v oblasti počítačového vidění a zpracování obrazu. Houghova transformace, která je dnes používána, byla vynalezena Richardem Dudou a Peterem Hartem navazující na patent Paula Hougha z roku 1962. Pro nás je důležitá klasická Houghova transformace, která se využívá pro detekci čar v obraze.

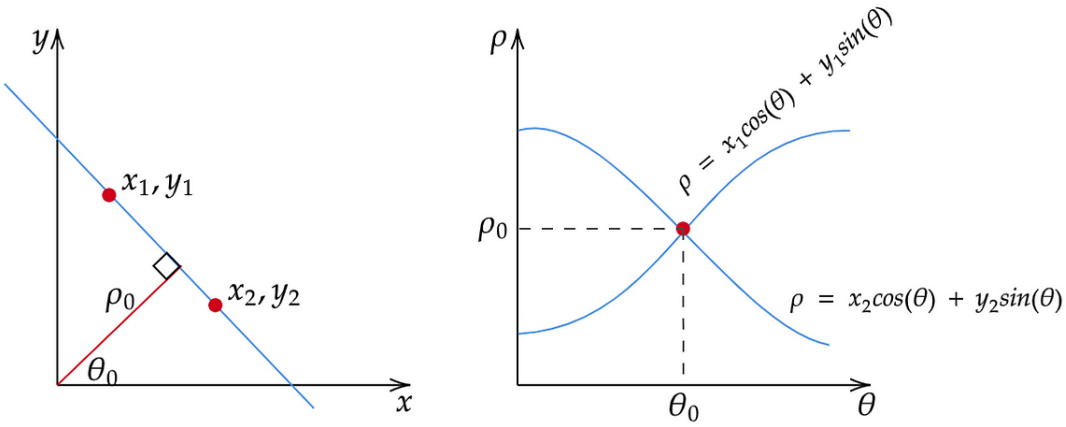
Obecně lze jakoukoliv přímku popsat pomocí směrnice přímky, která může být reprezentována v parametrickém prostoru pomocí jednoho bodu. U směrnice ale nastává problém u vertikálních čar, které nejsou ohraničené, a proto se využívá normální parametrizace. Normální parametrizace popisuje přímku pomocí úhlu θ a vzdálenosti od středu souřadného systému r (nebo také ρ). Po úpravě základního tvaru dostáváme:

$$r = xcos\theta + ysin\theta \tag{2.4}$$

Pro každý bod x_0, y_0 lze najít množinu přímek procházejících tímto bodem, definovanou v parametrickém prostoru pomocí sinusoidy jako:

$$r = x_0cos\theta + y_0sin\theta \tag{2.5}$$

Nyní, pokud máme množství bodů, jsme schopni zjistit přímku, pokud body mohou tvořit přímku. Dochází k protnutí sinusoid v určitém bodě, který reprezentuje parametry přímky.



Obr. 2.7: Houghova rovina se dvěma body[42]

Díky tomu můžeme rozdělit Houghovu rovinu na menší části pomocí mřížky, a aplikovat princip hlasování. Hlasování funguje tak, že každé protnutí bodu mřížky sinusoidou je zaznamenáno jako jeden hlas. My si definujeme minimální počet hlasů. Tento minimální počet hlasů představuje námi definovanou přímku v obraze [43][44].

Obr. 2.8: Obraz s nalezenými čárami po Houghově transformaci

2.4.5 Filtrace horizontálních čar

Z předchozího obrazu je zřejmé, že se v obraze vytvářejí i horizontální čáry, které nevypovídají o hranici cesty. Tyto čáry byly sice správně vytvořeny z hranového obrazu, který jsme poskytli pro Houghovu transformaci, ale víme, že jsou pro nás nepodstatné, proto je budeme filtrovat.

Pro filtraci využijeme znalosti sklonu přímky. U každé čáry, známe počáteční a koncové body, což nám umožňuje vypočítat sklon přímky.

Sklon přímky:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.6}$$

m - sklon přímky, (x_1,y_1) - souřadnice počátečního bodu, (x_2,y_2) - souřadnice koncového bodu

Nyní mohou nastat různé situace. Dostaneme-li kladný sklon $m > 0$, dostáváme rostoucí přímku, která ohraničuje levou stranu cesty. Dostaneme-li záporný sklon $m < 0$, dostáváme klesající přímku, která ohraničuje pravou stranu cesty. Pro nás

jsou v tento moment důležité přímky se sklonem $m = 0$, které označují horizontální čáry. Z praktického hlediska je nutné vytvořit určitý přechod mezi horizontálními čárami a čárami s jiným sklonem, jelikož nedostáváme úplně přesný sklon pro horizontální čáry. V našem případě jsme zvolili interval pro horizontální čáry jako $m \in (-0,2; 0.2)$. Posledním typem přímky, kterou můžeme dostat, je přímka s nekonečným sklonem, která označuje vertikální přímku. Vertikální přímky nijak nevyužíváme.

Díky znalosti sklonu všech přímek můžeme vyfiltrovat všechny horizontální čáry v obraze a pokračovat v dalším kroku.

Obr. 2.9: Obraz bez horizontálních čar

2.4.6 Průměrování čar

Po filtraci horizontálních čar vytvoříme ze všech získaných čar pouze dvě čáry, které ohraničují cestu. Využijeme znalosti sklonu přímky. Jak již bylo zmíněno, pokud máme kladný sklon $m > 0$, můžeme zařadit tyto čáry jako čáry levého okraje cesty, a naopak pokud máme záporný sklon $m < 0$, zařadíme tyto čáry jako čáry pravého okraje cesty.

Dále zprůměrujeme samostatně čáry levé strany a čáry pravé strany. Díky tomu dostáváme dvě přímky, které vykreslíme od počátku do přibližně středu obrazu. Tyto dvě vykreslené čáry nám tvoří ohraničení cesty a jsou na výsledném obrazu vybarveny červeně.

Krajní body obou přímek tvoří rohy lichoběžníku, který vykreslíme na výsledný obraz a dostáváme zobrazení s rozpoznanou silnicí na obraze. Nyní dostáváme výsledný obraz po všech provedených operacích.

Obr. 2.10: Výsledný obraz

2.5 Výsledky testování algoritmu

V této podkapitole se podíváme na výsledky testování algoritmu na dostupném datasetu, kde otestujeme robustnost algoritmu pro různé typy cest, světelné podmínky a rozmanitost prostředí. Zde ověříme vhodnost použití algoritmu. Dále se podíváme na výpočetní náročnost algoritmu, která je důležitá pro možnost praktického použití algoritmu v aplikacích reálného času.

2.5.1 Robustnost

Robustnost programu nám určuje schopnost algoritmu odolat vlivům, které by mohly poškodit kvalitu výsledného obrazu s označením silnice.

Námi navržený algoritmus má určité vlastnosti, které je nutné brát v potaz. Důležité je, že pro detekci silnice používáme hranový detektor a průměrování čar. To v praxi znamená, že je velmi důležité vhodné označení okrajů silnice pro úspěšný výsledek algoritmu. Bez výrazných okrajů nejsme schopni najít okraj silnice a následně vytvořit masku silnice. Proto je algoritmus velmi vhodný pro dobře označené

silnice s jasnými okraji vyznačenými čárami nebo výrazným okrajem. To samozřejmě ale také znamená, že program není vhodný pro silnice s nevýrazným nebo neoznačeným okrajem.

Světelné podmínky v programu hrají částečnou roli, ale při dobře označeném okraji je algoritmus schopný si poradit. Zhoršené světelné podmínky nám snižují jas obrazu v případě stínů nebo noční jízdy. Snížený jas vytváří menší rozdíly mezi prahy na obraze, což vede k menšímu množství hran v obraze. Tento problém je ale možné vyřešit v případě, že algoritmus je navržen přímo do nočního prostředí, protože jsme schopni snížit hranice minimálního a maximálního prahu u Cannyho hranového detektoru.

Pokud se zaměříme na různé typy cest, algoritmus má problémy, pokud se nejedná o jízdu rovně. Algoritmus není schopný si poradit s ostrými zatáčkami, protože nemají jasně definované okraje silnice. Algoritmus je navržen pro jízdu rovně nebo pro mírné zatáčky.

Kde algoritmus selhává, jsou parkové cesty. Parkové cesty obsahují velké množství hran v obraze, které označují prvky povrchu cesty, a jsou v obraze téměř nahodilé. Dále vzniká problém detekovat okraj cesty. Náš typ algoritmu není vhodný pro detekci parkových cest.

Algoritmus byl také testován za různých povětrnostních podmínek. Byly použity záběry s napadlým sněhem. Sníh by tvořil problémy programu v případě, kdyby celá silnice byla pokrytá sněhem. V našem případě jsme ale testovali, kdy sníh byl už pouze mimo průjezdné části silnice. S touto situací byl algoritmus schopný si poradit velmi dobře, jelikož sněhová pokrývka často tvořila dobrý okraj silnic pro detekci cesty. Částečně docházelo také k zhoršení detekce, jelikož se ve sněhu nacházely mírné hrany, ale detekce nebyla zásadně poškozena.

Obr. 2.11: Úspěšná detekce. Zleva: (1) Jízda se sněhem na silnici (2) Noční jízda s jasným okrajem silnice (3) Jízda do mírné zatáčky s dobře označeným okrajem

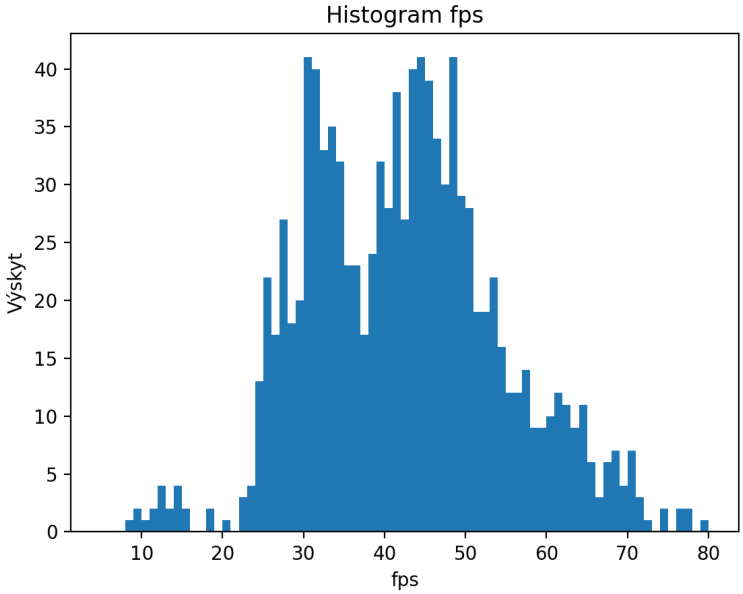
Obr. 2.12: Neúspěšná detekce. Zleva: (1) Jízda s nedostatečným okrajem na pravé straně (2) Parková cesta s příliš mnoha hranami v obraze (3) Jízda do ostré zatáčky s nedostatečným okrajem

2.5.2 Výpočetní náročnost

Výpočetní náročnost určuje množství výpočetní techniky potřebné pro efektivní vykonání algoritmu.

V našem případě je vhodné otestovat rychlost zpracování snímků za sekundu na specifickém hardwaru. Jako testovací hardware byl zvolen notebook Macbook Air ze série roku 2020 s procesorem CPU 1,1 GHz Dual-Core Intel Core i3 a grafickou kartu GPU Intel Iris Plus Graphics 1536 MB. A testovací video je s rozlišením 1920x1200.

Rychlost zpracování snímků se nazývá snímková frekvence, anglicky frame rate, která je v jednotkách fps, zkratka z anglického názvu frames per second. Jak již napovídá název, jedná se o jednotku určující počet zpracovaných snímků za sekundu.



Obr. 2.13: Histogram fps

Při testování výpočetní náročnosti nelze přesně určit hodnotu fps, protože samotná hodnota je ovlivněna mnoha faktory. Při testování na příslušných datech docházelo ke kolísání hodnoty fps, která je především způsobena rozdílnou komplexností každého snímku pro zpracování. Průměrná hodnota naměřená na videích z datasetu byla 40 fps.

Rychlost 40 fps je již vhodná pro praktické aplikace v reálném čase. Je však nutné dát pozor na hardwarové parametry, které se v praktických aplikacích budou lišit od parametrů testovaného zařízení.

2.5.3 Testovaní na KITTI benchmarku

Proto, abychom mohli přesně otestovat vlastnosti algoritmu, využijeme dataset KITTI benchmarku, na kterém porovnáme rychlost a robustnost algoritmu s nejlepšími existujícími pracemi.

Pro testování jsme zvolili část datasetu obsahující označené silnice, která obsahuje 95 obrazů. Označené silnice byly zvoleny z důvodu lepší schopnosti algoritmu správně rozpoznat silnice s jasně vyznačeným okrajem a proto představují vhodnější výsledky pro srovnání rychlosti a robustnosti.

Pokud se pokusíme srovnat robustnost narazíme na problémy spojené s hodnocením správného rozpoznání. KITTI benchmark testuje procentuální úspěšnost rozpoznání silnice po jednotlivých pixelech. V tomto ohledu náš algoritmus není schopný konkurovat řešením založeným na konvolučních neuronových sítích, které byly schopny dosáhnout F-skóre až v okolí 97 %. Proto jsme zvolili otestování pouze počtu správně rozpoznaných silnic z datasetu označených silnic, kde jsme dosáhli správného rozpoznání u 63 obrazů z 95. Lepší výsledek jsme nedosáhli z důvodu nízké kvality předloženého obrazu, která zhoršovala prahování, a kvůli častým stínům na silnici.

Rychlost jsme již ale schopni otestovat. Otestovali jsme čas vykonání programu od začátku do konce. Důležitou informací je, že obrazy mají rozlišení 1242x375. Pro získání reprezentativních výsledků jsme odstranili některé částečně nadbytečné prvky, jako je vypsání fps nebo zobrazení snímku po určitou dobu. Průchod všemi snímky byl algoritmus schopný vykonat za 1,14 s, což odpovídá 12 ms na jeden snímek. Nyní srovnáme s některými řešeními na KITTI benchmarku[11]. Zde je nutné brát v potaz, že ostatní testovali na odlišných zařízeních, takže výsledky by se lišily při stejném hardwaru. Specifikace našeho zařízení jsou uvedeny v předchozí podsekci 2.5.2.

V tabulce 2.1 lze vidět, že náš algoritmus, nazvaný metoda hran, patří mezi jedny z nejrychlejších dostupných metod. Dále je zřejmé, že při odstranění některých

nadbytečných funkcí a rozdílném rozlišení obrazu jsme schopni dosáhnout rychlosti 83 fps.

Tab. 2.1: Srovnání rychlosti doby běhu programu

Metoda	Doba běhu programu [ms]	Testované na zařízení (prog. jazyk)
Hran	12	MacBook Air 2020 (Python)
SNE-RoadSeg+	80	GPU na 2.5 Ghz (Python)
RoadFormer	40	GPU na 2.5 Ghz (Python)
RoadNet-RT	8	GPU na 2.5 Ghz (Python)
LoDNN	18	GPU na 2.5 Ghz (Python)
Klasická	28	Intel i5-3230 CPU (C++)

Závěr

Tato bakalářská práce se zabývala tématem segmentace cest pro autonomní jízdu. Cílem práce bylo vytvořit funkční algoritmus pro detekci silnic a cest s kamerou ve směru jízdy vozidla.

V teoretické části byly představeny základní principy zpracování obrazu a segmentace obrazu. Následně bylo zjištěno, že v praxi se nejčastěji využívají dva typy metod detekce cest. První jsou metody konvolučních neuronových sítí, které umožňují konzistentní a robustní řešení s vysokou rychlostí, proto jsou v dnešní době nejčastěji využívané ve praktických aplikacích. Druhým typem jsou metody klasické, které využívají práci s obrazem pomocí geometrických modelů. Mohou být založeny na mnoha principech. V dnešní době jejich praktické použití upadá kvůli nižší robustnosti, ale přesto zůstává výhodou jejich nízká výpočetní náročnost pro některé specifické aplikace.

V praktické části jsme se přesunuli k praktické implementaci vlastního řešení. Nejprve jsme získali vhodný dataset pro testování různých světelných podmínek nebo různých typů cest. Následně byl zvolen Python jako vhodný programovací jazyk, díky jednoduchosti a vysoké podpoře knihoven. Důležitou roli při výběru hrála podpora knihovny OpenCV, která se věnuje aplikacím počítačového vidění a zpracování obrazu. Následně byl navržen algoritmus založený na principu Cannyho hranového detektoru s Houghovou transformací pro vytvoření okrajů masky detekce.

V poslední části se testovaly výsledky algoritmu. Robustnost algoritmu byla převážně určena typem cest. Algoritmus byl velmi schopný si poradit se silnicemi s výrazně označeným okrajem. Světelné podmínky částečně ovlivňovaly výsledky, kde částečně zhoršovaly detekci hran, proto některé stíny mohly poškodit detekci. Velké problémy měl algoritmus s parkovými cestami, kde algoritmus detekoval cestu pouze v případě výrazného okraje, jinak si ale nebyl schopný poradit, kvůli detekci hran na povrchu cesty. Dalším důležitým parametrem byla výpočetní náročnost. Na testovaném zařízení byl algoritmus schopný dosáhnout průměrné rychlosti 40 fps, kde ale hodnota kolísala v závislosti na komplexnosti obrazu. Rychlost 40 fps je vhodná rychlost pro praktické využití. Pro poslední srovnání byl využit KITTI benchmark, na kterém jsme otestovali rychlost a robustnost. Náš algoritmus byl schopný dosáhnout úspěšného rozpoznání silnice v 63 z 95 případů, což je hodnota výrazně nižší než u řešení založených na principu konvolučních neuronových sítí. Doba zpracování jednoho snímku byla 12 ms, což je vysoká rychlost, která dokáže překonat široké množství dostupných metod.

Literatura

[1] JANÁKOVÁ, Ilona. *Počítačové vidění - Úvod a motivace* [online]. In: [cit. 2023-11-13]. Dostupné z: http://vision.uamt.feec.vutbr.cz/?course=POV.

[2] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image Processing, analysis, and machine vision.* 4th ed. Stamford: Cengage Learning, 2015. ISBN 978-1-133-59360-7. [cit. 2023-10-20]

[3] SPRING, Kenneth R., Thomas J. FELLERS a Michael W. DAVIDSON. *Introduction to Charge-Coupled Devices (CCDs)* [online]. In: MicroscopyU [cit. 2024-05-10]. Dostupné z: https://www.microscopyu.com/digital-imaging/introduction-to-charge-coupled-devices-ccds.

[4] CANON. *Capturing the image CCD and CMOS sensors* [online]. In: Canon [cit. 2024-05-10]. Dostupné z: https://web.archive.org/web/20180428122601/http://cpn.canon-europe.com/content/education/infobank/capturing_the_image/ccd_and_cmos_sensors.do.

[5] STRAKA, Stanislav. *Segmentace obrazu* [online]. Brno, 2009, Diplomová práce. Masarykova univerzita. [cit. 2023-10-21]. Dostupné z: https://is.muni.cz/th/72784/fi_m/dp.pdf.

[6] EDITORIAL TEAM OF TOWARDS AI. Semantic Segmentation: A Complete Guide. In: *Towards AI* [online]. 2019 [cit. 2023-10-20]. Zveřejněno dne: 6. 10. 2021. Dostupné z: https://towardsai.net/p/l/machine-learning-7.

[7] Thresholding. In: *Skicit-image* [online]. [cit. 2023-10-21] Dostupné z: https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_thresholding.html.

[8] VIKRAM M. Comprehensive Guide to Edge Detection Algorithms: 9. 8. 2022. In: *Analytics Vidhya* [online]. c2013-2023 [cit. 2023-10-22]. Dostupné z: https://www.analyticsvidhya.com/blog/2022/08/comprehensive-guide-to-edge-detection-algorithms/.

[9] JIRSÍK, Václav. Strojové vidění - Segmentace. *Umělá inteligence* [interní materiály VUT předmětu BPC-UIN]. 2022. [cit. 2023-11-15].

[10] CHAKRABORTY, Debasish, Goutam SEN a Sugata HAZRA. Image segmentation Techniques. In: *ResearchGate* [online]. 2012 [cit. 2023-11-15]. Dostupné z:
https://www.researchgate.net/publication/236655552_Image_segmentation_Techniques?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlIjoiX2RpcmVjdCJ9fQ.

[11] Road/Lane Detection Evaluation 2013. *The KITTI Vision Benchmark Suite* [online]. 2012 [cit. 2023-11-27]. Dostupné z:
https://www.cvlibs.net/datasets/kitti/eval_road.php.

[12] RAWAT, Waseem a Zenghui WANG. *TDeep Convolutional Neural Networks for Image Classification: A Comprehensive Review* [online]. Department of Electrical and Mining Engineering, University of South Africa, Florida 1710, South Africa, 2017 [cit. 2023-11-27]. Dostupné z:
https://www.researchgate.net/profile/Zenghui-Wang-6/publication/317496930_Deep_Convolutional_Neural_Networks_for_Image_Classification_A_Comprehensive_Review/links/59f814630f7e9b553ebefe27/Deep-Convolutional-Neural-Networks-for-Image-Classification-A-Comprehensive-Review.pdf.

[13] ZIKOVÁ, Jana. *Klasifikace fotografií pomocí hlubokých neuronových sítí* [online]. [online]. Brno, 2018 [cit. 2023-12-22]. Dostupné z:
https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=181335. Bakalářská práce. Vysoké učení technické v Brně.

[14] SHELHAMER, Evan, Jonathan LONG a Trevor DARRELL. *Fully Convolutional Networks for Semantic Segmentation* [online]. 2015 [cit. 2023-12-22]. Dostupné z:
https://arxiv.org/pdf/1605.06211.pdf.

[15] RONNENBERGER, Olaf, Philipp FISCHER a Thomas BROX. *U-Net: Convolutional Networks for Biomedical Image Segmentation* [online]. 2015 [cit. 2023-12-23]. Dostupné z:
https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28.

[16] BADRINARAYANAN, Vijay, Alex KENDALL a Roberto CIPOLLA. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation* [online]. 2017 [cit. 2023-12-23]. Dostupné z:
https://ieeexplore.ieee.org/document/7803544.

[17] WANG, Hengli, Rui FAN, Peide CAI a Ming LIU. *SNE-RoadSeg+: Rethinking Depth-Normal Translation and Deep Supervision for Freespace Detection*

[online]. 2021 [cit. 2023-12-23]. Dostupné z:
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9636723&
casa_token=AvQ-Ev06BUoAAAAA:qACC6u5rLpo838PWnRAYfv0zpyvCMw-HD_
N0hhhxyBnwAE6E7DRf7bmHTSQi-4M9wwAj0RRthjHW&tag=1.

[18] LI, Jiahang, Yikang ZHANG, Peng YUN, Guangliang ZHOU, Qijun CHEN a Rui FAN. *RoadFormer: Duplex Transformer for RGB-Normal Semantic Road Scene Parsing* [online]. 2023 [cit. 2023-12-23]. Dostupné z:
https://arxiv.org/pdf/2309.10356.pdf.

[19] BAI, Lin, Yecheng LYU a Xinming HUANG. *RoadNet-RT: High Throughput CNN Architecture and SoC Design for Real-Time Road Segmentation* [online]. 2021 [cit. 2023-12-23]. Dostupné z:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9266112.

[20] CALTAGIRONE, Luca, Samuel SCHEIDEGGER, Lennart SVENSSON a Mattias WAHDE. *Fast LIDAR-based road detection using fully convolutional neural networks* [online]. 2017 [cit. 2023-12-23]. Dostupné z:
https://ieeexplore.ieee.org/abstract/document/7995848.

[21] KONG, Hui, Jean-Yves AUDIBERT a Jean PONCE. *General road detection from a single image* [online]. 2009 [cit. 2023-12-28]. Dostupné z:
https://www.di.ens.fr/willow/pdfs/tip10b.pdf.

[22] DENG, Fucheng, Xiaorui ZHU a Chao HE. *Vision-Based Real-Time Traversable Region Detection for Mobile Robot in the Outdoors* [online]. 2017 [cit. 2024-01-02]. Dostupné z:
https://www.mdpi.com/1424-8220/17/9/2101.

[23] MOGHADAM, Peyman, Janusz A. STARZYK a W. S. WIJESOMA. *Fast Vanishing-Point Detection in Unstructured Environments* [online]. 2012 [cit. 2023-12-28]. Dostupné z:
Http://oucsace.cs.ohio.edu/~starzyk/network/Research/Papers/Fast%
20vanishing%20point%202012.pdf..

[24] MIKSIK, Ondrej. *Rapid Vanishing Point Estimation for General Road Detection* [online]. 2012 [cit. 2023-12-28]. Dostupné z:
http://miksik.co.uk/files/miksik2012icra.pdf.

[25] MIKSIK, Ondrej. *Rapid Vanishing Point Estimation for General Road Detection* [online]. 2012 [cit. 2023-12-28]. Dostupné z:
http://miksik.co.uk/files/miksik2012icra.pdf.

[26] PHANG, Son Lam, Manh Cuong LE a Abdesselam BOUZERDOUM. *Pedestrian lane detection in unstructured scenes for assistive navigation* [online]. 2016 [cit. 2024-01-02]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1077314216000369.

[27] TAN, Ceryen, Tsai HONG, T. CHANG a M. SHNEIER. *Color model-based real-time learning for road following* [online]. 2006 [cit. 2023-12-28]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/1706865.

[28] RAMSTROM, Ola a Henrik CHRISTENSEN. *A Method for Following Unmarked Roads* [online]. 2005 [cit. 2023-12-28]. Dostupné z: https://www.researchgate.net/profile/Henrik-Christensen-6/publication/4171865_A_method_for_following_unmarked_roads/links/02e7e514b45b2f165a000000/A-method-for-following-unmarked-roads.pdf.

[29] ÁLVAREZ, José M. a Antonio M. LOPEZ. *Road Detection Based on Illuminant Invariance* [online]. 2011 [cit. 2024-01-02]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5594640.

[30] LU, Xiqun. *Self-Supervised Road Detection From a Single Image* [online]. 2015 [cit. 2024-01-02]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7351351.

[31] MIKSIK, Ondrej, Petr PETYOVSKY, Ludek ZALUD a Pavel JURA. *obust Detection of Shady and Highlighted Roads for Monocular Camera Based Navigation of UGV* [online]. 2011 [cit. 2024-01-03]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5979773.

[32] ÁLVAREZ, Jose M., Antonio M. LÓPEZ a Theo GEVERS. *Combining Priors, Appearance, and Context for Road Detection* [online]. 2014 [cit. 2024-01-02]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6719504.

[33] LEE, Honggu, Kiho KWAK a Sungho JO. *An incremental nonparametric Bayesian clustering-based traversable region detection method* [online]. 2016 [cit. 2024-01-02]. Dostupné z: https://link.springer.com/article/10.1007/s10514-016-9588-7.

[34] LIGOCKI, Adam, Aleš JELÍNEK a Luděk ŽALUD. *Robotics BUT - Brno Urban Dataset* [online]. [cit. 2024-03-15]. Dostupné z: https://github.com/Robotics-BUT/Brno-Urban-Dataset.

[35] PYTHON. *About Python* [online]. In: . [cit. 2024-05-12]. Dostupné z: https://www.python.org/about/.

[36] STROUSTRUP, Bjarne. *The C++ Programming Language* [online]. Třetí vydání. Wesley publishing company, 1997 [cit. 2024-05-12]. ISBN 0-201-88954-4. Dostupné z: https://archive.org/details/cprogramminglang00stro_0/page/n5/mode/2up.

[37] OPENCV TEAM. *OpenCV* [online]. [cit. 2024-05-12]. Dostupné z: https://opencv.org.

[38] NUMPY TEAM. *NumPy* [online]. [cit. 2024-05-12]. Dostupné z: https://numpy.org.

[39] GRT. *Konvoluce 2rozm diskretni* [online]. [cit. 2024-04-09]. Dostupné z: https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg.

[40] FISHER, Robert, Simon PERKINS, Ashley WALKER a Erik WOLFART. *Gaussian Smoothing* [online]. 2003 [cit. 2024-04-09]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

[41] OPENCV. *OpenCV: Canny Edge Detection* [online]. [cit. 2024-05-12]. https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.

[42] LEE, Socret. *Lines Detection with Hough Transform* [online]. In: . 2020 [cit. 2024-04-07]. Dostupné z: https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549.

[43] OPENCV. *OpenCV: Hough Line Transform* [online]. In: . [cit. 2024-04-07]. Dostupné z: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[44] STEJSKAL, Jan. *Houghova transformace a její varianty* [online]. Brno, 2021 [cit. 2024-04-07]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=227518. Bakalářská práce. Vysoké učení technické.