VŠB - Technická Univerzita Ostrava
Fakulta Elektrotechniky a Informatiky
Katedra informatiky

Rozpoznávání dopravních značek pro řízení modelu auta
Recognition of traffic Signs for Car Model

2017 Bc. Jan Frydrych

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Frydrych**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozpoznávání dopravních značek pro řízení modelu auta
Recognition of traffic Signs for Car Model

Jazyk vypracování: čeština

Zásady pro vypracování:

Navrhněte program pro embeded zařízení, který bude během jízdy malého modelu automobilu rozpoznávat několik základních dopravních značek. Program musí být navržen pro malý počítačový systém na platformě ARM (i.MX6) s OS Linux nebo Android. Pro rozpoznávání je doporučeno využít knihovnu OpenCV.

1. Seznamte se s aktuálním stavem rozpoznávání dopravních značek.
2. Navrhněte vlastní model značek pro model automobilu v měřítku 1:18.
3. Nasnímejte značky v okolí testovací dráhy a detekujte je v obraze.
4. Aplikujte detekci značek za pohybu auta.
5. Ověřte spolehlivost v závislosti na rychlosti a světelných podmínkách.

Seznam doporučené odborné literatury:

[1] Knihovna OpenCV: http://www.opencv.org
[2] Pavel Čip, DP, Detekce a rozpoznávání dopravních značek, VUTBR
[3] Jakub Sochor, BP, Rychlá detekce dopravních značek v obraze, VUTBR
[4] Sojka Eduard, Digitální zpracování obrazu, Skripta VŠB, ISBN 80-7078-746-5

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Olivka, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017

.....

podpis studenta

Rád bych poděkoval vedoucímu diplomové práce Ing. Petrovi Olivkovi, Ph.D. za odbornou pomoc, cenné rady a poskytnutý hardware pro účely tvorby práce.

Abstrakt

Tato diplomová práce se zabývá detekcí a rozpoznáváním dopravních značek v obraze. Cílem práce bylo navrhnout program pro rozpoznávání modelů dopravních značek v reálném čase. Program má být navržen pro vestavěné zařízení postavené na platformě ARM. V první části práce budou popsány způsoby, kterými mohou být dopravní značky detekovány, a algoritmy, které se k tomu používají. Dále bude popsán používaný klasifikátor, pro klasifikaci dopravních značek. V další části práce se budu věnovat implementaci programu a použitým algoritmům. Nakonec budou vyhodnoceny výsledky, kterých se podařilo dosáhnout.

Klíčová slova

dopravní značky, detekce, Cannyho hranový detektor, support vector machines, OpenCV, Raspberry Pi

Abstract

This master's thesis deals with detection and recognition of traffic signs in the image. The aim of the thesis was to design a program for real-time identification of model traffic signs. The program should be designed for an embedded device on ARM based platform. The first part of the thesis will describe the ways in which traffic signs can be detected and the algorithms used for this purpose. Further, will be described the classifier used for the classification of traffic signs. In the next part I will deal with the implementation of the program and the algorithms used. Finally, the results that have been achieved will be evaluated.

Key words

Traffic signs, detection, Canny edge detector, Support Vector Machines, OpenCV, Raspberry Pi

Obsah

Seznam použitých symbolů a zkratek

SURF Speeded-up robust features
SIFT Scale-invariant feature transform
SVM Support Vector Machines

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešních moderních autech je stále více využíváno vestavěných počítačových systémů pro asistenci nebo spoluúčasti na řízení. Například asistenční systém udržování jízdního pruhu, parkovací asistent, systém rozpoznávání dopravních značek, adaptivní kontrola vzdálenosti nebo kontrola (a zamezení) mikrospánku. Většinou se jedná o snímání scény v okolí automobilu a následné zpravování pořízených záběrů. Systém pro udržování jízdního pruhu sleduje vozovku před vozidlem a detekuje čáry v obraze. Systém rozpoznávání a detekce dopravních značek snímá obraz před autem, a může tak řidiče upozornit na blížící se značku nebo zpětně zobrazit značky, které minul. Například lze v úseku od poslední značky detekovat nejvyšší povolenou rychlost a informovat řidiče. Systémy detekce dopravních značek mají většinou speciální kamery namířené ideálním směrem, kde by se přibližně měly značky vyskytovat. Profesionální systémy mají kvalitní kamery podpořené dobře fungujícím softwarem.

Pro všechny tyto systémy se používá počítačové vidění. Počítačové vidění je odvětví výpočetní techniky. Základní pilíře tvoří ještě tři další vědní obory, teorie signálů, rozpoznávání a lidské vidění. Aby počítač viděl a vnímal jako člověk, museli bychom vyřešit většinu úkolů umělé inteligence a porozumět biologickému vidění (což je velmi těžké, blízké k nemožnému). Vznikají tak zajímavé mezioborové vazby. Počítačové vidění usiluje o vytváření strojů schopných „vidět a vnímat“.

Dnes už existují inteligentní automobily i autobusy, které umí jezdit samostatně bez asistence řidiče. Ale pouze rozpoznáváním dopravních značek pro řízení takového auta není dostačující. Auto, které řídí samo, je vybaveno mnoha senzory a kamerami. Jedná se o laserové skenery různých dosahů, s různými zornými úhly a také řadu kamer, které sledují předměty a značky před vozidlem. Jako doplňkový systém slouží GPS navigace, která pro samotné řízení není přesná a spolehlivá.

V kapitole 2 této práce popíšu aktuální stav rozpoznávání dopravních značek. Představím používané metody a algoritmy pro detekci a klasifikaci objektů v obraze. Popíšu také, co je to počítačové vidění a čím se zabývá. V kapitole 3 stručně popíšu dopravní značky, jejich vlastnosti a rozměry. Následovat budu možnosti, kam lze podle normy značky umístit, v jaké mají být výšce a jak daleko od cesty se mají nacházet. Ke konci uvedu tvorbu zmenšených modelů dopravních značek, které snímám a rozpoznávám. V kapitole 4 se budu věnovat tvorbě programu a věcem s tím spojeným. Představím používaný hardware, popíšu výhody a nevýhody, podle kterých jsem se rozhodoval. Následně popíšu používané algoritmy a metody, modely pro digitální reprezentaci obrazových dat a použité knihovny. Na konci kapitoly popíšu funkce vlastního programu. V kapitole 5 popíšu, jak a čím byly značky snímány a jaké problémy nastaly při pohybu auta. V poslední kapitole popíšu a zhodnotím dosažené výsledky, rychlost zpracování a úspěšnost detekce.

2 Detekce dopravních značek

V této kapitole popíšu algoritmy, které se používají pro detekci dopravních značek v obraze. Na začátku stručně popíšu počítačové vidění. Informace jsem čerpal z knih zabývajících se počítačovým viděním [1] a [2].

2.1 Počítačové vidění

„Cílem počítačového vidění je dělat užitečná rozhodnutí o reálných fyzických objektech a scénách, na základně snímaných obrázků.“ Definice počítačového vidění podle autorů z knihy Computer Vision [1].

Aby bylo možné rozhodovat o reálných objektech a zpracovat obraz v počítači, je nutné vytvořit nějaký popis nebo model daného objektu. Můžeme se tak setkat s tvrzením, že cílem počítačového vidění je budování popisu scény z obrazu. Problémy počítačového vidění shrnují a definují následující otázky.

Snímání: Jak senzory získají představy o světě? Jak do obrazu zakódovat vlastnosti světa, jako je materiál nebo tvar předmětu, osvětlení scény nebo prostorové uspořádání?
Zakódované informace: Jak z obrazu získat informaci pro pochopení trojrozměrného světa, když se obraz ukládá dvojrozměrně? Jak získat například informaci o geometrii, textuře, pohybu nebo identitě objektu v obraze?
Reprezentace: Jak uložit popis objektů, jejich části, vlastnosti a vztahy?
Algoritmy: Jaké existují metody pro zpracování informací z obrázků a vybudování popisu světa a jeho objektů?

Obrázek 2.1: Počítačové vidění

Počítačové vidění vyrostlo přinejmenším na čtyřech pilířích (ilustrace na obrázku 2.1):

- věda o počítačích,
- teorie signálů,
- rozpoznávání,
- porozumění lidskému vidění (biologické vidění).

Poskytuje a využívá tak zajímavé mezioborové vazby. Je to relativně nový obor. Počítačové vidění vznikalo v 60. letech 20. století na univerzitách, které byly průkopníky v umělé inteligenci. Systém byl určen pro napodobení lidského oka, jako příprava pro tvorbu robotů s inteligentním chováním. Vznik se datuje od roku 1970. V těchto počátcích vznikl zajímavý projekt připojit k počítači kameru, zaznamenávat obraz a potom „říct počítači co viděl“. Dnes však už víme, že tenhle problém je o něco složitější.

2.2 Detekce založená na tvaru dopravní značky

V článku [3] je popisován postup detekce dopravních značek založený na jejich tvarech. Výhodou je, že jsou tyto tvary ve většině států světa stejné. Při změně světelných podmínek může dojít ke změně výsledné barvy, ale tvar zůstane stejný.

Popis algoritmu

Algoritmus je založený na principu vyhledávání kružnic a trojúhelníků v obraze. K tomuto účelu je použita Houghova transformace. Jelikož tato metoda není založena na barvě značky, funguje dobře ve dne i v noci. Různý odstín barvy značky za proměnlivých světelných podmínek tak nemá vliv.

Pro detekci hran je využita Cannyho metoda detekce hran. V článku používají mírně upravenou metodu s dynamicky volenými hodnotami pro prahování. Jednotlivé hodnoty nejsou určeny pevně, ale pro každý obrázek zvlášť. Hodnoty jsou získávány na základě histogramu obrázku. Histogram je rozdělený do osmi oblastí, a každé oblasti je přidělena dvojice hodnot, minimum a maximum. Podle šířky histogramu, se vždy vyberou odpovídající hodnoty. Díky tomuto přístupu lze algoritmus použít za dobré viditelnosti ve dne a také při zhoršení viditelnosti v noci nebo za deště.

Z nalezených hran se v obraze dále detekují obrysy. Každý obrys je otestován, zda splňuje podmínky pro to, být obrysem dopravní značky. Obrys je akceptován, když je uzavřený nebo téměř uzavřený, a délky stran obdélníku, který obrys ohraničuje, musí být v určitém poměru. Obrysy, které tyto podmínky nesplňují, jsou ignorovány. Houghova transformace je aplikována pouze na obrysy, které vyhoví všem filtrům. Tím se ušetří výpočetní čas.

Klasický Houghův algoritmus může být jednoduše rozšířen, aby našel všechny křivky obrázku, které mohou být vyjádřeny vztahem $f(x,p) = 0$. Kde x je bod v obraze a p je vektor parametrů.

Houghova transformace pro rovné přímky se používá pro detekci trojúhelníků. Přímka je vyjádřena rovnicí (2.1), kde ρ je délka kolmice k dané přímce z počátku soustavy souřadnic obrázku a θ je úhel kolmice k ose x.

$$x \times \cos(\theta) + y \times \sin(\theta) = \rho \tag{2.1}$$

Cílem je najít tři přímky, které se navzájem protínají a svírají úhly 60 stupňů. Takových přímek je v celém obrázku velké množství. Při použití Houghovy transformace neznáme začátek ani konec přímky. Mohou být proto nalezeny nereálné trojúhelníky, které v originálním obrázku nejsou. Z tohoto důvodu je zde Houghova transformace aplikována na každý obrys zvlášť, jeden po druhé. Tímto způsobem jsou detekovány pouze trojúhelníky, které ve skutečnosti v obraze existují, což opět sníží výpočetní čas.

Podobný princip se používá také pro kruhové značky. Používá se Houghova transformace pro detekci kružnic. Kromě kruhových značek, se také detekuje značka STOP. Ačkoli je značka STOP osmiúhelník, rozdíly mezi kruhem jsou nepatrné a algoritmus je také příjme. Kružnici se středem (χ, ψ) a poloměrem ρ lze vyjádřit rovnicí (2.2).

$$(x - \chi)^2 + (y - \psi)^2 - \rho^2 = 0 \tag{2.2}$$

Pro detekci kruhových objektů jsou použita stejná kritéria jako v případě přímek. Houghova transformace se také aplikuje na obrys po obrysu, aby ty, které neodpovídají značkám, neovlivňovaly detekci ostatních obrysů. Klasická Houghova transformace analyzuje všechny body jako možné středy kružnice. To znamená, že je detekováno více kružnic, než jich je ve skutečnosti. Aby se tomu zabránilo, ukládají se informace o nalezených kružnicích. Při detekci dalších kružnic, se prohledají dosud uložené informace a hledá se, zda nový střed neleží v nějaké jiné kružnici. Všechny tyto kroky značně sníží počet iterací a výpočetní čas, a to bez ztráty spolehlivosti.

Dosažené výsledky

V článku [3] dosáhli výsledků, kdy byl algoritmus schopný detekovat značky v reálném čase. Testovací sada obsahovala téměř 750 dopravních značek a byly v ní zastoupeny dva typy značek. Jeden typ byl kruhový - značky omezující maximální rychlost a druhý typ trojúhelníkový - výstražné značky. Dosažené výsledky lze vidět v tabulce 2.1.

Tabulka 2.1: Dosažené výsledky detekce podle tvaru

typ značky	počet	detekování	rozpoznání
kruhová	435	97.2%	98.5%
trojúhelníková	312	94.3%	97.2%

2.3 Detekce dopravních značek využívající vizuálních slov

V článku [4] je popisovaný odlišný algoritmus, který používá pro detekci dopravních značek metodu SURF. Zkratka vychází s anglického slovního spojení Speeded-Up Robust Features. Je to metoda, která dokáže popsat obrázek pomocí deskriptorů. Algoritmus je tvořen s cílem běžet na vestavěných zařízeních s podporou paralelního zpracování, měl by tedy být relativně rychlý.

Popis algoritmu

Rozpoznávání dopravních značek v reálném čase je problém, který se často řeší ve dvou fázích. První fáze představuje detekci objektu v obraze a druhá pak klasifikaci detekovaného objektu. Popisovaný algoritmu řeší obě fáze najednou, je zde tedy předpoklad, že algoritmus bude rychlejší než jiné. Algoritmus nespoléhá na barvy, detekci hran, detekci geometrických útvarů nebo hledání malé

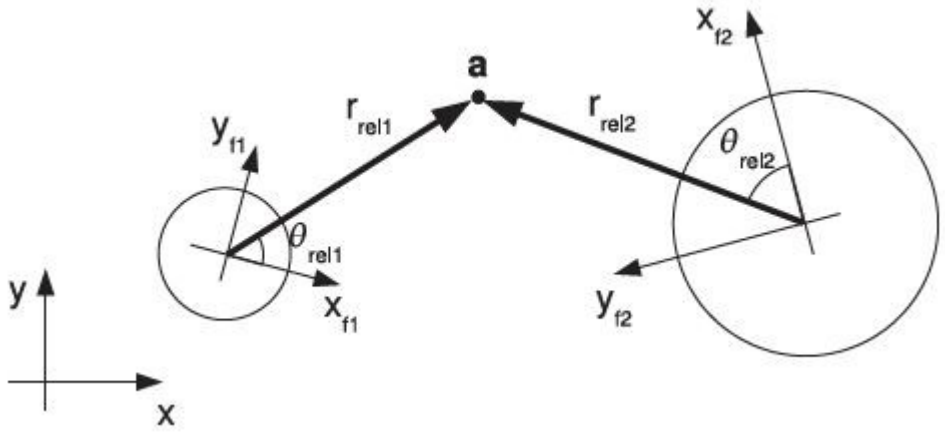
předlohy ve velkém obraze. Navíc není omezený pouze na určité tvary dopravních značek. Pro důkladný popis vzhledu značky používá to, jak jsou uspořádané rysy tvaru dopravní značky.

Pro obrázek z předlohy je spočítán a vytvořen model, který daný obrázek popisuje. V obrázku jsou nalezeny a označeny zájmové body. Následně je vytvořen slovník vizuálních slov, který je tvořen shlukem podobných deskriptorů, které popisují nalezené zájmové body. Ze slovníku je vytvořen model. Ke každému vizuálnímu slovu se uloží relativní pozice nalezených bodů a také úhel natočení. Pozice je daná vzdáleností od určeného bodu, většinou se volí střed obrazu.

Jako detektor lokálních rysů je použita metoda SURF. Jedná se o novější a vylepšenou metodu, než je metoda SIFT. Metoda SURF je několikrát rychlejší a je odolnější vůči různým transformacím obrazu. V první fázi jsou vytvořeny kruhové oblasti kolem nalezených rysů za účelem přiřazení orientace každému bodu. Orientace je počítána pomocí Haarovy vlnky. Postupným natáčením, vždy o úhel $\frac{\pi}{3}$, jsou získány hodnoty. Úhel, kterému odpovídá maximální hodnota, je vybraný jako orientace daného rysu. Ve druhé fázi jsou vytvořeny čtvercové oblasti kolem nalezených bodů pro vytvoření popisu SURF rysů. Orientace čtvercových oblastí odpovídá vypočítané orientaci z předchozího kroku.

Dalším krokem algoritmu je vytvoření slovníku vizuálních slov. Je důležité vytvořit slovník za použití velkého množství rysů, aby byl reprezentativní pro všechny obrázky, které mají být později zpracovány. To je dosaženo shlukováním velkého množství SURF rysů. Samotné shlukování je prováděno algoritmem k-means. Vzdálenosti jsou počítány pomocí Euklidovské vzdálenosti mezi odpovídajícími SURF rysy. Toto předzpracování se dá provést dopředu, při zpracování v reálném čase se informace využívají k porovnávání.

Všechny příznaky nalezené na obrázku předlohy jsou spojeny s příslušným vizuálním slovem ve slovníku. Ke každému příznaku je navíc uložena informace o umístění v obraze. Každý řádek v popisu modelu obsahuje symbolický název vizuálního slova a relativní souřadnice (r_{rel}, θ_{rel}) od určeného kotevního bodu. Kde r je vzdálenost příznaku od kotevního bodu a θ je úhel natočení vzhledem k přímce vzdálenosti. Jako kotevní bod se většinou používá střed obrázku. Příklad určení souřadnic, vzdálenosti a úhlu pro dva body, lze vidět na obrázku 2.2



Obrázek 2.2: Ukázka rozmístění dvou bodů

(převzato z článku [4])

Fáze detekce objektů je časově náročná. Pro realizaci algoritmu pro vestavěné systémy je potřeba tuhle část optimalizovat a co nejvíce zrychlit.

Pro vstupní obraz jsou spočítány SURF příznaky stejným způsobem jako pro modely. Příznakům jsou přiřazena vizuální slova ze slovníku, který vznikl při vytváření modelů. Ke každému nalezenému příznaku lze spočítat jeho kotvící bod. V ideálním případě by se pro konkrétní objekt všechny body promítly do středu hledaného objektu. Kvůli chybám a šumu se toto často nestává. Navíc některé příznaky mohou být v modelu nalezeny vícekrát, a to s odlišnou vzdáleností a úhlem. V detekovaném obraze jsou uvažovány všechny možné polohy tohoto objektu, ale pouze jedna může být správná.

Dalším úkolem je najít, kde hledaný objekt leží. Nelze však použít jednoduchý průměr všech nalezených kotvících bodů, protože určitý objekt může být v obraze nalezen vícekrát, a tvořil by tak falešné body. Pro tento účel je vytvořen Houghův prostor, který je inicializován hodnotami nula. Každý nalezený bod zvýší hodnotu v matici Houghova prostoru. Hodnoty jsou zvyšovány na základě souřadnic kotvících bodů. Lokální maximum výsledného Houghova prostoru je považováno za střed hledaného objektu.

Dosažené výsledky

Autoři článku [4] testovali prototyp algoritmu pro software Octave. Obrázky použité jako modely značek měly rozlišení přibližně 100 × 100 pixelů. Z každého obrázku bylo vygenerováno v průměru 52 příznaků. Testování probíhalo na reálných obrazech značek s rozlišením 640 × 480 pixelů.

Testování proběhlo na 35 obrázcích s výslednou úspěšností 82% detekovaných značek. Selhání detekce bylo způsobeno většinou tím, že značka byla daleko, byla proto příliš malá, nebo značku překrýval jiný objekt. Rychlost detekce v článku uvedena není, možná z toho důvodu, že se jednalo o prototyp.

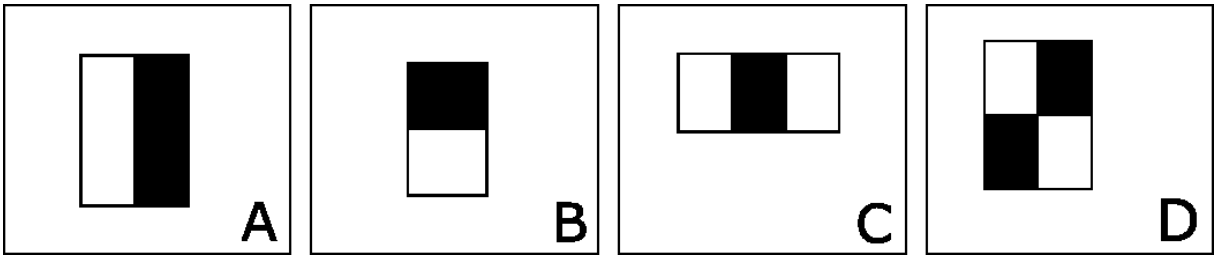
2.4 Detekce objektů v obraze pomocí posuvného okna

Algoritmus představený v článku [5] je navržený pro detekci nejen dopravních značek, ale také pro detekci jiných objektů. Článek popisuje přístup k detekci objektů pomocí strojového učení. Pomocí toho je možné zpracovat obraz velmi rychle a dosáhnout vysoké míry spolehlivosti.

Příznaky

Postup detekce objektů v následujícím algoritmu je založen na hodnotě jednoduchých příznaků. Ve většině případů bývá výhodnější použít pro detekci speciální příznaky, než pracovat přímo s pixely. Příznaky lze použít pro rozeznání speciálních případů. Další motivace pro použití příznaků je, že pracují mnohem rychleji než systémy založené na zpracování pixelů.

V algoritmu jsou používány tři různé příznaky. Jedná se o oblast, která se skládá ze dvou, tří nebo čtyř obdélníků. Oblasti mají stejnou velikost a tvar, jsou orientovány vodorovně nebo svisle. Hodnota celého příznaku je rovna rozdílů součtu pixelů v rámci jednotlivých obdélníkových oblastí. Počet pixelů pod bílým obdélníkem je odečten od počtu pixelů pod černým obdélníkem. Používané příznaky jsou vidět na obrázku 2.3. Ve výřezu o velikosti 24 × 24 bodů je 180 000 takovýchto příznaků.

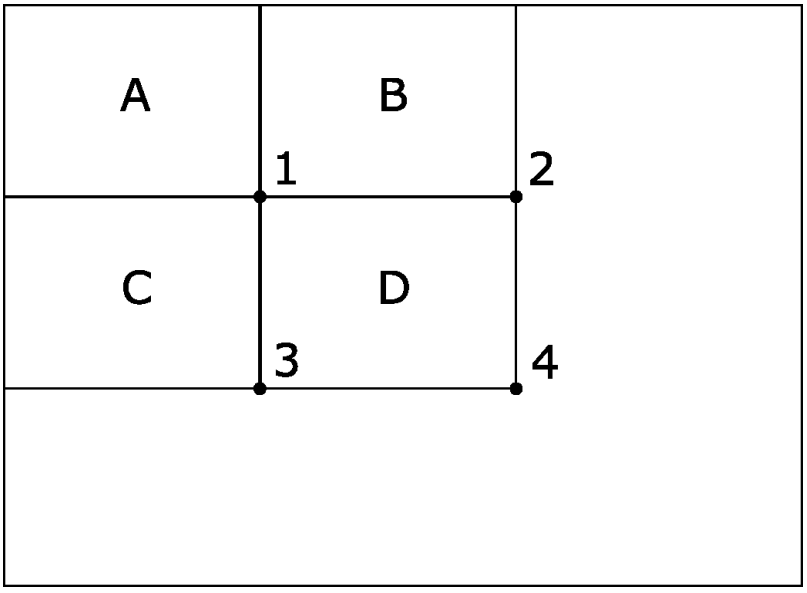


Obrázek 2.3: Příklady obdélníkových příznaků

Hodnotu příznaku lze velmi rychle vypočítat pomocí integrálního obrazu. Integrální obraz je stejně velký jako obraz původní. Součet pixelů integrálního obrazu na souřadnicích x, y se rovná počtu všech pixelů na ploše vlevo a nahoře od uvedených souřadnic. Hodnotu integrálního obrazu lze spočítat podle vzorce (2.3), kde $ii(x, y)$ je integrální obraz a $oi(x, y)$ originální obraz.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} oi(x', y') \tag{2.3}$$

Díky použití integrálního obrazu lze pomocí čtyř hodnot zjistit počet bodů v jakémkoli obdélníku, viz obrázek 2.4. Součet pixelů v obdélníku D lze snadno spočítat pomocí ostatních. Hodnota integrálního obrazu v bodě 1 je součet pixelů v obdélníku A. Hodnota v bodě 2 je A + B, v bodě 3 pak A + C a v bodě 4 je hodnota rovna A + B + C + D. Součet pixelů v obdélníku D lze vyjádřit jako 4 + 1 - (2 + 3).



Obrázek 2.4: Výpočet bodů v integrálním obraze

Rozdíl v počtu pixelů mezi dvěma obdélníky lze spočítat pomocí osmi bodů. Jelikož obdélníky spolu sousedí, stačí pro výpočet pouze šest bodů. Pro tři obdélníky je potřeba osm bodů, pro čtyři sousedící obdélníky bodů devět.

Popis algoritmu

Algoritmus pro detekci se skládá z kaskády klasifikátorů. Jednotlivé klasifikátory jsou trénovány pomocí algoritmu AdaBoost (zkratka z anglického Adaptive Boosting). Následně jsou prahové hodnoty přizpůsobeny pro minimální počet falešných detekcí. Obraz, ve kterém chceme objekt detekovat, je postupně předáván klasifikátorům. Každý možný výřez obrazu je předán prvnímu klasifikátoru. Pozitivní výřez, který splní podmínky prvního klasifikátoru, je předán druhému. Ten daný výřez opět vyhodnotí, a pokud splňuje podmínky, je předán dalšímu klasifikátoru a tak dále. Negativní výřez může vypadnout v jakémkoli kroku, pozitivní projde celou kaskádou klasifikátorů.

Dosažené výsledky

V článku [6] byl představený algoritmus využit pro detekci dopravních značek. Algoritmus pracuje s mírně pozměněnými příznaky. Používá jich více typů a více variant rozmístění bílých a černých obdélníků. Počet těchto příznaků ve výřezu 30×30 je 2^{30}. Proto je v algoritmu využíváno evolučního algoritmu AdaBoost.

Celková úspěšnost detekce byla větší než 90%. Algoritmus dokáže rozpoznat značky i v extrémních světelných podmínkách. Pro trojúhelníkové značky a značky omezující rychlost byla úspěšnost 92%. Pro ostatní kruhové značky dokonce 98%. Rychlost detekce v článku uvedena není.

3 Dopravní značení

Dopravní značení je neodmyslitelnou součástí silničního provozu. Je určeno k řízení a regulaci dopravy na pozemních komunikacích. Slouží především účastníkům silničního provozu. Značky jsou různého typu. Výstražné upozorňují na nebezpečí, zákazové nařizují, jak se má účastník chovat nebo upravují pravidla provozu. Na křižovatkách se pak setkáváme se značkami upravujícími přednost nebo příkazovými značkami. Dopravní značky mají svůj specifický tvar a barvu:

- značky označující nebezpečí jsou trojúhelníkové podoby s červeným okrajem,
- příkazové značky jsou typicky kruhové s bílými symboly na modrém podkladě,
- zákazové značky jsou kruhové s černými symboly na bílém podkladě s červeným okrajem,
- značky upravující přednost na křižovatkách mají specifický tvar.

Tvaru a barvy dopravních značek lze využít pro jejich detekci. Tyto vlastnosti jsou relativně konzistentní po celém světě a pravděpodobně nedojde k jejich zásadní změně. Proto jsem si jako vzor modelů značek pro svoji práci vybral klasické dopravní značení České republiky. Dopravní značení v České republice je v současné době na základě zákona č. 361/2000 Sb., o pravidlech provozu na pozemních komunikacích stanoveno vyhláškou ministerstva vnitra č. 30/2001 Sb. a vyhláškami, které ji novelizují. Podoba značky je stanovena zákonem, a to jak tvar, barva nebo text uvnitř značky. Je zde tedy předpoklad, že se nebudou měnit a budou shodné na celém území.

Rozměry značek a také výška a vzdálenost od cesty, kam se značky mohou umisťovat, jsou dané normou ČSN EN 12899-1. Norma například udává, že spodní okraj dopravní značky je nejméně 1,2 metrů nad úrovní vozovky. V případě průchozího prostoru pak minimálně 2,2 m nad úrovní chodníku. Maximální výška spodního okraje je 2,5 metrů nad úrovní vozovky. Rozměry dopravních značek lze vidět v tabulce 3.1.

Tabulka 3.1: Rozměry dopravních značek (zdroj: norma ČSN EN 12899-1)

Velikost (mm)	Trojúhelník	Kruh	Osmiúhelník	Čtverec	Obdélník
zmenšená	700	500	-	-	-
základní	900	700	700	500	500 × 700
zvětšená	1250	900	900	750	1000 × 1500
měřítko 1:18	50	39	39	28	-

V mé práci budu pracovat pouze se základním dopravním značením. Vybral jsem několik značek, které upravují přednost, přikazují směr a značky nejvyšší povolené rychlosti.

1. Stůj, dej přednost v jízdě.
2. Dej přednost v jízdě.
3. Hlavní pozemní komunikace.
4. Nejvyšší povolená rychlost 30 km/h.
5. Nejvyšší povolená rychlost 50 km/h.
6. Nejvyšší povolená rychlost 100 km/h.
7. Přikázaný směr jízdy přímo.

8. Přikázaný směr jízdy zde vlevo.
9. Přikázaný směr jízdy zde vpravo.

3.1 Model dopravních značek

Pro svou práci jsem potřeboval vytvořit modely dopravních značek. Značky měly být následně detekovány malým modelem automobilu, který je v měřítku 1:18. Pro vybrané typy značek jsem vyrobil jejich modely ve stejném měřítku. Zvolil jsem základní velikost značek a zmenšil je v poměru 1:18. Spodní okraj modelu značky je ve výšce 13 cm, to odpovídá výšce přibližně 2,3 m. Velikosti modelů značek lze vidět v tabulce 3.1.

Při výrobě modelu značek jsem bral ohled na případnou srážku auta se značkou. Modely značek jsou proto lehké, aby při nárazu auta do značky nedošlo k jeho poškození. Počítá se spíš s poškozením značky. Podstavec je z větvového suku, do kterého jsem vyvrtal díru a zalepil matičku. Sloupek je vyroben ze závitové tyče o průměru 3mm. Samotná značka je vytištěna na samolepku a nalepena na plastovou desku a na plastový distanční sloupek se závitem. Značka se tak dá snadno rozebrat a opravit v případě poškození. Vytvořené modely značek lze vidět na obrázku 3.1.

Obrázek 3.1: Modely dopravních značek

4 Implementace

4.1 Použitý hardware

V této kapitole popíšu hardware použitý v mé práci. Zvolený hardware plyne z požadavků práce. Popíšu výhody a nevýhody používaných kamer, které lze připojit k počítači Raspberry Pi. Informace jsem čerpal z knihy [7] a z oficiální dokumentace na webových stránkách projektu [8].

4.1.1 Raspberry Pi

Požadavek na mojí práci byl navrhnout program pro malý počítačový systém na platformě ARM. Pro tento účel jsem zvolil počítač Raspberry Pi. Je to malý jednodeskový počítač. Jeho velikost jen trochu přesahuje velikost platební karty. To umožňuje technologie systém na čipu (anglicky SoC - system on a chip), kdy jsou všechny součásti počítače integrované do jediného čipu. Můžeme na něm tedy spustit plnohodnotný operační systém. Počítač má svá omezení, například nedisponuje obvodem reálného času. Když se zapne bez připojení k internetu, nezná tak přesný čas, to může způsobit určité problémy. Software je připravovaný na míru a musí splňovat požadavky jednotlivých integrovaných obvodů. Cena nejnovějšího modelu je 35$, v České republice dnes kolem 1 200 Kč.

Počítač Raspberry Pi byl původně vyvinut s cílem podpořit studenty na univerzitách ve výuce informatiky. První verze, Model A, byla vydána v únoru roku 2012. Měl 256 MB paměti RAM, jednojádrový procesor o frekvenci 700 MHz. Cena byla ještě před výrobou stanovena na 25 dolarů. Autoři chtěli, aby počítač stál asi tolik co učebnice.

Dnes už je k dostání aktuální verze Raspberry Pi 3 Model B z února roku 2016. Tento model je osazen čtyř jádrovým 64 bitovým procesorem o frekvenci 1,2 GHz a 1 GB paměti RAM. To už je dostatek pro běh složitějších programů a zpracování obrazu. Tabulka 4.1 shrnuje základní parametry Raspberry Pi 3 Modelu B.

Tabulka 4.1: Parametry počítače Raspberry Pi 3 Model B

Architektura	SoC	Procesor	GPU	Pamět RAM
ARM Cortex A53 (ARMv8)	Broadcom BCM2837	4×1,2 GHz 64-bit	Broadcom VideoCore IV	1 GB

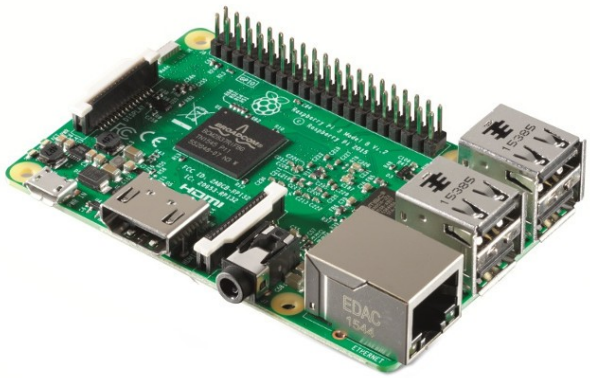
Novinkou v posledním modelu kromě 64 bitového procesoru jsou Wi-Fi a Bluetooth moduly. Počítač Raspberry Pi se tak dá připojit bezdrátově k internetu, případně může sloužit jako přístupový bod. Toho lze využít také pro účely správy na dálku a případné programování. Odpadá nutnost připojovat Raspberry Pi kabelem k směrovači nebo počítači.

Raspberry Pi obsahuje také HDMI výstup pro monitor a poslední model také 4 USB porty, ke kterým je možné připojit klávesnici a myš. Na desce je k dispozici 17 vstupně/výstupních (GPIO) pinů, pro ovládání různých zařízení. Nechybí také sériový port UART, sběrnice I^2C nebo SPI.

Dále je k dispozici rozhraní MIPI pro připojení externího displeje, DSI konektor a vstup pro připojení externí kamery CSI konektor.

4.1.1.1 Operační systém

Primární operační systém pro počítač Raspberry Pi je Raspbian. Jedná se o operační systém odvozený z Debianu. Raspbian tak jako Debian používá jádro Linux. Debian GNU/Linux je jedna z nejstarších doposud vyvíjených distribucí GNU/Linux. Debian má velkou podporu a funguje na většině počítačů. Každá nová verze podporuje větší množství počítačových architektur. Jeho stabilní verze jsou pečlivě testovány a zbaveny chyb. To může někdy znamenat zastaralý software. Pro tyto stabilní verze jsou ale pravidelně vydávány záplaty řešící bezpečnostní problémy a chyby.



Obrázek 4.1: Počítač Raspberry Pi 3 Model B

4.1.2 Kamery pro snímání videa

Podle níže popsaných vlastností jsem se rozhodl pro moji práci zvolit kamerový modul navržený přímo pro Raspberry Pi. Webkamery poskytují stejné rozlišení jako modul a rozměrově jsou větší. Akční kamery jsou také rozměrově větší, ale už poskytují velké rozlišení. Pro požadavky zpracování v reálném čase se použít nedají, nelze je přímo připojit pomocí USB.

4.1.2.1 Kamerový modul pro Raspberry Pi

Kamerový modul se k Raspberry Pi připojuje pomocí již zmiňovaného konektoru CSI. Je to samostatný modul o rozměrech přibližně 25 × 25 mm. Rozlišení senzoru modelu v1 je 2592 × 1944 pixelů. Video je možné přehrávat v největším rozlišení 1080p s 30 snímky za sekundu. To je rozlišení 1920 × 1080 pixelů, známé také jako Full HD.

4.1.2.2 Kamery připojitelné přes USB

Místo používání modulu kamery můžeme použít standardní webkameru připojenou přes USB. Kvalita ani míra nastavení však nebude lepší, než při použití kamerového modulu. Kamera, která umí streamovat video přes USB rozhraní, musí splňovat standard UVC (USB video device class). Běžné, ale i dražší webkamery, které umí streamovat přes USB, poskytují maximální rozlišení videa 1920×1080 pixelů. Stejné rozlišení poskytuje také kamerový modul.

4.1.2.3 Akční (outdoorové) kamery

Akční kamery už nabízí podstatně větší rozlišení videa. Využívají se nejen pro natáčení sportovních zážitků, ale mohou sloužit pro záznam jízdy v autě, nebo záznam videa z letu helikoptéry

či dronu. Výkonnější typy už nemají problém s videem ve 4k (4096×2160) při 30 snímcích za sekundu. Tyto kamery už ale neumí posílat video přímo přes USB. Mají většinou svoje úložiště, kde se video ukládá pro pozdější zpracování. Pro přenos videa z dronu se používá bezdrátový přenos. Nejčastěji se používá frekvence 2,4 GHz a 5,8 GHz. Při přímé viditelnosti není problém přenést video až na několik kilometrů.

4.2 Použité algoritmy

V této kapitole popíšu použité algoritmy pro detekci značek a objektů v obraze. Popíšu modely pro ukládání a reprezentaci obrazů v digitální podobě.

4.2.1 Barevné modely

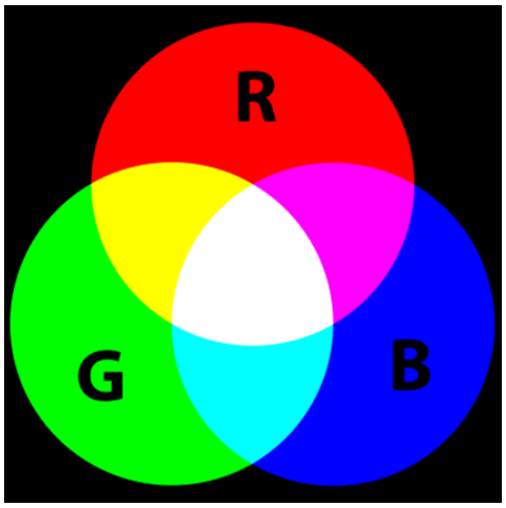
V této části popíšu základní barevné modely, pro reprezentaci barev v obraze. Popíšu výhody a nevýhody. Jednotlivé modely jsou pak používány v popisovaných algoritmech.

4.2.1.1 Model RGB

Barevný model RGB je jeden z nejčastějších. Používá se v barevných televizích, monitorech a podobných zobrazovacích zařízeních, kde je základní barvou barva černá. Jedná se o aditivní způsob míchání barev. Jednotlivé světelné složky barev se sčítají a vytváří světlo větší intenzity. Složíme-li, červenou (R, red), zelenou (G, green) a modrou (B, blue) barvu v plné intenzitě, získáme barvu bílou. Adaptivní míchání barev lze vidět na obrázku 4.2.

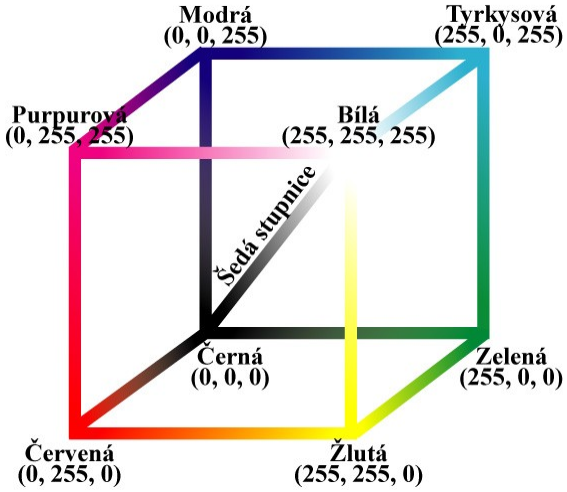
Barvy lze vyjádřit trojicí (barevným vektorem), jejíž složky nabývají hodnoty z intervalu $\langle 0, 1 \rangle$ nebo v celočíselném rozsahu 0 - 255. Hodnota 0 znamená, že daná barva není zastoupena, maximální hodnota pak, že barva nabývá největší intenzity. Hodnotu takové trojce lze zakódovat pomocí 3 bytů. To je dnes nejběžnější způsob. Používá se ale i jiná hodnota, např. 12 nebo 16 bitů. Barevný vektor si lze představit v prostoru RGB krychle, ta je znázorněna na obrázku 4.3.

Počet barevných odstínů, který lze reprezentovat třemi byty je $256^3 = 16\,777\,216$. Ne všechna obrazová zařízení jsou schopna takové množství barev současně zobrazit, proto bývá počet barev před vykreslením uměle snižován.



Obrázek 4.2: Aditivní míchání barev

(převzato z https://en.wikipedia.org)



Obrázek 4.3: Krychle představující RGB prostor

(převzato z http://www.prumyslova-kamera.cz/content/12-barvy-teorie)

4.2.1.2 Šedotónový obraz

Šedotónový obraz, nebo také obraz ve stupních šedi, je reprezentován pomocí barevného modelu, ve kterém je hodnota každého bodu určena jedním vzorkem, nese pouze informaci o intenzitě daného bodu. Obrazy tohoto typu jsou také známé jako černobílé.

Stupně šedi získáme skládáním všech tří barev se stejnou intenzitou. Šedých barev je přesně 255. Jsou to ty odstíny, když jsou všechny tři základní barvy zastoupeny ve stejném poměru. Převod barevného obrazu na šedotónový ale není tak jednoduché. Nemůžeme barvy pouze nahradit odstínem šedi získaným prostým průměrem ze tří základních barev. Lidské oko vnímá intenzitu jednotlivých barevných složek různým způsobem (nejcitlivější je na zelenou). Pro převedení obrazu v RGB modelu do obrazu v odstínech šedi se proto používá empirický vztah vyjádřený rovnicí (4.1).

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{4.1}$$

Pro jeden pixel obrazu ve stupních šedi stačí na uložení do paměti jeden byte. Obrázek lze chápat jako dvourozměrnou matici čísel. Tento model se často využívá pro jeho jednoduchost. Používají ho některé detektory hran. Příklad obrázku převedeného do odstínu šedi viz obrázek 4.4.

Obrázek 4.4: Porovnání barevného a šedotónového obrazu

4.2.1.3 Model HSV

Barevný model HSV (známý také jako HSB) definuje barvu trojicí složek. Tyto složky však tentokrát nepředstavují základní barvy. Třemi hlavními složkami modelu HSV jsou (H - hue) barevný tón, (S - saturation) sytost a (V - value) jasová složka. Barevný tón označuje, která barva převládá, sytost určuje příměs jiné barvy (množství šedi v poměru k odstínu) a hodnota jasu udává množství bílého světla.

Výhodou tohoto modelu je oddělení informace o jasu (složka V) od informace o vlastní barvě (složky H a S). To lze využít pro zkoumání barvy jednotlivých bodů. Barva bodu tak nebude ovlivněna jasovou složkou.

Existují ještě dva podobné modely: HSL a HSI. V prvním z nich L znamená světlost (anglicky lightness) a v druhé I znamená intenzitu.

4.2.1.4 Převod z RGB do HSV

Nevýhodou tohoto modelu je nutnost převodu obrazu z RGB modelu na HSV. Převod je znázorněn rovnicemi (4.2), (4.3), (4.4) převod je už trochu složitější než v případě šedotónového obrazu.

$$\boldsymbol{V} = M = \max(R, G, B) \quad m = \min(R, G, B) \tag{4.2}$$

$$\boldsymbol{S} = \begin{cases} \dfrac{M - m}{M} & V \neq 0 \\ 0 & V = 0 \end{cases} \tag{4.3}$$

$$\boldsymbol{H} = \begin{cases} \dfrac{60 \cdot (G - B)}{M - m} & V = R \\ \dfrac{120 + 60 \cdot (B - R)}{M - m} & V = G \\ \dfrac{240 + 60 \cdot (R - G)}{M - m} & V = B \end{cases} \tag{4.4}$$

4.2.2 Detekce hran v obraze

Detekci hran v obraze používám pro nalezení dopravní značky. Je důležité kvalitně detekovat hrany, pro co nejpřesnější nalezení značky. Popíšu používaný algoritmus pro detekci hran.

Proces detekce hran v obraze slouží pro získání užitečných strukturálních informací o hranách objektů v obraze a snížení množství dat, které se mají dále zpracovávat. Používá se v různých systémech počítačového vidění.

4.2.2.1 Cannyho hranový detektor

Cannyho hranový detektor je více stupňový algoritmus vyvinutý v roce 1986, jeho autorem je John Canny. Je jedním z nejpoužívanějších detektorů hran, protože se snaží splňovat podmínky pro dobrou detekci. Informace jsem čerpal z článku Johna Cannyho [9] a z knihy [10].

Algoritmus se skládá z několika po sobě jdoucích kroků, které umožňují získat co nejlepší výsledek a detekovat co největší počet hran. Canny se snažil objevit optimální algoritmus pro detekci hran. Ve svém článku stanovil tři podmínky pro takový algoritmus.

1. Dobrá detekce: Měla by existovat malá pravděpodobnost selhání detekce hrany a také malá pravděpodobnost detekce špatné hrany.
2. Dobrá lokalizace: Body označené jako hrana se budou nacházet co možná nejblíže středu opravdové hrany.
3. Jednoznačnost: Každá nalezená hrana bude označena právě jednou, nebude docházek ke zdvojení. Šum v obraze nebude vytvářet falešné hrany.

Cannyho hranový detektor je navržen spíše pro obecný signál, který může být jednorozměrný nebo dvourozměrný, lze pomocí tohoto detektoru hledat hrany pouze v obraze s jedním kanálem. Z tohoto důvodu se většinou používá obraz v odstínech šedi. Ten obsahuje pouze jeden kanál, a je to jeden z nejjednodušších způsobů, jak získat z barevného obrazu se třemi kanály obraz s jedním kanálem. Fáze Cannyho algoritmu pro detekci hran lze rozdělit do následujících čtyř kroků.

1. Vyhlazení obrazu za účelem odstranění šumu.
2. Nalezení přechodů v intenzitě obrazu.
3. Aplikování funkce „potlačení nemaximálních hodnot“.
4. Provedení prahování s hysterezí.

Popis algoritmu:

První krok algoritmu spočívá v odstranění šumu z obrazu. Výsledky detekce hran mohou být ovlivněny obrazovým šumem, proto je vhodné tento krok aplikovat. Odstraněním šumu se zabrání detekci falešných hran. Pro vyhlazení obrazu se používá filtr na základě první Gaussovy derivace. Výsledkem je mírně rozmazaný obraz, který je zbavený rušivých pixelů. Příklad filtru o rozměrech 5×5 lze vidět v rovnici (4.5), kde * označuje operaci konvoluce.

$$B = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * A \tag{4.5}$$

Druhý krok je vyhledání přechodů v intenzitě barvy, neboli určení gradientu. Hrana v obraze může směřovat libovolným směrem, proto Cannyho algoritmus používá čtyři filtry pro detekci horizontální, vertikální hrany a diagonálních hran. Pro detekci hran se používají operátory jako například, Roberts cross, Prewitt, Sobel. Nejvhodnější je použít Sobelův operátor, který není příliš citlivý na šum. Operátor vrátí hodnotu první derivace v horizontálním směru (G_x) a ve vertikálním směru (G_y). Z těchto hodnot lze pomocí rovnic (4.6) a (4.7) vypočítat gradient a úhel vektoru, který určuje směr hrany. Úhel vektoru směru hrany se zaokrouhluje na jeden ze čtyř úhlů, které představují vertikální, horizontální směr a dvě diagonály $(0°, 45°, 90°, 135°)$.

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{4.6}$$

$$\theta = atan2(G_y, G_x) \tag{4.7}$$

Další krok algoritmu je aplikování funkce „potlačení nemaximálních hodnot“ (anglicky: non-maximum suppression). Jedná se o techniku ztenčování okraje hrany. Úkolem této funkce je vybrat z hodnot gradientů jen lokální maxima, neboli odebrat body, které nejsou maximem. Tím se zajistí, že hrana bude detekována v místě největšího gradientu. Hrany jsou zpracovávány na základě úhlu vektoru, který udává směr hrany. Zpracování probíhá postupným průchodem mřížky 3 × 3 nad mapou intenzity obrázku. Zpracování probíhá již se zaokrouhlenými úhly. Výpočet může probíhat jedním z následujících způsobů:

- pro úhel 0° platí: bod bude považován za bod hrany, pokud je jeho intenzita větší než intenzita bodů ve směru doleva a doprava,
- pro úhel 90° platí: bod bude považován za bod hrany, pokud je jeho intenzita větší než intenzita bodů ve směru nahoru a dolů,
- pro úhel 45° platí: bod bude považován za bod hrany, pokud je jeho intenzita větší než intenzita bodů vpravo nahoře a vlevo dole,
- pro úhel 135° platí: bod bude považován za bod hrany, pokud je jeho intenzita větší než intenzita bodů vlevo nahoře a vpravo dole.

Poslední krok algoritmu je provedení takzvaného prahování s hysterezí, které slouží k vyznačení nalezených hran. Tato metoda je vhodná, protože zachovává kontury. To je velmi důležité pro detekci tvarů, které se detekují pomocí uzavřených obrysů.

Pro rozhodování se používá hodnota gradientu každého bodu. Pokud je hodnota velká, je zde velká pravděpodobnost, že daný bod je bodem hrany, než když je hodnota malá. Ve většině případů není možné jednoznačně určit prahovou hodnotu, kdy je daný bod ještě bodem hrany. Proto se v algoritmu používá prahování s hysterezí. Jsou tak zapotřebí dvě prahové hodnoty, dolní (T1) a horní (T2). Mezi těmito prahy může gradient kolísat.

Všechny body s hodnotou gradientu větší než hodnota horního prahu T2 jsou označeny jako hranové. Poté se aplikuje dolní práh. Algoritmus rekurzivně postupuje ve směru hrany, která byla určena v dřívějším kroku. Když narazí na bod, jehož hodnota gradientu leží mezi prahy T1 a T2, má na výběr ze dvou možností. Pokud bod sousedí s jiným, který byl jako hrana označen dříve, pak je i tento bod označen jako hranový. Pokud ne, je bod označen jako nehranový. Všechny ostatní body s hodnotou gradientu pod úrovní prahu T1 jsou označeny jako nehranové. Díky této metodě můžeme najít slabé úseky hran.

Dosažené výsledky:

Na obrázku 4.5 lze vidět detekované hrany dopravní značky. Protože Cannyho detektor hran pracuje pouze s obrázky, které mají jeden kanál, byl barevný obrázek napřed převeden do odstínu šedi.

originální obrázek detekované hrany

Obrázek 4.5: Detekované hrany pomocí Cannyho hranového detektoru

4.2.2.2 Detekce hran v barevném prostoru RGB

V článku [11] je představený algoritmus, který přistupuje k detekci hran odlišným způsobem. Pro detekci využívá celého barevného prostoru RGB. Typické algoritmy pro detekci hran v barevném obraze používají Euklidovskou vzdálenost dvou vektorů nebo podobné metriky. Pro přesnější zachycení údajů o hraně se používá také úhel mezi vektory, ale pouze pro složku odstínu a sytosti barevného obrazu. V článku jsou popsané dvě metody, které kombinují Euklidovskou vzdálenost dvou vektorů a úhel mezi vektory. Obě metriky mají určitá omezení, v článku se snaží využít výhody obou těchto metrik.

Euklidovská vzdálenost se obvykle používá v n-rozměrném vektorovém prostoru. Je definována jako vztah (4.8). Pro barevný model, který si lze představit jako trojrozměrný prostor, se vzdálenost počítá vztahem (4.9) kde $\overrightarrow{v_1} = \left[v_{1,1}, v_{1,2}, v_{1,3}\right]^T$ je vektor trojice definující barvu.

$$D(\overrightarrow{v_1}, \overrightarrow{v_2}) = \|\overrightarrow{v_1} - \overrightarrow{v_2}\| \tag{4.8}$$

$$D(\overrightarrow{v_1}, \overrightarrow{v_2}) = \sqrt{\left(v_{1,1} - v_{2,1}\right)^2 + \left(v_{1,2} - v_{2,2,}\right)^2 + \left(v_{1,3} - v_{2,3}\right)^2} \tag{4.9}$$

V barevném prostoru RGB Euklidovská vzdálenost neumí rozlišit body, které mají podobnou barvu. Na rozdíl v barevném prostoru LUV to dělá poměrně správně. Dá se říct, že Euklidovská vzdálenost je citlivější na změny v intenzitě, ale není citlivá na změny v odstínu a sytosti.

Druhá používaná metrika je velikost úhlů vektorů. Na rozdíl od Euklidovské vzdálenosti, je tato metrika citlivější na rozdíly v odstínu a sytosti, ale necitlivá na změnu v intenzitě. Úhel θ, který vektory svírají, respektive kosinus daného úhlu, je definován rovnicí (4.10). Používání funkce kosinus ale není příliš vhodné, a to nejen kvůli složitému výpočtu inverzní funkce. Při nepatrné změně v barevném odstínu dojde k nepatrné změně úhlu θ a funkce kosinus se pro malé hodnoty úhlu příliš nemění. Protože chceme detekovat i malé změny v barevném odstínu, je tomto případě výhodnější použít funkci sinus. Ta totiž pro malé hodnoty úhlu roste relativně rychle. Převod lze provést za použití známého vzorce pro goniometrické funkce $\sin^2 x + \cos^2 x = 1$. Výsledek vyjadřuje rovnice (4.11).

$$\cos\theta = \frac{\vec{v}_1^T \, \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \cdot \|\overrightarrow{v_2}\|} \tag{4.10}$$

$$\sin\theta = \sqrt{1 - \left(\frac{\vec{v}_1^T \, \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \cdot \|\overrightarrow{v_2}\|}\right)^2} \tag{4.11}$$

V obou výše popsaných metrikách lze použít pro detekci hran lokální funkci vektor gradient. Jsou spočítány maximální vzdálenosti vektoru středového bodu od všech vektorů bodů v osmiokolí. Funkce následně vrátí maximální vzdálenost. Funkce pro euklidovskou vzdálenost je vyjádřena rovnicí (4.12) a funkce pro využití úhlu mezi vektory je vyjádřena rovnicí (4.13). Počítadlo *i* reprezentuje všech osm sousedních bodů.

$$E_{VG} = \max_{i=1\ldots8} \{\|\vec{v}_i(x,y) - \vec{v}_0(x,y)\|\} \tag{4.12}$$

$$S_{VG} = \max_{i=1\ldots8} \left[\sqrt{1 - \left(\frac{\vec{v}_i^T(x,y) \cdot \vec{v}_0(x,y)}{\|\vec{v}_i(x,y)\| \cdot \|\vec{v}_0(x,y)\|}\right)^2}\right] \tag{4.13}$$

Každá metrika se hodí lépe pro určení vzdálenosti z jiné informace. Jedna umí lépe pracovat s intenzitou obrazu, druhá lépe s barevným tónem neboli odstínem. Proto je vhodné kombinovat určitým způsobem obě, aby se využilo konkrétních vlastností. V článku jsou představeny dva způsoby kombinace.

První způsob kombinace je založený na intenzitě barev, druhý způsob na sytosti barvy. V článku je používaný právě druhý způsob, založený na sytosti barvy. Pro tuhle metodu je použitý převod RGB obrázku na reprezentaci pomocí modelu HSI. Provádí se to z toho důvodu, že rozptyl šumu v rámci barevného tónu je větší než v intenzitě barvy, když je sytost barvy nízká.

Obě metriky lze použít přímo v RGB prostoru. Metrika založena na úhlech vektorů poskytuje dobrou míru v rozdílu odstínů. Euklidovská metrika zase dobrou míru v rozdílu intenzity. Proto, když jsou oba porovnávané body vysoce nasycené, použije se metrika založená na úhlu vektorů. V případě, že jsou oba body málo nasycené, bude použita Euklidovská vzdálenost.

To, která metrika bude mít větší váhu a ovlivní výsledek, se dá vypočítat. Pro každou dvojici bodů, kterou budeme porovnávat, je potřeba přechodová funkce. Takzvaný sigmoid (4.14) definuje převodní vztah. Hodnota *offset* definuje střed přechodu a *slope* definuje sklon v tomto bodě. Oba body jsou závislé na aplikaci, v článku byly nastaveny experimentálně. Kombinační funkce pro obě metriky je pak definována rovnicí (4.15).

$$\alpha(S) = \frac{1}{1 + e^{-slope(S-offset)}} \tag{4.14}$$

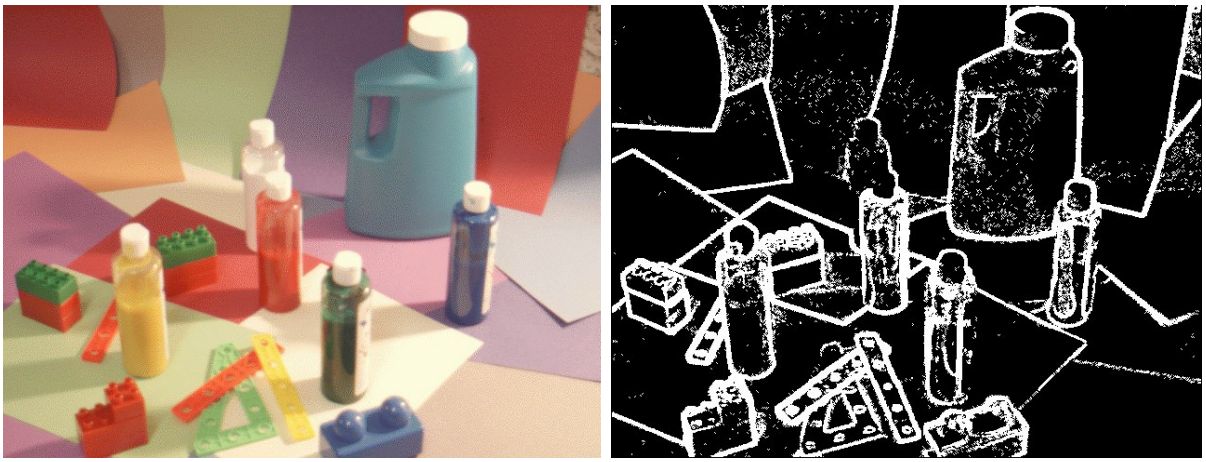
$$\rho(S_1, S_2) = \sqrt{\alpha(S_1) \cdot \alpha(S_2)} \tag{4.15}$$

Výsledná kombinace obou metrik, Euklidovské vzdálenost a velikostí úhlu mezi vektory, je při použití lokální funkce vektor gradient definována jako (4.16). Akorát je nutné normalizovat obě metriky, aby vracely výsledky ve stejném rozsahu. Euklidovská metrika byla přizpůsobena té druhé, obě tak vrací výsledek v rozsahu 0-1.

$$C_{VG} = \max_{i=1\ldots8} \begin{pmatrix} \rho(S_1, S_2) \sqrt{1 - \left(\frac{\vec{v}_i^T(x,y) \cdot \vec{v}_0(x,y)}{\|\vec{v}_i(x,y)\| \cdot \|\vec{v}_0(x,y)\|} \right)^2} + \\ \left(1 - \rho(S_1, S_2)\right) \cdot \|\vec{v}_i(x,y) - \vec{v}_0(x,y)\| \end{pmatrix} \tag{4.16}$$

Dosažené výsledky

Okraje získány pomocí metody založené na Euklidovské vzdálenosti byly mnohem více přeplněné než metoda velikosti úhlu vektorů nebo kombinace obou. Metody založené na velikosti úhlu vektorů vytvářely slabší hrany založené na intenzitě. Nejslibnější výsledky byly dosaženy při použití metody vektor gradient založené na intenzitě a kombinací obou zmíněných metrik. V obrázku se ale také vyskytuje malé množství šumu. Zmíněný výsledek je zobrazen na obrázku 4.6.



Obrázek 4.6: Ukázka detekovaných hran v barevném RGB obrázku

(Obrázky převzaty https://www.semanticscholar.org/paper/Color-Edge-Detection-in-Rgb-Using-Jointly-Wesolkowski-Jernigan/97998fd3f092c50a66edab6e91456b0a8cb761a8)

4.2.3 Detekce obrysů

V následujícím textu popíšu algoritmus, který používám pro vyhledávání obrysů v obraze. Algoritmus je založený na principu, který v článku [12] prezentuje Satoshi Suzuki. Algoritmus je také známý pod názvem sledování pomezí (border following).

Co je to obrys?

Obrys lze popsat jednoduše jako křivku spojující všechny spojité body, které mají stejnou barvu nebo intenzitu. Obrysy jsou užitečné pro pozdější analýzu tvarů, detekci a rozpoznávání objektů.

Popis algoritmu

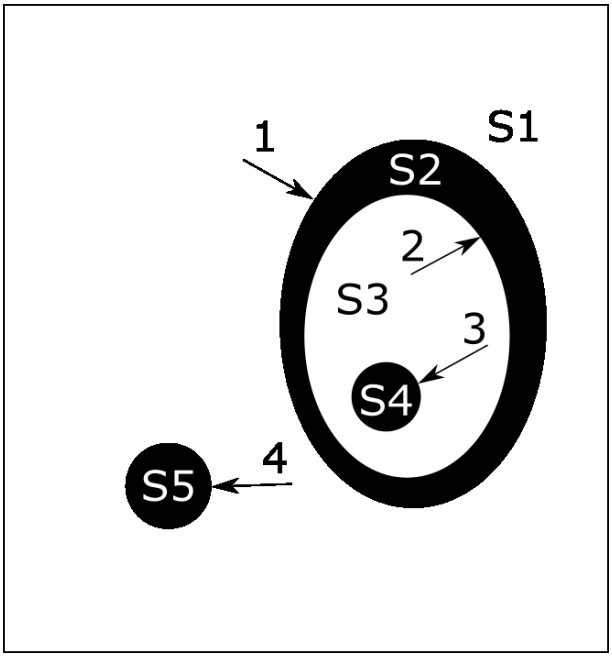
Algoritmus je postaven na principu sledování pomezí. To je jedna ze základních technik při zpracování digitálního binárního obrazu. Algoritmus je schopný zpracovat pouze s tímto obrazem. Obraz je uložený jako body v obdélníkové mřížce. Krajní sloupce a řádky obrazu tvoří jeho rám. V článku je pro algoritmus zavedený systém souřadnic, uvádí se v pořadí (řádek, sloupec). Souřadnice (0, 0) tak představují levý horní roh.

Pro účely vyhledávání obrysů jsou jednotlivé body obrázku v průběhu algoritmu označovány čísly. Význam čísel je následující:

- Body označené jako 0 jsou buď to pozadí obrázku nebo díra.
- Body označené jako 1 jsou body ještě nezpracovaných hran.
- Body označené jinou hodnotou jsou již detekované obrysy. Hodnota může být kladná nebo záporná.

Dále se rozlišují dvoje komponenty. Nulové komponenty složené z bodů označených jako 0. A kladné komponenty, složené z kladných bodů nebo bodů označených jako 1.

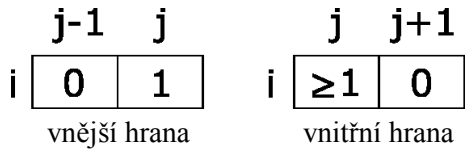
Obrysy se dělí na dva typy, vnější obrys a vnitřní obrys. Vnější obrys je definován jako skupina hraničních bodů mezi nulovou komponentou a kladnou komponentou. Nulová komponenta musí v tom případě kladnou komponentu přímo obklopovat. Vnitřní obrys je pak definován jako skupina hraničních bodů mezi dírou a kladnou komponentou, kde kladná komponenta díru přímo obklopuje. Vnitřním obrysům se někdy také říká obrys otvoru (hole border). Příklad obrysů je znázorněný na obrázku 4.7.



Obrázek 4.7: Typy obrysů a komponent

Obrysy 1, 3 a 4 jsou vnější a obrys 2 je vnitřní. S1 - S5 jsou komponenty. S1 je nulová komponenta, která sousedí s rámem obrazu, je nazývána pozadí. S3 je také nulová komponenta, ale už nesousedí s rámem obrazu, jedná se o takzvanou díru. S2, S4 a S5 jsou kladné komponenty.

Samotný algoritmus probíhá následovně. Vstupní obrázek je procházen postupně po řádcích zleva doprava a začíná se v levém horním rohu. Procházení je zastaveno, pokud je nalezen bod (i, j), který splňuje podmínky pro počáteční bod vnějšího nebo vnitřního okraje, viz obrázek 4.8. Pokud bod splňuje obě podmínky, pak musí být považován za vnější hranu. Následně je bodu přiřazeno jedinečné identifikační číslo označované jako NBD.



Obrázek 4.8: Podmínky pro typ hrany

Poté jsou procházeny a označovány všechny body nalezeného obrysu. Systém procházení obrysu je standardní, významnou vlastností algoritmu je speciální označování bodů. Při procházení se udržuje hranice dvou komponent na jedné straně a pokračuje se tak dlouho, než se opět narazí na počáteční bod procházeného obrysu:

a) Pokud je sledovaný obrys hranicí mezi nulovou komponentou, která obsahuje bod (p, q + 1), a kladnou komponentou, která obsahuje bod (p, q), změní se hodnota aktuálního bodu (p, q) na mínus NBD.
b) V opačném případě je hodnota bodu (p, q) nastavena na NBD, ale pouze pokud daný bod už nebyl dříve označený jako obrys.

Tímto způsobem je obraz procházen tak dlouho, dokud není dosaženo pravého spodního rohu. V tom okamžiku algoritmus končí.

Obrysy jsou reprezentovány pomocí všech jeho bodů. Obrysy lze také aproximovat pomocí úseček, které budou určeny všemi jeho body.

4.2.4 Klasifikace objektů

Klasifikace, nebo také statistické rozpoznávání vzorů, je oblast strojového učení. Zabývá se rozdělením vstupních dat do několika tříd. Každá třída představuje množinu stejných vzorků s podobnými vlastnostmi. V případě klasifikace dopravních značek budou třídy reprezentovat jednotlivé typy značek. Budu se věnovat klasifikátoru Support Vector Machines (SVM), protože ho používám ve své práci. Informace jsem čerpal z knihy [13] a článku [14].

4.2.4.1 Support Vector Machines

Support Vector Machines v překladu Algoritmy podpůrných vektorů je metoda strojového učení původně vynalezená V. Vapnikem. Metoda patří do kategorie tzv. jádrových algoritmů (kernel machines). Tyto metody se snaží využít výhody efektivních algoritmů pro nalezení lineárních hranic a jsou schopny reprezentovat vysoce složité nelineární funkce. Jedním ze základních principů je převod problému do vícedimensionálního prostoru, ve kterém lze od sebe jednotlivé třídy lineárně oddělit.

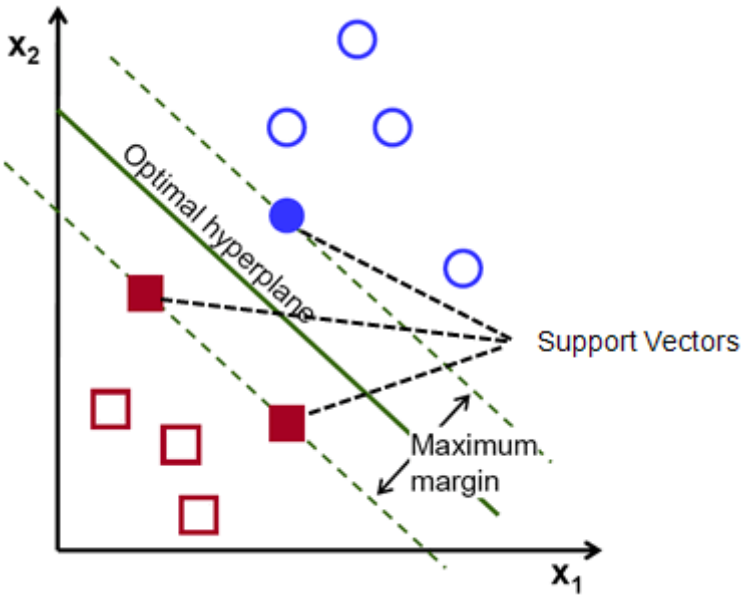
Vstupní množina pro trénování představuje množinu vzorků dat, kdy je každý vzorek **x** označen svým štítkem **y**. Podle štítku se data dělí do dvou tříd, na pozitivní, označené (+1), a negativní označené (-1). Množina tak tvoří vstupní vektor $\big((x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\big)$. Hodnota $x_i = (x_1 \dots x_m)$ představuje vektor vstupních dat pro jeden vzorek a $y_n \in \{-1,1\}$ je označení pozitivní nebo negativní třídy.

Využití SVM pro lineárně oddělitelné datové množiny

Pro tyto dvě třídy se metoda SVM snaží najít dělící rovinu (hyperrovinu). Pokud je datová množina lineárně oddělitelná, existuje rovina, která třídy od sebe oddělí. Pro každý bod z datové množiny tak musí platit vztah (4.17), kde **w** je váhový vektor definující rovinu a **b** je práh. Tím se zaručí, že všechny negativní vzorky budou na jedné straně roviny a pozitivní vzorky na straně druhé.

$$y_i(w \cdot x_i + b) > 0 \tag{4.17}$$

Protože máme konečnou množinu vstupních dat, existuje velké množství těchto dělících rovin. Cílem je vyhledat takovou rovinu, která se maximalizuje minimální vzdálenost od obou shluků dat. Vzdálenost vychází s konvexního obalu obou shluků. Spojnice dvou nejbližších bodů definuje minimální vzdálenost mezi shluky. Dělicí rovina se bude nacházet v polovině spojnice a bude na ní kolmá. Nalezená rovina je tak v maximální vzdálenosti od obou shluků. Příklad je znázorněn na obrázku 4.9.



Obrázek 4.9: Dělící rovina SVM

(převzato z http://www.listendata.com/2017/01/support-vector-machine-in-r-tutorial.html)

Změnou měřítka **w** a **b** (vynásobením kladným číslem) se neporuší platnost vztahu (4.17) a můžeme zvolit **w** a **b** takové, že platí vztah (4.18). Rovnost pak platí pro body s minimální vzdáleností na každé straně. Takových bodů může být více, ale vždycky minimálně jeden. Body, které leží v minimální vzdálenosti od dělící roviny, se nazývají support vectors.

$$y_i(w \cdot x_i + b) \geq 1 \tag{4.18}$$

Součet vzdáleností dvou bodů x_k a x_l, krajních bodů obou množin, od dělící roviny je $\frac{2}{\boldsymbol{w}}$. Tuto vzdálenost chceme při trénování SMV maximalizovat. Maximalizace této vzdálenosti se dá převést na problém minimalizovat $\frac{1}{2}w \cdot w$, přitom stále platí omezení (4.18). Tento problém se dá převést na duální problém pomocí Lagrangeova formalismu, který lze řešit pomocí kvadratického programování. Výslednou podobu duálního problému reprezentuje rovnice (4.19), kde α_i představuje Lagrangův násobitel. Při řešení duálního problému bylo zjištěno, že body které nejsou support vector, se na výsledné podobě dělící roviny nepodílejí.

$$\begin{gathered} maximalizuj \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \left(y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j\right) \alpha_j \\ podmínky\ \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0 \end{gathered} \tag{4.19}$$

Využití SVM pro datové množiny, které nejsou lineárně oddělitelné

Pokud data, která v lineárním prostoru nejsou oddělitelná, používá SVM jinou techniku, jedná se o mapování dat do prostoru s vyšší dimenzí označovaného jako $F(\boldsymbol{x})$. Mapování se provádí pomocí tzv. jaderných funkcí. Skalární součin $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$ ve výrazu (4.19) může být nahrazen funkcí $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Tato funkce se nazývá jádrem (kernel function). Je to funkce, která může být aplikována na dvojice vstupních dat k vyhodnocení skalárního součinu v odpovídajícím prostoru. Výpočtem jádrové funkce lze najít lineární oddělovač ve vícerozměrném prostoru.

Jader existuje velké množství, je také možné definovat vlastní jadernou funkci. Skalární součin je někdy nazýván jako lineární jádro. Nalezené lineární oddělovače ve vyšší dimenzi lze mapovat zpět do původního prostoru, čímž lze získat libovolně zvlněné, nelineární hranice mezi pozitivními a negativními případy.

Výsledná funkce pro klasifikaci s použitím libovolného jádra a s vypočítaným optimálním parametrem α_i je vyjádřena (4.20). Násobitel α_i je nulový pro všechny hodnoty, které nepředstavují support vectors. Takže pouze body, které jsou support vectors, jsou použity při výsledné klasifikaci.

$$h(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x} \cdot \boldsymbol{x}_i)\right) \tag{4.20}$$

4.3 Používaná knihovna

OpenCV (Open Source Computer Vision) je volně dostupná knihovna s otevřeným zdrojovým kódem. Je šířena pod licencí BSD (Berkeley Software Distribution). BSD je licence pro svobodný software, a ve své skupině je jednou z nejsvobodnějších. Knihovnu původně vyvíjela společnost Intel, později zpravována společností Willow Garage, nyní se o podporu a vývoj stará společnost Itseez.

Knihovna se především zaměřuje na počítačové vidění a zpracování obrazu v reálném čase. Knihovna je multiplatformní a je vyvíjena v jazyce C/C++. Lze ji využit z prostředí nejen jazyků C, C++ ale také z prostředí jazyka Python a Java. Podporuje operační systémy Microsoft Windows, Linux, Mac OS, iOS a také Android. Knihovna je navržená tak, aby podporovala zpracování na procesorech s více jádry. Například při převodu barevného obrázku z modelu RGB na obrázek v odstínech šedi může cyklus převodu běžet paralelně. Na procesoru se čtyřmi jádry tento cyklus tak zabere teoreticky čtvrtinu času. Podporuje také spolupráci se standardem OpenCL nebo architekturou CUDA. V takovém případě knihovna může využívat hardwarové akcelerace výpočetní platformy.

Knihovna obsahuje řadu funkcí pro použití v počítačovém vidění, detekce objektů v obraze a hledání obrysů. Například algoritmus pro detekování obrysů, který je implementován v knihovně, je postaven na principu popsaném v téhle kapitole.

Verze knihovny OpenCV

Už v historii vedle sebe existovaly dvě verze OpenCV 1.x a 2.x. Starší verze 1.x byla původně vyvíjena pro jazyk C. Když začal být jazyk C++ více populární, vznikla nová verze 2.x. Verze pro jazyk C byla využívána na vestavěných zařízeních, které nepodporovaly jazyk C++. Proto byla stále podporována i tato starší verze. Novější verze se čím dál více zaměřovala na jazyk C++ a přístup pomocí jazyka C se stal pouze zálohou. S ohlášením vydání nové verze 3.0 se starší verze sloučily pod

označení 2.4. Verze 2.4.x se přestala vyvíjet, respektive přestaly se přidávat nové funkcionality. Stále má ale podporu na odstraňování chyb a vylepšování efektivnosti.

Verze 3.x, která je teď aktuální, stále se vyvíjí a jsou zde přidávány nové funkcionality. Verze 3.0 byla v porovnání se staršími mnohem rychlejší a měla o hodně více funkcí.

Na začátku mé práce byla aktuální verze 3.1. Proto jsem ji začal používat pro vývoj programu. Podporuje plně přístup z jazyka C++, je rychlá a je zde předpoklad aktualizování.

4.4 Vlastní implementace a popis programu

Na návrh programu pro detekci dopravních značek bylo kladeno pár omezení. Program měl fungovat za jízdy malého modelu automobilu. Z toho plyne požadavek na zpracování v reálném čase. Program měl být také vytvářen pro vestavěné zařízení, stavěné na platformě ARM (i.MX6) a operačním systémem Linux.

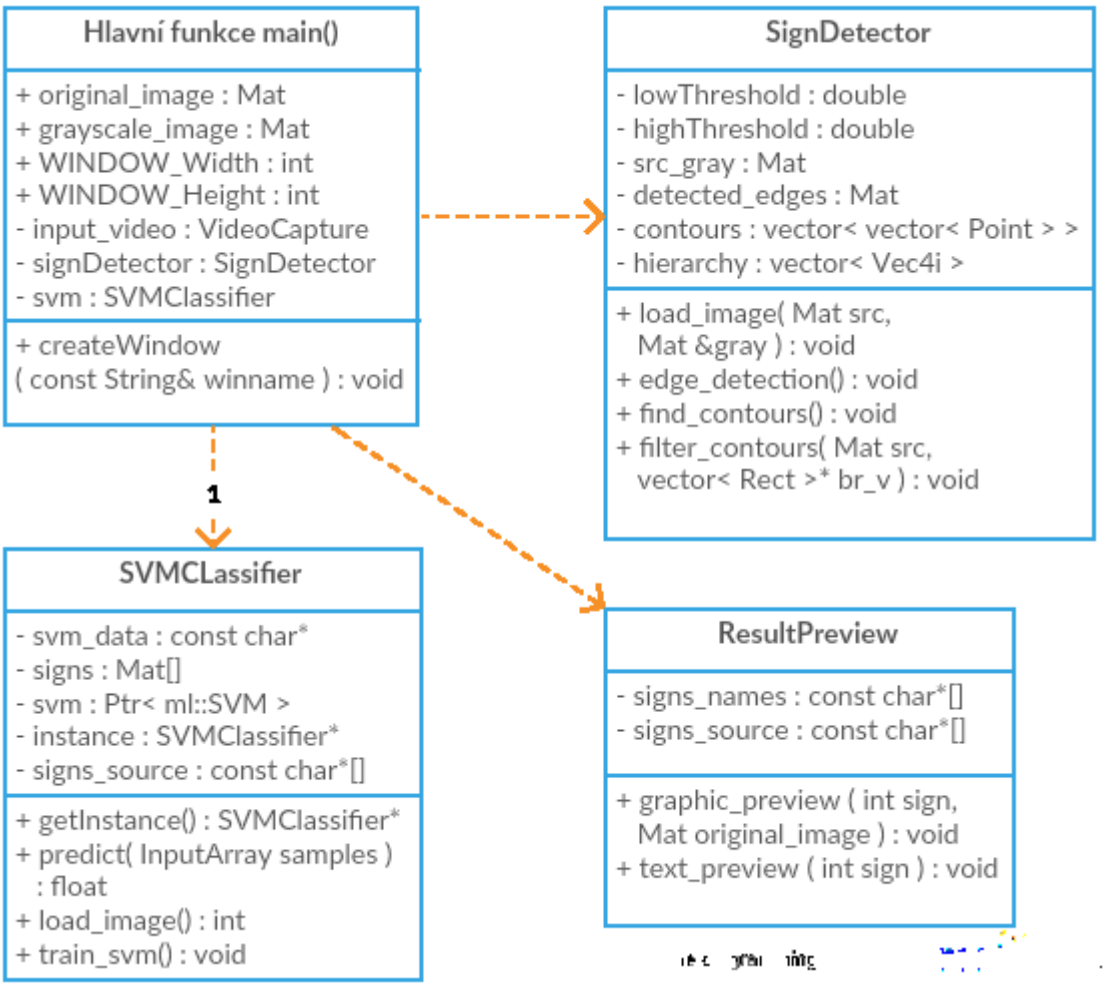
Z toho důvodu jsem pro tvorbu programu zvolil programovací jazyk C++. Program tam může být velmi rychlý, aby splňoval požadavky zpracování v reálném čase. Jazyk C++ také nabízí objektový přístup, což může být výhodou při navrhování programu.

Pro implementaci algoritmů jsem využil knihovny OpenCV, která byla popsána výše. Přestože je knihovna psaná právě v jazyce C/C++, lze jí použít i v jiném programovacím jazyce. Rychlost samotných algoritmů by tak při využití například jazyka Java byla stejná, ale rychlost celého programu by se snížila. I z tohoto důvodu jsem volil právě jazyk C++.

Pro tvorbu programu jsem si vybral vývojové prostředí Code::Blocks. Je to multiplatformní nástroj stavěný pro programovací jazyk C++. Potřeboval jsem testovat program na klasickém počítači a také na malém počítači Raspberry Pi. Code::Blocks je rychlý i na méně výkonném počítači s ním nebyl problém. Nevýhoda tohoto vývojového prostředí je, že nevytváří soubor zvaný makefile. Soubor slouží mimo jiné pro překlad zdrojového kódu z příkazové řádky. To byl zásadní problém při vzdáleném překladu kódu na malém počítači Raspberry Pi, který byl připevněn k modelu auta. V tomto případě se dá použít křížový překlad, kdy upravený kompilátor generuje kód spustitelný na jiném typu procesoru. Ale je nutné mít na počítači, na kterém probíhá překlad, nainstalovanou knihovnu OpenCV. Z tohoto důvodu jsem nakonec přešel na vývojové prostředí Eclipse pro C++. V prostředí Eclipse se k projektu před překladem vytvoří soubor makefile. Při následném zkopírování zdrojových kódů na jinou platformu lze provést překlad bez nutnosti spouštět tam vývojové prostředí.

4.4.1 Struktura programu

Program jsem rozdělil do třech tříd. První třída má název `SignDetector`. Má na starost detekci dopravní značky v obraze. Druhá třída se jmenuje `SVMClassifier` a má za úkol klasifikovat značku, která byla detekována předchozí třídou. Třetí třída `ResultPreview` se stará pouze o zpracování výsledku a jeho prezentaci. Hlavní funkce programu zajišťuje načítání videa z kamery po jednotlivých snímcích. Snímek je dále předán třídě pro detekci. Ta snímek zpracuje a vrátí údaje o možném výskytu dopravní značky. Tento výskyt je předán druhé třídě, která zajišťuje klasifikaci dopravní značky. Výsledek klasifikace je číslo, které je předem definované. Číslo je předáno poslední třídě, která ho dekóduje a vhodně prezentuje výsledek detekce, a to buď graficky, nebo textově. Diagram tříd lze vidět na obrázku 4.10.



Obrázek 4.10: Diagram tříd

4.4.1.1 Třída SignDetector

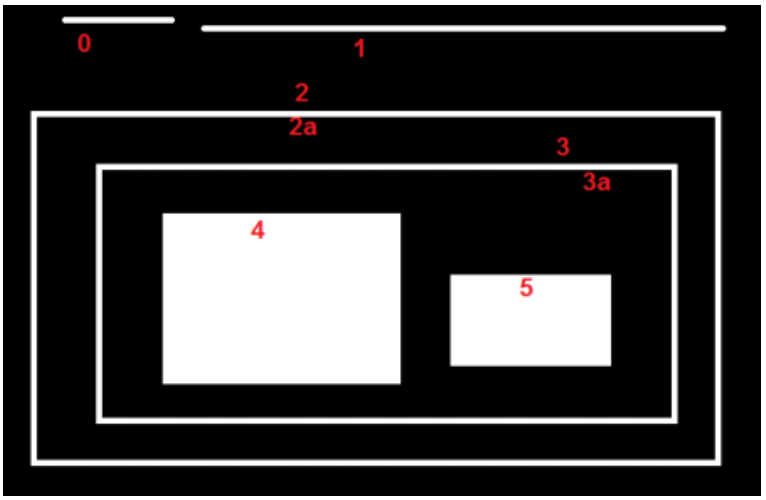
Z kamery je získán obraz reprezentovaný barevným modelem RGB. Tento obraz je předán třídě SignDetector. Pro pozdější potřeby je nutné převést obrázek na model s jedním kanálem. Používám zde převod do odstínu šedi. Je to jeden z nejjednodušších způsobů. Používám k tomu funkci cvtColor z knihovny OpenCV. Vestavěná funkce se ukázala jako nejrychlejší, je totiž optimalizována pro použití na více jádrových procesorech. Zkoušel jsem i experimentální převod, který použil ve své bakalářské práci Jakub Sochor [15] . Hodnota výsledného bodu se skládá z průměru modrého a zeleného bodu v barevném obraze. Ruční převod pomocí procházení celého obrázku byl přibližně 20× pomalejší než při použití vestavěné funkce. Proto jsem se rozhodl používat převod na obraz v odstínech šedi. V tabulce 4.2 je znázorněno porovnání průměrných časů při převodu jednotlivých snímků videa s rozlišením 1920 × 1080 bodů (Full HD).

Tabulka 4.2: Časy převodu obrazu

Funkce	Čas převodu v ms
cvtColor, převod do odstínů šedi	1,7
ruční převod do odstínu šedi	50,9
ruční převod, průměr zelené a modré	43,4

Následně jsou v obrázku detekovány hrany. Obraz je zbaven šumu Gaussovým filtrem s velikostí matice 3×3. Samotná detekce probíhá ve funkci Canny pomocí Cannyho detektoru hran, který byl popsán výše. Pro detekci je potřeba nastavit dva parametry, dolní a horní práh. Pro tyto hodnoty se doporučuje poměr 2:1 až 3:1. Po několika pokusech jsem se vrátil k hodnotám 100 pro dolní a 200 pro horní práh. Jsou to běžně používané hodnoty, v případě potřeby je lze změnit.

Po detekci hran přichází na řadu detekce obrysů. Funkce používá algoritmus popsaný v této kapitole, který se nazývá sledování pomezí. Algoritmus vyhledá a vrátí seznam všech možných obrysů. Každý obrys má své pořadové číslo, číslování začíná od 0. Společně s obrysy je vygenerována hierarchie, která obsahuje informace o topologii všech obrysů. Je zde na výběr více možností výstupní podoby. Já používám pro popis topologie kompletní strom, který představuje něco jako rodokmen. Jsou zde zachyceny všechny vztahy, předchůdce, následník, potomek, rodič. Jedná se o vektor čtyř hodnot, kde hodnota znamená příslušný obrys podle pořadového čísla. Následník, předchůdce představuje další, respektive předchozí obrys ve stejné úrovni. Potomek je obrys zanořený do jiného obrysu. Rodič je pak obrys, který zcela obklopuje obrys svého potomka. Grafický příklad je zobrazen na obrázku 4.11.



Obrázek 4.11: Příklad hierarchie obrysů

(převzato z http://docs.opencv.org/trunk/d9/d8b/tutorial_py_contours_hierarchy.html)

Všechny nalezené obrysy jsou poté filtrovány na základě několika kritérií. Každý obrys musí projít celým filtrem podmínek, aby byl prohlášen za obrys dopravní značky. Napřed vypočítám obsah obrysu. Pokud je hodnota menší, než stanovená hranice, je obrys zahozen. Hranici lze nastavit při vytváření v konstruktoru třídy SignDetector. Výchozí hodnota je nastavena 500,0. Všechny obrysy menší než přibližně 23×23 bodů jsou tak ignorovány. V dalším kroku jsou odstraněny vnitřní okraje. Zde pracuji s vygenerovaným hierarchickým stromem pro nalezené obrysy. Každý uzavřený obrys má dva okraje,

vnější a vnitřní. Každý vnější okraj uzavřeného obrysu má potomka, je jím právě vlastní vnitřní okraj. Když obrys není uzavřený, pak nemůže mít žádného potomka. Neexistuje totiž žádný obrys, který by zcela obklopoval. Obrys, který nemá potomka, je zahozen.

Pro obrys je nyní vypočítán ohraničující obdélník. Obdélník má všechny strany rovnoběžné se souřadnicovým systémem. U obdélníku se ověřuje, jestli nemá příliš rozdílnou výšku a šířku. Pro obraz značky vychází ohraničující obdélník téměř jako čtverec. Výsledný poměr stran musí být v rozmezí od 0,65 do 1,35. Tím se eliminují například detekované obrysy sloupku dopravní značky.

Obrysy, které vyhověly všem podmínkám, jsou považovány za dopravní značku. Ohraničující obdélníky těchto obrysů jsou vráceny. Jelikož se samotná značka skládá z více různých hran, případně písmen a čísel, je pro jednu značku vráceno většinou více než jeden obdélník. Pro další potřeby mi stačí použít pouze největší obdélník, který představuje hlavní obrys nalezené značky. Všechny ostatní, které se nacházejí uvnitř jiného obdélníku, jsou tedy jeho potomky, jsou ignorovány.

Je zde předpoklad, že v obraze bude právě jedna značka. Klasifikovat postupně více značek není problém, ale pro samotné řízení modelu by to představovalo jistou komplikaci.

Z původního obrázku z kamery, který byl převeden do odstínu šedi, je udělán výřez. Velikost a pozice výřezu je stanovena získaným obdélníkem. Ten přesně kopíruje okrajové hrany značky a vybírá tak ideální pozici. Získaný výřez je předán třídě SVMClassifier, která provede klasifikaci.

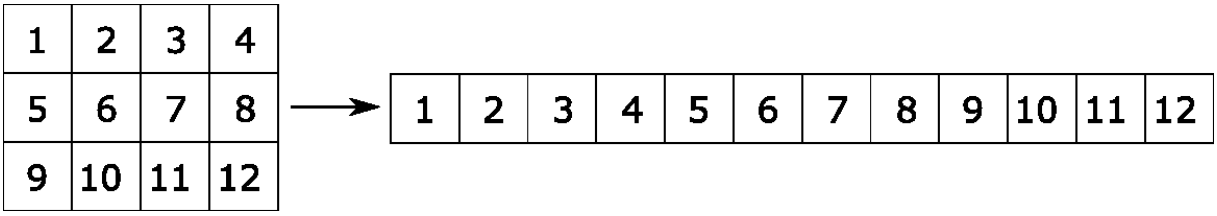
4.4.1.2 Třída SVMClassifier

Třída je navržena podle návrhového vzoru jedináček (Singleton) z důvodu případného načtení stejných dat dvakrát. Do paměti se totiž ukládají natrénovaná data algoritmu. Načítání obrazů a učení algoritmu probíhá na začátku programu. Poté je teprve načteno video a spuštěno rozpoznávání.

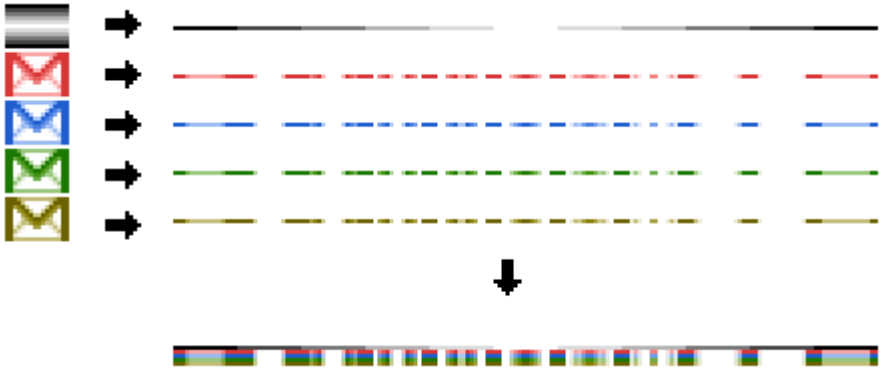
Jako klasifikátor dopravních značek používám algoritmus Support Vektor Machines zkráceně SVM. Princip, na kterém klasifikátor funguje, byl popsán výše. Po vytvoření je nutné algoritmus SVM nastavit. Nastavuje se typ samotného algoritmu a typ používaného jádra (kernel).

Pro klasifikátor je nutné sestavit matici pro trénování. Napřed je nutné načíst všechny obrazy, které jsou určeny jako předloha pro trénování algoritmu. Všechny obrazy jsou převedeny do odstínu šedi a uloženy v klasické matici v poli.

Další důležitý krok spočívá v převedení načtených obrazů do podoby řádkové matice pro účely natrénování algoritmu. Princip je znázorněný na obrázku 4.12. Jednotlivé řádkové matice jsou pak naskládány na sebe, kdy každý řádek představuje jeden obraz předlohy. Tím je vytvořena matice pro trénování 4.13.



Obrázek 4.12: Princip převodu na řádkvou matici



Obrázek 4.13: Skládání matice pro trénování

Pro každý obraz předlohy je vygenerované jednoznačné číslo, takzvaný štítek, které ho identifikuje. Algoritmu je předaná matice pro trénování spolu se seznamem štítků. Štítky slouží pro natrénování předané matice a poté pro zpětnou odpověď při následné klasifikaci. Když je všechno připravené, může se spustit trénování. Natrénovaná data lze následně uložit do souboru ve formátu YAML. To je formát pro uložení strukturovaných dat. Při dalším spuštění programu už nemusí probíhat proces trénování, ale stačí načít uložený soubor. Spuštění je tak trochu rychlejší. Když dojde ke změně vstupních dat pro trénování, je nutné soubor smazat nebo ignorovat a celé trénování provést znova. Poté může být zase uložený nový soubor.

Nyní se dostávám k funkci pro klasifikaci obrazu. Aby bylo možné provést porovnání, je potřeba obraz vzniklý z výřezu upravit na stejné rozměry, jako jsou natrénované obrazy. Testovaný obraz je následně převeden na řádkovou matici. Ta je předložena funkci algoritmu pro predikování. Testovací obraz je porovnán s naučenou maticí vzorků. Je vrácena hodnota odpovídajícího štítku předaná při učení.

4.4.1.3 Třída ResultPreview

Poslední třída slouží k zobrazení výsledků. Podle hodnoty vráceného štítku je vybrána příslušná značka, která byla identifikována v obraze. Pokud je k dispozici pouze textový výstup, je vypsán název nalezené značky. V případě, kdy je možné zobrazit grafický výstup, jsou výsledky zakresleny do aktuálního snímku. Kolem detekované značky je vykreslen zelený obdélník. Nad značku je do obrazu také vložen název značky tak, jak byla značka klasifikována.

Typ detekované značky se dá následně použít pro řízení modelu auta. Ke každé značce je potřeba vytvořit pravidla, podle kterých by se model ovládal.

4.4.2 Vlastnosti programu

4.4.2.1 Klasifikátor SVM

Ze začátku jsem pro trénování algoritmu SVM používal pouze originální obrazy značek. Během testování jsem postupně přidával další výřezy značek z obrazů získaných z videí. Zvažoval jsem použít dostupnou Německou sadu výřezů dopravních značek, ale nakonec jsem od toho upustil. V Německu mají totiž trochu odlišné značky pro přikázaný směr. Negativní sadu jsem vytvářel jako náhodné nebo špatně detekované výřezy.

Testoval jsem také použití různého typu jádra (kernelu). Ze začátku jsem používal polynomiální jádro, u kterého je nutné nastavit parametr stupeň a gama. Později jsem přešel na jádro s názvem průnik histogramu. Nakonec jsem zkusil lineární jádro, které je podle specifikace nejrychlejší. Poskytovalo přibližně stejné výsledky, tak jsem se rozhodl ho nadále používat

4.4.2.2 Více značek v obraze

V reálném provozu mohou být na jednou sloupku pouze dvě značky. Je také stanovena minimální vzdálenost značek mezi sebou. Neměly by se v jednom snímku objevovat více než dvě značky. Tato situace ale může někdy nastat, konkrétně v případě přenosných značek nebo v případě uzavírek a objížděk, kvůli stavebním pracím.

V programu jsem toto omezení neřešil. Pokud je na jednom snímku detekováno více značek, jsou postupně klasifikovány všechny značky. Do výstupního obrázku jsou zakresleny všechny obdélníky detekovaných značek. Všechny značky jsou také klasifikovány a zjištěný název je vypsán nad značku.

V případě používání programu pro následné řízení modelu auta by bylo situaci potřeba vyřešit. Buď omezit počet značek vždy na jednu, což je asi nejlepší řešení. Odpadne starost s rozhodováním, co když bude detekována více než jedna značka. Nebo určit priority pro případ detekce více značek současně.

5 Snímání a detekce značek

Obsahem této kapitoly je popis jak probíhalo snímání dopravních značek. Popisuji zde nastavení pozice kamery nebo volbu rozlišení videa. Dále pak do jaké vzdálenosti od auta a od kraje cesty umístit model dopravní značky a porovnám situaci s reálným světem. Rozeberu vzniklé problémy a navrhnu postup možného řešení.

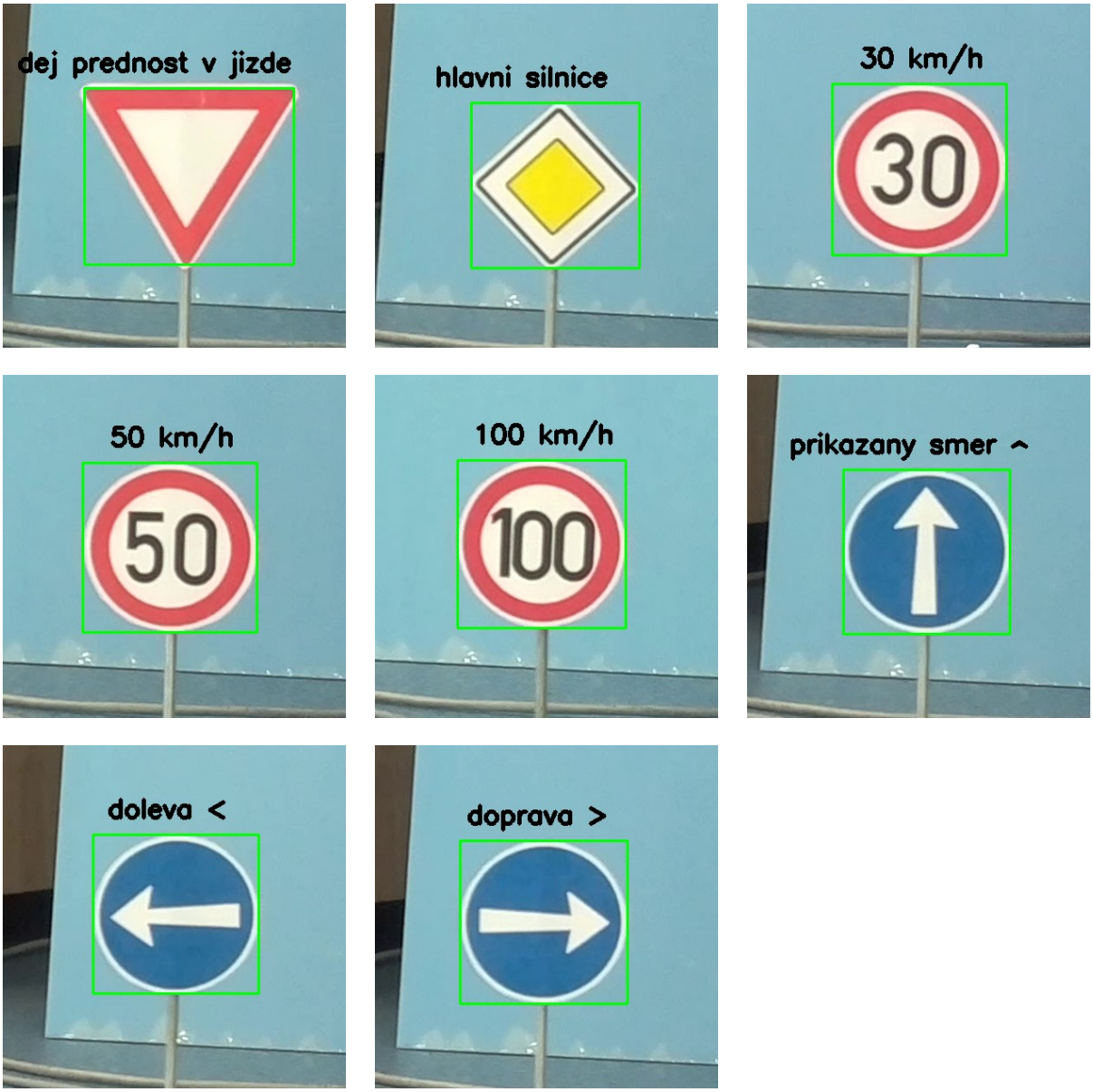
Na začátku kapitoly je ukázka statického rozpoznání značek a na konci jsou ukázky z natočených videí.

5.1 Detekce značek bez pohybu

Než jsem začal s detekcí značek v pohyblivém obraze, vyzkoušel jsem detekci na statických záběrech. Model auta jsem postavil na testovací dráhu a před kameru pokládal jednotlivé značky. Pozadí jsem záměrně zvolil jednolité, abych otestoval detekci a následnou klasifikaci v ideálním případě. Značky byly snímány ze vzdálenosti okolo 30 cm, to odpovídá přibližně vzdálenosti od reálné značky 5,5 metru. Přikládám obrázky všech modelů dopravních značek. První obrázek 5.1 je celkový pohled z kamery. Další obrázky, jsou už jenom ořezané značky získané ze stejného záběru, viz obrázek 5.2

Na obrázcích je vidět kolem značky zelený rámeček. To značí, že byla značka detekována. A černý text nad značkou je označení značky po klasifikaci.

Obrázek 5.1: Statický záběr, značka „STOP“



Obrázek 5.2: Statický záběr, ostatní značky

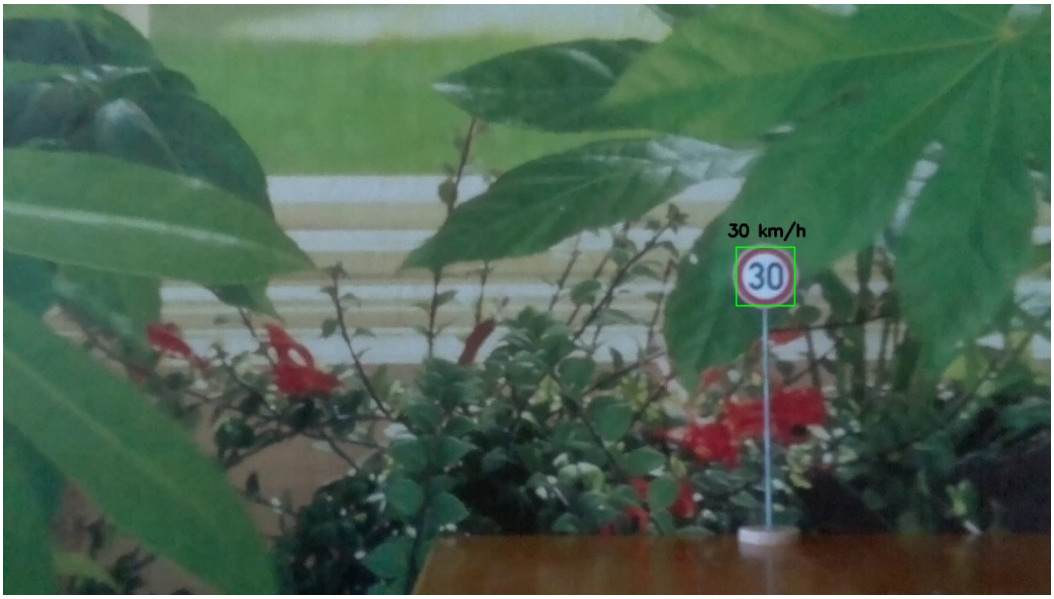
5.2 Detekce značek při pohybu

Další testy jsem prováděl už za pohybu. Počítač Raspberry Pi jsem upevnil k prototypu modelu auta. Auto s kamerou jsem postavil přibližně na vzdálenost 150 cm od značky. To představuje reálnou vzdálenost 27 metrů. Auto jsem posouval ke značce a zaznamenával video. Následující snímky ukazují, kdy byla značka detekována, a detekci v průběhu videa.

Pro značku „30 km/h“ vkládám celé obrázky, aby šlo poznat, že se jedná o pohyb. Tato značka byla detekována ze všech nejlépe. Značka byla zachycena z původní vzdálenosti, tj. 150 cm. Vzdálenost dalších snímku neznám. Šla by přibližně vypočítat podle velikosti značky v obraze.

Třetí obrázek zachycuje poslední detekci značky, dále už byla značka rozmazaná a zmizela ze záběru. Viz obrázek 5.3.


30 km/h
30


30 km/h
30


30 km/h
30

Obrázek 5.3: Detekce při pohybu, značka „30 km/h“

Následující obrázky vkládám oříznuté, vypouštím část, která je nepodstatná, a zanechávám pouze tu potřebnou, na které je zachycena značka.

Značka „100 km/h“ byla detekována o něco později než „30 km/h“, přibližně 70 cm, zato byla detekována těsně na okraji záběru, před tím než část značky zmizela. Viz obrázek 5.4.

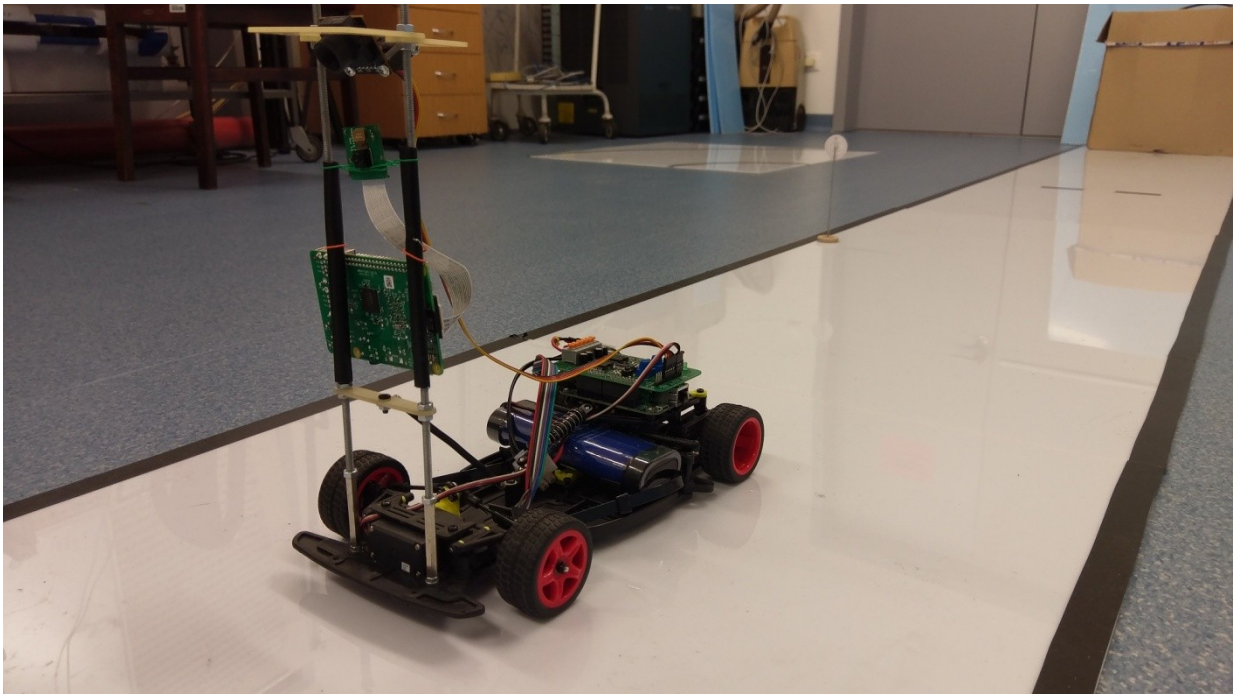
Obrázek 5.4: Detekce při pohybu, značka „100 km/h“

Značka „přikázaný směr jízdy vpravo“, je poslední příklad, který zde uvádím. Značka byla detekována přibližně z dálky jednoho metru, ale na prvním snímku byla špatně klasifikována. Další zachycení značky už bylo v pořádku a byla označená správně. Viz obrázek 5.5.

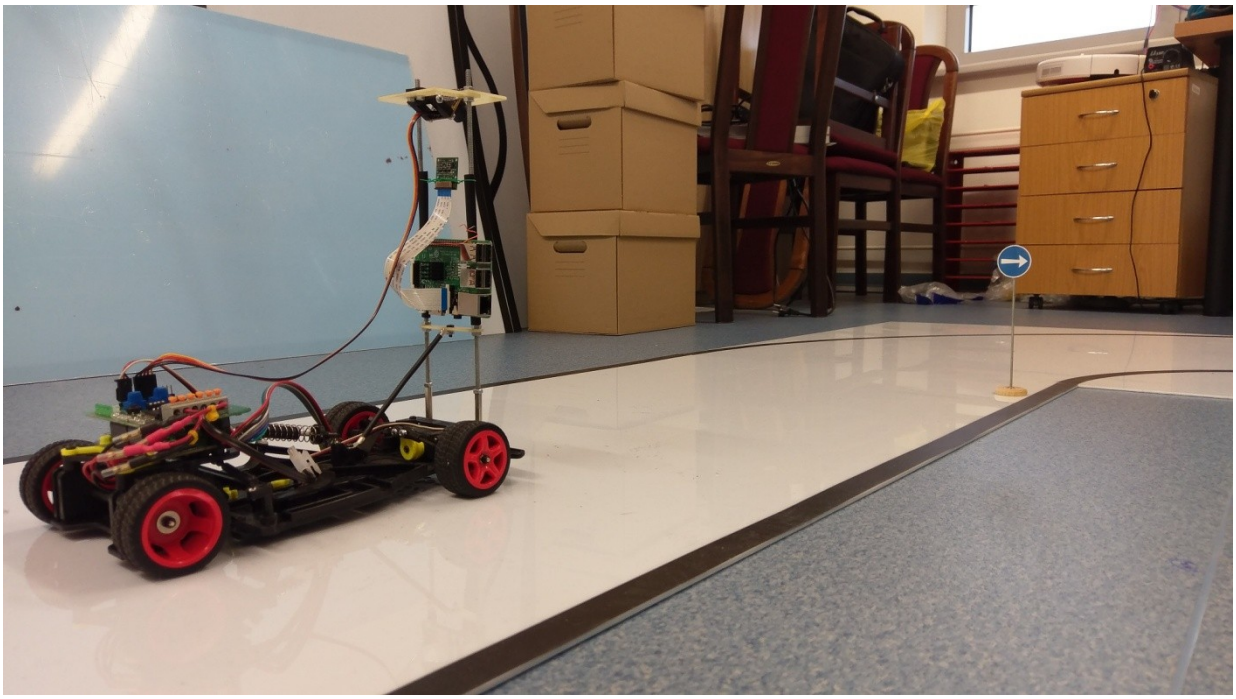
Obrázek 5.5: Detekce při pohybu, značka „přikázaný směr jízdy vpravo“

5.3 Snímání značek v okolí testovací dráhy

Snímání modelů dopravních značek bylo prováděno na testovací dráze. Na modelu auta v měřítku 1:18 byl připevněný malý počítač Raspberry Pi. K počítači byl připojen kamerový modul, který je určený přímo pro Raspberry Pi. Kamera byla namířena tak, aby snímala obraz před autem přibližně tak, jako v reálném autě. Model jezdil po dráze a byly snímány rozestavěné značky v okolí dráhy. Videa se ukládala pro pozdější detekci značek v obraze. Model auta a kamery lze vidět na obrázku 5.6. Model se značkami pak na obrázku 5.7.



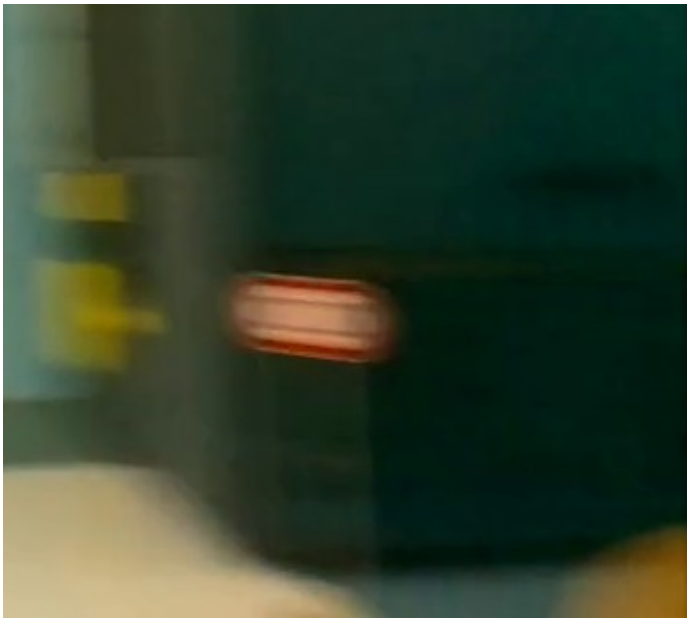
Obrázek 5.6: Model auta na dráze zepředu



Obrázek 5.7: Model auta na dráze zezadu

5.3.1 Umístění značek

Kolem testovací dráhy jsem postupně rozestavoval jednotlivé značky. Pozice jsem napřed vybíral náhodně, ale ukázalo se, že některé pozice jsou zcela nepoužitelné. Například v zatáčce nebyla šance značku zachytit, viz obrázek 5.8.



Obrázek 5.8: Rozmazaná značka v zatáčce

5.3.2 Vzdálenost značek

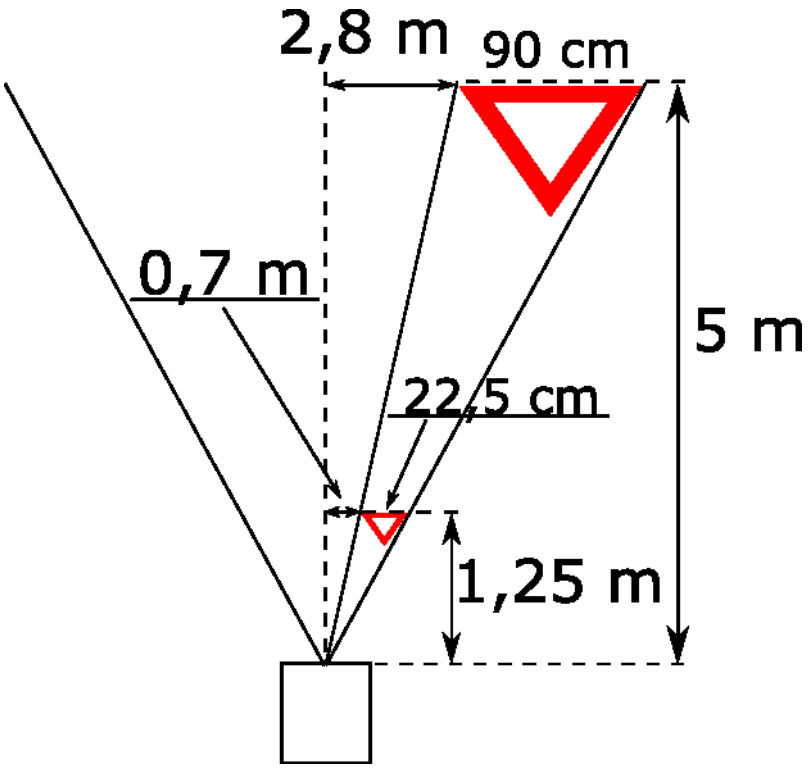
Jeden z problému byla vzdálenost značky, a to jak od kamery ve směru jízdy tak také do strany. Když byla značka umístěna příliš daleko, byla ve výsledném videu malá. A když byla značka umístěna příliš do strany, nebyla v záběru kamery, nebo tam byla jen na okraji.

Testovací dráha je široká 56 cm. Tato hodnota odpovídá reálné cestě o šířce 10 metrů. Podle normy ČSN 73 6110 je šířka jednoho jízdního pruhu na dálnici 3,5 metru. Pro sběrné nebo obslužné komunikace norma udávaná šířku jen 3 metry. Testovací dráha je tedy široká přibližně jako tři normální jízdní pruhy. Za předpokladu že je kamera umístěn uprostřed modelu auta a model jede uprostřed dráhy má tak k okraji vzdálenost 28 cm. To je v reálném měřítku 5 metrů. Kdyby byla kamera umístěna na normálním autě jedoucím také uprostřed jízdního pruhu na cestě široké 3 metry, tak je od krajnice vzdálena 1,5 metru. Model auta tak je od krajnice vzdálen o 3,5 metrů víc, než auto na normální cestě. To by odpovídali situaci, když by auto jelo v protisměru a snímalo značky, které jsou u pravého okraje cesty.

Značky mohou být umístěny podle následujících pravidel. Nejmenší vzdálenost okraje značky od krajnice musí být minimálně 0,5 metru, největší vzdálenost je povolena 2 metry. V měřítku 1:18 to odpovídá vzdálenosti 2,7 - 11,1 cm. Značka umístěná 2 centimetry od okraje dráhy je od jejího středu vzdálena 30 cm. To odpovídá vzdálenosti 5,4 metrů od středu normální cesty. Od krajnice by tak značka byla vzdálená 3,9 metrů. To je skoro dvojnásobek povolené vzdálenosti. Tak vzdálenou značku už by byl větší problém zachytit kamerou na projíždějícím voze, nehledě na úhel natočení zachycené značky, který by byl hodně malý.

Vzdálenost umístění značky hraje také svou roli. Značky se v obci umisťují přibližně kolmo na směr jízdy ve vzdálenosti 50 - 100 metrů. To odpovídá přibližně vzdálenosti 2,8 - 5,6 metrů v měřítku. Myslím si, že detekovat 50 metrů vzdálenou značku za jízdy je velmi obtížné. S kvalitní kamerou se stabilizací obrazu to asi bude možné, ale obyčejnou kamerou ne. Řekl bych, že reálná vzdálenost je přibližně 25 metrů. To odpovídá vzdálenosti přibližně 1,4 metry pro model značky.

Další problém je úhel, ve kterém je kamera schopna zachytit obraz. Takzvaný zorný úhel. Značka, umístěna příliš na straně by nemusela být vůbec zaregistrována. Během jízdy, se snižující se vzdálenosti, dochází ke zmenšování úhlu, ve kterém lze značku pozorovat. Proto je problém jak vzdálenost do strany tak vzdálenost od kamery. Aby se snížila vzdálenost, musí auto přijet blíž, tím se zase zmenší úhel, ve kterém je značka vidět. A značka příliš vzdálené od kraje cesty může být tak při ideální vzdálenosti od kamery mimo zorný úhel. Na příklad značka v měřítku 1:4 je čtyřikrát menší než reálná značka. Aby došlo k zachování velikosti na snímku, musí se vzdálenost čtyřikrát zmenšit. A to i vzdálenost do strany. Situace je zakreslena na obrázku 5.9. (Velikost značky neodpovídá měřítku, je pouze ilustrativní.)



Obrázek 5.9: Příklad zorného úhlu kamery

5.3.3 Rozlišení videa

Rozlišení videa hraje roli v rychlosti zpracování jednotlivých snímků. V době procesorů s jedním jádrem a slabších procesorech s více jádry bylo rozlišení důležité. Ve starších článcích o rozpoznávání dopravních značek se používá rozlišení 640×480. Na procesoru s jedním jádrem trvá zpracování dvakrát většího obrázku čtyřikrát déle. Na dnešních procesorech, většinou se čtyřmi nebo více jádry a vyššími frekvencemi, je zpracování mnohem rychlejší. Navíc lze využít výhody více jader k paralelnímu zpracování.

Pro otestování jsem natočil videa ve třech rozlišeních. 640 × 480, 1280 × 1024 a největší možné rozlišení, které poskytuje modul kamery pro Raspberry Pi, 1920 × 1080 bodů.

5.3.4 Detekce značek ve videu

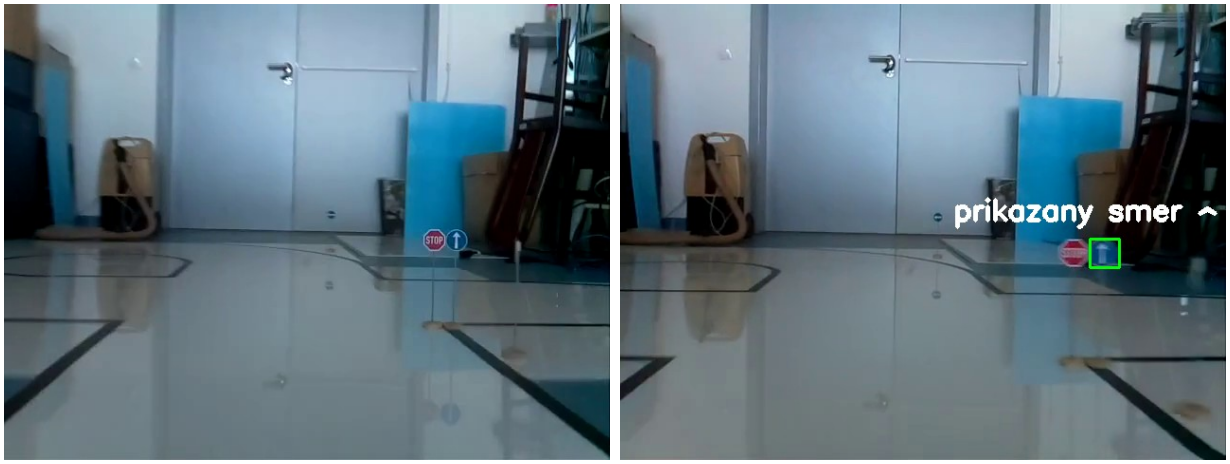
5.3.4.1 Rozlišení 640 × 480

Jako první jsem zkoušel detekovat značky ve videu s rozlišením 640 × 480. Výsledky byli poměrně špatné, hodně značek nebylo detekováno. V natočených záběrech bylo celkem 11 značek. Z těchto 11 možných byly detekovány pouze 3. Značky byly na jednotlivých snímcích hodně rozmazané, a to prakticky znemožňuje detekci hran a obrysů. Některé rozmazané značky byly i přesto detekovány. Přikládám dva příklady detekce značek ve videu v tomto rozlišení.

Obrázek 5.10 zobrazuje situaci detekování značky „hlavní silnice“. Na prvních dvou záběrech je vidět značka „dej přednost v jízdě“, která detekována nebyla. Na prvním snímku nebyl detekovaný obrys, z důvodu velmi podobného pozadí za značkou, a na druhém snímku už je poměrně rozmazaná. Na třetím snímku již detekovaná značka „hlavní silnice“.

Obrázek 5.10: detekce značek na dráze, rozlišení 640 × 480, značka „hlavní silnice“

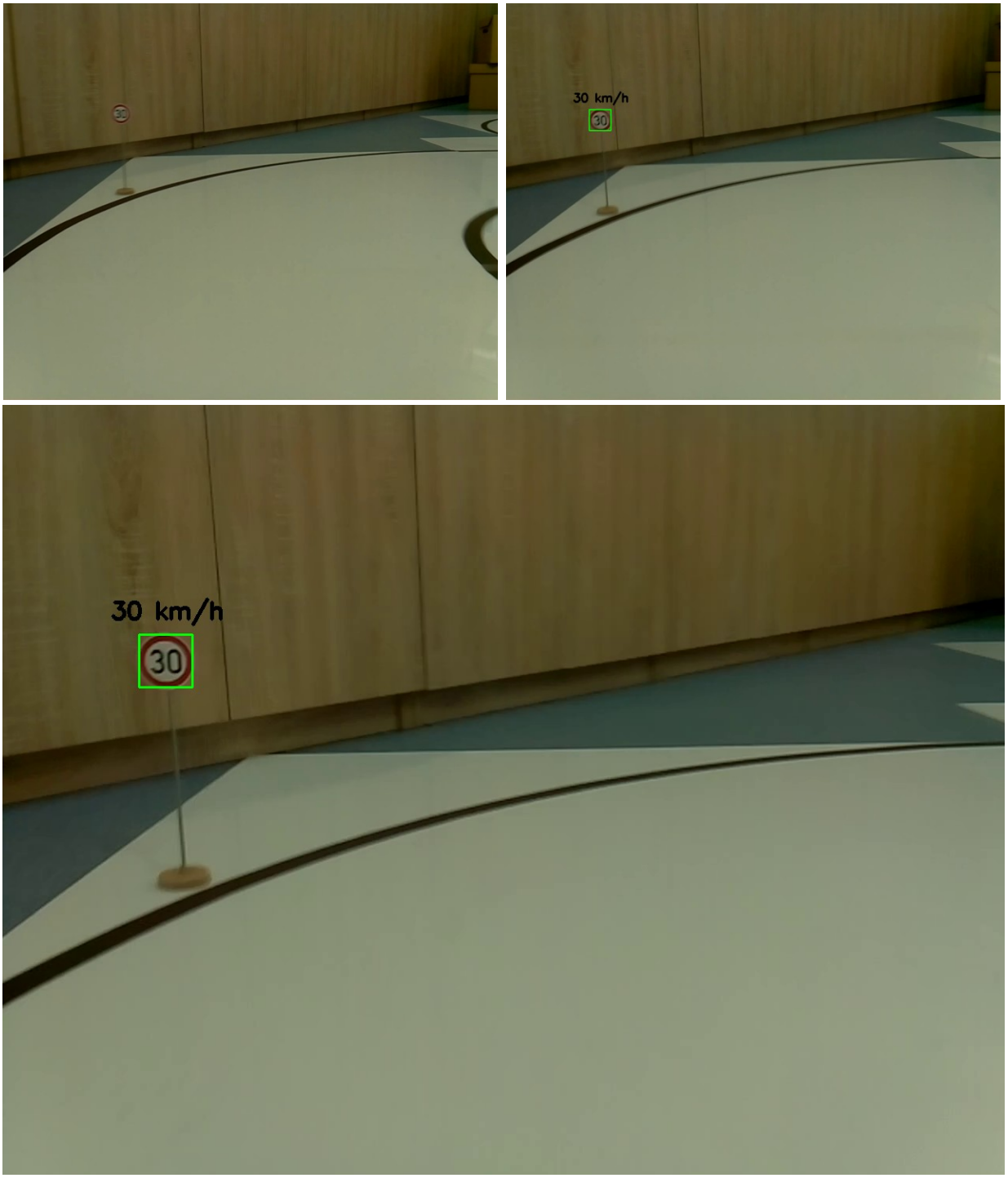
Další obrázek 5.11 zobrazuje detekci značky „přikázaný směr jízdy přímo“. Na prvním záběru značka detekovaná nebyla, i když je ostřejší než na dalším snímku, ale pozadí má velmi podobný odstín jako značka. Značka „stop“ nebyla detekovaná vůbec, zřejmě také kvůli podobnému pozadí, a na druhém snímku je rozmazaná.



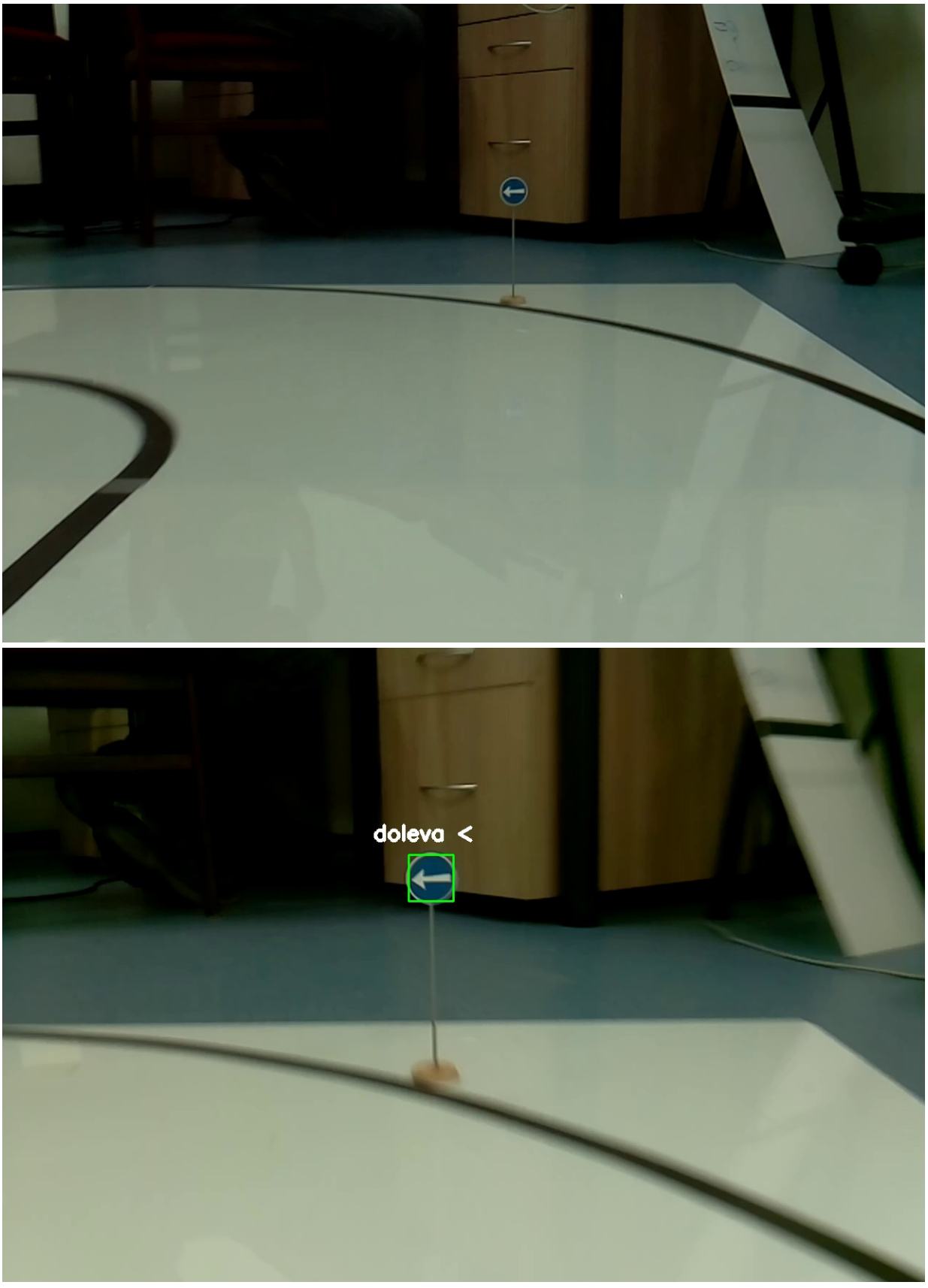
Obrázek 5.11: detekce značek na dráze, rozlišení 640 × 480, „přikázaný směr jízdy“

5.3.4.2 Rozlišení 1280 × 1024

Ve větším rozlišení už byla detekce dopravních značek o poznání lepší. Ze 31 možných značek jich bylo detekováno 19. Značky jsou ve větším rozlišení také větší a dařilo se lépe detekovat okraje. Občas se stalo, že byla značka chybně klasifikovaná. Vkládám dva příklady detekované značky a správně klasifikované. Obrázek 5.12 značka „30 km/h“. Obrázek 5.13 značka „přikázaný směr jízdy vlevo“.



Obrázek 5.12: detekce značek na dráze, rozlišení 1280 × 1024, „30 km/h“



Obrázek 5.13: detekce značek na dráze, rozlišení 1280 × 1024, „přikázaný směr jízdy vlevo“

5.3.4.3 Rozlišení 1920 × 1080

Poslední největší rozlišení už nepřineslo zásadně lepší výsledky. Ty byly přibližně stejné jako v předchozím rozlišení.

Značky jsem natáčel postupně ve třech skupinách podle jejich druhu. Nejlepší výsledky byly u značek přikázaný směr. Přikládám pár detekovaných značek i z tohohle rozlišení viz obrázek 5.14.

Obrázek 5.14: detekce značek na dráze, rozlišení 1920 × 1080, „značky přikázaný směr jízdy“

6 Dosažené výsledky

Jelikož měl být program navržen pro malý počítač a detekce značek měla být prováděna v reálném čase, byla důležitá celková rychlost programu. Navržený algoritmus jsem následně upravoval a snažil se zjednodušit. Odebral jsem například detekci geometrických útvarů, která nebyla využívána.

Další možnost byla provádět výpočty paralelně. Knihovna OpenCV kterou jsem používal, paralelní zpracování plně podporuje. Algoritmy v knihovně jsou také optimalizovány, aby bylo co nejrychlejší.

Problém byl i ten, že jenom načtení jednoho snímku videa z kamery na počítači Raspberry Pi trvalo déle než zpracování celého snímku na výkonnějším procesoru, s kterým jsem rychlost porovnával. Zpracování v reálném čase se dosáhnout nepodařilo.

6.1 Rychlost zpracování

Rychlost zpracování jsem testoval na čtyřech různých rozlišeních a v každém tři různá videa. Výsledky u videí ve stejném rozlišení se nijak zvlášť nelišily. Testoval jsem na počítači se čtyřjádrovým procesorem Intel Core i5 s frekvencí 3,5 GHz. Počítač Raspberry Pi disponuje také čtyřjádrovým procesorem ale s frekvencí 1,2 GHz. Průměrné časy zpracování jsou znázorněny v tabulce 6.1.

Tabulka 6.1: Porovnání rychlosti zpracování

Rozlišení	**Čas Snímky / s**	**Intel Core i5-4690K 3,5 GHz**	**ARM Cortex-A5 1,2 GHz**
640×480	čas zpracování	2,9ms	57 ms
	počet snímků za sekundu	351	18
800×600	čas zpracování	3,4ms	97 ms
	počet snímků za sekundu	294	10
1280×1024	čas zpracování	9,7 ms	228 ms
	počet snímků za sekundu	103	4
1920×1080	čas zpracování	16 ms	355 ms
	počet snímků za sekundu	63	3

Z výsledků je patrné, že pro zpracování v reálném čase by se dalo využít pouze rozlišení 640 × 480. Ale rychlost 18 snímků za sekundu je pořád málo. Pro malé rychlosti by to bylo pravděpodobně možné.

6.2 Kvalita detekce a klasifikace

Detekci hran naruší nebo znemožní rozmazaný obraz značky. Při jízdě je obraz většinou roztřepaný a značky tak nejsou dokonale ostré. Některé značky byly detekovány i mírně rozmazané, jak ukazují příklady v předchozí kapitole. Nejlépe rozpoznané značky byly šipky pro přikázaný směr jízdy.

Při klasifikaci značek docházelo k problémům u rozmazaných značek. Značka byla špatně klasifikována z důvodu špatné detekce.

Značky umístěné v zatáčce se nepodařilo detekovat vůbec, místo značky byla na obraze pouze rozmazaná šmouha. Na reálné cestě se v zatáčce značky nevyskytují, jsou buď před zatáčkou nebo za ní. V zatáčce jsou pouze červenobílé šipky, které upozorňují na zatáčku hlavně v noci.

Datová sada klasifikátoru

Postupným přidáváním výřezů detekovaných značek do klasifikátoru došlo ke zlepšení klasifikace některých značek, na druhou stranu došlo ke zhoršení klasifikace jiných značek. To může být dané tím, že každá značka má jako vzor obrázky v jiném rozlišení a na jiném pozadí. Ideální by bylo vytvořit pro všechny značky sadu obrázků, vždy ve stejném rozlišení a na stejném pozadí. Další příčina může být, že jsou některé značky na obrázku mírně rozmazané.

Tabulka 6.2 zobrazuje dosažené výsledky detekce a klasifikace. Klasifikace udává počet správně určených značek ze značek detekovaných. V případě nejmenšího rozlišení byly správně klasifikovány obě značky, proto 100%.

Tabulka 6.2: Úspěšnost detekce a klasifikace dopravních značek

Rozlišení	Počet značek	Počet detekovaný značek	Úspěšnost detekce	Úspěšnost klasifikace
640 × 480	11	2	11%	100%
1280 × 1024	31	19	61%	46%
1920 × 1080	25	21	84%	62%

7 Závěr

Cílem práce bylo navrhnout program pro detekci modelu dopravních značek. Značky měly být rozpoznávány v okolí testovací dráhy během jízdy malého modelu automobilu. Jeden z požadavků byl zpracování v reálném čase. Detekce měla probíhat přímo za jízdy na malém počítači připevněném k modelu auta. Vytvořený a upravený program ukázal, že použitý počítači není dostatečně rychlý pro zpracování videa v reálném čase.

Pro běh programu na modelu automobilu jsem se rozhodl používat malý počítač Rapsberry Pi. Podle specifikace jsem předpokládal, že bude dostatečně rychlý a zvládne zpracovat video ve větším rozlišení, například HD (1280 × 1024 bodů) nebo Full HD (1920 × 1080 bodů). To se nakonec ukázalo jako mylný předpoklad.

Pro návrh programu jsem zvolil postup detekce založené na tvaru dopravní značky. Během testování se tento přístup moc neosvědčil. Jenom zvyšoval čas výpočtu a neprodukoval dobré výsledky. Pro detekci značek tak používám obrys dopravní značky. Když je detekován obrys, který je uzavřený, je velká pravděpodobnost, že se jedná o dopravní značku. Nalezený obrys následně vyhodnocuje klasifikátor dopravních značek.

Pro rozlišení obrazu 1920 × 1080 se mi podařilo dosáhnout výsledné rychlosti zpracování 63 snímků za sekundu. Rychlost byla ale naměřená na procesoru se čtyřmi jádry o frekvenci 3,5 Ghz. Na malém počítači s procesorem o frekvenci 1,2 GHz a také čtyřmi jádry byla rychlost zpracování pouze 3 snímky za sekundu. To je pro řízení modelu auta v reálném čase nedostačující.

Další vývoj práce by mohl spočívat ve využití lepšího hardware. Například použít lepší kameru se stabilizací obrazu nebo výkonnější malý počítač.

Literatura

[1] Shapiro, Linda G. a Stockman, George C. *Computer vision.* Upper Saddle River, New Jersey : Prentice Hall, Inc., 2001. ISBN 0-13-030796-3.

[2] Szeliski, Richard. *Computer vision: algorithms and applications.* New York : Springer, 2011. ISBN 978-1-84882-934-3.

[3] Garcia-Garrido, M. A., Sotelo, M. A. a Martin-Gorostiza, E. Fast traffic sign detection and recognition under changing lighting conditions. *Intelligent Transportation Systems Conference.* Toronto : IEEE, 2006. stránky 811-816. ISSN 2153-0009.

[4] Goedemé, Toon. Traffic Sign Recognition With Constellations Of Visual Words. *Proceedings of the Fifth International Conference on Informatics in Control.* 2008. stránky 222-227. ISBN 978-989-8111-30-2.

[5] Viola, Paul a Jones, Michael. Rapid object detection using a boosted cascade of simple features. *Computer Society Conference on Computer Vision and Pattern Recognition.* místo neznámé : IEEE Comput. Soc, 2001. Sv. 1, stránky 511-518. ISBN 0-7695-1272-0.

[6] Baró, X., a další. Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification. *IEEE Transactions on Intelligent Transportation Systems.* 2009. stránky 113-126. ISSN 1524-9050.

[7] Upton, Eben a Halfacree, Gareth. *Raspberry Pi Uživatelská příručka.* Brno : Computer Press, 2016. 978-80-251-4819-8.

[8] Raspberry Pi Foundation. Raspberry Pi Documentation. *Raspberry Pi.* [Online] Raspberry Pi Foundation. [Citace: 4. duben 2017.] https://www.raspberrypi.org/documentation/.

[9] Canny, John. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 1986. stránky 679-698. ISSN 0162-8828.

[10] H., M. *Algorithm Designs.* místo neznámé : PediaPress. https://books.google.cz/books?id=98Tx_6SH0kwC.

[11] Wesolkowski, Slawo a Jernigan, Ed. Color Edge Detection in Rgb Using Jointly Euclidean Distance and Vector Angle. Trois-Rivières : Vision Interface, 1999.

[12] Suzuki, Satoshi a Abe, Keiichi. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing.* 1985. stránky 32-46. ISSN 0734189x.

[13] Forsyth, David a Ponce, Jean. *Computer vision: a modern approach.* London : Prentice Hall, 2003. stránky 615-626. ISBN 0-130-85198-1.

[14] Žižka, J. *Support vector machines (SVM).* [Online] 21. 10 2005. [Citace: 15. 4 2017.] https://is.muni.cz/el/1433/podzim2005/PA034/09_SVM.pdf.

[15] Sochor, Jakub. Rychlá detekce dopravních značek v obraze. *bakalářská práce.* Brno : FIT VUT v Brně, 2015.

Přílohy

I. Příloha na CD/DVD

obsah CD/DVD

- Diplomová práce
- Zdrojové kódy programu pro rozpoznávání dopravních značek
- Návod na spuštění programu