VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

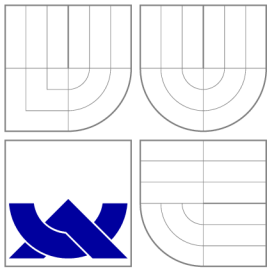
DETEKCE DOPRAVNÍCH ZNAČEK Z KAMERY VE VOZIDLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

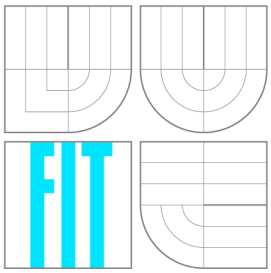
AUTOR PRÁCE
AUTHOR

JAN DUŠEK

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE DOPRAVNÍCH ZNAČEK Z KAMERY VE VOZIDLE

ROAD SIGN DETECTION FROM CAMERA IN CAR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN DUŠEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá detekcí dopravních značek v obraze nebo videu. Nejprve budou popsány algoritmy běžně využívané k tvorbě obecného detektoru dopravních značek. Následně popíšu detekci dopravních značek využívající metod histogram orientovaných gradientů a support vector machines. Nakonec zhodnotím výsledky, kterých se podařilo dosáhnout.

Abstract

This bachelor's thesis is focused on detection of traffic signs from image or video. Algorithms common for object detection will be introduced in the beginning. Description of object detection using histogram of oriented gradients and support vector machines will follow. Last part will present accomplished results.

Klíčová slova

dopravní značky, detekce, histogram orientovaných gradientů, support vector machines, posuvné okno

Keywords

traffic signs, detection, histogram of oriented gradients, suppot vector machines, sliding window

Citace

Jan Dušek: Detekce dopravních značek
z kamery ve vozidle, bakalářská práce, Brno, FIT VUT v Brně, 2015

Detekce dopravních značek z kamery ve vozidle

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D.

.....
Jan Dušek
29. července 2015

Poděkování

Rád bych poděkoval vedoucímu práce, Ing. Vítězslavu Beranovi, Ph.D, za jeho odbornou pomoc a vedení. Dále bych také rád poděkoval Michalu Jurčovi za poskytnuté výstupy jeho bakalářské práce.

Obsah

Kapitola 1

Úvod

Detekce dopravních značek v obraze je již několik let zkoumaný problém, související s potřebou vzniku autonomních vozidel, která budou schopna provozu bez zásahu člověka. K tomuto problému se vztahuje obrovské množství publikací a mnoho projektů již bylo nasazeno do běžného provozu — ať už se jedná o vozidlo bez řidiče, nebo o asistenty zabudované v systémech vozidel vyšších či středních tříd. Takový systém pak například umožní řidiči zapnout jakýsi mód „autopilota“ a věnovat se po krátkou dobu něčemu jinému, než řízení.

Motivací pro vznik systému detekce dopravních značek je jednak zmiňované vozidlo bez řidiče, jehož součástí pak takový systém bude. Dalším vhodným využitím je nápověda pro běžného řidiče, který může značku přehlédnout, nebo se může na neznámých a nepřehledných místech více věnovat okolnímu provozu. Systém mu pak během jízdy bude napovídat jaká značka je před ním, nebo jakou právě minul. Uplatnění může takový modul najít také v navigačních systémech, které můžou na základě zpozorovaných značek upravit trasu pro ostatní řidiče.

Ve své bakalářské práci se zabývám návrhem a implementací systému, který bude schopen dopravní značky ve snímcích detekovat a označovat. Práce obsahuje také náhled na metody počítačového vidění běžně využívané v úlohách detekce objektů. V poslední kapitole se věnuji výsledkům detekce a hodnotím fungování celého systému.

Kapitola 2

Detekce objektů

Počítačové videní jako rozsáhlý problém zasahuje do mnoha vědních odvětví. Existují proto stovky existujících řešení zabývajících se touto problematikou. Pravděpodobně nejčastěji je možné se setkat s detekcí obličejů v obraze, detekcí související se stavebnictvím nebo detekcí související s dopravou — dopravní značky, poznávací značky, chodci, vozidla, ...

V této kapitole představím detekci objektů v obraze všeobecně. Návrh detektoru rozdělím do jednotlivých částí a zmíním různé metody, kterých je při řešení dílčích kroků obvykle používáno. Jelikož jsou předmětem této práce dopravní značky, zmíním jejich vlastnosti a odlišnosti. Také popíši datové sady, které jsou využívány pro trénování a testování.

2.1 Dopravní značka

Dopravní značky se z historického hlediska změnily hlavně svým vzhledem. První značky měly podobu dřevěných nebo kamenných milníků, které určovaly hlavně vzdálenost mezi městy nebo udávaly směr. V pozdější době se značky začaly objevovat například u křižovatek — byly tedy větší a určovaly více směrů, než jeden. První skutečně dopravní značky sloužily hlavně cyklistům, kteří je sami stavěli aby varovali ostatní účastníky provozu před nebezpečím na cestě. S rozvojem automobilové dopravy se značky vyvíjely až do současné podoby. Dopravní značky jsou podobné ve státech sousedících — například státy Evropy podepsaly v roce 1968 smlouvu, která vedla k jisté standardizaci dopravního značení. Skupiny států používající podobné značky jsou samozřejmě k nalezení v Asii nebo Severní Americe.

2.1.1 Dělení značek

Značky lze roztřídit do mnoha různých kategorií, představím proto pouze rámcově ty nejpoužívanější. Kategorizace značek je důležitá zejména v úloze detekce a také se podle ní řídí finální testování detektoru. Nejčastěji se značky dělí následovně:

- Podle tvaru — trojúhelník, čtverec, kruh, osmiúhelník, ... – některé ze základních tvarů, podle kterých je možné rozdělit velkou většinu všech značek. Detekce na základě tohoto třídění je velice výhodná, jelikož se v přírodě například pravidelných kružnic moc nevysktytuje.
- Podle barvy. Dalším velice často využívaným dělením je dělení značek podle barev. Značky většinou tvořeny tak, aby se jejich vzhled výrazně lišil od běžného prostředí.

Ať už zvolenými barvami (sytě modrá, červená, žlutá), nebo kontrastem mezi těmito barvami (bílá s červenou, žlutá s černou).

- Podle typu — příkazové, zákazové, výstražné, ostaní. Pokud opomineme, že se každá kategorie může (a většinou dělí) ještě do několika konkrétnějších kategorií, je toto zjednodušené dělení značek také velice oblíbené. Značky spadající do stejné třídy se pak často vyznačují charakteristickým rysem, podle kterého je možné provádět detekci nebo klasifikaci.[1]

Jak je patrné, volba tvaru nebo barvy značky slouží k tomu, aby značka vizuálně vyčnívala z běžného prostředí a lidské oko ji rozpoznalo i na velkou vzdálenost. Stejným způsobem pak takové informace může využít aplikace počítačového vidění. Většina detekčních úloh se pak opírá o vícero takových informací - například práce [6], která vznikla jako výstup týmu, účastnícího se soutěže v detekci značek [8], se opírá o tvar i barevné schéma značky.

Obrázek 2.1: Značky zachycené v obtížných světelných podmínkách.[2]

2.1.2 Datové sady

V průběhu let bylo ve světě vytvořeno velké množsví datových sad obsahujících značky některých států. Výběr kvalitní a obsáhlé datové sady hraje podstatnou roli v celkové úspěšnosti detektoru. Kvalitnější datové sady obsahují jak neklasifikované značky, tak určitý počet značek s anotací. Za zmínku tedy stojí větší projekty (převážně z Evropy nebo Spojených Států Amerických), obsahující až desetitisíce dopravních značek.

Datová sada LISA [7], pocházející z americké Laboratoře pro Inteligentní a Bezpečná vozidla, obsahuje 6500 barevných i černobílých snímků ze 47 různých amerických značek.

Linköping Universtiy ve Švédsku také představila svou vlastní datovou sadu, nazvanou jednoduše *Traffic Signs Dataset*. Datová sada použitá například v [5] patří k nejrozsáhlejší vůbec. Obsahuje 20 000 snímků, z nichž je 20% anotovaných. Jedná se o snímky posbírané ze švédských silnic a dálnic. Tvůrci uvádějí, že obsluha automobilu spustila nahrávání kamery kdykoli byla značka v dohledu, poté kameru opět vypnula. Dá se tedy přepokládat, že téměř všechny snímky datové sady obsahují značku.

Německá datová sada *German Traffic Sign Detection Benchmark* [3], vytvořena Německým Institutem Neuroinformatiky, se přímo zaměřuje na výzkumné skupiny. Poskytuje celkem 900 obrázků, z nichž slouží 600 k trénování detektoru a zbylých 300 k hodnocení. K datové sadě je také přiložen soubor formátu *csv*, obsahující takzvanou *ground truth* - očekávané výstupy, které mohou sloužit k porovnání dosažených výsledků. Nižší počet snímků

[1] Například české příkazové značky jsou vždy vyobrazeny na výrazně modrém pozadí. Také jsou všechny kruhového tvaru.

[2] Převzato z [3].

je zapříčiněn hlavně tím, že se tvůrci zaměřili na detekci značky v na sobě nezávislých snímcích. Nejedná se tedy o datovou sadu složenou ze snímků pořízených z videa, což například znemožňuje využití předchozích snímků pro detekci značky na snímku aktuálním.

Zmiňovaný institut má na svědomí ještě jednu datovou sadu — *German Traffic Sign Recognition Benchmark* [8]. Ta obsahuje trénovací sadu pro aplikace strojového učení, která se skládá z bezmála 50 000 snímků obsahujících výřez se značkou. Výřezy jsou navíc rozděleny do více než 40 tříd. K těmto výřezům jsou k dispozici také vypočítané vektory deskriptorů.

Bohužel jsem nenalezl žádný oficiální zdroj datové sady, který by poskytoval dostačující množství snímků a zároveň vycházel pouze z českých dopravních značek. Při experimentech a vyhodnocení detekce jsem tedy použil obě německé datové sady zmíněné výše s vědomím, že se jedná o blízkého souseda naší republiky, a tak nebude odchylka ve vzhledu a typech dopravních značek tak markantní. Pravdivost tohoto předpokladu je ostatně potvrzena na obrázku 2.2.

(a) Přechod pro chodce (CZ)

(b) Zákaz vjezdu (CZ)

(c) Přikázaný směr jízdy (CZ)

(d) Přechod pro chodce (GER)

(e) Zákaz vjezdu (GER)

(f) Přikázaný směr jízdy (GER)

Obrázek 2.2: Srovnání českých značek a jejich německých ekvivalentů.[3]

Význam datové sady

Z práce [1] vyplývá, že datová sada, nad kterou je detektor trénován, hraje velkou roli v celkovém výkonu detektoru. Rozdíly v detektorech trénovaných nad jednou datovou sadou se lišily až o 30% při použití kvalitnější datové sady. Důležitým faktorem není jen objem dat v datové sadě, ale také její obsah. Vysoká různorodost prostředí, počasí, nebo světelných podmínek je pro kvalitní trénování detektoru kritická.

[3] Převzato z http://www.dopravni-znaceni.eu/ a https://en.wikipedia.org/wiki/Road_signs_in_Germany.

2.2 Extrakce příznaků z obrazu

Účelem extrakce příznaků je získání číselné informace z obrazu. Tyto informace jsou pak velice často vstupem pro aplikace strojového učení, rozpoznání tvarů nebo zpracování obrazu všeobecně. Extrakce tedy začíná načtením vstupních dat, ze kterých jsou získány informační hodnoty. Během tohoto procesu je navíc ignorována redundance, což zevšeobecňuje poskytnutá vstupní data. Metoda extrakce příznaků se používá hlavně v případech, kdy je zjevné, že nebude možné zpracovat celkový objem vstupních dat, nebo že vstupní data obsahují velkou redundanci. Výsledkem takového procesu je zpravidla sada příznaků (často nazývána příznakový vektor).

Není úplně možné jednoznačně definovat, jak výsledný příznak bude vypadat. Často se jeho podoba liší v závislosti na problému, ke kterému se aplikace vztahuje. Obrazové příznaky jsou také rozděleny do různých kategorií podle svých vlastnosti a výstupů. Například mezi metody označované za detekční se řadí detekce hran, detekce rohů nebo detekce kapek. Příznaky, které jsou vypočteny z takových metod je pak možné vizualizovat v původním obrazu, ať už třeba vyznačením rohů, nebo vykreslením všech hran.

Popisné příznaky jsou například výstupem metody histogram orientovaných gradientů nebo metody Scale Invariant Feature Transform[4]. Pravděpodobně nejznámnějším zástupcem této kategorie jsou Haarovy příznaky, které byly použity v prvním detektoru, který byl schopen detekovat objekty v reálném čase. Příznaky pak zpravidla nabývají podoby vektoru čísel. Takový vektor může být dále postoupen některému z algoritmů strojového učení.

2.2.1 Haarovy příznaky

Tyto příznaky dostaly své jméno kvůli podobnosti s *Haarovými vlnkami*. Haarovy příznaky mohou sloužit například při detekci objektu pomocí posuvného okna.

Příznaky jsou získávány z obrazu pomocí sčítání a odčítání hodnot pixelů. Pro každý příznak je vypočítán součet bodů nad bílým a černým obdélníkem (viz. 2.3) a tyto hodnoty jsou odečteny. Je patrné, že opakování takových výpočtů pro každý příznak by bylo časově neúnosné a často by se opakovalo pro již zpracované body. Integrální obraz (2.2.1) vhodně odstraňuje tento problém a výpočet pro jakkoliv velký výřez posuvného okna je proveden v konstantním čase.

V detekční úloze práce [9] jsou využity slabé klasifikátory[5] složené v kaskádě, která dohromady tvoří silný klasifikátor. Každý stupeň kaskády vyhodnotí příznaky pro výřez, ve kterém se nachází potenciální objekt — pokud vyhoví, zašle výřez dalšímu stupni kaskády. V případě, že výřez jednomu stupni kaskády nevyhoví, je zahozen.

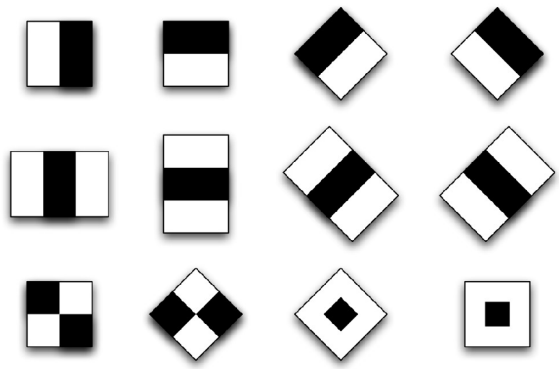
Integrální obraz

Hlavním důvodem pro použití integrálního obrazu je velice rychlé provedení konvoluce obrazu, nebo extrakce jednoduchých příznaků (jako bylo použito, a poprvé představeno, v práci [9]) nezávisle na velikosti nebo pozici výřezu, ze kterého budeme příznaky extrahovat.

Samotné sestavení integrálního obrazu je provedeno pouze v jednom průchodu obrazem, což logicky ještě více zvyšuje efektivitu finálního výpočtu.

[4] Respektive pouze část této metody - Scale Invariant Feature Detection.

[5] Pravděpodobnost, s jakou určují, zda se jedná o hledaný objekt, je mírně vyšší, než hádání.



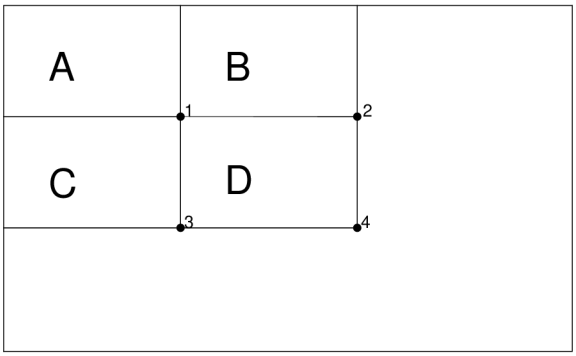
Obrázek 2.3: Příklad některých Haarových příznaků, které jsou využívány pro posuvné okno.[6]

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l) \tag{2.1}$$

Ze vzorce je patrné, že se jedná o postupný součet všech hodnot pixelů původního obrazu. Toho pak lze využít při rekurzivním výpočtu zbytku obrazu:

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j) \tag{2.2}$$

Výsledkem výpočtu je pak matice součtů hodnot všech pixelů původního obrazu vlevo-vzhůru od dané pozice.



Obrázek 2.4: Příklad použití integrálního obrazu. Pokud bychom chtěli vypočítat součet hodnot všech pixelů pod písmenem D za pomoci integrálního obrazu, výpočet by vypadal takto: $D_{sum} = s(4) + s(1) - (s(2) + s(3))$

[6] Převzato z http://fileadmin.cs.lth.se/graphics/theses/projects/facerecognition/.

2.2.2 Template matching

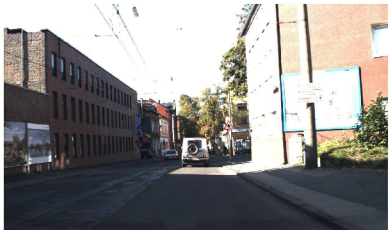
Jedná se o metodu zpracování obrazu, která srovnává části obrazu se vzorem a hledá shodu. Pokud lze z obrazu extrahovat příznaky, je srovnání provedeno na bázi těchto hodnot. V opačném případě je provedeno srovnání na základě vizuální podobnosti ve vzhledu daných dvou výřezů.

Ve své základní verzi využívá tato metoda konvoluční masku složenou z příznaků hledaného vzoru. Výstup konvoluce pak bude značit místa v obraze, která se shodují s konvoluční maskou. Tento postup je snadno použitelný ve snímcích převedených do odstínů šedi, nebo snímcích obsahujících pouze hrany.

Porovnávání na základě podobnosti většinou vyžaduje zpracování velkého množství bodů, je proto vhodné uvažovat o snížení počtu takových bodů — například poměrným snížením rozlišení obrazu a vzoru. Porovnávaný obraz je zmenšen vícekrát — tak vznikne sada obrazů různých velikostí. V této sadě je provedeno srovnání se zmenšenými vzory. Výsledky tohoto srovnání pak znázorňují možné oblasti výskytu hledaného objektu, které jsou využity při novém hledání — teď už v obraze plné velikosti.

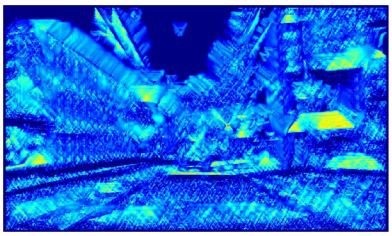
Zpracovávaný obraz může být samozřejmě různě natočen nebo špatně osvícen. V takovém případě je často využíváno vlastních prostorů (*angl. eigenspaces*). Používaný vzor je pozměněn, aby při porovnávání pokryl co největší množství variací v obraze — je například několikrát natočen, je změněna perspektiva, ze které je vzor zachycen, nebo jsou změněny světelné podmínky.

Tato metoda je využita například v práci [6]. Graficky je pak její fungování znázorněno na obrázku 2.5.

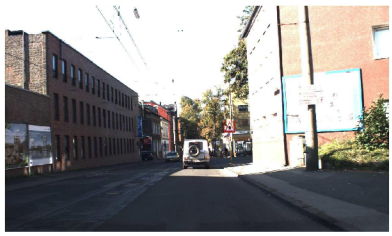


(a) Původní obraz.

(b) Obraz převedený do odstínů šedi.



(c) Konvoluce obrazu za použití vzoru pro výstražné značky.



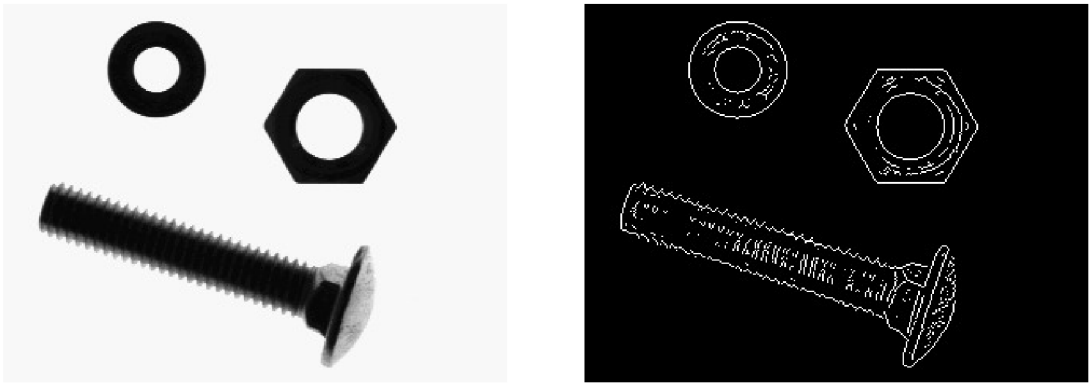
(d) Prahováním vzniká zájmová oblast.

Obrázek 2.5: Fungování modulu z práce [6] pro detekci zájmových oblastí, využívajícího template matching.

2.2.3 Cannyho detektor hran

Detekce hran spadá do kategorie detekce příznaků. John Canny vymyslel tento detektor už v roce 1986, přesto je stále hojně používán v úlohách počítačového vidění. Tato technika se prokázala velice užitečnou při extrakci strukturované informace z různých objektů. Také vysokou měrou snižuje množství dat, která je nutné prozkoumat a vyhodnotit (viz. 2.6). Canny také určil tři základní kritéria, které by měl detektor hran splňovat, na jejichž základě pak výsledný detektor vznikl.

- Detektor by měl detekovat pouze hrany, které v obraze opravdu existují
- Vzdálenost (v pixelech) mezi detekovanou hranou a hranou reálného objektu by měla být minimální
- Detektor by měl na každou hranu obrazu reagovat pouze jednou



(a) Původní obraz (b) Obraz s detekovanými hranami

Obrázek 2.6: Porovnání původního obrazu a obrazu s detekovanými hranami.[7]

2.3 Histogram orientovaných gradientů

Tato metoda extrakce příznaků z obrazu byla poprvé představena v práci [2]. Z nastudovaných publikací vyplývá, že valná většina kvalitních detektorů využívá tuto metodu — ať už ve své původní verzi, nebo mírně vylepšenou ([1], [10]). Rozhodl jsem se proto ji ve své práci také využít.

Metoda je založena na histogramu orientovaných gradientů. Základní myšlenka spočívá v tom, že prestože neznáme přesnou polohu objektu v obraze, může být objekt pomocí svého vzhledu nebo tvaru charakterizován intenzitou gradientu.

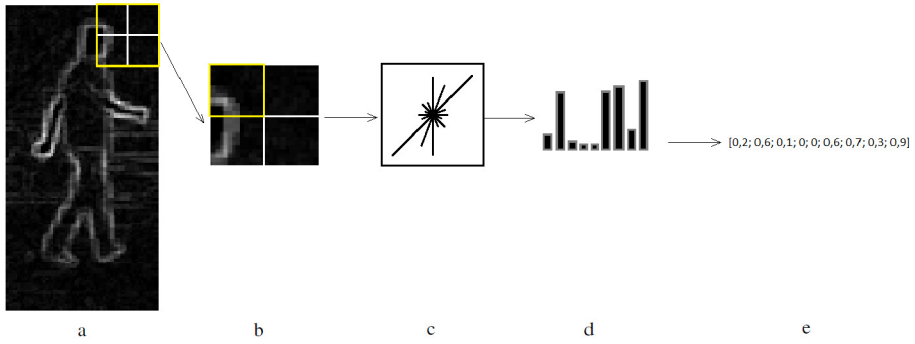
Obraz je rozdělen na buňky pevné velikosti a pro každou z nich je vypočítan jednorozměrný histogram. Obraz je vhodné před jakýmikoli výpočty normalizovat. Toho je docíleno shromažďováním buňek do tzv. bloků. Histogram orientovaných gradientů je pak buňka normalizovaná blokem.

[7] Převzato z http://www.ele.uri.edu/~hansenj/projects/ele585/lab8/.

Buňka jakožto hlavní nositel informace zachycuje určitou oblast snímku. V rámci každé buňky je určen směr gradientů, který by pak měl vymezit hledaný objekt. Detekci gradientu lze provést využitím funkce pro detekci významných hran v obraze — jednoduchého detektoru hran. V takových místech totiž dochází ke skokové změně v jasu obrazu.

Orientace hran se pak logicky pohybuje v rozmezí 0° - 360°. Není ale nutné využívat celý kruhový prostor, jelikož hrany nejsou orientované, prochází hrana na imaginární jednotkové kružnici jak bodem $\frac{\pi}{2}$ tak bodem $\frac{3\pi}{2}$. Využívaný stupňový prostor lze tedy zmenšit na 0° - 180°. Tento rozsah by ale pořád obsahoval 181 kategorií, proto jsou vytvořeny tzv. kanály, které seskupují gradienty.

Součet stejně orientovaných gradientů náležících do stejných kanálů pak tvoří výslednou velikost kanálu. V mé práci jsem využil 8 kanálů, což znamená, že například do prvního kanálu jsou zařazeny gradienty s hodnotami 0° - 22.5°. Z takových kanálů se pak vytvoří lokální histogram, který znázorňuje zastoupení gradientů v jednotlivých kanálech. Výsledný vektor je tedy složen ze všech histogramů jednotlivých buněk, normalizovaných dle buněk v jejich bezprostřední blízkosti.



Obrázek 2.7: Ukázka extrakce příznaku v metodě histogram orientovaných gradientů.[8]

Předzpracování obrazu

Může se zdát, že je před výpočtem vhodné provést změnu vstupního obrazu a vylepšit tak výkon výsledného detektoru. Autoři metody [2] zkoumali různé úpravy vstupního obrazu — například redukcí barevného prostoru RGB do stupní šedi, nebo použitím filtrů pro úpravu osvětlení a jasu. Autoři zjistili, že tyto úpravy však mají minimální vliv na výsledný výkon detektoru, natrénovaného za pomoci vylepšených deskriptorů.

Výpočet gradientů

Gradient se v kontextu zpracování obrazu používá nejčastěji pro detekci hran v obraze. Gradient f v bodě (x, y), označený jako Δf je definován vektorem:

$$\Delta f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \tag{2.3}$$

[8] Převzato z http://darkpgmr.tistory.com/116.

Tento vektor udává největší změnu intenzity f v bodu (x, y). Velikost změny ve směru gradientu vektoru je pak:

$$|\Delta f| = \sqrt{{g_x}^2 + {g_y}^2} \tag{2.4}$$

Směr gradientu je dán úhlem:

$$\alpha(\Delta f) = tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{2.5}$$

K získání gradientu obrazu se používá parciální derivace $\frac{\partial f}{\partial x}$ a $\frac{\partial f}{\partial y}$. Aproximace parciálních derivací je nejčastěji prováděna za pomoci sousedních pixelů zjišťovaného bodu. Rovnice

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1, y) - f(x, y) \tag{2.6}$$

a

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x, y+1) - f(x, y) \tag{2.7}$$

jsou využity pro hodnoty x a y filtrace $f(x, y)$ s 1-D maskou. Pokud chceme při výpočtu gradientu zahrnout i pixely diagonálně sousedící se zjišťovaným bodem, je nutné využít 2-D masku.

Nechť existuje obecná maska, která ilustruje výpočty v rovnicích 2.9 a 2.10:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \tag{2.8}$$

Aproximace 2-D maskou o rozměru 3x3 nazývanou Sobelův operátor je pak vypočtena:

$$g_x = \frac{\partial f}{\partial x} = (a_7 + 2a_8 + a_9) - (a_1 + 2a_2 + a_3) \tag{2.9}$$

a

$$g_x = \frac{\partial f}{\partial x} = (a_7 + 2a_8 + a_9) - (a_1 + 2a_2 + a_3) \tag{2.10}$$

Samotný Sobelův operátor je často využíván. Vychází z aproximace dle Prewittové — jediným rozdílem je to, že střednímu koeficientu přiřazuje dvojnásobnou váhu — dochází pak k většímu vyhlazení obrazu.

Normalizace bloků

Výsledné histogramy se významně liší v rámci celého obrazu — to je způsobeno nerovnoměrným osvětlením a rozdíly v kontrastu mezi popředím a pozadím. Pro použitelné výsledky je nutné tyto vlivy odstranit. Většina normalizačních schémat je založena na seskupování buňek do bloků a jejich následné normalizaci. Překrývání bloků a vícenásobné zpracování jednotlivých buněk je velice žádoucí pro získání kvalitního deskriptoru.

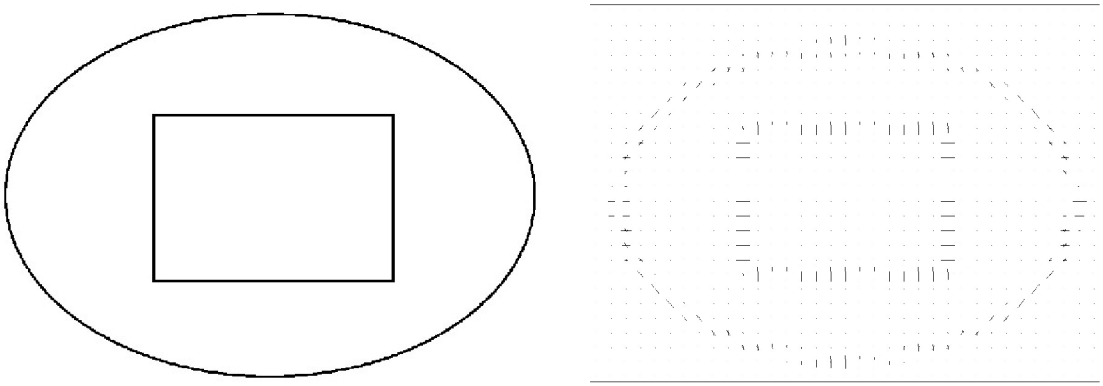
Normalizace pak může probíhat následujícím způsobem. Nechť $\vec{v}$ je nenormalizovaný vektor obsahující všechny histogramy v určitém bloku a nechť ϵ je malá konstanta. Pak $\| v \|_k$ je jeho k-norma pro $k = 1, 2$. Různá normalizační schémata pak vypadají takto:

$$L2-norm: \qquad v \longrightarrow \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}} \tag{2.11}$$

$$L1-norm: \qquad v \longrightarrow \frac{v}{\|v\|_1 + \epsilon} \tag{2.12}$$

Často využívané schéma *L2-hys* využívá *L2-norm* — po první normalizaci zahodí hodnoty vyšší než je zvolený práh a normalizuje je ještě jednou. Dochází tak k potlačení nežádoucího šumu.

Výsledný vektor se skládá z deskriptorů — proto je také nazýván popisný vektor (*angl. descriptor vector*). Jeho délka je závislá na počtu kanálů, buněk, velikosti bloku a také posunu bloku při normalizaci. Posledním krokem je předání tohoto vektoru nějakému algoritmu strojového učení.



(a) Původní obraz

(b) Výsledné deskriptory a jejich orientace

Obrázek 2.8: Grafické znázornění orientací a velikostí výsledných deskriptorů.[9]

2.4 Strojové učení a výsledný model

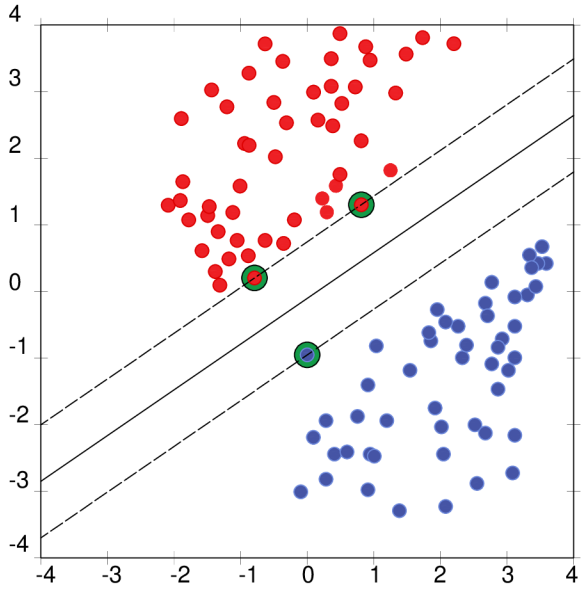
Výstupem algoritmů strojového učení je většinou model, který provádí odhady nebo rozhodnutí. Existuje mnoho algoritmů strojového učení — neuronové sítě, rozhodovací stromy, lineární regrese, . . . , ve své práci zmíním pouze jeden — *Support Vector Machines*. Ač ve své práci tento klasifikátor přímo neimplementuji, využívám výstupy tohoto algoritmu poskytnuté z [4].

2.4.1 Support Vector Machines

Ve své základní verzi je SVM diskriminativním klasifikátorem, který dokáže oddělit data dvou tříd (pokud jsou lineárně oddělitelná). Jeho výpočetní složitost nezávisí v takové míře na velikosti vstupního vektoru (narozdíl od neuronových sítí). Trénování SVM závisí na množině dat, ve které je každý vzorek označen za pozitivní (+1) nebo negativní (-1). Mezi těmito dvěma třídami hledá SVM dělící rovinu, která má od všech prvků co největší vzdálenost. Podpůrné vektory (*support vectors*) jsou pak vzorky, které mají v rámci jedné

[9] Převzato z http://answers.opencv.org/question/33689/hog-descriptor-orientation.html.

třídy nejmenší vzdálenost od takové roviny. Na jejich základě je pak tvořen model, který lze použít pro následnou klasifikaci. Nalezená rovina lineárně oddělující dvě třídy je vidět na obrázku 2.9.



Obrázek 2.9: Lineárně oddělitelné třídy. Podpůrné vektory jsou vyznačeny zelenou tečkou.[10]

2.5 Detekce objektu v obraze

Posledním krokem je samotná detekce značky v obraze. Opět existuje mnoho metod, kterých se využívá při hledání potenciálních oblastí se značkou. V této kapitole zmíním pouze posuvné okno, které jsem využil i ve své práci.

2.5.1 Posuvné okno

Princip této metody procházení obrazu začíná vytvořením detekčního okna. Okno musí být stejně velké, jako okno použité při tvorbě příznakových vektorů (délky těchto vektorů se musí shodovat). Detekční okno je pak posouváno ve vertikálním i horizontálním směru přes celý obraz. Délka kroku rozhoduje o kvalitě samotné detekce – jemný krok má potenciál najít mnohem větší množsví značek. Na druhou stranu takový krok negativně ovlivňuje rychlost detekce, jelikož je nutné vyhodnotit mnohem více pozic detekčního okna.

Pro každou pozici detekčního okna je pak vypočten nový histogram orientovaných gradientů, který je předán k porovnání modulu strojového učení popsaném v předchozí kapitole.

[10] Převzato z http://www.mblondel.org/journal/2010/09/19/support-vector-machines-in-python/.

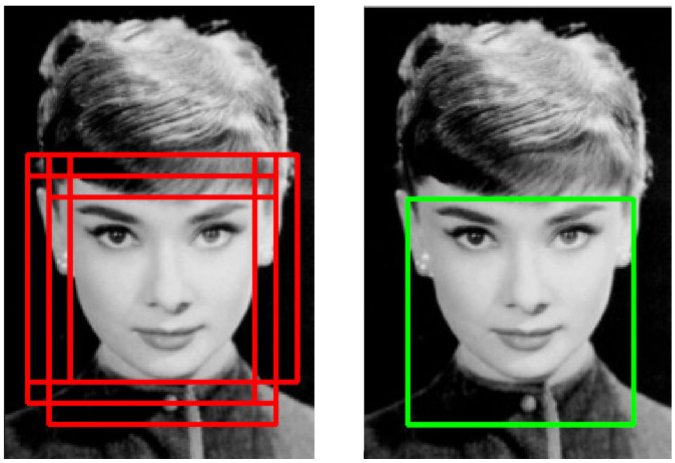
Ten pak rozhodne, zda se v aktuálním výřezu obrazu nachází nebo nenachází hledaný objekt.

Když detekční okno dojde na konec obrazu, je obraz o malé procento zmenšen a celý postup se několikrát opakuje. V obraze je pak možné nalézt objekty, které jsou větší, než velikost detekčního okna.

Non-maxima suppression

Tato metoda je hojně využívána právě spolu s posuvným oknem. Při použití posuvného okna je totiž často vyhodnoceno více než jedna pozice detekčního okna nad hledaným objektem jako platná (viz. 2.10). Je proto využita metoda pro potlačení ostatních oken a výběr pouze jednoho výsledného čtyřúhelníku, který by měl správně ohraničit daný objekt.

Zároveň může být nalezen objekt i na místě, kde se ve skutečnosti nenachází. Tato chyba je odhalena například tím, že na stejném místě jen málo dalších oken vyhodnotilo oblast jako platnou. Taková oblast je pak zahozena a není zobrazena ve výsledném obrazu s detekovanými objekty.



(a) Kladně vyhodnocené pozice posuvného okna.

(b) Výsledná oblast.

Obrázek 2.10: Ilustrace metody *non-maxima suppresion*.[11]

[11] Převzato z http://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/.

Kapitola 3

Funkce a nástroje pro detekci dopravních značek

Cílem práce bylo vytvořit aplikaci, která bude ve snímcích z kamery ve vozidle detekovat naučenou sadu dopravních značek. V této kapitole blíže popíšu samotnou implementaci mého programu, která přímo vychází z informací nashromážděných a popsaných v kapitole 2. Zmíním nástroje, které jsem využil během implementace, a také podpůrné skripty a vedlejší části programu.

3.1 Architektura výsledného programu

Výsledný program má dvě hlavní části. První poskytuje uživateli možnost ze zadných snímků vytvořit vektory deskriptorů, které pak mohou být použity v algoritmech strojového učení pro vytvoření nějakého modelu. Druhá pak předpokládá existenci takového modelu a používá ho. Uživatel předloží jednotlivé snímky nebo videosoubor, ve kterých jsou nalezeny potenciální oblasti obsahující dopraví značku a jsou označeny. Obě části programu budou blíže popsány v kapitolách 3.2 a 3.3.

Program je složen ze sedmi tříd — jejich závislosti a vzájemné interakce jsou zobrazené na obrázku 3.1.

Třída `Options` poskytuje metody pro zpracování parametrů předaných příkazovou řádkou a kontroluje jejich obsah. Na základě předaných parametrů nastavuje hodnoty odpovídajících proměnných.

Třída `Support_Files` nejprve vybere přípony souborů podle toho, zda uživatel nějaké zakázal a následně za pomoci nástrojů knihovny *Boost* zpracuje cesty ke vstupním souborům a připraví je pro následné vyhodnocení. Poskytuje také metody pro uložení vypočtených deskriptorů nebo obrazu či videa s označenými značkami. Ukládání videa je implementováno pomocí *OpenCV* třídy `cv::VideoWriter`.

`Support_Detection` obsahuje deklaraci struktury, do které jsou ukládány informace o videosouboru. Struktura obsahuje informaci o velikosti snímku, snímcích za vteřinu nebo počtu všech snímků videa. Tato třída obsahuje také metodu pro inicializaci zmiňované struktury informacemi o videu. Dále obsahuje třída deklaraci struktury pro nalezený objekt, obsahující ohraničující obdélník a také informaci o skóre. V poslední řadě obsahuje deklaraci struktury trackeru a také funkci pro sledování objektů v navazujících snímcích videosoboru (kapitola 3.2.1).

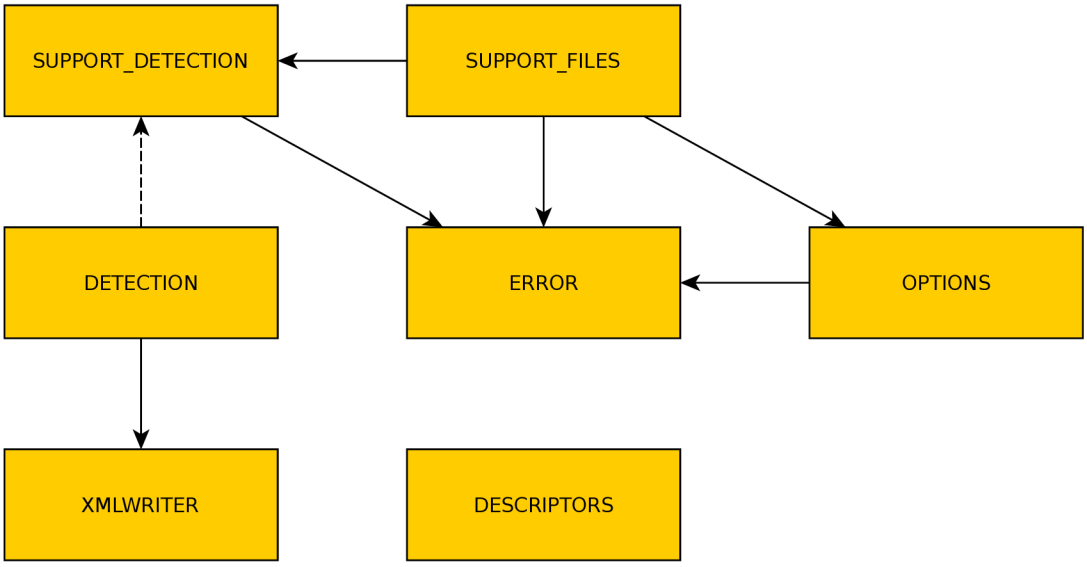
Jednou ze dvou hlavních tříd je třída `Detection`. Třída poskytuje metody, které na zá-

kladě předaného *SVM modelu* detekují značky jak v obraze tak ve videu. Další metody pak značky ve snímcích označí. V průběhu detekce jsou předávány informace třídě `XmlWriter`, která vypisuje informace o detekci na standartní výstup.

Druhou hlavní třídou je třída `Descriptors`. Ta obsahuje pouze jednu metodu, která přijme vstupní obrázek, vypočte a vrátí jeho vektor deskriptorů.

Třída `XmlWriter` tedy poskytuje uživateli textový výstup týkající se detekce. Vypisuje názvy procházených souborů spolu se souřadnicemi nalezených potenciálních značek, nebo třeba dobu zpracování jenoho snímku i celé várky souborů.

Nakonec se třída `Error` stará o hlášení o chybách, které vypisuje na stadnartní chybový výstup. Konstruktoru třídy je předána chybová zpráva a její typ (chyba v parametru, jiná chyba), která je následně vypsána.



Obrázek 3.1: Diagram vztahů mezi třídami. Plná čára značí používání, čárkovaná dědičnost.

Knihovna OpenCV 2.4.10

Tato knihovna[1] poskytuje rozhraní pro zpracování obrazu a videa. Je volně šiřitelná a multiplatformní. Určena je jak pro akademické účely, tak komerční využití. Poskytuje rozhraní pro jazyky *C, C++, Python, Java*. Celá knihovna je napsána v jazyce *C/C++* a je funkční pod operačnímy systémy Linux, Microsoft Windows a Mac OS. Distribuována je pod BSD licencí.

Ve své práci jsem využíval zejména knihovny *core, highgui, imgproc, objdetect*.

Knihovna Boost 1.58.0

Boost[2] je soubor více než 80 knihoven pro jazyk *C++*, který poskytuje podporu pro různé úkoly — od lineární algebry, vícevláknového zpracování po zpracování obrazu a regulární výrazy. Knihovny Boost jsou také zařazeny do standartu *C++11* a je plánováno

[1] http://www.opencv.org/
[2] http://www.boost.org/

zařazení dalších do standartu *C++17*. Distribuována je pod vlastní licencí *Boost Software Licence*, která je dost podobná licenci BSD.

Ve své práci jsem využil zejména knihovnu *system* a *filesystem*.

IDE CodeLite 7.0

CodeLite[3] je multiplatformní vývojové prostředí pro jazyky *C, C++, PHP*, které běží pod všemi hlavními operačními systémy. Poskytuje klasické doplňky, jako verzovací systém, generování částí kódu, kontrolu správy paměti nebo *debugger*.

3.2 Program pro detekci dopravních značek

Parametr `--detection` logicky vyvolá detekční část programu. Ke svému fungování vyžaduje cestu k souboru nebo složce obsahující snímky nebo videa a navíc cestu k *SVM modelu*, který je převeden do formátu pro použití s knihovnou *OpenCV*. Stejně jako v části pro tvorbu deskriptorů jsou nejprve rekurzivně zpracovány vstupní složky obsahující snímky nebo videa. Ty jsou pak připraveny ke zpracování. Pro snímky to znamená pouze uložení cesty, videosoubory jsou načteny do struktury `videoFileStruct`, do které se ukládají například informace o velikosti snímku, počtu snímků ve videu nebo počtu snímků za sekundu. Do struktury jsou také připraveny instance tříd `cv::VideoCapture` a `cv::VideoWriter`, které slouží k načtení, respektivne uložení videosouboru s označenými značkami.

Dalším krokem je vytvoření instance třídy `cv::HOGDescriptor`. Konstruktoru třídy jsou předány tyto parametry:

- Velikost buňky — 5x5 pixelů
- Velikost bloku — 10x10 pixelů
- Posun bloku při normalizaci — 5 pixelů v obou směrech
- Počet kanálů v buňce — 8
- Typ normalizace — L2Hys s hranicí 0.5

Metodou `cv::HOGDescriptor::setSVMDetector` je třídě předán model, který je následně využit pro detekci.

Samotná detekce je implementována metodou `cv::HOGDescriptor::detectMultiScale`, která využívá již zmíněného posuvného okna a metody non-maxima suppression (2.5.1). Parametry předané metodě jsou:

- Hranice pro vzdálenost mezi příznaky a SVM rovinou — -100.0
- Posun posuvného okna — 10 pixelů v obou směrech
- Koeficient zvětšení detekčního okna — 1.05
- Koeficient seskupení čtyřúhelníků nad jedním objektem — 10.0

[3] http://www.codelite.org/

Metoda vyžaduje jako další parametry také snímek a dva vektory. Do prvního vektoru uloží všechny nalezené čtyřúhelníky. Do druhého pak váhy, které symbolizují jistotou, s jakou označila metoda ten který čtyřúhelník.

Třetím krokem je pak označení všech čtyřúhelníků v obraze — všechny čtyřúhelníky jsou ještě před označením mírně zmenšeny, jelikož jsou větší, než hledaný objekt.

V posledím kroku je snímek uložen. Pokud byl vstupem detektoru obrázek, je uložen (opět v závislosti na parametru výstupní cesty) ve formátu `puvodni_jmeno_processed.jpg`.

Pokud se jednalo o videosoubor, všechny vyjmenované kroky jsou provedeny najednou — tedy je načten jeden snímek videa, použita metoda pro detekci objektů v obraze, všechny nalezené výskyty jsou ve snímku označeny a snímek je zapsán do výstupního souboru. Soubor je uložen s příponou *avi* v kodeku *DIVX*.

3.2.1 Tracking detekovaných objektů ve videu

Jelikož detekce ve videosouborech vykazovala také vysoké množství falešené detekovaných oblastí, pokusil jsem se implementovat metodu pro sledování objektů v po sobě jdoucích snímcích. Cílem bylo odfiltrovat případný šum a snížit počet falešných detekcí.

Z každé detekované oblasti je vytvořen nový *tracker*. Před začátkem porovnání je doba života všech existujících trackerů snížena o konstantní hodnotu. Pokud detekovaná oblast překrývá již existující tracker, je jeho doba života zvýšena o konstantní hodnotu a jeho poloha je aktualizována na polohu dané detekce. V opačném případě je na místě detekce vytvořen nový tracker a je mu přiřazena hodnota reprezentující dobu života. V mé implementaci vydrží tracker dva snímky bez aktualizace — pak zaniká. Detekce, která nepřekryla žádný tracker, není zobrazena v aktuálním snímku.

3.3 Extrakce deskriptorů

Tok programu je do této části přesměrován pomocí parametru `--descriptors`. Ke svému fungování potřebuje pouze cestu k souboru nebo složce, obsahující snímky. Složka je rekurzivně zpracována a všechny cesty k nalezeným snímkům s podporovanou příponou jsou uloženy do vektoru. Během procházení jsou brány v potaz přípony, které se uživatel rozhodl explicitně vynechat.

Jak bude zmíněno v kapitole 3.4.1, při výpočtu vektoru deskriptorů, které budou následně použity pro trénování modelu, jsem se snažil přiblížit hodnotám, které byly přiloženy k datové sadě [8]. Jelikož velká většina výřezů poskytnutých k této sadě měla rozměry blížící se 40x40 pixelů, autoři změnili rozměry všech výřezů na tuto hodnotu. Pro potřeby testovacího skriptu pro kontrolu správnosti z kapitoly 3.4.1 jsem tedy všechny zpracované snímky zmenšil také. Při trénování modelu bylo také však nutné vytvořit vektory deskriptorů pro negativní snímky[4]. Ty už měly různé velikosti, bylo proto nutné upravit velikost buněk a bloků použitých při výpočtu histogramu orientovaných gradientů tak, aby byla výsledná délka vektoru vždy 1568.

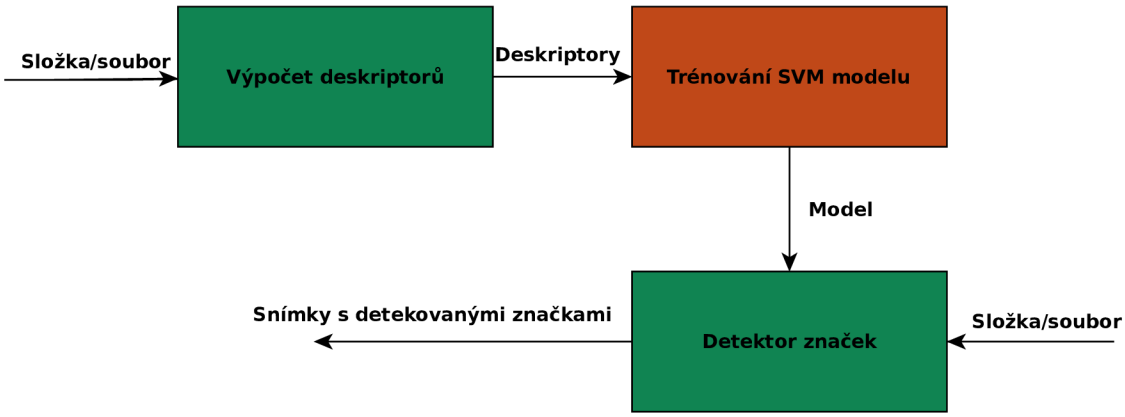
Velikost snímku jsem tedy měnil tak, aby se každý z jeho rozměrů rovnal násobku 40 (a byl minimální velikosti 40x40 pixelů). Každou z velikostí (buňka, blok, posun bloku) jsem pak násobil poměrem mezi velikostí obou rozměrů nového snímku a výchozí velikostí snímku.

Samotná extrakce příznakového vektoru je implementována pomocí metody z knihovny *OpenCV* `cv::HOGDescriptor::compute`.

[4] Snímky neobsahující hledaný objekt.

Posledním krokem je pak zapsání deskriptorů po řádcích do souboru. Soubor s deskriptory má stejný název, jako soubor vstupní — pouze jeho koncovka je změněna na *txt*. Cesta výstupního souboru se řídí parametrem pro výstup. Pokud uživatel zvolil složku, do které chce soubory ukládat, jsou uloženy do ní, v opačném případě je soubor s deskriptory uložen vedle vstupního souboru.

Soubory s deskriptory byly předány kolegovi Michalu Jurčovi [4], který se ve své bakalářské práci zaměřil zejména na problém strojového učení a také trénování modelu, který jsem využil v detekční části své práce. Tato spolupráce je znázorněna na obrázku 3.2.



Obrázek 3.2: Pipeline mezi mými moduly (zelená) a modulem z [4] (oranžová).

3.4 Skripty

Všechny skripty byly implementovány v jazyce *Python2*.

3.4.1 Kontrola správnosti výpočtu deskriptorů

Jak jsem uvedl v kapitole 2.1.2, využil jsem ve své práci datovou sadu [8] a [3]. Ze snímků první datové sady jsem vypočítal vektory deskriptorů, které byly dále použity pro trénování modelu. K této datové sadě byly přiloženy také již vypočítané vektory deskriptorů — se třemi různými nastavení. První z nich jsem zvolil jako referenční a při používání metody pro extrakci deskriptoru z knihovny *OpenCV* se jim snažil přiblížit.

Autoři datové sady zmenšily všechny výřezy na velikost 40x40 pixelů, velikost buňky nastavili na 5x5 pixelů, velikost bloku byla zvolena na 10x10 pixelů (4 buňky na jeden blok). Posun bloku při normalizaci byl zvolen 5 pixelů horizontálně i vertikálně. Počet kanálů jedné buňky se nakonec rovnal 8.

Blok o velikosti 10x10 pixelů je tedy posouván o 5 pixelů v obou směrech přes obraz velikosti 40x40. Vznikne tak 49 různých pozic bloku v obraze, každá obsahující 4 buňky. Buňka se navíc skládá z 8 kanálů. Délka výsledného vektoru je tedy:

$$v = 7 * 7 * 4 * 8 = 1568 \tag{3.1}$$

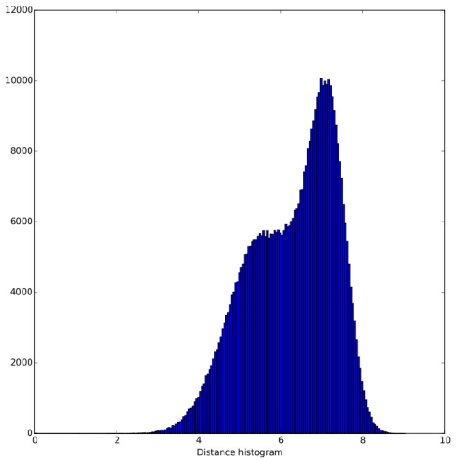
Vypočítal jsem tedy deskriptory pro výřezy stejným způsobem a zkontroloval, zda mají mnou vypočítané vektory nad stejnými daty podobné vlastnosti, jako deskriptory referenční.

Jinými slovy zda jsou stejná obrazová data popsána způsobem se stejným rozložením.

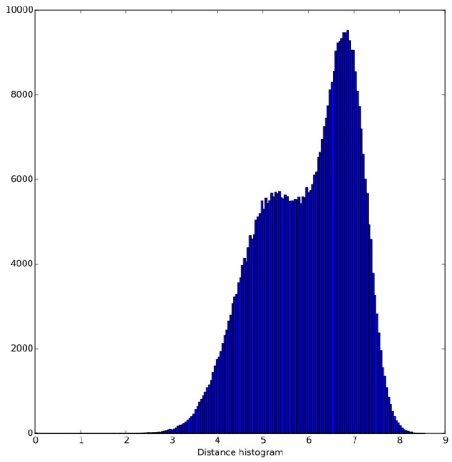
Jedno spuštění skriptu se skládalo z 20 běhů, v každém běhu jsem ze 40 000 referenčních vektorů náhodně zvolil 1000. Pak jsem vybral odpovídajících 1000 mnou vypočítaných vektorů. Pro referenční vektory jsem vypočítal euklidovskou vzdálenost každého vektoru s každým. Tuto akci jsem provedl i pro mnou vypočítané vektory. Z obou matic vzdáleností jsem pak vytvořil histogramy a srovnal je metodou pro srovnání histogramů nazývanou průnik (*angl. intersection*). Výpočet průniku je následující:

$$H_a, H_b = \sum_{i=1}^{n} min(a_i, b_i) \tag{3.2}$$

Kde H_a a H_b jsou dva histogramy (viz. 3.3) složené ze stejného počtu vstupních hodnot a n je počet sloupců histogramu. Nakonec a_i a b_i reprezentují hodnoty histogramů H_a a H_b v odpovídajících sloupcích.



(a) Histogram vzdáleností mezi mnou vypočítanými vektory



(b) Histogram vzdáleností mezi referenčními vektory

Obrázek 3.3: Vizuální porovnání dvou histogramů vzdáleností, které byly použity při srovnání.

Součet je nakonec nutné normalizovat počtem hodnot, které byly použity pro tvorbu histogramu. Průnik identických histogramů je pak roven 1. Průnik histogramů, které nemají žádné společné hodnoty je 0.

Průměr průsečíků histogramů 20 běhů po 1000 hodnotách je pak vidět v tabulce 3.1.

Intersection	**MY-MY**	**REF-REF**
MY-MY	1	0.967917
REF-REF	0.967917	1

Tabulka 3.1: Výsledné hodnoty průsečíků mezi histogramy.

Kde *MY-MY* a *REF-REF*, značí histogram vytvořený z matice, složené ze vzdáleností mezi mnou vypočítanými vektory, respektive referenčními vektory. Z výsledku je patrné, že shoda obou matic je 96.8%, což jsem považoval za dostačující.

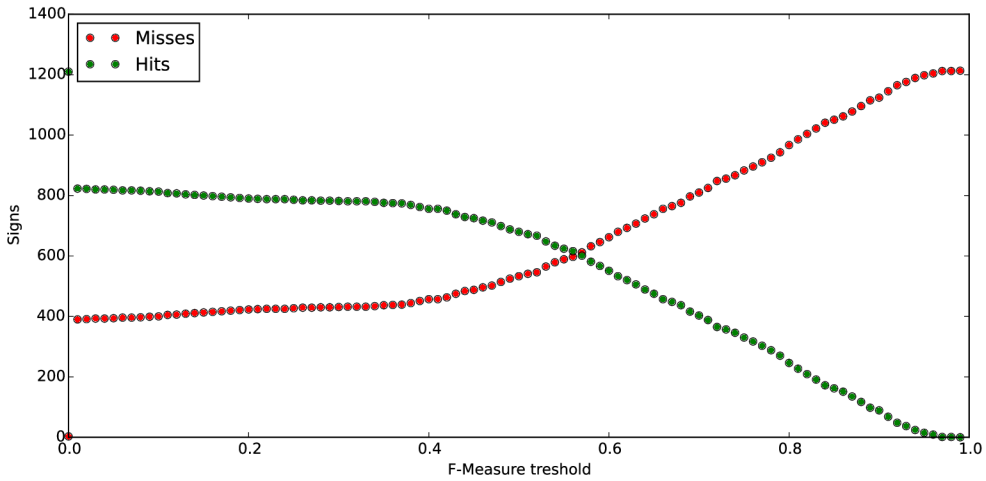
3.4.2 Vyhodnocení detekce

Prvním krokem vyhodnocení je načtení souboru (nebo složky obsahující soubory) typu *csv*, který je výstupem detekce. Do slovníku jsou pak načteny všechny souřadnice nalezených objektů a jejich skóre.

Souřadnice jsou předány konstruktoru nového datového typu `Rect`, který se snaží kopírovat datový typ poskytovaný knihovnou *OpenCV* – `cv::Rect`. Ohraničení značky je zde reprezentováno souřadnící levého horního rohu, šířkou a výškou čtyřúhelníku. Operace odčítání a sčítání nad tímto datovým typem pak implementuje průnik, respektive sjednocení dvou čtyřúhelníků. Operátor & pro průnik dvou čtyřúhelníků je podporován pouze v jazyce *C++*, musel jsem proto tuto operaci implementovat ručně.

Následuje načtení souboru (nebo celé složky) obsahující *ground truth* — hranice objektů jsou opět převedeny do připraveného datového typu.

Existují tedy dva slovníky obsahující informaci o názvu souboru a odpovídajících nalezených, respektive hledaných souřadnicích. Každý čtyřúhelník z hledané sady je pak porovnán se všemi nalezenými čtyřúhelníky odpovídajícího souboru metrikou F-Measure, která bude blíže popsána v kapitole 4.1. Aby byla značka označena za nalezenou, musí být skóre metriky alespoň 0.5. V opačném případě značka nalezena nebyla. Vztah zvolené hranice F-Measure a počtu značek označených jako *hit/miss* je vidět v grafu 3.4.



Obrázek 3.4: Počet značek označených jako hit/miss ve vztahu k hranici F-Measure.

Skript shromažďuje mimo jiné informace o počtu souborů, správně a chybně označených výřezů nebo správně a chybně odmítnutých výřezů. Informace vypisuje na standartní výstup.

Pro potřeby vyhodnocení jsou generovány ROC[5] křivky. V případě, že se jednalo o jeden soubor, je vygenerovaná křivka pouze jedna. Pokud byla skriptu předána složka, jsou

[5]Reciever Operating Characteristic

všechny křivky seskupeny do jednoho grafu, aby bylo možné výsledky vizuálně porovnat. Analýzou křivky je možné nalézt hranici detekčního skóre shodujícího se s požadovaným chováním detektoru (například vysoký počet správně označených oblastí (*true positive*) za cenu vysokého množství chybně označených oblastí (*false positive*)).

3.4.3 Anotace snímků

Pro potřeby vyhodnocení detektoru jsem shromažďoval snímky z vlastní datovoé sady. Tato datová sada musela být anotována, aby bylo možné porovnat nalezené výsledky s těmi správnými. K tomuto účelu jsem vytvořil skript, který takovou anotaci umožňuje.

Nástroj přijme jako parametr složku obsahující snímky z kamery ve vozidle a volitelně obsahující již vytvořený soubor s anotacemi. Jeden po druhém otevírá snímky a umožňuje uživateli myší označit hledanou oblast. Jednotlivé akce jsou pak prováděny stiskem klávesy. Každá označená oblast se může nacházet ve čtyřech stavech:

- Předem označená (červená barva) - oblast byla načtena z existujícího souboru s anotacemi a pokud uživatel nezvolí jinak, bude beze změny uložena znovu
- Aktivní (zelená barva) - nově vytvořená oblast, se kterou uživatel právě pracuje. V tomto módu může uživatel dané oblasti přiřadit třídu, GPS souřadnice, může danou oblast smazat nebo změnit její rozměry za pomoci šipek
- Vybraná (modrá barva) - v případě, že je ve snímku více označených oblastí a uživatel potřebuje jednu z nich změnit, přejde do módu selekce a pomocí šipek přejde na tuto oblast
- Změněná (žlutá barva) - tato oblast bude po zavření snímku uložena do anotačního souboru (pokud uživatel nerozhodne zavřít snímek bez uložení změn)

Kompletní návod k ovládání je k dispozici uvnitř skriptu. Je vypsán na požádání a také ihned po spuštění.

Nástroj je možné také spustit v módu „video“ (za pomoci přepínače `-v` nebo `--video`). V tomto módu se předpokládá existence po sobě jdoucích snímků, které byly shromážděny z kamery ve vozidle. Pokud uživatel označí oblast v jednom snímku a rozhodne se uložit změny, je daná oblast zobrazena na snímku následujícím.

Výsledky jsou do anotačního *csv* souboru zapsány v následujícím formátu:

```
název_snímku;x1;y1;x2;y2;skóre_detekce;třída;zeměpisná_šířka;zeměpisná_délka
```

Kapitola 4

Vyhodnocení

V této kapitole se budu věnovat vyhodnocení výstupů detektoru dopravních značek, jehož implementace byla popsána v kapitole 3, a jehož návrh přímo vycházel z informacích poskytnutých v kapitole 2.

Nejprve popíšu metriku F-Measure, kterou jsem využil při porovnání výsledků s referenčními hodnotami a následně popíšu výstup tohoto srovnání. V dalším bodě zmíním detekci provedenou nad videosoubory. Nakonec provedu celkové zhodnocení všech těchto výstupů.

4.1 F-Measure

Metrika F-Measure (někdy nazývána F-Score), je statistický test správnosti, který určuje, s jakou přesností byly získány informace. V kontextu počítačového vidění jej lze využít pro měření přesnosti detekce objektů v obraze. Rovnice pro výpočet metriky jsou následující:

$$precision = \frac{D \cap C}{D} \tag{4.1}$$

$$recall = \frac{D \cap C}{C} \tag{4.2}$$

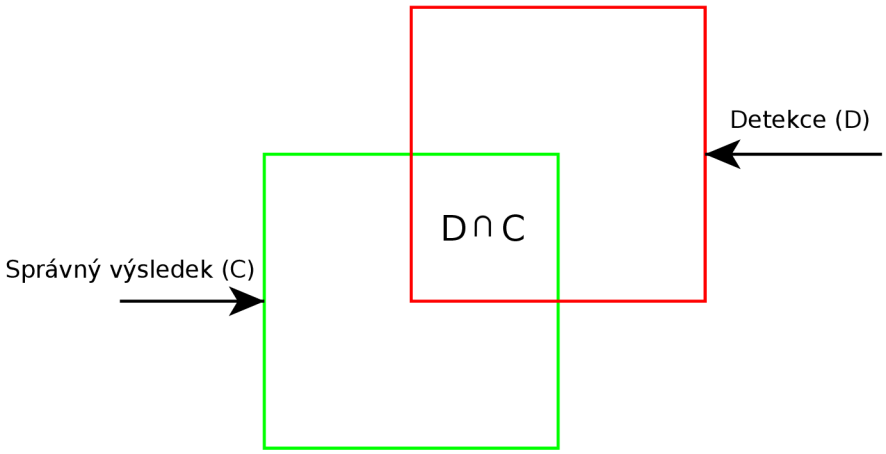
$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.3}$$

Kde D značí detekovaný čtyřúhelník a C čtyřúhelník z referenční sady — tj. správný výsledek. *precision* určuje jaké procento označeného objektu je relevantní, zatímco *recall* určuje jaké procento hledaného objektu bylo označeno. $F1$ je pak skóre metriky F-Measure. Výpočet je ilustrován obrázkem 4.1.

Pokud tedy existuje referenční hodnota obsahující rozměry a pozici čtyřúhelníku ohraničujícího hledaný objekt, je možné srovnat detekovaný čtyřúhelník s referenčním a zjistit jak přesná detekce byla.

4.2 Porovnání výsledků s referenčními hodnotami

Pomocí skriptu zmiňovaného v 3.4.2 jsem tedy provedl srovnání výsledků mého detektoru s výsledky referenčními. Z datové sady [3] jsem použil všech 900 snímků a předal je (spolu s převedeným *SVM modelem* z [4]) detektoru implementovanému v mé práci.



Obrázek 4.1: Ilustrace výpočtu metriky F-Measure.

Vyhodnocení bylo provedeno na notebooku Sony Vaio VPCF2 s procesorem Intel Core i7-2670QM (@ 2.20GHz) a pamětí 6GB. Všechny snímky měly velikost 1360x800 pixelů. Naměřené hodnty jsou znázorněny v tabulce 4.1.

Soubory	**Značky**	**True positive**	**False positive**	**False negative**	**Čas**	**FPS**
900	1213	680	8035	533	1369s	0.656

Tabulka 4.1: Výsledek detekce značek na snímcích referenční datové sady

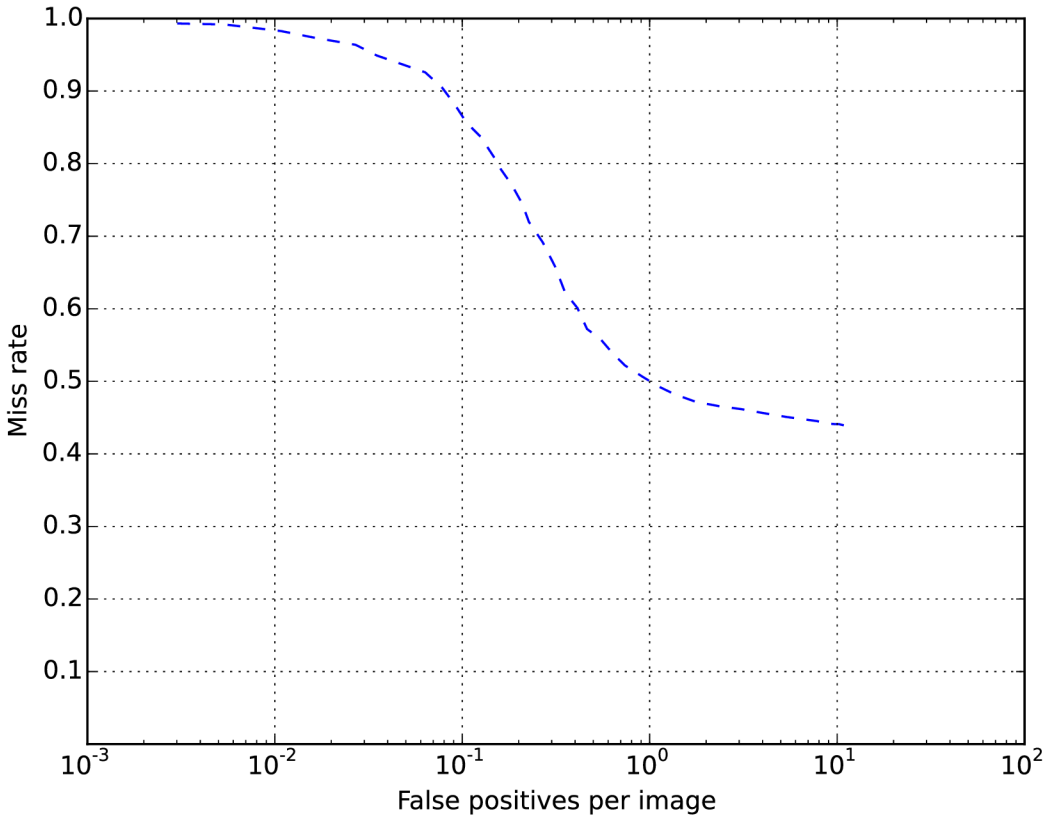
Kde *Soubory* značí celkový počet souborů předaných detektoru, *Značky* celkový počet značek nacházejících se na snímcích, sloupec *True positive* značí počet správně označených oblastí, *False positive* značí počet chybně označených oblastí, sloupec *False negative* pak ukazuje počet chybně neoznačených oblastí, celkový čas potřebný pro detekci je uveden ve sloupci *Čas* a nakonec *FPS* značí počet snímků zpracovaných za vteřinu. Aby byla oblast označena za nalezenou, muselo být skóre metriky F-Measure alespoň 0.5.

Výsledná křivka ukazující chování detektoru nad touto datovou sadou je k dispozici na obrázku 4.2.

4.3 Detekce značek ve videosouborech

Videosoubory přiložené k datové sadě byly posbírány v okolí Klášterce nad Ohří a přilehlých obcí. Jak je vidět v tabulce 4.2, rychlost zpracování snímků (sloupec *FPS*) je znatelně vyšší, než u snímků z předchozí kapitoly. Tento fakt přičítám zejména nižší kvalitě jednotlivých snímků.

Jak bylo zmíněno v kapitole 3.2.1 implementoval jsem jednoduchou metodu trackingu pro detekci ve videosouborech. Vybral jsem tedy dva videosoubory a porovnal výsledky detekce s vypnutým a zapnutým trackingem. Výsledky jsou k vidění v tabulce 4.3.



Obrázek 4.2: ROC křivka pro referenční datovou sadu

Název souboru	**Délka videa**	**Rozměry**	**FPS**
20150512_145429.mp4	88s @ 30fps	720x480 px	9.7
20150512_143819.mp4	72s @ 30fps	720x480 px	8.8
20150512_142952.mp4	72s @ 30fps	720x480 px	9.2
20150512_143113.mp4	51s @ 30fps	720x480 px	9.8
20150512_143946.mp4	34s @ 30fps	720x480 px	8.7
20150512_145632.mp4	32s @ 30fps	720x480 px	10.4
20150512_142459.mp4	31s @ 30fps	720x480 px	8.7
20150512_144043.mp4	30s @ 30fps	720x480 px	9.3
Průměr	51.2s @ 30fps	720x480 px	9.325

Tabulka 4.2: Statistické údaje a výsledky detekce některých videí z datové sady

Soubor	Označené oblastí	Označené oblasti / snímek
20150512_142952.mp4	5071	2.346
20150512_142952.mp4 (tracking)	4614	2.135
20150512_145429.mp4	10343	3.914
20150512_145429.mp4 (tracking)	8095	3.371

Tabulka 4.3: Porovnání výsledků detekce ve videosouboru s a bez trackingu

Jak je vidět, sledování nalezených objektů v po sobě jdoucích snímcích značné zlepšení nepřineslo a detekce vykazuje stále velké množství označených oblastí, z nichž je valná většina chybná. S největší pravděpodobností je tento fakt zapříčiněn skutečností, že téměř všechny chybně označené oblasti přetrvávají i v po sobě jdoucích snímcích — nejedná se pouze o náhodný šum. V tomto případě by bylo vhodné vytvořit nový model strojového učení.

4.4 Vyhodnocení detekce

V tabulce 4.1 je vidět, že doba zpracování jednoho snímku je velice vysoká. Tento fakt je jednak zapříčiněn poměrně vysokou kvalitou snímků datové sady — dá se předpokládat, že při využití kamery ve vozidle budou snímky řádově menší. Dalším důvodem je s největší pravděpodobností využití posuvného okna. V tomto případě je nutné ohodnotit obrovské množství (v tomto případě až 100 000) různých poloh takového okna. Počet pozic posuvného okna potřebných k vyhodnocení by bylo vhodné snížit například využitím dalšího algoritmu pro detekci zájmových oblastí potenciálně obsahujících značku.

Další zrychlení by mohlo přinést zmenšení zpracovávaného snímku nebo vylepšení snímku, které by alespoň z části snížilo vliv okolí (nadměrné osvětlení, ...) na výslednou detekci. Na základě pozorování jsem objevil značnou nelinearitu v době zpracování jednotlivých snímků — snímky obsahující velké množství jasně osvětlených ploch (nebe) — byly zpracovávány znatelně déle, než snímky ostatní. Tyto plochy také většinou obsahovaly největší množství chybně označených oblastí.

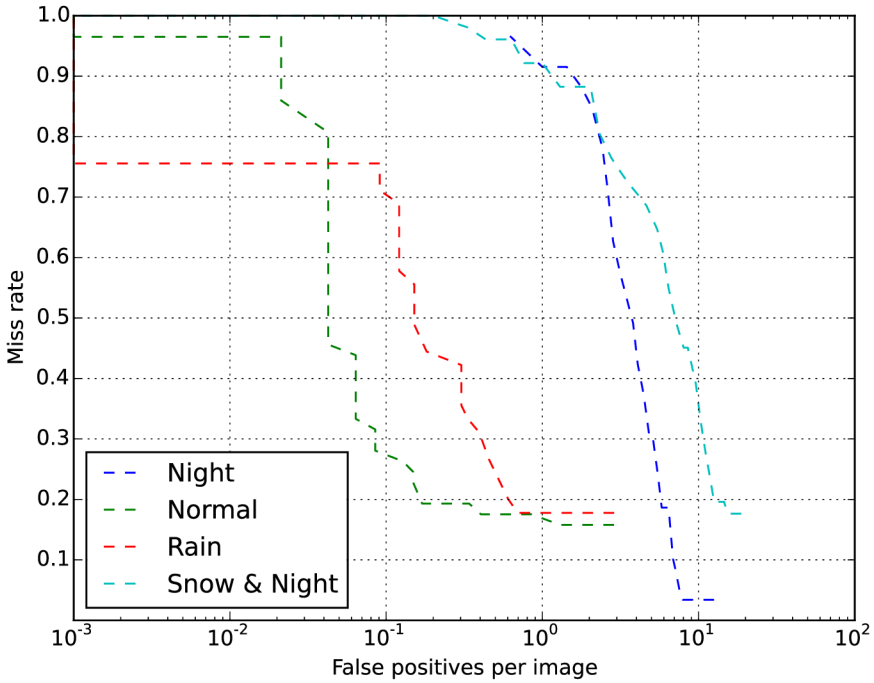
Jak jsem popisoval, rozhodnutí, zda jsem správně nalezl dopravní značku, záviselo na skóre metriky F-Measure. Vypozoroval jsem, že značné množství značek bylo mým detektorem ve snímku skutečně označeno, ale čtyřúhelník ohraničující tuto značku byl větší, než samotná značka. Jelikož ohraničující čtyřúhelníky referenční datové sady těsně přiléhaly k hledané dopravní značce, často bylo výsledné skóre metriky porovnávající tyto dva čtyřúhelníky nižší, než očekávané.

Ve své práci jsem nevyužíval žádného klasifikátoru, který by všechny detekované oblasti zařazoval do tříd podle kategorie dopravní značky. Pokud bych takovou techniku využil, výsledná detekce by pravděpodobně vykazovala nižší až nulové množství chybně označených objektů a zároveň by byla část značek označených větším čtyřúhelníkem správně zařazena do některé z kategorií. To vše samozřejmě na úkor dalšího snížení rychlosti detekce.

Dalším faktorem, ovlivňující výsledky celé detekce, je použitý model strojového učení. Tento model byl předán „jak je“ — je tedy možné, že rozsáhlejším trénováním nad jinou datovou sadou mohlo být dosaženo jiných výsledků.

Zaměřil jsem se také na vyhodnocení výkonu detektoru za různého počasí a různých

světelných podmínek. Výsledky jsou k vidění na obrázku 4.3 a také v tabulce 4.4. Snímky využité v tomto vyhodnocení pochází jednak z mnou vytvořených videosouborů a také z datové sady vytvořené Doc. Ing. Adamem Heroutem, Ph.D, která je k dispozici na jednom ze školních serverů. Převzaté snímky byly využity pro vyhodnocení detektoru ve sněhových podmínkách a v noci.



Obrázek 4.3: ROC křivka srovnávající různé podmínky

Třída	**Značky**	**True positive**	**False positive**	**False negative**
Night	59	57	465	2
Normal	57	48	135	9
Rain	45	37	95	8
Snow & Rain	51	42	563	9

Tabulka 4.4: Výsledek detekce značek v různých podmínkách

Přesnost detekce ve videosouborech lze ohodnotit vizuálně po shlédnutí přiložených videí. Ukázka detekce značek ve videu je na obrázku 4.4. Zrychlení detekce při zpracování videosouborů by spočívala například v nezpracování některých snímků. Za předpokladu, že je běžná kamera schopna nahrávat více než 40 snímků za vteřinu, je teoreticky možné zpracovat pouze každý druhý, třetí, ..., snímek (v závislosti na rychlosti jedoucího vozidla). Souřadnice čtyřúhelníků s potenciálně nalezenou značkou by pak byly přeneseny do následujících nezpracovaných snímku.

Obrázek 4.4: Ukázka zpracování snímků z videa

Kapitola 5

Závěr

Cílem bakalářské práce bylo vytvořit a implementovat systém pro detekci dopravních značek (především z kamery ve vozidle). Cíl se s drobnými výhradami podařilo splnit. Výhradou je myšlena hlavně rychlost zpracování jednotlivých snímků. Zpracování obrazu z kamery ve vozdile by mělo probíhat v reálném čase, což znamená zpracování alespoň 30 snímků za vteřinu (nebo více, v závislosti na rychlosti, s jakou kamera ve vozidle snímá). Nastudoval jsem také několik odborných článků a publikací, na základě kterých jsem vytvořil výsledný systém. Možná rozšíření tohoto systému jsou zmíněna v kapitole 4.4.

Systém je schopen hledat a označovat značky ve snímcích nebo ve videu. Vedlejší částí projektu je výpočet a extrakce deskriptorů z poskytnutých výřezů, které jsou předány [4] k trénování *SVM modelu*. Implementoval jsem také jednoduchý *tracking* nalezených objektů ve videosouboru za účelem snížení počtu falešných detekcí. Experimentoval jsem s parametry pro detekci značek v obraze za účelem nalezení nejlepšího poměru mezi kvalitou detekce a rychlostí zpracování. Pro potřeby vyhodnocení detekce jsem také vytvořil skoro dvacítku krátkých videosouborů, na kterých je zachycen provoz vozidla na pozemních komunikacích z pohledu spolujezdce. Zároveň jsem vytvořil čtyři anotované datové sady obsahující snímky z kamery ve vozidle za různých podmínek.

Literatura

[1] Benenson, R.; Omran, M.; Hosang, J.; aj.: Ten years of pedestrian detection, what have we learned? In *Computer Vision for Road Scene Understanding and Autonomous Driving (CVRSUAD, ECCV workshop)*, September 2014.

[2] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In *In CVPR*, 2005, s. 886–893.

[3] Houben, S.; Stallkamp, J.; Salmen, J.; aj.: Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 1288, 2013.

[4] Jurča, M.: *Detekce dopravních značek*. Bakalářská práce, FIT VUT v Brně, 2015.

[5] Larsson, F.; Felsberg, M.; Forssen, P.-E.: Correlating Fourier descriptors of local patches for road sign recognition. *IET Computer Vision*, ročník 5, č. 4, 2011: s. 244–254.

[6] Liang, M.; Yuan, M.; Hu, X.; aj.: Traffic sign detection by ROI extraction and histogram features-based recognition. In *IJCNN*, IEEE, 2013, s. 1–8.

[7] Mogelmose, A.; Trivedi, M. M.; Moeslund, T. B.: Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey. In *IEEE Transactions on Intelligent Transportation Systems*, 2012.

[8] Stallkamp, J.; Schlipsing, M.; Salmen, J.; aj.: The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, 2011, s. 1453–1460.

[9] Viola, P.; Jones, M.: Robust real-time face detection. *International Journal of Computer Vision*, ročník 57, 2004: s. 137–154.

[10] Wang, G.; Ren, G.; Wu, Z.; aj.: A robust, coarse-to-fine traffic sign detection method. In *IJCNN*, IEEE, 2013, s. 1–5.

Příloha A

Obsah DVD

K mé práci je přiloženo jedno DVD, které obsahuje tyto složky:

- **Dataset** - složka obsahující videa z kamery ve vozidle, snímky Německé datové sady [3] a anotované snímky v různých podmínkách — vše využité při testování detekce
- **Poster** - složka obsahující demonstrační plakát mé bakalářské práce
- **Report** - složka obsahující zdrojový text technické zprávy a také její verzi ve formátu *pdf*
- **SourceCodes** - složka obsahující zdrojové kódy implementovaného programu, *SVM model* vytvořený v [4], podpůrné skripty, knihovny potřebné pro přeložení programu a také návod k překladu
- **Video** - složka obsahující demonstrační video mé bakalářské práce

Příloha B

Plakát