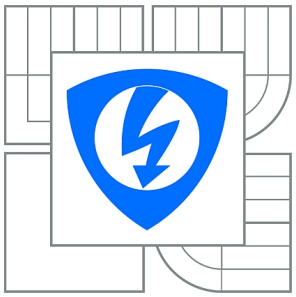
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DETEKCE A ROZPOZNÁNÍ DOPRAVNÍHO ZNAČENÍ

TRAFFIC SIGNS DETECTION AND RECOGNITION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

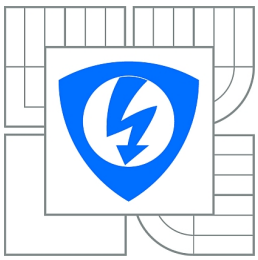
AUTOR PRÁCE — Bc. MICHAL DVOŘÁK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETER HONEC, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Michal Dvořák	***ID:***	140223
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Detekce a rozpoznání dopravního značení

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce bude vytvoření algoritmů pro spolehlivou detekci a rozpoznání dopravních značek v obraze. Zadání bude rozděleno do následujících bodů:

1. Proveďte rešerši používaních metod.
2. Navrhněte knohovnu algoritmů pro detekci a klasifikaci dopravního značení.
3. Implementujte a optimalizujte pro použití algoritmů v reálném čase.
4. Otestujte na datech pořízených v reálném provozu.
5. Vyhodnoťte spolehlivost.

DOPORUČENÁ LITERATURA:

Hlaváč, Šonka, Počítačové vidění.
Šonka, Hlávač, Boyle - IMAGE PROCESSING, ANALYSIS, AND MACHINE VISION,

Termín zadání:	9.2.2015	***Termín odevzdání:***	18.5.2015

Vedoucí práce: Ing. Peter Honec, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem práce je vytvoření algoritmů pro detekci a rozpoznání dopravních značek v obraze. Výsledný program bude zpracovávat reálná data z kamery zabudované v automobilu. Z tohoto důvodu je důraz kladen na spolehlivost detekce a optimalizaci, která povede k minimalizaci využití výpočetních prostředků a schopnost rozpoznávání v reálném čase.

KLÍČOVÁ SLOVA

počítačové vidění, dopravní značení, histogram orientovaných gradientů, segmentace, HSL, detektor Viola-Jones

ABSTRACT

The goal of this thesis is the utilization of computer vision methods, in a way that will lead to detection and identification of traffic signs in an image. The final application is to analyze video feed from a video camcorder placed in a vehicle. With focus placed on effective utilization of computer resources in order to achieve real time identification of signs in a video stream.

KEYWORDS

Computer vision, traffic signs, histogram of oriented gradients, segmentation, HSL, Viola-Jones detector

DVOŘÁK, M. Detekce a rozpoznání dopravního značení. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 62s. Vedoucí diplomové práce Ing. Petr Honec, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Detekce a rozpoznání dopravního značení jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Petru Honcovi, Ph.D. za pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Metody počítačového vidění nalézají v dopravě mnohá uplatnění, ať už se jedná o složky dopravu monitorující či řídící. Oblastí ve které počítačové vidění nalézá velké uplatnění, je pak mimo jiné snaha o monitorování okolí vozidla a poskytování informací o důležitých vnějších událostech, do této kategorie pak bezesporu můžeme zařadit i extrakci informací spojenou s dopravním značením podél, nad, či na dopravních komunikacích.

Systémy zabývající se získáváním těchto informací pak mohou naplňovat službu jak asistenční, kde o způsobu naložení s těmito daty rozhoduje uživatel systému, alternativně pak takový systém může být součástí autonomního zařízení, které tyto informace zohledňuje ve svém rozhodování. V současné době se objevující autonomní silniční vozidla jsou pak bez takovéhoto systému nemyslitelná. Jedním z přístupů k řešení této úlohy je částečné, či úplné spolehnutí na počítačové vidění.

První část této práce si klade za cíl seznámit se s metodami, které mohou být použity k detekci a následnému rozpoznání dopravního značení v obrazu. Druhá část práce, je pak věnována zužitkování těchto informací za účelem vytvoření aplikace, která bude schopná v reálném čase zpracovávat obrazová data. Uvažuje se, že zdrojem obrazových dat bude videokamera a snímky budou pořízeny za jízdy.

Zadání této práce jsem si zvolil právě kvůli skutečnosti, že se tento problém počítačového vidění v posledních letech řešil, tudíž se stále jedná o aktuálně se rozvíjející oblast vývoje. Absence definitivního řešení pak navíc představuje možnost seznámení se a vyhodnocení vícera možných přístupů k řešení jednotlivých částí tohoto zadání.

1 DOPRAVNÍ ZNAČENÍ

Dopravní značení v České republice představuje prostředek k řízení a regulaci silničního provozu na pozemních komunikacích. Dopravní značení v České republice je dle zákona č. 361/2000 Sb., o pravidlech provozu na pozemních komunikacích, stanoveno vyhláškou ministerstva vnitra č. 30/2001 Sb. [1] a vyhláškami, které ji novelizují. Vzhled značek je daný zákonem a to jak barva, tvar, tak případný piktogram nalézající se vně značky.

Tato práce se zabývá především identifikací svislých dopravních značek. Tato skupina může být dále roztříděna, některé možnosti třídění odpovídají platné legislativě ČR a jiné dělíme s ohledem na apriorní informace, s ohledem na které budou dále rozvíjeny principy detekce či identifikace.

1.1 Rozdělení svislých dopravních značek dle vyhlášky

Vyhláška Ministerstva dopravy a spojů 30/2001 Sb. dělí svislé dopravní značky na

- Výstražné dopravní značky – Upozorňující na místa kde hrozí nebezpečí
- Značky upravující přednost – Stanovující přednost jízdě v provozu na pozemních komunikacích
- Zákazové dopravní značky – Ukládající účastníku provozu zákaz nebo omezení
- Příkazové dopravní značky - Ukládající účastníku provozu příkazy
- Informativní značky provozní – Poskytující účastníku provozu nutné informace, slouží k jeho orientaci, nebo mu ukládají povinnosti stanovené tímto zákonem nebo zvláštním právním předpisem

Značky výstražné, upravující přednost, zákazové a příkazové lze nalézt v příloze této práce. Značkami informativními se práce v tomto bodě nezabývá.

Prohlédnutím značek můžeme dále pozorovat opakující se významné vlastnosti, které umožní dělení značek podle jiného systému třídění.

1.2 Rozdělení svislých dopravních značek dle tvaru

Přestože následující pozorování nelze vztáhnout na všechny katalogizované značky, tak můžeme jednoznačně pozorovat, že významnou množinu značek lze identifikovat pomocí jednoho z definovaných tvarů. Využití těchto vlastností je vhodné díky skutečnosti, že rozměry jednotlivých značek, a tím potažmo poměry a úhly, jsou pevně dány a definovány Vzorovým listem VL. 6.1. Rozměry jsou definovány pro trojúhelník, kruh, čtverec, obdélník a osmiúhelník. Rozměry jsou uvedeny v Tab. 1.1.

Tab. 1.1: Základní vnější rozměry pro značky v mm[2]

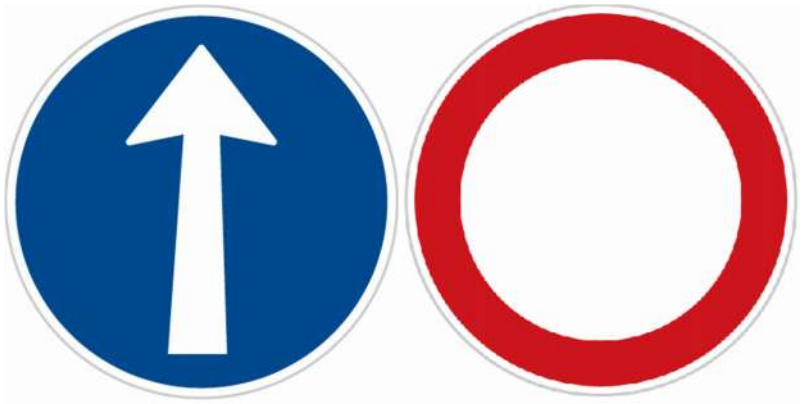
Velikost	Trojúhelník	Kruh	Čtverec	Obdélník	Osmiúhelník
Zmenšená	700	500	-	-	-
Základní	900	700	500	500x700	700
Zvětšená	1250	900	750	dle VL. 6.1	900

Značky výstražné a některé ze značek upravujících přednost odpovídají tvarem trojúhelníku.

Obr. 1.1: Dopravní značky trojúhelníkového tvaru

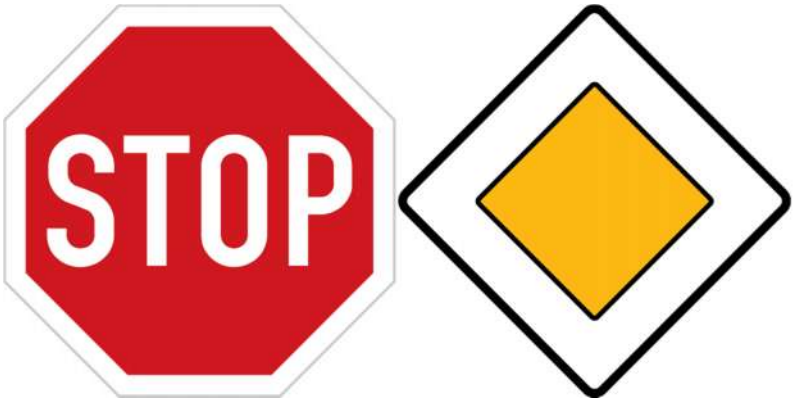
Značka trojúhelníkového tvaru se nachází i v o 180° otočeném tvaru.

Dalším často zastoupeným tvarem je pak kruh. Obvod příkazových a zákazových značek má vždy tvar kružnice, značka *Přednost protijedoucích vozidel* je jediným zástupcem kruhových značek ze skupiny značek upravujících přednost.



Obr. 1.2: Dopravní značky kruhového tvaru

Při návrhu metody, která by využívala apriorní znalost tvaru, by bylo třeba navíc zohlednit alespoň dva další tvary, které jsou pro pohyb po dopravních komunikacích důležité a to diamant a šestiúhelník. Diamant je čtverec otočený o 45°.



Obr. 1.3: Šestiúhelníková a diamantová dopravní značka

1.3 Rozdělení svislých dopravních značek dle charakteristické barvy

Stejně jako generalizovat dopravní značení tvarem, můžeme podobně roztřídit velké množství dopravních značek na základě výrazné barvy. I zde čerpáme z výhody, že barvy jsou pevně definovány.

Mezi nejvýraznější charakteristické barvy existujících svislých dopravních značek patří červená, která je intenzivně zastoupena jak mezi zákazovými a výstražnými dopravními značkami, tak i značkami upravujícími přednost. Můžeme si všimnout, že s výjimkou diamantu je červená zastoupena pro všechny obecné dříve popsané tvary.

Obr. 1.4: Červené značky

Druhou nejčastěji zastoupenou barvou je modrá, kterou můžeme identifikovat valnou většinu příkazových dopravních značek a velkou část informačních dopravních značek. Značka *Přednost před protijedoucími vozidly* je jedinou značkou ze skupiny značek upravujících přednost, jejíž charakteristická barva je modrá.

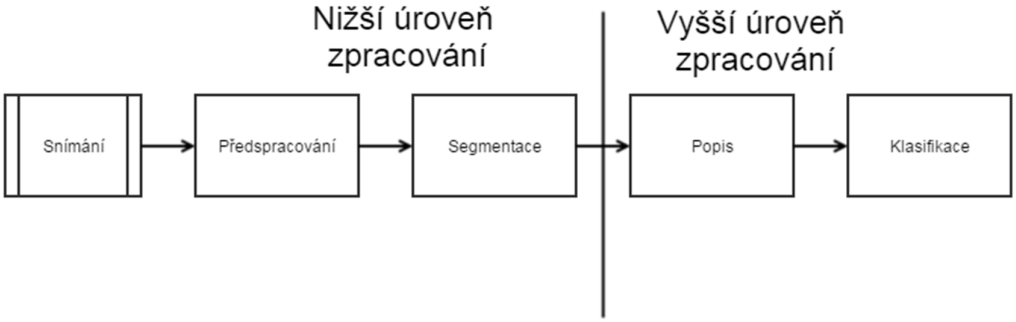
Obr. 1.5: Modré značky

Další barvy, které je třeba sledovat, jsou žlutá, kterou jsme mohli vidět na diamantové značce *Hlavní pozemní komunikace.*

Překážkou pro využití barevné informace je skutečnost, že mezi všemi skupinami značek se nacházejí značky bez charakteristické barvy. Jedná se především o značky ukončující platnost informační či zákazové barvy. Například *Konec nejvyšší dovolené rychlosti* nebo *Konec zóny s dopravním omezením.* Další nesnází je pak skutečnost, že barva značek se vlivem vnějších podmínek mění a různé světelné podmínky mohou vést k různé zjevné barvě. Těmito záležitostmi se bude zabývat část práce rozebírající přístup k řešení problému.

2 METODY DETEKCE DOPRAVNÍHO ZNAČENÍ

Standardní řetězec zpracování obrazu se skládá z jednotlivých operací, které vedou ke kýženému výsledku. Na Obr. 2.1 můžeme pozorovat obecný řetězec operací, kterými při zpracování obrazu projdeme, a které při úspěšné implementaci, vedou k získání žádané informace. Pod pojmem žádané informace v této práci uvažujeme úspěšnou klasifikaci dopravního značení nalezeného v obraze.



Obr. 2.1: Řetězec zpracování obrazu

Detekce dopravního značení tak představuje mezikrok, který je třeba za účelem

úspěšné klasifikace učinit. Tato část práce si klade za cíl popsat metody, s pomocí kterých bude provedeno nízko úrovňové zpracování obrazu především pak operace sloužící k předzpracování a segmentaci.

S ohledem na první část práce, kde jsou identifikovány apriorní informace, které mohou být využity pro segmentaci, budou nyní představeny možné způsoby segmentace. Jak již bylo naznačeno, dvě základní metody uvažují detekci pomocí segmentace na základě známé barvy hledaného objektu a detekci založené na s využití informací tykající se tvaru hledaného objektu.

Za účelem konečného ověření hotové aplikace byla zajištěna databáze snímků z projektu The German Traffic Sign recognition Benchmark[3], ve kterém se nalézá 600 snímků pořízených z vozidla. V těchto snímcích jsou hledané objekty identifikované a jejich souřadnice katalogizované dále databáze obsahuje 300 obrázků bez těchto informací. Snímky mají rozlišení 1360x800 pixelů. Z popisu databáze víme, že velikost jednotlivých značek ve snímku je v rozpětí 16x16 – 128x128px.

Pro učení byla ze stejného ústavu zajištěna databáze GTSRB German Traffic Sign Recognition Benchmark. Tato databáze již neobsahuje obrazy celé scény, nýbrž pouze výřezy s identifikovanými značkami. Databáze obsahuje 43 typů dopravních značek v zastoupené mezi 360 až 2250 značek na kategorii. Vzhledem k chybějícímu pozadí, budou tyto snímky použity pro účely učení.

2.1 Segmentace s ohledem na tvar

Díky skutečnosti, že tvar dopravního značení je definován zákonem s ohledem na snadnou zpozorovatelnost ve scéně, dá se předpokládat, že využitím metod hranové detekce a následného hledání objektů splňující požadavky na předpokládaný tvar, povede k úspěšné detekci dopravních značek.

Metody pro detekci hran, jak název napovídá, slouží k hledání významných hran v obraze. Hrana je v optimálním případě reprezentovatelná informací, která definuje hranice objektu. Matematicky vzato pak hrana představuje místo v obrazu, kde mezi dílčími částmi dochází ke strmé změně. Tato strmost nám následně určuje magnitudu hrany, zatímco normála pak její směr. Šum v obraze a skutečnost, že magnituda může nabývat jak skokových změn, tak pozvolných, činí z problému záležitost netriviální.

Dvourozměrný gradient může být reprezentován vektorem dle následujícího zápisu.

$$\nabla f(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f(x,y)}{\delta x} \\ \frac{\delta f(x,y)}{\delta y} \end{bmatrix} \quad (1)$$

Kde složky G_x a G_y udávají změnu podél osy x a osy y. Absolutní velikost gradientu pak v závislosti na požadavky na přesnost, může být vypočítána jako délka přepony, kde Gy a Gx představují odvěsny, alternativně lze tuto hodnotu aproximovat, například jako maximální hodnotu složek, či jejich součet.

Protože u úloh počítačového vidění nejčastěji pracujeme s diskrétními obrazy, gradient změny je pak nejčastěji aproximován jako diference mezi jednotlivými

obrazovými body.

$$G_x = \frac{\delta f(x,y)}{\delta x} \approx \Delta_x f(x,y) = f(x,y) - f(x-n,y) \quad (2)$$

$$G_x = \frac{\delta f(x,y)}{\delta x} \approx \Delta_x f(x,y) = f(x,y) - f(x-n,y) \quad (3)$$

Často se pak detekce hran realizuje pomocí konvoluce a vybraných maticových operátorů. Diskrétní konvoluce pro práci s obrazy je aproximována následovně.

$$g(x,y) = f(x,y) * h = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) \cdot h(i,j) \quad (4)$$

Jako maticový operátor můžeme zvolit jeden z existujících hranových detektorů či podobný vycházející z obecného principu. Níže následují nejznámější maticové operátory. Je také třeba pamatovat na to, že hranové detektory založené na 1. derivaci nejsou hranově invariantní a úspěšnou detekci hran je třeba operátory rotovat a aplikovat pro hrany více orientací.

$$h_x = \begin{bmatrix} -1 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad (5)$$

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (6)$$

$$L_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (7)$$

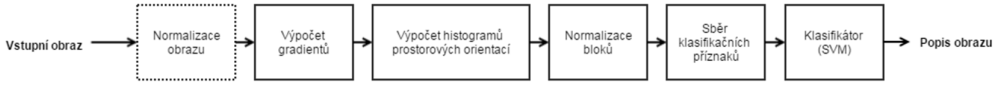
Konvoluční maska (5) představuje operátor pro obecnou diferenci, operátor (6) je pak zástupcem hranových detektorů aproximující 1. derivaci, známý jako operátor Prewittové, a operátor (7) pak hranový detektor aproximující 2. derivaci, jmenovitě se jedná o Laplaceův operátor, v tomto případě pracující pro čtyřokolí obrazového bodu. Hrana se u prvních dvou operátorů nachází v oblastech maxima, třetí operátor využívající 2. derivaci pak hranu indikuje průchodem nulou.

Při volbě vhodného operátoru je třeba zvážit, jestli a do jaké míry se bude v obrazu nacházet šum. Ten je třeba vhodnou metodou potlačit, aby nebyl identifikován jako falešná hrana. Velké konvoluční jádro může na jednu stran potlačit šum, ale na druhou stranu zvyšuje výpočetní náročnost této operace, což je obzvlášť u aplikací s požadavkem na zpracování v reálném čase, třeba zohlednit.

Na základě rešerše byla identifikována metoda Histogramu orientovaných gradientů (HOG) jakožto postup vycházející z hranové detekce, která díky využití jednoduchého konvolučního jádra může být dostatečná pro splnění rychlostních požadavků, a navržené

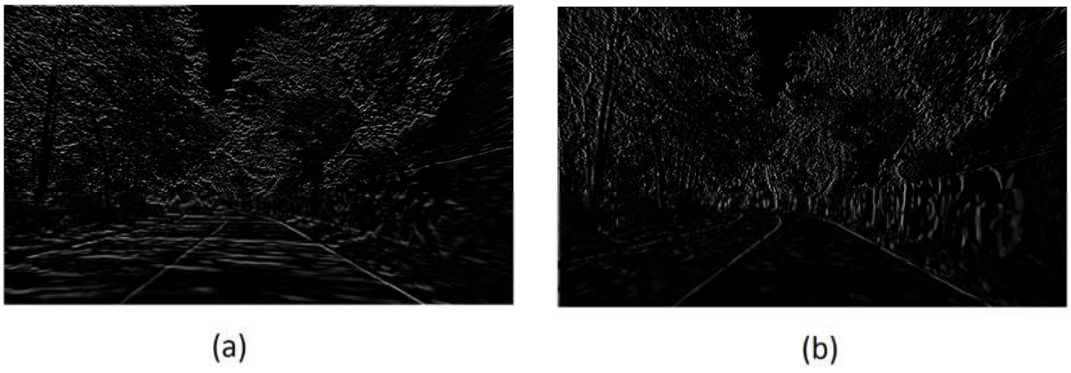
metody zpracování a učení pak mohou vést ke spolehlivé detekci tvarů [5].

Původně navržený algoritmu, prochází sérií kroků, které obsahují metody předzpracování, výpočty příznaků a následný popis jakým způsobem lze zjištěné příznaky využít k detekci hledaného tvaru. Přestože původně byl algoritmus navržen jako prostředek k detekci chodců, poslední krok jeho implementace, je binární klasifikátor a pouze zvolené positivní obrazy následně určí, které objekty budou detekovány. Blokové schéma HOG je uvedeno na Obr. 2.5, v následující části jsou jednotlivé části tohoto schématu popsané.



Obr. 2.2: Schéma HOG

- Normalizace obrazu je uvedena jako první krok, ale autoři této metody konstatují [5] že efekt normalizace na výslednou úspěšnost má pouze marginální efekt.
- Výpočet obrazů gradientů se provádí pomocí operace konvoluce. Výsledné obrazy derivací pro směr x a y nabízejí informace o hranách, které metoda následně využívá. Pro výpočet může být použita libovolná konvoluční maska navrhnutá pro hranovou detekci, leč autoři v práci [5] došli k závěru, že konvoluční masky na Obr. 2.4 vedou k nejkvalitnějším výsledkům. Na obrázku vidíme obrazy gradientů dx a dy



Obr. 2.3: Obrazy gradientů

- Histogramy jsou vypočítány v několika krocích. V první řadě zvolíme počet orientací, do kterých budeme deriváty dx a dy dělit. Na rozsahu 0-360° nejčastěji volíme 9 tzv. binů a v závislosti na orientaci gradientů rozdělíme váhu hrany do příslušného binu. Nad obrazem těchto vektorů uvažujeme síť o zvolené velikosti bloků, např. 8x8. Pro každý krok zkonstruujeme histogram hodnot pro jednotlivé biny. Je možné pracovat jak s šedotónovým obrazem, tak s barevným. V případě práce s barevnými obrazy, autoři použili při konstrukci histogramu tu složku, která vracela nejsilnější vektor.

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad [-1 \quad 0 \quad 1]$$

(a) (b)

Obr. 2.4: Konvoluční maska pro gradient dy (a) a dx (b)

- Zvolenou množinu bloků v předchozím kroku následně normalizujeme. Nejčastěji se využívá množina 2x2 či 3x3.

Pro jasnější představu jaké příznaky můžeme z obrázku metodou HOG získat, jsme použili funkci extractHOGFeatures(), která provede všechny kroky metody HOG až na finální klasifikaci pomocí SVN klasifikátoru. Na Obr. 2.5 si můžeme prohlédnout vstupní obrázek zpracovaný metodou HOG. Scéna byla zvolena, protože obsahuje jak kruhovou dopravní značku zákazu, tak trojúhelníkovou výstražnou značku.

V ukázce si můžeme všimnout, že příznakový vektor v částech pozadí nevykazuje významnou charakteristiku. Příznakové oblasti, ve kterých se nachází hrana, naopak významnou charakteristiku vykazují. Účelem vhodně zvoleného klasifikátoru je detekce shluků příznaků, které s vhodnou tolerancí identifikují příznaky odpovídající hledanému objektu.

Obr. 2.5: Ukázka Histogramu orientovaných gradientů

2.2 Segmentace s ohledem na barvu

2.2.1 RGB prostor

Přestože hlavní zabarvení dopravního značení je definováno legislativou, nelze tuto informaci využít triviálním způsobem. Za ideálních podmínek by v prostoru RGB bylo možné vstupní snímek prahovat pomocí hodnot popsaných v Tab. 2.1. Leč protože pracujeme s obrazovými snímky z reálného světa, podmínky nejsou ideální. Největším problémem při práci s barevným modelem RGB jsou světelné podmínky a vliv vnějších podmínek na barvu dopravního značení.

Tab. 2.1: Teoretické ideální hodnoty barevných složek pro jednotlivé třídy dopravního značení

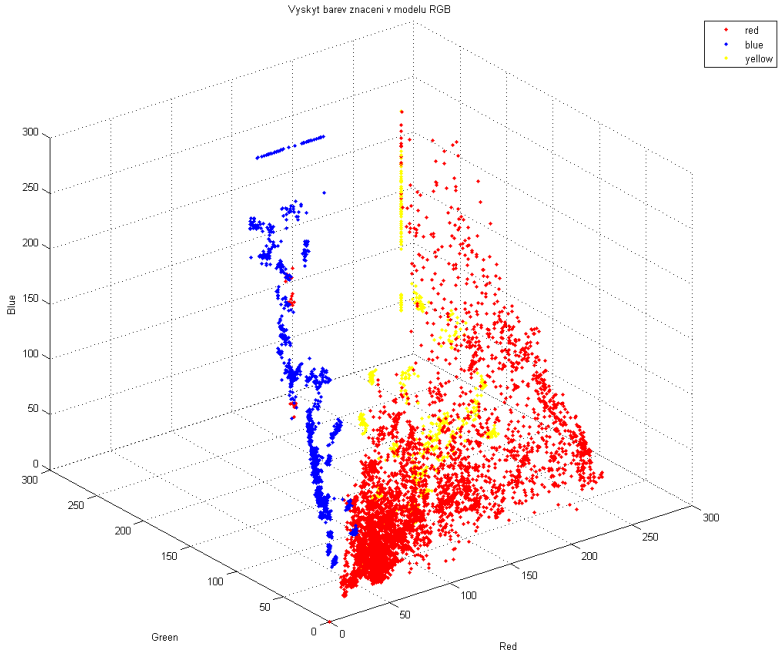
	Hodnoty barevných složek		
Barva značky	**R**	**G**	**B**
Červená	255	0	0
Modrá	0	0	255
Žlutá	255	255	0
Bílá	255	255	255

Za účelem teoretického rozboru byl proveden průzkum s cílem určit, v jakých částech barevného modelu RGB (a posléze HLS) se nachází barvy dopravního značení na snímcích. Na každém z 250-ti snímků byly zvoleny 3 body, pro které byly zjištěny hodnoty jednotlivých barevných složek. Aby se minimalizoval vliv šumu pro případné vybrané pixely a množina datových sad byla co největší, byly zároveň zjištěny hodnoty barevných složek v osmiokolí zvoleného pixelu.
Tímto způsobem byla získána datová sada obsahující následující počet barevných informací.

Tab. 2.2: Datová sada pro účely segmentace s ohledem na barvu

Barva	**Počet hodnot**
Modrá	1242
Červená	4590
Žlutá	756

Datová sada byla následně projektována do 3D prostoru a do všech kombinací 2D rovin RGB, jak si můžeme na Obr. 2.6 a obrázcích v příloze A Obr. 7.1, Obr. 7.2, Obr. 7.3 všimnout. Byť můžeme pozorovat, že modrá barva od žluté a červené diverguje, žlutá a červená mají v projektovaném prostoru překryv, je také třeba konstatovat, že žlutá a hlavně pak červená složka vykazuje v prostoru RGB velký rozptyl.



Obr. 2.6: Zastoupení barev značení v RGB prostoru

2.2.2 IHLS prostor

Datová sada byla taktéž transformována do barveného modelu IHLS[4]. Výhoda tohoto barevného modelu spočívá v oddělení světelné složky od barevné.
Rovnice (8), (9), (10) a (11) byly použity pro transformaci z RGB do IHLS.

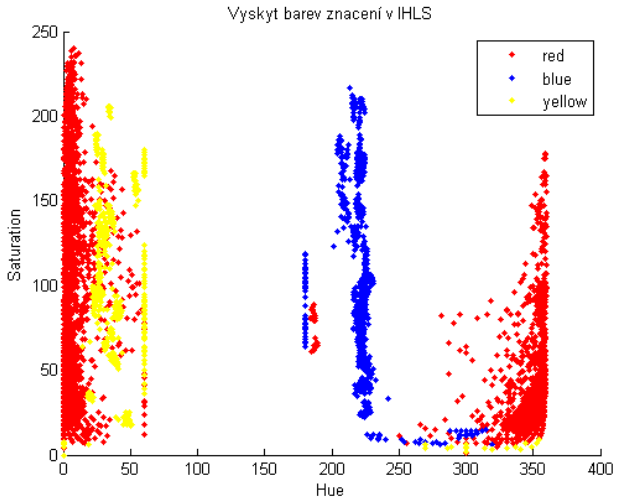
$$Y(c) = 0.2126R + 0.7152G + 0.0722B \quad (8)$$

$$S(c) = \max(R,G,B) - \min(R,G,B) \quad (9)$$

$$H'(c) = \cos^{-1}\left[\frac{R - \frac{1}{2}G - \frac{1}{2}B}{(R^2 + G^2 + B^2 - RG - RB - BG)^{\frac{1}{2}}}\right] \quad (10)$$

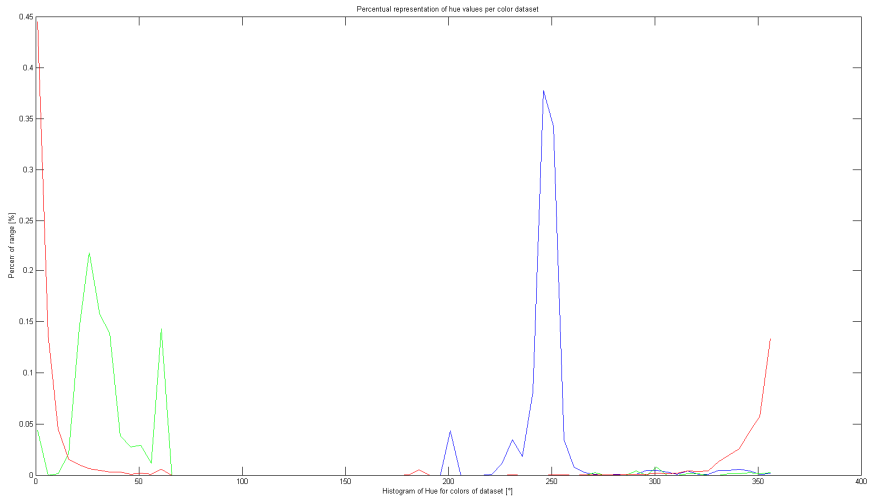
$$H(c) = \begin{cases} 360° - H' if\ B > G \\ else\ H' \end{cases} \quad (11)$$

Transformovaná data byla následně projektována do 2D roviny hue a saturation. Jak je z obrázku Obr. 2.7 zřejmé, rozdělení hlavních barev zájmu je zde mnohem jednoznačnější. Data z červených, modrých a žlutých bodů značek se nalézají v úzkých intervalech saturace.



Obr. 2.7: Zastoupení barev značení v prostoru IHLS (složky HS)

Vypočítáním histogramu, kde osa x reprezentuje hodnotu odstínu (hue), v intervalu po 5ti stupních, získáme následující graf Obr. 2.8. Výskyty na interval byly poděleny celkovým množstvím prvků v datové sadě, osa y tak reprezentuje relativní zastoupení odstínu pro respektivní datovou sadu. V grafu můžeme pozorovat, že pro červené a modré značky jsme schopni identifikovat dominantní odstín a skutečnost že v blízkém okolí barvy ztrácejí na výskytu. Žlutá barva se pak vymyká více výskyty v širším intervalu odstínů a jako jediná obsahuje dva dominantní odstíny.



Obr. 2.8: Histogram zastoupení složek Hue

Jako koncepční příklad je na obrázku zobrazeno prahování s použitím získaných informací, na obrázku je využit rozsah odstínů <355,6°>, je vhodné podoktnout, že se v obraze nachází velké množství nechtěného šumu. Tento šum by pak bylo třeba dalším způsobem odstranit. Například morfologickými operacemi, či odstraněním objektů

nesplňujících vybrané radiometrické deskriptory. Kdyby tak nebylo učiněno, nastala by situace, kde by se popisná část algoritmu věnovala i těmto nechtěným částem. Vzhledem k tomu, že se jedná pouze o průkaz konceptu, můžeme konstatovat, že hledané objekty jsou v obraze jednoznačně zvýrazněné, viz Obr. 2.9 k nahlédnutí.

Obr. 2.9: Ukázka obrázku filtrovaného v IHLS prostoru

2.3 Detektor Viola-Jones

Metoda detekce objektů navrhnutá Michelem Jonesem and Paulem Violou byla původně navržena za účelem detekce obličejů v reálném čase [6]. Detektor Viola-Jones vychází z myšlenky, že množina části lidského obličeje zachyceného na snímku je jednoznačně odlišitelná od pozadí, problémem však je, že jednoznačná jednokroková detekce je výpočetně příliš náročná, než aby se dala provádět v reálném čase, namísto toho využívá detektor velké množství jednoduchých příznaků spojených do jednoho silného detektoru. Za účelem vyšší rychlosti jsou využity následující principy.

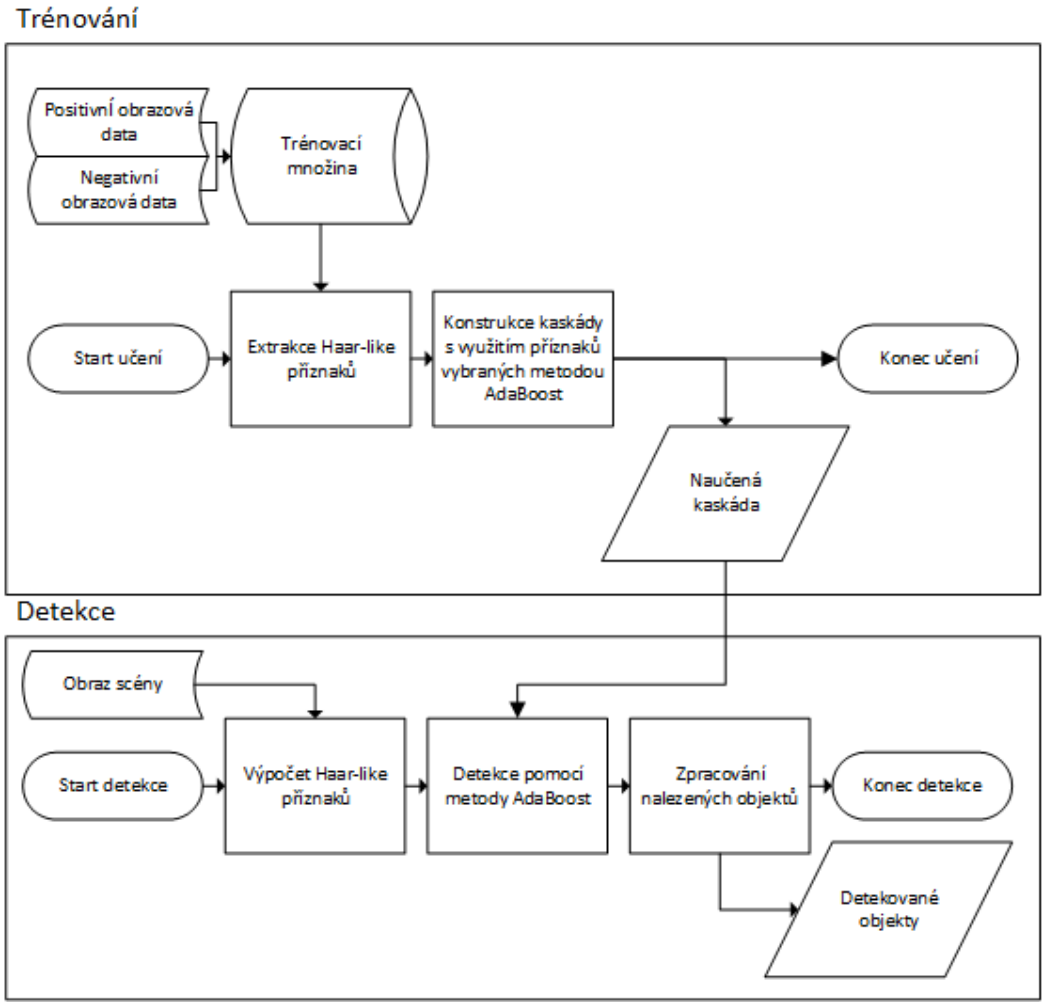
Obraz je v prvním kroku převeden do šedotónového obrazu, což na jednu stranu vede ke ztrátě dvou třetin obrazové informace, ale metoda počítá s tím, že hledaný objekt je detekovatelný i v šedotónovém obrazu a ztráta informace je přijatelná s ohledem na následné snížení nároků na výpočetní výkon.

Metoda pro detekci pravděpodobných částí lidského obličeje využívá Haarových filtrů. Tato jednoduchá obrazová primitiva jsou využita v lokální jasové transformaci pro výpočet obrazových příznaků. V této kapitole se s nimi seznámíme blíže a na úvod tak stačí říct, že za účelem efektivnějšího výpočtu těchto příznaků, je šedotónový obraz převeden do takzvaného integrálního obrazu. V tomto obrazu jsou tyto jasové transformace konstantně výpočetně náročné. Jinými slovy ať výpočet lokální transformace provádíme pro libovolně velkou část původního obrazu, doba výpočtu by měla zůstat zachována pro všechny zvolené velikosti.

Rychlost výpočtu je dále dána vhodným výběrem příznaků, detektor Viola-Jones využívá pro tento účel metodu strojového učení AdaBoost. Tato metoda identifikuje vhodnou množinu příznaků, která pro množinu tréninkových obrazových vzorů bude splňovat požadavky na dostatečně vysokou pravděpodobnost detekce skutečně pozitivních případů a dostatečně nízkou pravděpodobnost detekce falešně pozitivních případů. Tímto způsoben je nalezeno množství snadno vypočitatelných příznaků, skrze které je hledán, v porovnaní s nimi, komplexní objekt.

Konečně kaskádová architektura detektoru umožňuje vyloučení sub oken obrazu hned jak je rozhodnuto, že se nejedná o hledaný objekt. Tímto způsobem je možno algoritmus výpočtu pro vybrané sub okno ukončit předčasně a dále tak celkový výpočet zrychlit.

Na Obr. 2.10 si můžeme prohlédnout blokové schéma ilustrující způsob, jakým detektor Viola-Jones pracuje. Stejně jako algoritmus je obrázek rozdělen na dvě fáze, fázi trénovací a detekční respektive.



Obr. 2.10: Blokové schéma detektoru Viola-Jones

2.3.1 Haarovy filtry a integrální obraz

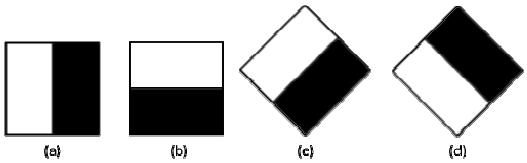
Haarovy filtry představují primitivní plošné detektory, kde vypočítaný příznak představuje rozdíl sum jasových hodnot pod světlou a tmavou částí Haarova filtru. Původní sada Haarových filtrů byla navržena ve článku [7], ve které byly popsány základní obdélníkové filtry. S touto sadou pracuje i detektor v původně popsaném detektoru Viola-Jones. Následně však byla sada rozšířena o 45° natočené příznaky v práci R. Lienharta a J. Maydta [8]. Tyto upravené filtry byly v tomto články popsány včetně jejich vlivu na výsledky Viola Jones detektoru.

Haarovy filtry podle svého návrhu dělíme na několik skupin. Obr. 2.11 zobrazuje filtry, které slouží k hranové detekci, Obr. 2.12 a Obr. 2.13 pak filtry určené k detekci čáry. S pomocí filtrů vyobrazených na Obr. 2.14 detekujeme silné středové příznaky.

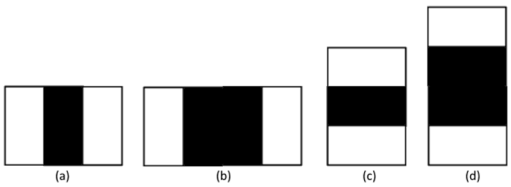
Metoda uvažuje, že výpočtem Haarových příznaků všemi či vybranými typy filtrů a jejich zvětšených variant budeme schopni najít dostatečné množství slabých detektorů, tedy detektorů, jejichž úspěšná detekce je vyšší než 50%, ze kterých bude možné postavit

detektor silný.

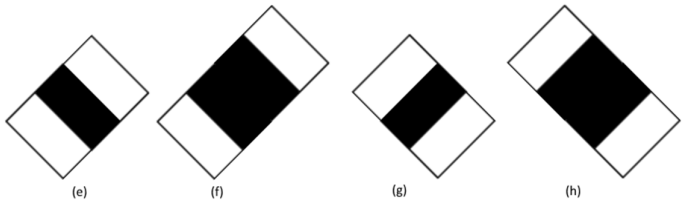
Článek Prof. Lienharta demonstruje, že pro okno o velikosti pouze 24x24 pixelů lze aplikací níže uvedených filtrů a jejich variací vypočítat 117,941 příznaků [8].



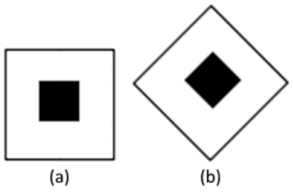
Obr. 2.11 Hranové Haarovy filtry



Obr. 2.12: Čárové Haarovy filtry 1



Obr. 2.13: Čárové Haarovy filtry 2



Obr. 2.14: Středové Haarovy filtry

Avšak jak jsme již v teoretické části podotkli, výpočet diferencí pro obraz například použitím konvoluce by pro takto obrovské množství zamýšlených příznaků představovalo neúnosné požadavky na výpočetní výkon. Především s ohledem na skutečnost, náročnost pro výpočet příznaků by rostla s velikostí použitého filtru.

Aby výpočetní náročnost byla co nejnižší, je za tímto účelem využit integrální obraz. Integrální obraz je reprezentací původního obrazu. Každý bod integrálního obrazu je jako součet všech předcházejících jasových hodnot nalevo a nad ním, včetně bodu samotného. Výpočet integrálního obrazu pak lze matematicky definovat následujícím způsobem.

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y') \qquad (12)$$

Kde I(x, y) představuje integrální obraz a jeho indexy a i(x', y') obraz původní. Výše uvedená definice je naprosto správná, avšak půvabem integrálního obrazu je, že jeho výpočetní náročnost lze při implementaci snížit na tři aritmetické operace. Již v původním článku z roku 1984, ve kterém jsou výhody integrálního obrazu využity k detekci objektů [10], je popsán způsob, jak lze integrální obraz vypočítat jedním průchodem původního obrazu s pomocí již vypočítaných částí obrazu integrálního. Výpočet je definován následovně.

$$I(x,y) = i(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1) \qquad (13)$$

Takto vytvořený integrální obraz lze použít pro rychlé zjištění součtu hodnot ve zvolené ploše obrazu. Součet jasových hodnot pro podokno obrazu lze poté vypočítat s pomocí pouhých čtyř přístupů do paměti třech aritmetických operací.

$$\sum_{\substack{x0<x\le x1 \\ y0<y\le y1}} i(x,y) = I(x0,y0) + I(x1,y1) + I(x1,y0) - I(x0,y0) \qquad (14)$$

Vzhledem k tomu, že obdélníkové Haarovy filtry právě s těmito hodnotami pracují, stačí pro výpočet libovolně velkého příznaku pouze relativně malý počet přístupů do paměti a triviální matematické operace. Jak si na Obr. 2.15 můžeme všimnout, díky rigidnosti dopravního značení, představují Haarovy filtry velice atraktivní možnost detekce, která bude v práci dále prozkoumána.

Obr. 2.15 Ukázka RGB, šedotónového a integrálního obrazu

2.3.2 Algoritmus učení

Detektor využívá modifikovanou verzi algoritmu AdaBoost jako metodu strojového učení. Účelem algoritmu je vybrat vhodný počet příznaků a zkonstruovat silný detektor, jehož úspěšnost bude splňovat námi zadané parametry.

Jak bylo uvedeno, algoritmus využívá množinu slabých detektorů. Každý slabý detektor je vytvořen na základě právě jednoho Haarova příznaku[6]

$$h_j(x) = \begin{cases} 1 \; když \; p_j f_j < p_j \theta_j \\ 0 \; jinak \end{cases} \quad (15)$$

Následující rovnice popisuje matematický zápis operace spojující slabé klasifikátory do silného

$$h(x) = \begin{cases} 1 \; když \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \; jinak \end{cases} \quad (16)$$

Kde T představuje všechny vypočítané Haarovy příznaky. Hodnota prahu θ a koeficient α jsou vypočítány v průběhu algoritmu AdaBoost, *p* představuje paritu určující směr nerovnosti.

Původní návrh detektoru Viola Jones pracuje s následovně upraveným algoritmem[6]

- Na vstupu algoritmu máme trénovací množinu pozitivních a negativních vzorů x_i čítající *m* pozitivních a *n* negativních vzorů. Společně s nimi popis, zda se jedná o představitele obrazu s hledaným objektem, v tom případě $y_i = 1$, či obraz bez něj, pak $y_i = 0$.
- Provedeme inicializaci vah $\omega_{1,i}$ pro každý trénovací obraz na hodnotu $\frac{1}{2m}$ či $\frac{1}{2n}$ podle toho zda se jedná o positivní či negativní vzor.
- Následně pro t=1,…, T kde T představuje počet příznaků provedeme následující:
    1. Provedeme normalizaci vah, aby jejich součet byl roven jedné, pro obě kategorie
    2. Pro každý příznak *j,* učíme detektor h_j tvořený právě jedním příznakem. Chyba je vyhodnocena vůči ω_t vztahem

$$\epsilon_j = \sum_i \omega_i \left| h_j(x_i - y_i) \right| \quad (17)$$

    3. Jako detektor h_t zvolíme ten s nejnižší chybovostí ε_t
    4. Váhy jsou přepočítány s pomocí vztahu

$$\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i} \quad (18)$$

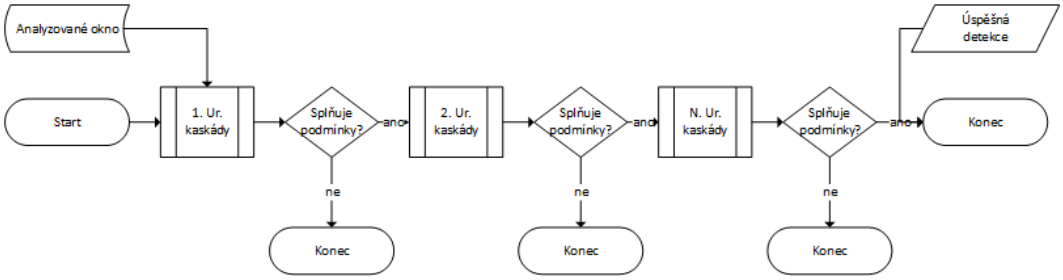
kde $e_i = 0$ v případě správné klasifikace a 1v případě chybné. $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$

2.3.3 Kaskádová architektura

Při běžném využití detekce pomocí Haarových příznaků, běžně využíváme posuvného okna o určité velikosti. Na každé toto okno aplikujeme detektor a jeho binární výstup nás informuje o případné detekci hledaného objektu. Tuto operaci musíme postupným posunem aplikovat na celý analyzovaný obraz, což znamená velké množství použití detektoru a v případě že vyžadujeme invariantnost, co do velikosti hledaného objektu, se tento počet dále zvyšuje.

Jak už bylo řečeno v podkapitole o algoritmu AdaBoost, metoda hledá velké množství slabých detektorů, v případě že pro každé okno provádíme všechny tyto výpočty, opět dochází k neefektivnímu využití výpočetních prostředků kvůli skutečnosti, že v drtivé většině případů obraz objekt neobsahuje a odlišnost od vzoru je natolik velká, že i mnohem slabší podmínky by jej byly schopny bezpečně vyloučit. Právě tento problém řeší takzvaná kaskádová architektura.

Kaskádová architektura je založena na myšlence kaskády středně silných detektorů. Silný detektor může být naučen s pravděpodobností úspěšné detekce například 90% a chybovostí blížící se nule. Aby bylo dosaženo nízké chybovosti, bude zapotřebí použít velké množství slabých detektorů. Na druhou stranu detektor, který chybně označí 50% obrazů tak i s přísným požadavkem na pozitivní detekci až 99.5, bude pro první úrovně vyžadovat pouze malé množství příznaků a jejich složitost a počet příznaků bude stoupat s přibývajícími úrovněmi. Síla pak spočívá v myšlence, že v kterémkoliv okamžiku, je v případě nesplnění podmínek možné kaskádu opustit a pokračovat v detekčním algoritmu pro další okno. Zobecněné fungování N úrovňové kaskádní architektury si můžeme prohlédnout na Obr. 2.16.



Obr. 2.16: Blokové schéma kaskádové architektury

Pravděpodobnost falešně pozitivních detekcí a skutečně pozitivních detekcí je matematicky definována následovně.

$$FP = \prod_{i=1}^{N} f_i \qquad SP = \prod_{i=1}^{N} s_i \qquad (19)$$

Kde *N* představuje počet úrovní kaskády a *f* a *s* pak dílčí pravděpodobnosti pro stupeň kaskády *i*. Pro lepší ilustraci následuje tabulka, která demonstruje efektivitu detektoru v závislosti na počtu úrovní kaskády. Koeficient f = 0.995 a s = 0.5 pro každou úroveň.

Tab. 2.3: Ukázkový kaskádový detektor

Ur. kaskády	Skutečně pozitivní	Falešně pozitivní
1	99.50%	50.00%
2	99.00%	25.00%
3	98.51%	12.50%
4	98.01%	6.25%
5	97.52%	3.13%
6	97.04%	1.56%
7	96.55%	0.78%
8	96.07%	0.39%
9	95.59%	0.20%
10	95.11%	0.10%

3 KLASIFIKACE DOPRAVNÍHO ZNAČENÍ

Kapitola 2 se zabývala detekcí dopravního značení. S ohledem na zkoumané metody, očekávaným výstupem z této části bude za ideálních podmínek okno, ve kterém se bude nacházet zástupce hledaného objektu.

Logickou druhou částí pak musí být popis nalezeného objektu a jeho přiřazení k jednomu z existujících vzorů dopravního značení. Tato část se zabývá popisem dvou přístupů, které by dle rešerše mohly vést ke kýženému výsledku.

3.1 Porovnání se vzorem

Tato metoda představuje nejpřímočařejší způsob klasifikace. Předpokládá se, že jako metoda použité detekce bude buďto použita detekce založená na barvě, či detekci hrany. V obou případech získáme masku hledané značky. Díky předem známým barevným a tvarovým vlastnostem jednotlivých dopravních značek, jsme schopni provést geometrické transformace vedoucí ke změně detekovaného tvaru na tvar a rotaci standardní, následně ji porovnat proti ideálnímu vzoru a konečně identifikovat, pro který vzor vykazuje detekovaný objekt nejvyšší podobnost.

Klasifikace by tak proběhla v několika krocích. V prvním kroku bychom detekovali geometrický tvar. [14] Objekt, který by splňoval naše prahovací podmínky, v případě barevné detekce by to mohl být například červený, uvnitř prázdný trojúhelník, který bychom normalizovali na zvolenou velikost. Tento nově vzniklý objekt bychom řádek po řádku porovnávali proti vzoru o stejné velikosti a počítali jeho odchylku podle (20)

$$D(n) = |X - T(n)|^2 \qquad (20)$$

V případě, že odchylka je nižší než zvolený práh, bude tvar považován za určený.

V druhém kroku, můžeme počítat odchylku objektu proti značkám spadajícím do identifikovaného tvaru. Aby tato metoda byla fungující, je třeba zajistit, aby normalizace proběhla přesně. Alternativně pak připravit knihovnu vzorů, která by kromě ideálních tvarů a piktogramů obsahovala vzory zarušené či jinak geometricky upravené.

3.2 Sekundární využití detektoru Viola Jones

Princip fungování detektoru Viola Jones byl již blíže popsán v kapitole 2.3. V této části tak již pouze budeme konstatovat, že vzhledem k tomu, že detektor je obzvlášť vhodný pro detekci objektů s rigidní geometrií, jedná se o teoreticky velice příhodný způsob klasifikace dopravního značení.

Je zde však problém, který je třeba adresovat. Detektor obličejů dle svého původního návrhu [6] byl vyvinut za účelem rychlé detekce. Avšak rychlost je uvažována při použití pouze jednoho detektoru na obraz, jmenovitě detektoru obličeje. Rozšířit metodu na jiné tvary je pochopitelně přímočaré, avšak vzhledem k tomu, že dopravních značek jsou desítky a nové stále přibývají, je nereálné připravit pro každou vlastní detektor, ten aplikovat na každý obrázek a přitom dodržet real-time.

Řešení této situace je dle rešerše málo diskutovanou záležitostí. Avšak všechny přístupy vycházejí ze stejného principu. Jak již nadpis této podkapitoly napovídá, detektory Viola Jones pro jednotlivé dopravní značky nemohou být prvním krokem klasifikačního řetězce. Aby byl splněn požadavek na real-time, je třeba nejprve co nejvíce vymezit množství oblastí, ve kterých se hledaný objekt může nacházet a teprve na tato okna aplikovat detektory, ideálně pak jen menší množinu z nich. Metody schopné výběru těchto oblastí byly popisovány v předchozí kapitole.

4 VOLBA METODY

Na základě rešerše bylo rozhodnuto, že jako nejslibnější metody se jeví použití detektorů ve spolupráci s algoritmy strojového učení. S použitím dostatečné sady obrazových vzorů pro učení, tento postup slibuje robustní algoritmus, a dle rešerše, i příslib splnění podmínky real-time.

Metody barevného prahování představují dva vzájemné nepřímo úměrné problémy. Pro jejich spolehlivé fungovaní je třeba nastavit prahy takové, aby vykazovaly malou míru falešně pozitivních nálezů, ale zároveň musí být natolik přísné, aby nalezené objekty byly snadno rozlišitelné na objekty hledané a ostatní. V průběhu testování praktické části této práce, bylo zjištěno, že statisticky zpracované prahy, jak v prostoru RGB, tak IHSL by mohly dobře sloužit k detekci modrých dopravních značek. Bohužel rozsah jasových hodnot červených dopravních značek vykazuje tak vysokou míru variace, která se navíc překrývá s rozsahem žlutých jasových hodnot, že autor této práce nebyl schopen vymyslet, jak tuto metodu prakticky zužitkovat pro detekční algoritmus.

Vzhledem k pozdější snaze této práce dále optimalizovat klasifikační algoritmy, byla provedena úvaha, jestli by barevná segmentace nemohla být použita jakožto část algoritmů předzpracování. V této části by byly vyloučeny části obrazu, ve kterých se hledaný objekt s jistotou nevyskytuje. Pokud by se tímto způsobem podařilo vyloučit například oblohu, silnici či stěny domů, vedlo by to ke zrychlení celého algoritmu. Realizace této funkcionality je však pouze námětem na budoucí zpracování.

Metoda detekce s pomocí detektoru založeném na HOG byla koncepčně vyzkoušena v prostředí Matlab. Leč už výpočet příznakové mapy s pomocí implementovaných funkcí zabral více než vteřinu bez jakéhokoliv dalšího výpočtu. Přestože následně implementované algoritmy s pomocí knihoven OpenCV tuto dobu podstatně zkrátily na zlomky vteřiny, bylo nutné konstatovat, že vzhledem k tomu, že se jedná o první krok metody, navíc o pouhé vytvoření mapy příznaků, nejeví se tato metoda jako efektivní způsob více objektové detekce.

Nakonec byl pro účely této práce zvolen několika-úrovňový detektor Viola Jones.

4.1 Příprava tréninkových vzorů

Jak bylo řečeno, pro účely trénování kaskádního detektoru byla využita databáze GTSRB. Aby mohl být vytvořen víceúrovňový detektor Viola Jones, je pro každou úroveň třeba vytvořit tréninkovou sadu.

Databáze GTSRB obsahuje 43 unikátních druhů dopravního značení, a pro tyto dopravní značky je třeba připravit kaskádové detektory. Avizovaného víceúrovňového detektoru bude dosaženo pomocí rozdělení vizuálně podobných dopravních značek do osmi skupin.

Detektory budou vytvořeny jak pro každou značku zvlášť, tak pro typové skupiny. Následuje seznam skupin a značek, které do nich spadají.

- Zákazové
    - Nejvyšší dovolená rychlost 20
    - Nejvyšší dovolená rychlost 30
    - Nejvyšší dovolená rychlost 50
    - Nejvyšší dovolená rychlost 60
    - Nejvyšší dovolená rychlost 70
    - Nejvyšší dovolená rychlost 80
    - Nejvyšší dovolená rychlost 100
    - Nejvyšší dovolená rychlost 120
    - Zákaz předjíždění
    - Zákaz předjíždění pro nákladní automobily
    - Zákaz vjezdu všech vozidel (v obou směrech)
    - Zákaz vjezdu nákladních automobilů.
- Stůj, dej přednost v jízdě!
- Hlavní pozemní komunikace
- Výstražné
    - Křižovatka s vedlejší pozemní komunikací
    - Jiné nebezpečí
    - Zatáčka vlevo
    - Zatáčka vpravo
    - Dvojitá zatáčka, první vlevo
    - Nerovnost vozovky
    - Nebezpečí smyku
    - Zúžená vozovka
    - Práce
    - Světelné signály
    - Pozor, přechod pro chodce
    - Děti
    - Cyklisté
    - Náledí
    - Zvěř
- Dej přednost v jízdě!
- Zákaz vjezdu všech vozidel
- Příkazové
    - Přikázaný směr jízdy vpravo
    - Přikázaný směr jízdy vlevo
    - Přikázaný směr jízdy přímo a vpravo
    - Přikázaný směr jízdy přímo a vlevo
    - Přikázaný směr objíždění vpravo
    - Přikázaný směr objíždění vlevo
    - Kruhový objezd
- Konec
    - Konec nejvyšší dovolené rychlosti
    - Konec všech zákazů
    - Konec zákazu předjíždění
    - Konec zákazu předjíždění pro nákladní automobily

Aby bylo možné implementovat detektor v programovacím jazyku C++, budou

v této práci využity knihovny OpenCV. OpenCV mimo jiné obsahuje programy, které jsou navržené přímo pro tvorbu detektoru Viola Jones.

4.2 Tvorba kaskád pro detektor Viola Jones

Aby bylo možné vytvořit kaskádní detektor, je třeba data nejprve převést na tvar, se kterým umějí moduly OpenCV pracovat. To znamená, že negativní a pozitivní tréninkové vzory musí být přístupné a hlavně musí být přístupný seznam těchto souborů v definovaném formátu.

Negativní tréninkové vzory představují takové obrazy, na kterých se nenachází námi hledaný objekt. Protože modul opencv_createsamples je schopný vytvářet z velkého obrazu množství negativních vzorů o velikosti námi zvoleného okna, je v této fázi lepší této možnosti využít.

V průběhu testování byla často identifikována situace, ve které docházelo k falešně pozitivní detekci pro zcela odlišný typ značení. Aby příkazová značka nebyla falešně detekována jako zákazová, byly do negativní tréninkové sady přidány výřezy zákazových značek, doba učení kaskády se znatelně prodloužila, ale míra falešně pozitivních detekcí tohoto typu byla prakticky eliminována.

Obr. 4.1: Obraz bez značení pro negativní tréninkové vzory

Požadavky na seznam negativních tréninkových vzorů jsou nenáročné. Musí se jednat o textový soubor a na každém řádku je uvedena jedna relativní či absolutní cesta k souboru.

Pozitivní tréninkové vzory jsou naopak obrazy, na kterých se ideálně bude nacházet pouze hledaný objekt, vzhledem k tomu že algoritmus bude uvažovat jakékoliv zbytkové obrazové pozadí za součást objektu.

Na následujících dvou obrázcích jsou uvedeny ukázky sérií vzorů, které byly použity, mimo jiné, při trénování svých respektivních detektorů. Obr. 4.2 je ukázkou vzorů pro konečný klasifikátor, úkolem klasifikátoru bude označit nalezený objekt za ‚Maximální povolená rychlost 20' a tudíž to jsou pouze značky tohoto typu, které se budou v této množině vyskytovat.

Obr. 4.2: Ukázky obrazových vzorů jedné značky

Obr. 4.3 naopak reprezentuje sadu pro detektor, který bude hledat všechny ‚trojúhelníky' v obraze. Aby tato úloha byla co nejspolehlivější, je třeba, aby v této množině bylo zastoupeno množství všech značek tohoto typu. Ideálně pak rovnoměrně, aby jedna skupina nepropadla učícím algoritmem jako přijatelná ztráta.

Obr. 4.3: Ukázky obrazových vzoru jednoho typu

V případě pozitivních učících množin je nutné seznam cest k souboru navíc obohatit o poloze, počtu a velikosti značek v obraze. Formát obrazové informace musí mít následující tvar:

%název a cesta k souboru% n x0 y0 w0 h0 x1 y1 w1 h1…

Název a cesta k souboru opět mohou nabývat jak absolutního tak relativního tvaru. Parametr se používá ke sdělení, kolik hledaných objektů se v obraze nachází. Následují tolikrát opakované informace o značkách, kolik jsme zadali n. Poté, co máme množiny připraveny ve správném formátu, je můžeme předat společně s dalšími parametry modulům.

4.2.1 Modul opencv_createsamples

Tento modul je schopen pracovat ve dvou režimech. V prvním je na vstupu pouze vzorový obraz a modul vygeneruje tréninkovou obrazovou množinu využitím geometrických transformací, změnou jasových hodnot a dalšími metodami.

Ve druhém režimu pak využijeme náš předem vytvořený seznam souborů. Pro nás důležité parametry tohoto modulu tedy jsou:

- -info cesta k textovému souboru se seznamem pozitivních vzorů
- -num kolik obrazů bude z textového souboru použito. Výchozí hodnota je 1000, pokud chceme použít více, musíme tento parametr zadat, jinak bude

použito maximálně 1000 obrazů
- -h výška předzpracovaných vzorů
- -w šířka předzpracovaných vzorů
- -vec jméno vygenerovaného souboru vektorů

Soubor vektorů vec ve své hlavičce obsahuje informace o rozměrech vzorů, jejich počtu a dále pak samotné vzory, které byly zmenšeny na námi nastavenou velikost. Samotné nastavení velikosti je třeba zvážit, větší vzory umožní přesnější detekci s nižší chybovostí, ale jak bylo v teorii rozváděno, již malé rozměry dávají vzniknout velkému množství příznaků a dalším zvětšováním toto číslo dále stoupá.

V této práci bylo použito okno o velikosti 24x28 pro trojúhelníkové vzory, aby byl aproximován poměr stran trojúhelníků, a aby nedocházelo k distorzi. 24x24 pro všechny ostatní, jejichž poměr stran je roven jedné.

4.2.2 Modul opencv_traincascade

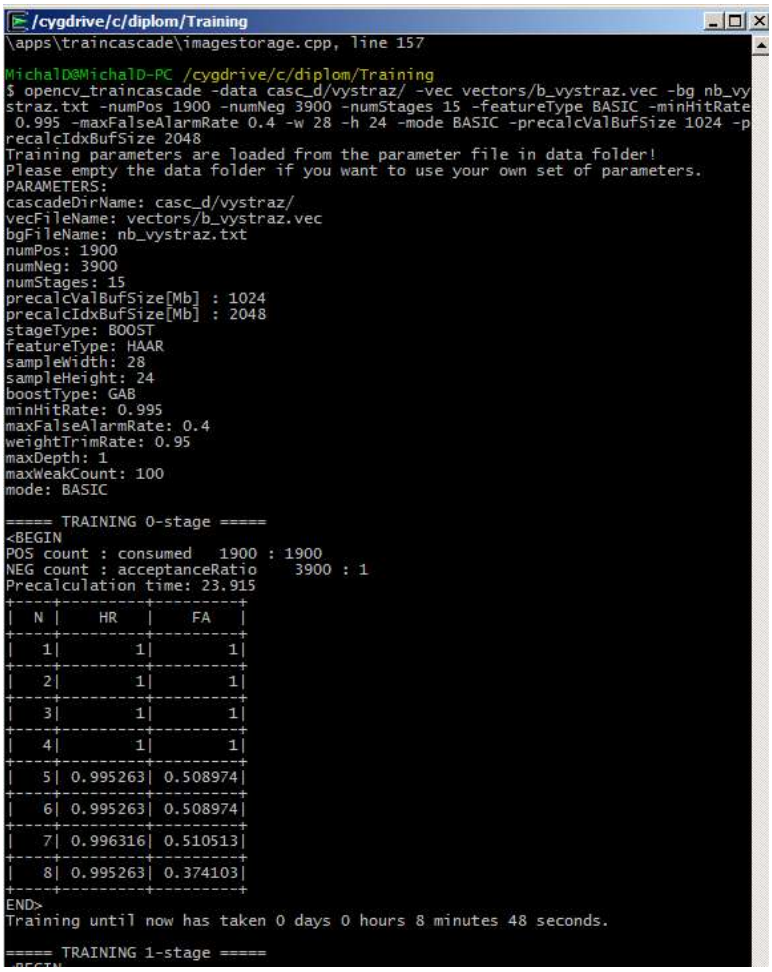
Program opencv_traincascade skrze algoritmus strojového učení je schopen vytvořit samotný kaskádový systém vytvořený z HAAR příznaků a uložitho do souboru xml.

Tento program je velice flexibilní co do nastavení vlastností učení a je silně ovlivnitelný skrze velké množství parametrů[11]. Zde následují ty nejdůležitější:

- -vec cesta k vektoru vytvořeného programem opencv_createsamples
- -bg cesta k souboru negativních tréninkových vzorů
- -data složka, do které budou ukládány úrovně kaskády a po skončení kompletní systém
- -numStage počet úrovní kaskády
- -minHitRate jak velký poměr pozitivních tréninkových vzorů má být klasifikováno jako správně pozitivní
- -maxFlaseAlarmRate maximální míra falešně pozitivní detekce na úroveň kaskády
- -w, –h výška a šířka vzorů musí souhlasit s velikostí v hlavičce vektoru

Výpočet kaskádového systému je, obzvláště pro mnoha úrovňové systémy, časově nesmírně náročná operace. 15ti úrovňové kaskády, které byli používány pro účely této práce zabrali 10-15 hodin na výpočet. Vzhledem ke skutečnosti že tato práce vyžadovala 51 kaskádových systémů, samotný výpočet zabral stovky hodin.

Na níže uvedeném obrázku si můžeme prohlédnout nastavení a vzhled programu opencv_traincascade před a za běhu.



Obr. 4.4: Ukázka opencv_traincascade

5 IMPLEMENTOVANÉ ŘEŠENÍ ÚLOHY

Pro účely této práce, nazývejme ty kaskádové systémy, které slouží k detekci jednoho z mnoha podobných objektů (zákaz, výstraha) detektory a ty kaskádové systémy, které slouží k detekci jednoho typu klasifikátory.

Výsledný algoritmus byl implementován v jazyce c++. Aby byla dodržena podmínka real-time, bylo třeba upravit algoritmus tak, aby detekce probíhala postupně. Vzorové obrázky mají rozlišení 1300x800px. Aplikace jednoho detektoru využívající Haarovy příznaky na celý obraz, s použitou velikostí detekčního okna od 24px do 200px, zabere v průměru 50ms. Z tohoto důvodu není možné aplikovat všech 43 klasifikátorů naráz.

Jako řešení byla zvolena metoda postupné detekce. Na základě podobnosti bylo vybráno 8 skupin, které sdílejí tolik společných visuálních prvků, že na ně kaskáda může být naučena i přes jejich rozdíly. Zvolené skupiny byly určeny následovně:

- Značky zákazové s výjimkou značek ukončující zákaz a zákaz vjezdu všech vozidel
- Zákaz vjezdu všech vozidel
- Značky výstražné a Křižovatka s vedlejší pozemní komunikací
- Značky příkazové
- Stůj, dej přednost v jízdě
- Dej přednost v jízdě
- Hlavní pozemní komunikace
- Značky ukončující zákaz

Čtyři z výše uvedených kategorií obsahují pouze jednu značku, ale na dále popsanou metodiku to nebude mít vliv.

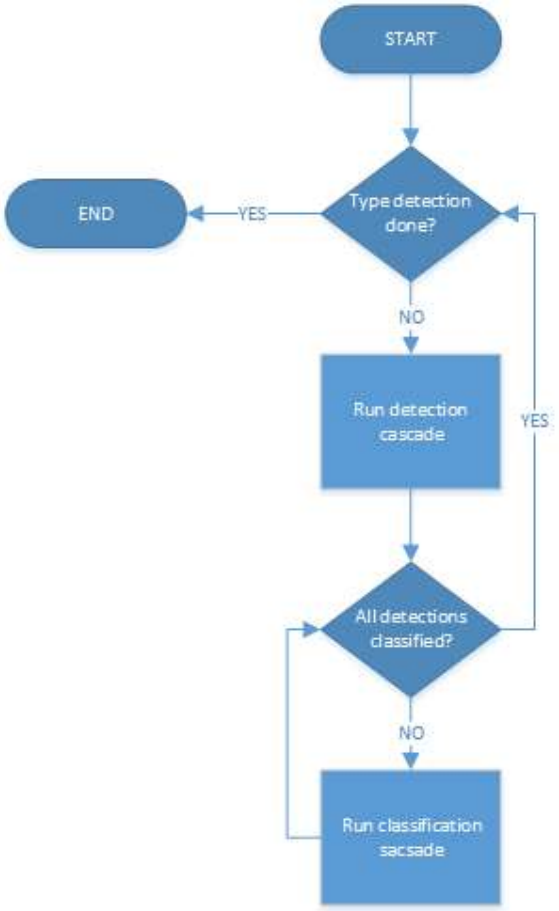
Pakliže detektory pro tyto kategorie aplikujeme v první řadě, tak docílíme výrazného omezení oblastí, na které budeme v druhém kroku aplikovat velké množství klasifikátorů. Na tyto oblasti aplikujeme pouze ty typy detektorů, které spadají do kategorie první sady detektorů.

S pomocí xml souboru kaskádového detektoru, jehož tvorba byla popsána v předchozí kapitole, může být v naší c++ implementaci použit pro inicializaci datového typu CascadeClassifier. Skrze proměnnou tohoto datového typu jsme pak schopni zavolat následující metodu[10].

```
void CascadeClassifier::detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size());
```

Kde parametry *image* slouží k výběru obrazu, na který bude detektor aplikován, *object* určuje, kam budou detekované oblasti uloženy ve formátu Rect (obdélníku). *scaleFactor* představuje koeficient, o který budou buďto příznaky zvětšeny či obraz zmenšen, aby mohla být provedena detekce pro více možných velikostí hledaného objektu v závislosti na hodnotě *flags*. Parametr *minNeighbors* určuje minimální počet sousedů detekcí v těsné blízkosti, aby byl objekt uznán jako detekovaný.

Výše uvedená funkce v algoritmu figuruje dvakrát, v prvním případě provádí detekce typů a v druhém používá detekované oblasti a v nich hledá objekty. Blokové schéma tohoto principu je ilustrováno na Obr. 5.1.



Obr. 5.1 Algoritmus detekce a klasifikace dopravního značení

Vlastnosti kaskády již není možné po vytvoření jednoduše modifikovat a jediná možnost, jak za běhu v programu ovlivňovat průběh detekce způsobem, který by měl na výsledky přímý vliv, je skrze parametry scaleFactor, minNeighbors. Obzvlášť díky tomu, že je možné je nastavit pro každý detektor i klasifikátor individuálně. Z tohoto důvodu byly do projektu přidány knihovny pro práci s konfiguračními ini soubory minIni [15].

V zdrojovém adresáři aplikace se nachází soubor param.ini, který je při volání konstruktoru pro třídu *detektor* otevřen, a hodnoty v něm jsou přečteny a použity mimo jiné k inicializaci vektorů, ale hlavně k načtení parametrů, které budou uplatněny při volání metody detectMultiScale();

Algoritmus lze spustit dvěma způsoby buďto voláním funkce

```
vector<result>& singleImageDetector(string path, char MODE = ALLDET)
```

nebo

```
vector<result>& multipleImageDetector(string path, char MODE = 0)
```

Funkce singleImageDetector(), jak název napovídá, zpracuje jeden obraz, k němuž

cesta je předána parametrem *path* a vrátí vektor s výslednými detekcemi. Detekce mají podobu jednoduché struktury:

```
/***********Structure for returning results*************/
struct result{
       int x, y, w, h;
       signinfo si;
       std::string img_name;
    };
```

Tato struktura drží pouze informace o detekovaném objektu, jmenovitě souřadnice levého horního bodu v obraze, šířku a výšku okna, dále pak index identifikované značky, uložený jako celočíselná hodnota 0-42 a konečně český ekvivalent značkového indexu.

Fungování detekčního algoritmu je ovlivněno flagem MODE. Flag může nabývat následujících hodnot.

```
/***********Detektor Modes*************/
#define DETANDCLAS 0
#define DETONLY 1
#define ALLDET 2
#define TEST 4
#define SIMAGE 8
```

Kde DETANDCLAS při každém spuštění provede detekci pouze dvěma detektory, při dalším volání použije další dva v pořadí. Po průběhu všemi osmi detektory se počítadlo resetuje.

ALLDET na každý obrázek je aplikováno všech osm detektorů

TEST Testovací flag, umožní načtení scv souboru s testovými daty, je diskutováno v kapitole o Testování řešení.

SIMAGE s tímto flagem, bude každý analyzovaný obrázek zároveň uložen do složky result.

6 TESTOVÁNÍ ŘEŠENÍ

Pro účely testování programu jsme použili databázi z GTSDB, která obsahuje 600 obrázků se seznamem, jehož součástí je poloha a typ všech značek, které se na nich nacházejí. V případě, že se v obraze nachází větší množství značek, je v seznamu uvedeno více záznamů, každý pro jednu značku. Abychom byli schopni tento seznam číst, byly vytvořeny mechanismy, které umožní soubor parsovat do formátu, který nám vyhovuje. Z dokumentace GTSRB víme, že přiložený soubor gt.txt obsahuje informace v následujícím formátu.

#ČísloObr.#.ppm;#levýSloupec#;##horníŘada#;#pravýSloupec#;#spodníŘada#;#index Značky#

Za tímto účelem byla vytvořena následující struktura

```
/***********Structure for test data entity**************/
struct testformat{
	std::string name;
	int x, y, w, h,sign,index, found;
};
```

Při inicializaci je vytvořen vektor vektorů s touto strukturou jako šablonou.

```
vector <vector <testformat>> tf_vec;
```

Pokud je uživatelem funkce detektoru volána v režimu TEST, následující část kódu přečte soubor definovaný v param.ini a uloží ho ve tvaru struktury testformat do vektoru příslušícího k analyzovanému obrázku.

```
while (getline(tf, line))
{
	if (line.empty())
		continue;
	sscanf(line.c_str(), "%[^;];%d;%d;%d;%d;%d", name, &x, &y, &x2, &y2, &k);
	index = atoi(name);
	t.name = name;
	t.x = x;
	t.y = y;
	t.w = x2 - x;
	t.h = y2 - y;
	t.index = index;
	t.sign = k;
	t.found = 0;
	tf_vec[index].push_back(t);

}
```

Využitím hodnoty index vytváříme vlastní vektor údajů pro každý obraz. Tato úprava silně zjednodušuje práci s obrazy, ve kterých se nachází 2 a více značek. Protože jsme schopni počet značek zjistit prostým voláním metody size().

V průběhu našeho měření uvažujeme, že detekce je skutečně pozitivní, když středobod naší detekce se nachází uvnitř obdélníku definovaného seznamem výsledků a zároveň námi vyhodnocený typ značky odpovídá tomu ze seznamu. Pakliže jedna ze dvou podmínek není splněna, výsledek je veden jako falešně pozitivní. Každá v seznamu vedená značka, která není správně klasifikována, je poté vedena jako falešně negativní.

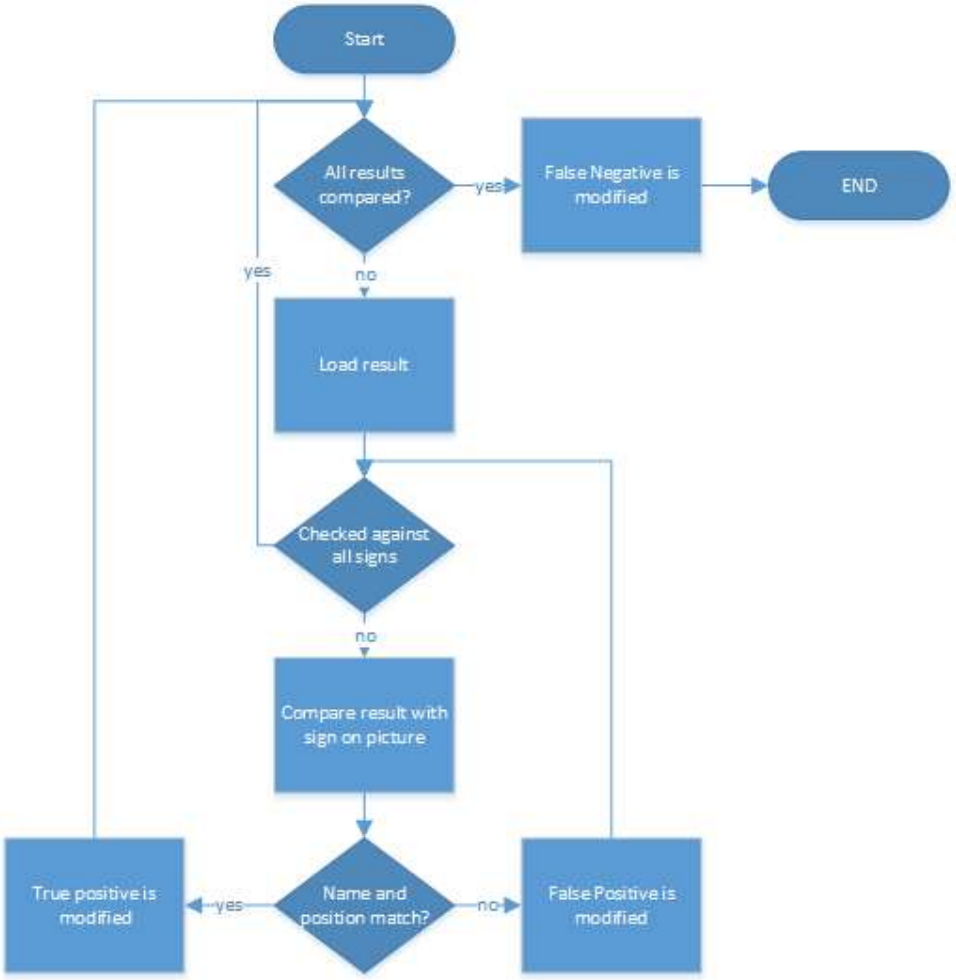
Pro účely ověřování výkonu algoritmu, byly do projektu přidány mechanismy, které umožňují schraňování informací o skutečně pozitivních detekcích, falešně pozitivních detekcích, a falešně negativních detekcích. Následující struktura slouží právě k průběžnému ukládání stavu detekce, tedy ke kolika skutečně pozitivním, falešně pozitivním a falešně negativním událostem došlo.

```
/***********Structure for storing test results*************/
struct testAnalysis{
	std::vector<int> FP_k;
	std::vector<int> TP_k;
	std::vector<int> FN_k;
	std::vector<clock_t> time;
};
```

Nejdůležitější část kódu obstarávající analýzu detekce je uvedena níže

```
for (int i = 0; i < results.size(); i++)
{
	tp = 0;
	fp = 0;
	fn = 0;
for (int j = 0; j < tf_vec[findex].size(); j++)
{
	Rect rect(tf_vec[findex][j].x, tf_vec[findex][j].y, tf_vec[findex][j].w,
	tf_vec[findex][j].h);
	Point p(results[i].x + results[i].w*0.5, results[i].y + results[i].h*0.5);
	if ((rect.contains(p)) && (tf_vec[findex][j].sign == results[i].si.type))
	{
		tp = 1;
		fp = 0;
		tf_vec[findex][j].found = 1;
		break;
	}
	else{fp = 1;}
}
test_ana.TP_k[results[i].si.type] += tp;
test_ana.FP_k[results[i].si.type] += fp;
}
for (int i = 0; i < tf_vec[findex].size(); i++)
{
	if (tf_vec[findex][i].found== 0)
		test_ana.FN_k[tf_vec[findex][i].sign]++;
}
```

Tento kód kontroluje všechny detekované oblasti ve vektoru *result* a porovnává je proti známým hodnotám ve vektoru *tf_vec*. Pakliže je detekována shoda v poloze a typu značení, vektor TP pro dané značení je inkrementován a prvek *tf_vect* je označen jakou *found*. Pokud shoda není nalezena, FP je inkrementován. V druhé smyčce pak provedeme kontrolu proti flagu *found*, každá nenalezená značka je vedena jak FN. Blokové schéma tohoto postupu si lze prohlédnout na Obr. 6.1.



Obr. 6.1: Algoritmus porovnání zjištěných dat se známými

Pro hodnocení testu provedeme také výpočet sensitivity dle vztahu (21) a prediktivní hodnoty pozitivního testu dle (22). TP reprezentuje výskyt skutečně pozitivních detekcí, FN falešně negativní a FP falešně pozitivní.

$$SE = \frac{TP}{TP+FN} \qquad (21)$$

$$PPV = \frac{TP}{TP+FP} \qquad (22)$$

Tab. 6.1 zobrazuje kumulované výsledky získané analýzou 600 obrazů a Tab. 6.2 pak výsledky podle typu.

Tab. 6.1: Kumulované výsledky všech detektorů

TP	FP	FN	TPR	PPV
678	212	180	0.79	0.76

Tab. 6.2: Kumulované výsledky detektorů podle typu

Typ	TP	FP	FN	SE	PPV
Zákaz	335	99	61	0.85	0.77
Stop	15	0	7	0.68	1.00
Hlavní u.	50	2	4	0.93	0.96
Výstraha	335	99	61	0.85	0.77
Dej přednost	15	0	7	0.68	1.00
Jednosm	50	2	4	0.93	0.96
Příkaz	73	50	41	0.64	0.59
Konec zákazu	0	0	33	0.00	

Podobný poměr mezi falešně pozitivními a falešně negativními je dobrou indikací skutečnosti, že podmínky, proti kterým byly kaskády při učení testovány, byly zvoleny dobře. Přísnější podmínka na skutečně pozitivní míru by dál zvyšovala falešně negativní detekce a zpřísnění podmínky pro skutečně negativní, by vedlo ke zvýšenému výskytu falešně pozitivních.

Největší abnormalitu představuje skupina značek, Konec zákazu, která selhala ve všech případech. Vzhledem k absolutnosti tohoto jevu, byla tato chyba zřejmě způsobena již při učení kaskádového systému. Tato kaskáda bude muset být v budoucnu naučena znova a lépe. Problém falešných detekcí je příhodně ilustrován na Obr. 6.2, kde je objekt chybně identifikován jako značka typu: Křižovatka s vedlejší pozemní komunikací. Je snadné si všimnout na Obr. 6.3, že tyto dva objekty skutečně sdílejí vysokou míru podobnosti.

Obr. 6.2: Falešná pozitivní detekce

Obr. 6.3: Skutečná pozitivní detekce

Pro zmíněných 600 analyzovaných obrázků bylo zároveň provedeno měření uplynulého času mezi začátkem a koncem detekčního cyklu. Těchto 600 intervalů měřených v milisekundách bylo uloženo do datového souboru a v tabulkovém procesoru Microsoft Excel pak byla vypočítána průměrná hodnota analýzy obrazu a to 395ms.

Získaná hodnota byla naměřena při spouštění algoritmu v režimu ALLDET, tedy režimu, ve kterém jsou na jeden obrázek aplikovány všechny dostupné detektory. Při použití režimu DETANDCLAS je použita pouze čtvrtina detektorů na každý obraz a výpočetní čas se úměrně zkrátí.

Při cestovní rychlosti vozidla 50km/h urazí vozidlo za 1s téměř 14 metrů, uvažujeme-li průměrnou dobu detekce 400ms, znamená to, že obraz zpracujeme každých 5.6 metru. V případě menšího obrázku by tato vzdálenost dále klesla. Pro účely této práce, může být požadavek na real-time za splněný.

Testování proběhlo na laptopu Eurocom M4 s CPU Intel® Core™ i7 – 4710MQ, s operační pamětí 8GB a OS Microsoft Windows 7 -64bit.

Aplikace byla vyvíjena a testována v IDE Visual Studio 2013 na x64 platformě.

7 ZÁVĚR

Tato práce se zabývala detekcí a klasifikací dopravního značení v obrazu. Tréninkové a testovací obrazové množiny pocházejí z databáze vzniklé v rámci projektu GTSDB a GTSRB, které jsou pro všech 43 učených vzorů identické s jejich českým ekvivalentem.

V rámci této práce došlo k seznámení se s metodami, které mohou být využity pro úlohu detekce objektu v obraze. Na základě prozkoumání dopravního značení v České republice a obeznámení se s pravidly, kterými se řídí, byly diskutovány přístupy k této úloze a jejich vhodnost.

Pro implementaci algoritmů segmentující hledané objekty s pomocí znalostí o barvě hledaných objektů byla vytvořena datová sada jasových hodnot pro zvolené typy značek z tréninkové množiny obrázků. Pomocí takto získaných dat byly zmapovány barevné roviny RGB a IHLS a poloha hledaných hodnot na nich.

Byla implementována metoda Histogramu orientovaných gradientů, ale časová náročnost výpočtu této metody vedla k porušení podmínky pro real-time a tento přístup byl opuštěn. Namísto HOG pokračoval vývoj detektoru založený na detektoru Viola Jones a jeho implementace v programovacím jazyku C++.

První fáze detektoru Viola Jones byly naprogramovány v jazyce C++ přímo, jmenovitě knihovny pro načítání obrazu ze souboru, převod do šedotónového obrazu a výpočet integrálního obrazu. Před implementací algoritmu AdaBoost bylo zjištěno, že je součástí knihoven OpenCV s podporou TBB a bylo rozhodnuto pokračovat dále s těmito knihovnami.

S pomocí programů OpenCV bylo vygenerováno 8 kaskádových systémů založených na Haarových příznacích, které byly učeny na typově podobné dopravní značky. Aby program byl schopný rozeznat i podobné značky s odlišným piktogramem mezi sebou, bylo dále vytvořeno dalších 43 kaskádových systémů, jejichž účelem je právě finální klasifikace.

Výsledkem práce je pak funkční algoritmus, napsaný v jazyce C++. Algoritmus je schopný detekce dopravního značení a její následné klasifikace. V tuto chvíli se jedná o obecné algoritmy, které výsledky detekce vracejí jakožto návratový parametr. Jak je v kapitole o programu napsáno, je možné změnou jedné definice program upravit tak, aby byl výsledný obraz s označenými detekovanými objekty uložen do souboru.

Systém je schopný v tuto chvíli rozpoznat až 39 druhů dopravních značek a díky nastavitelnosti programu skrze konfigurační soubor ini lze tento počet dále zvyšovat. Pro účely vyšetřování byly do systému implementovány metody schopné automatického porovnávání výsledků s pomocí csv souboru se skutečnými výsledky. Tímto způsobem byla také zjištěna průměrná sensitivita algoritmu 0.79 a prediktivní hodnota pozitivního testu 0.76 při průměrně rychlostí 400ms na obrázek.

Prediktivní hodnota pozitivního testu a sensitivita prvních kaskádních systémů se pohybovala pod 50% a skrze studium akademických článků a modifikace tréninkových množin se podařilo míru detekce zlepšit do konečného stavu. Velký pokles v míře výskytu falešných pozitiv přinesla úprava tréninkové množiny taková, kde množiny obsahovaly kromě obrazů pozadí navíc tréninkové pozitivní množiny ostatních značek.

V této práci byl otestován detektor Viola Jones jak pro účely detekce, tak pro účely klasifikace a tato metoda byla seznána jako platný přístup. Algoritmus byl vytvořen tak, aby bylo možné přidávat další dopravní značky. Pozornost by však v tuto chvíli bylo třeba věnovat značkám značícím konec zákazu, jejichž míra detekce je v tuto chvíli nejnižší ze všech skupin a vybraným zákazovým značkám, které vykazovaly vyšší míru falešně pozitivních výsledků, než ostatní.

Skrze tuto práci jsem teprve konečně docenil sílu strojového učení, o kterých jsem do této chvíle měl pouze teoretické povědomí. V tomto směru bych rád své znalosti dále rozšiřoval.

LITERATURA

[1] Vyhláška Ministerstva dopravy a spojů, kterou se provádějí pravidla provozu na pozemních komunikacích a úprava a řízení provozu na pozemních komunikacích. In: 30/2001 Sb. 2001. Dostupné z: http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=3589

[2] Zásady pro dopravní značení na pozemních komunikacích: technické podmínky - TP 65 : s účinností od 1.12.2002 [online]. Vyd. 2. Brno: Centrum dopravního výzkumu, 2002, 98 s. [cit. 2015-01-08]. ISBN 80-865-0204-X. Dostupné z: http://www.ibesip.cz/data/web/kampane/legislativa/besip-02-TP_65_2vydani.pdf

[3] HOUBEN, Sebastian, Johannes STALLKAMP, Jan SALMEN, Marc SCHLIPSING a Christian IGEL.Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. International Joint Conference on Neural Networks, 2013. Dostupné z: http://benchmark.ini.rub.de/?section=gtsdb&subsection=dataset

[4] HANBURY, Allan a Jean SERRA. A 3D-polar Coordinate Colour Representation Suitable for Image Analysis. Vienna AUSTRIA, 2003. Dostupné z: http://cmm.ensmp.fr/~serra/notes_internes_pdf/NI-230.pdf. Technical Report. Vienna University of Technology

[5] DALAL, et al., "Histograms of Oriented Gradients for Human Detection", Dostupné z http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf, CVPR, Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1-vol. 01, Jun. 20-26, 2005

[6] VIOLA, Paul a Michael JONES. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision* [online]. (57) [cit. 2015-05-14]. ISSN 1751-5858 15. Dostupné z: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

[7] PAPAGEORGIOU, C.P., M. OREN a T. POGGIO. 1998. A general framework for object detection. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)* [online]. Narosa Publishing House, : 555-562 [cit. 2015-05-14]. DOI: 10.1109/ICCV.1998.710772. ISBN 81-7319-221-9. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=710772

[8] LIENHART, R. a J. MAYDT. 2002. An extended set of Haar-like features for rapid object detection. *Proceedings. International Conference on Image Processing* [online]. IEEE, : I-900-I-903 [cit. 2015-05-14]. DOI: 10.1109/ICIP.2002.1038171. ISBN 0-7803-7622-6. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1038171

[9] MALACH, T., P. BAMBUCH a J. MALACH. 2011. Detekce obličeje v obraze s využitím prostředí MATLAB. In: *In Proceedings of the 19th Technical Computing Prague* [online]. [cit. 2015-05-14]. Dostupné z: http://dsp.vscht.cz/konference_matlab/MATLAB11/prispevky/078_malach.pdf

[10] Cascade Classification: Haar Feature-based Cascade Classifier for Object Detection. 2011. Opencv documentation [online]. [cit. 2015-05-15]. Dostupné z: http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html

[11] Cascade Classifier Training. 2011. Opencv documentation [online]. [cit. 2015-05-15]. Dostupné z: http://docs.opencv.org/doc/user_guide/ug_traincascade.html

[12] CROW, Franklin C. 1984. Summed-area tables for texture mapping. *ACM SIGGRAPH Computer Graphics* [online]. **18**(3): 207-212 [cit. 2015-05-15]. DOI: 10.1145/964965.808600. ISSN 00978930. Dostupné z: http://portal.acm.org/citation.cfm?doid=964965.808600

[13] CHEN, Zhilu, Xinming HUANG, Zhen NI a Haibo HE. A GPU-based real-time traffic sign detection and recognition system. In: 2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS) [online]. 2014 [cit. 2015-05-18]. DOI: 10.1109/civts.2014.7009470.

[14] ZÁMEČNÍK, D. Rozpoznání dopravních značek využitím neuronové sítě. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 80s. Vedoucí diplomové práce doc. Ing. Václav Jirsík, CSc.

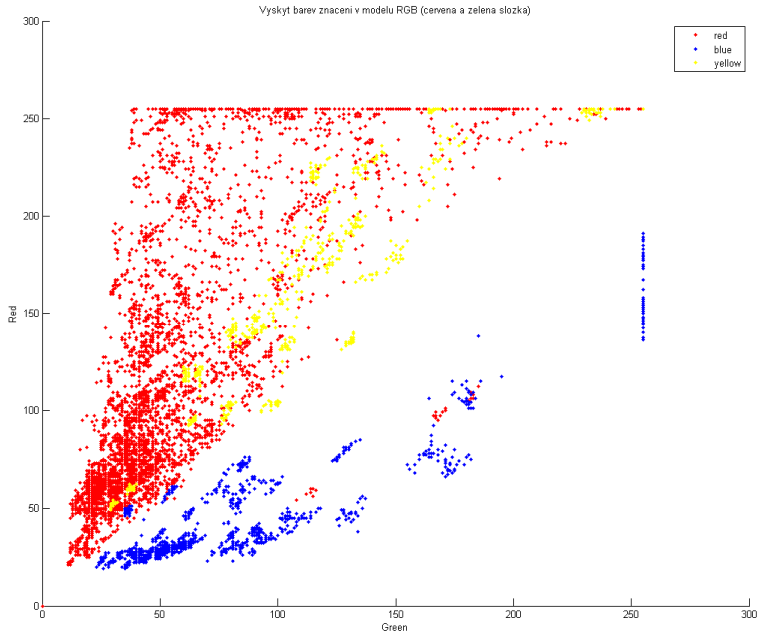
[15] RIEMERSMA, Thiadmer. MinIni. CompuPhase [online]. 2008 [cit. 2015-05-18]. Dostupné z: http://www.compuphase.com/minini.htm

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

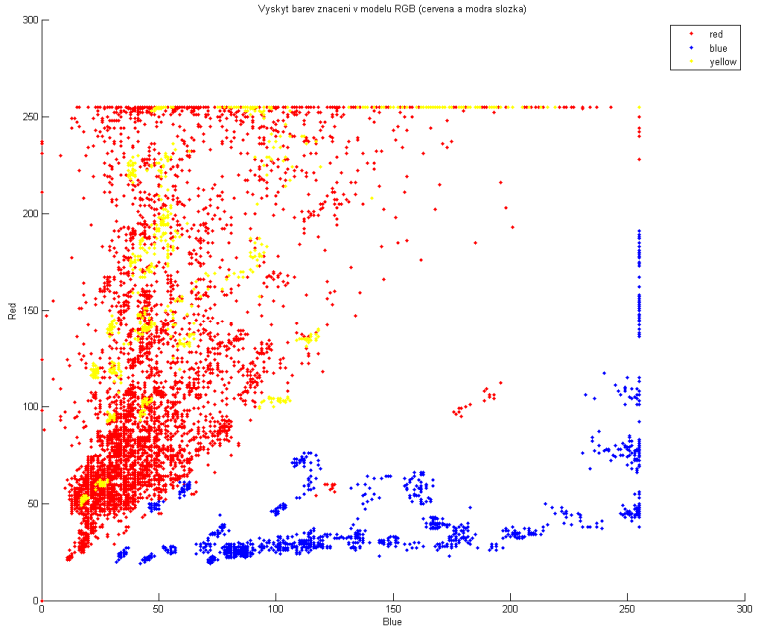
IHLS	Improved Hue Lightness Saturation
HOG	Histogram of oriented gradients
SVM	Support Vector Machine
TP	True positive – skutečna positivní
FP	False positive – falešná positivní
FN	False negative – falešná negativní
TBB	Threading Build Blocks

SEZNAM PŘÍLOH

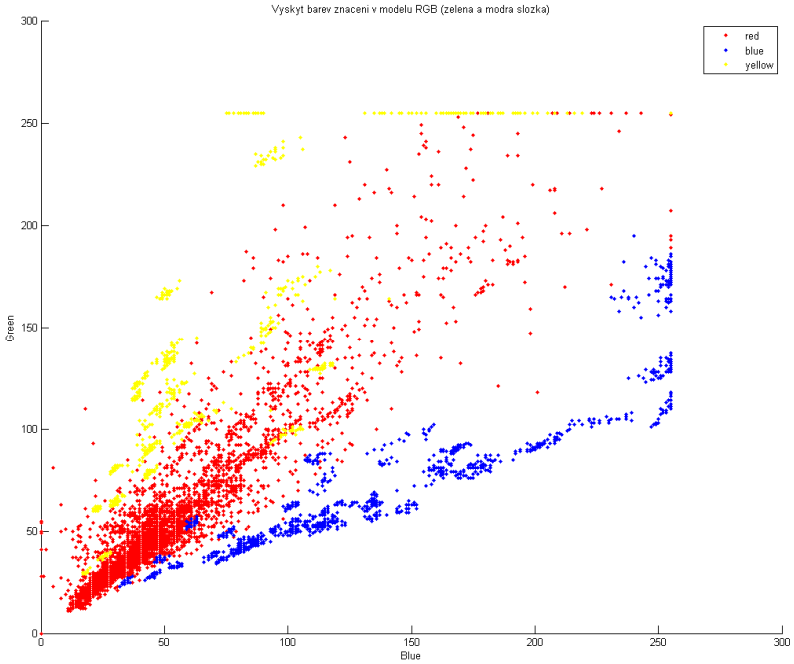
A DOPLŇUJÍCÍ OBRAZOVÉ INFORMACE



Obr. 7.1: Zastoupení barev značení v prostoru RG



Obr. 7.2: Zastoupení barev značení v prostoru RB



Obr. 7.3: Zastoupení barev značení v prostoru GB

B VÝSTUPY

B.1 Ukázky detekcí

Obr. 7.4: TP náledí

Obr. 7.5: TP dvojitá zatáčka, nej. povolená rychlost 50

Obr. 7.6: TP Zákaz předjíždění

Obr. 7.7: TP Nejvyšší povolená rychlost 50

Obr. 7.8: TP hlavní ulice, nej. povolená rychlost 50

Obr. 7.9: TP hlavní ulice

Obr. 7.10: FN Práce na silnici

Obr. 7.11: TP Kluzká vozovka


pedestrian crossing


slippery road
slippery road

B.2 Kompletní výsledky statistiky

Type	TP	FP	FN	TPR	PPV	F1
0	3	0	1	0.75	1	0.857143
1	39	24	9	0.8125	0.619048	0.702703
2	50	9	9	0.847458	0.847458	0.847458
3	19	1	2	0.904762	0.95	0.926829
4	24	2	7	0.774194	0.923077	0.842105
5	32	31	5	0.864865	0.507937	0.64
6	0	0	17	0	#	0
7	33	0	4	0.891892	1	0.942857
8	40	9	7	0.851064	0.816327	0.833333
9	25	2	7	0.78125	0.925926	0.847458
10	57	0	6	0.904762	1	0.95
11	26	9	1	0.962963	0.742857	0.83871
12	50	2	4	0.925926	0.961538	0.943396
13	51	9	2	0.962264	0.85	0.902655
14	15	0	7	0.681818	1	0.810811
15	7	21	3	0.7	0.25	0.368421
16	6	0	1	0.857143	1	0.923077
17	22	3	4	0.846154	0.88	0.862745
18	26	4	2	0.928571	0.866667	0.896552
19	2	6	0	1	0.25	0.4
20	7	5	2	0.777778	0.583333	0.666667
21	2	1	3	0.4	0.666667	0.5
22	7	2	2	0.777778	0.777778	0.777778
23	9	2	4	0.692308	0.818182	0.75
24	1	1	1	0.5	0.5	0.5
25	21	0	1	0.954545	1	0.976744
26	11	4	1	0.916667	0.733333	0.814815
27	1	1	2	0.333333	0.5	0.4
28	3	2	6	0.333333	0.6	0.428571
29	2	2	2	0.5	0.5	0.5
30	13	5	1	0.928571	0.722222	0.8125
31	1	5	0	1	0.166667	0.285714
32	0	0	3	0	##	0
33	11	11	2	0.846154	0.5	0.628571
34	6	7	3	0.666667	0.461538	0.545455
35	11	21	4	0.733333	0.34375	0.468085
36	5	0	3	0.625	1	0.769231
37	0	0	1	0	##	0
38	36	1	21	0.631579	0.972973	0.765957

39	2	2	2	0.5	0.5	0.5
40	2	8	5	0.285714	0.2	0.235294
41	0	0	6	0	##	0
42	0	0	7	0	##	0
average	678	212	180	0.79021	0.761798	0.775744

C OBSAH PŘILOŽENÉHO DVD

Elektronická verze diplomové práce

Zdrojové soubory projektu

Vygenerované kaskádní systémy

Testovací obrazy

Konfigurační soubor aplikace