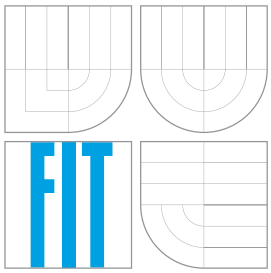
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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
AUTHOR

MICHAL JURČA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2016

Abstrakt

V rámci této práce byl navržen a implementován systém pro detekci dopravních značek z videa. Tento systém umožnuje detekovat a klasfikovat dopravní značky. Navržený systém pracuje s dobrou přestností detekce a klasifikace. Aplikace je schopná pracovat v reálném čase.

Abstract

A system for detection and classification road sign was designed and implemented during work on the thesis. The system is able to detect and classify road sign. The accuracy of detection and classification is good. The system runs in real time.

Klíčová slova

dopravní značky, klasifikace, detekce, support vector machines, kaskádový klasifikátor, LIBSVM

Keywords

road sign, classification, detection, support vector machines, cascade classification, LIBSVM

Citace

JURČA, Michal. *Detekce dopravních značek z kamery ve vozidle*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Beran Vítězslav.

Detekce dopravních značek z kamery ve vozidle

Prohlášení

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

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Jurča
18. května 2016

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Vítězslavu Beranovi, Ph.D, za jeho vedení, trpělivost, konzultace a pomoc při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Už od 20.století s rozmachem automobilového průmyslu rostla potřeba detekce dopravních značek. Nyní s rozvojem počítačů jsou do automobilů implementovány vestavěné a počítačové systémy, které umožnují detekovat a klasifikovat dopravní značky. Systém může tak detokovat hrozící nebezpečí. S rozvojem autonomních vozidel bude potřeba detekce dopravních značek stále požadována.

Dnešní počítačové systémy ADAS (Advanced Driver Assistance Systems) v automobilu hlídají skoro všechno, jestli je řidič připoutaný, neusíná a dává pozornost. S rozvojem techniky dnešní systémy umožnují noční vidění, sledování chodců a dopravních značení. A posledním bodem se bude ve své práci zabývat. Detekcí a klasifikací dopravních značek.

Ve druhé kapitole této práce bude popsána historie dopravního značení od samotného začátku až po současnost. Následně budou představeny existující řešení, použité datové sady a klasifikátory. Ve třetí kapitole budou rozebrány problémy týkající se řešení detekce a klasifikace. Následně bude popsán návrh řešení. Další kapitola bude věnována implementaci aplikace a pomocných scriptů. Další kapitola bude věnována představením charakteristik pro vyhodnocení natrénovaného modelu. A následně pomocí charakteristik budou diskutovány dosažené výsledky. Závěr kapitoly bude věnován experimentům. V poslední kapitole provedu zhodnocení řešení a představím návrhy na vylepšení.

Kapitola 2

Detekce dopravních značek

V této kapitole budou popsány dopravní značky, použité datové sady, existující řešení.

2.1 Dopravní značení

V dávných dobách funkci dopravního značení plnily pouze jen stopy, vyšlapané pěšiny a vyjeté koleje od vozů nebo koní. Již v Pompejích v době před 2 tisíci lety se objevily patníky, které oddělovaly peší zónu od ulice. Objevil se zde i přechod pro chodce v podobě vyvýšeného chodníku oproti silnici. Později Římané začali značit cesty pomocí svislých kamenů vyznačující vzdálenost od Říma. Ve středověku se pokračovalo ve značení směrů a vzdáleností. V roce 1868 byl použit první primitivní mechanický semafor. Potřeba dopravních značek vzrosla s rozmachem automobilního průmyslu ve 20.století. Od roku 1924 se koná každé čtyři roky silniční kongres, ale i přes veškeré snahy se dodnes nepovedlo všechny dopravní značky sjednotit.

V Československu v roce 1935 bylo zavedeno dopravní značení z pěti druhů výstražných značek. O pár let později přibyly další dopravní značky a byl zaveden pravostranný silniční provoz. Rychlost jízdy v obci nesměl být vyšší než rychlost koně v poklusu. Jelikož v té době, reflexní folie ještě nebyly, připevňovaly se ně odrazky pro dostatečnou viditelnost v noci. V současnosti se prosazují elektronické a promněné dopravní značení, které se využívají při regulaci dopravy a zajištění plynulého a bezpečného provozu. V reálném čase poskytují řidiči informaci o výskytu dopravní nehody, zhoršeném počasí na konkrétních úsecích nebo provádění údržbových prací. Dále se prosazuje osvětlení značek s časovým ovládáním, které se využívá zejména u škol a používá se reflexní žlutozelené orámování.

Dopravní značky jsou jednoduché piktogramy určené pro řízení a regulaci silničního provozu na pozemních komunikacích. Jedná se o zařízení, které upozorňuje řidiče na nebezpečná místa, ukládá jim příkazy, zákazy nebo omezení, poskytuje jim zpřesnění nebo informace, doplňuje nebo omezuje význam jiné dopravní značky. Rozlišují se dopravní značky svislé nebo vodorovné.

Podle zákona o provozu na pozemích komunikacích §63 odst. 1 svislé dopravní značky jsou:

- výstražné značky, které upozorňují na místa, kde účastníku provozu na pozemních komunikacích hrozí nebezpečí a kde musí dbát zvýšené opatrnosti
- značky upravující přednost, které stanoví přednost v jízdě v provozu na pozemních komunikacích

- zákazové značky, které ukládají účastníku provozu na pozemních komunikacích zákazy nebo omezení
- příkazové značky, které ukládají účastníku provozu na pozemních komunikacích příkazy
- informativní značky, které poskytují účastníku provozu nutné informace, slouží k jeho orientaci nebo mu ukládají povinnosti stanovené tímto zákonem nebo zvláštním právním předpisem
- dodatkové tabulky, které zpřesňují, doplňují nebo omezují význam dopravní značky, pod kterou jsou umístěny.

V úloze detekce dopravních značek, se používá kromě dělení podle typu taky dělení podle tvaru, který může být určen čtvercem, kruhem, trojúhelnékem, osmiúhelníkem a obdelníkem. Tvar a velikost jsou pevně definovány a nesmí se měnit.

Kompletní přehled všech dopravních značek a více inforací o dopravním značení lze nalést na stránkách Ministerstva dopravy[1].

2.2 Detekce dopravních značek

Detekcí a klasifikací dopravních značek se zabývají vědecké skupiny a stále vytvářejí nové postupy, pro přesnější a rychlejší detekování a klasifikování značek. Dostupné řešení může být použito v autonomním vozidlech. Detekce značek by se dala rozdělit do 4 skupin - barevná segmentace, detekce tvarů, extrakce příznaků a strojové učení.

Barevná segmentace

Při barevné segmentaci se využívá, že mají značky zákonem definovanou barvu. A dochází tedy k výběru oblastí s barvou určené dopravními značkami. Nejčastěji červená, žlutá a modrá.

Toho využívají autoři práce [7], kteří ke segmentaci používají barevný model HSV. Vytváří se dvě LUT tabulky na základě získaných barených složek z modelu a to barevný tón a sytost. Po vytvoření tabulek jsou snímky normalizovány na maximální hodnotu 255. V následujícím kroku jsou využívány genetické algoritmy pro vyhledání a určení pozice dopravní značky v obraze. Jako klasifikátor použili neuronové sítě se dvěmi vrstvami. Řešení bylo testováno na stroji s AMD Duron o 1 GHz s úspěšností 70%-90%.

Dalším řešením, které využívá segmentaci je práce od Hassan Shojania [17], která se skládá ze čtyř hlavních fází - barevná segmentace, detekce hran, detekce tvaru a klasifikace. Segmentace je založena na převedení vstupního obrázku do binárního obrázku. Autor sestrojil řadu filtračních masek k nalezení rohů v obraze. Následně je snímek dál zpracováván a testován jestli splňuje tvar dopravní značky. Klasifikátorem jsou použity neuronové sítě. Metoda je úspěšná pro detekci kruhových červených značek, čtverců, trojúhelníků a osmiúhelníků. Ale má jistá omezení, snímky se nesmí otáčet a je povolen jen menší náklon zhledem k pozici kamery. V podstatě stejný náklon jako to, co řidič vidí normálně z vozidla.

[1] http://www.ibesip.cz/

Detekce tvarů

Metody založené na detekci tvarů jsou více stabilní a nejsou ovliňovány podnebními podmínkami.

Toho využívají autoři práce [15]. Práce detekuje a rozpoznává evropské a americké výstražní značky. Model je založený na rozpoznávání kruhu nebo trojúhelníku. Pro rozpoznávání potencionální oblasti umístění značky používá šedotonový obraz a Houghovy transformace. Klasifikátorem jsou použity neuronove sítě a na rozpoznávání číslic se používá ODR. Datová sada byla manuálně vytvořena z dopravních značek Francie a Ameriky.

Úspěšnost při detekování dosahovala přibližně 90%. Program zpracovával přibližně 20 snímků/s.

Strojové učení

Metody založené na strojovém učení dosahují lepších a stabilnějčích výsledků. Řešení je možné používat v aplikacích zprácovávající obraz v reálném čase. Algoritmy strojového učení, které používám ve své práci budou popsány v kapitole 2.4.

Mathias [13] aplikoval Integral Channel Features detector. Detektor je založený na histogramu orientovaných gradientů (HOG) s rozhodovacími stromy. Vycházejí ze svého výzkumu [1] detekce osob, kde z analyzoval víc jak 40 detektorů použitých během posledních 10 let a na základě nasbíraných zkušeností vytvořili vlastní detektor, který dosahoval lepších výsledků než ostatní. A patří nyní k nejlepším řešení na vědecké úrovňi.

K natrénování detektoru autoři použili Quad-Core AMD Opteron 8360 SE s 64GBytes RAM. Vstupní snímky byly zmenšeny na 28x28 pixelů a převedeny na šedotonový obraz. Jako datové sady byly použity Německé dopravní značky [11] a Belgické dopravní značky [19], které dohromady obsahují více jak pěttisíc dopravních značek. Detekovaní značek trvalo 45 minut a byl použit Intel Core i7 870 o 2.5 MHz společně s Nvidia GeForce GTX 470 GPU.

Datovou sadu rozdělil do tří skupin podle typu dopravních značek. Příkazové, zákazové a výstražné s výsledky vyhodnocení 96.98% , 100% a 100%. Metoda dosahuje nejlepších výsledků. Ale ve své práci neuvažuje vliv nepříznivých povětrnostních podmínek např. sníh, déšť, mlha, noc.

Autoři této práce [16] trénují a rozpoznávají Japonské dopravní značky. Využívají barevný model HSV s využitím klasifikátoru AdaBoost a integralní obraz . K natrénování metody autoři použili 4125 značek , k detekci použili 9036 značek. Vstupní obrázky byly zmenšeny na rozměr 60 x 60 pixelů.

K rozpoznání dopravních značek bylo použito 210 značek, které autoři pořídili sami při rychlosti 40 až 70 km/h s rozlišením videa 640 x 480. Úspěšnost rozpoznání dosahovala v rozmězí 80%-97%.

2.3 Obrazové příznaky

Obrazové příznaky jsou velmi důležitou součástí detekce objektů. Ovlivňují přesnost a také rychlost detekování. Účelem extrakce příznaků je získání informace z obrazu, které jsou pak vstupem klasifikátoru.

Haarovy příznaky

Jedním typem nejačastěji používaných příznaků jsou Haarovy příznaky též znány pod názvém Haarovy vlnky, které jsou založeny na rozdílu jasu mezi obdélníkovými oblastmi. Byly použity v práci Viala & Jones [20].

Hodnota příznaku je získána z obrazu s vypočítáním suma pixelů obrazu odpovídající bílé části. A suma pixelů v černé části. Tyto hodnoty jsou pak odsebe odečteny. Ten to postup je značně časově náročný a proto se používá integrální obraz 2.3.

Práce Viala & Jones byla navrhnuta za účelem detekce obličejů v reálném čase za použití stojového učení AdaBoost.

Integrální obraz

Integrální obraz (summared area table) se používá jako rychlý a efektivní způsob výpočtu součtu hodnot v obdelníkové oblasti v daném obrazu. Integralní obraz se používá při posledním kroku extrakce příznaků, protože je extrakce často časově náročná kvůli častým přístupům do paměti. Tato metoda byla použita v práci [20] pro rychlou extrakci příznaků pro detekci obličejů.

Výpočet integrálního obrazu pro pixel o souřadnicích (x,y) se provede pomocí vzorečku 2.1. Jedná se tedy o sumu hodnot jednotlivých pixelů v obdélníku, který je podmnožinou původního obrázku.

$$I(x,y) = \sum_{x' \leq x, y' \leq' y} i(x^{'}, y^{'}) \tag{2.1}$$

Integrální obraz lze spočítat jediným průchodem původního obrazu v konstantním čase. Složitost O(1). Výpočet hodnoty sumy zabere vždy pouze 4 přístupy do paměti. Například pro výpočet hodnoty sumy na obrazku stačí znát oblasti určené body A, B, C, D odpovídajícího integrálního obrazu. Pro výpočet stačí tři aritmetické operace sčítání a odčítání:

$$i(x^{'}, y^{'}) = s(A) + s(D) - s(B) - s(C) = 5 + 16 - 7 - 8 = 6 \tag{2.2}$$

Histogramu orientovaných gradientů

Dalším typem používaných příznaků je histogramu orientovaných gradientů, který se používá zejména při detekovaní dopravních značek s použitím SVM klasifikátoru. Myšlenkou je, že hledaný objekt v obraze, lze pomocí tvaru a vzhledu popsat intenzitou gradientu. A lze použít pro rozpoznávání objektů a klasifikaci. V případě kdy objekty nemění své natočení v obraze. Histogram gradientů lze počítat i z černobílých obrazů, ale tím se získá jenom jedna hodnota. A nebo lze počítat i pro barevný obraz, kde se počítá pro každou barevnou složku zvlášť. Obraz se rozdělí na malé oblasti - buňky a pro každou buňku je vypočítaný jednorozměrný histogram, který se počítá pro všechny pixely z buňky. Obraz před výpočty je vhodné znormalizovat. Shomažďováním informací z buňky, ale i z okolí vzniká tzv. blok.

Celý obraz je tedy rozdělen do bloků o stejné velikosti a vzniklé buňky jsou nositely hodnot. Při výpočtu histogramu s využitím kruhové nebo pravoúhlé obblasti je nutné rozdělit vypočítaný úhel na několik rozsahů. Velikostí rozsahů se může dodcílit a přijetí malého natočení klasifikovaného objektu.

LBP

Dalším typem používaných příznaků jsou LBP(Local Binary Pattern), které jsou jednoduché a efektivní. Označí se jednotlivé pixely obrazu a pomocí prahovaní se získá výsledek v binárním tvaru. Díké své jednoduché výpočetní složitosti jsou často používany v aplikacích. které analyzují snímky v reálném čase.

Zpracovávaný obrázek musí být převeden do stupní šedi a následně se začně procházet pixel po pixelu. Originální LBP operátor (Ojala et al. 1996) prováděl prahování nad 3x3 okolí a bod ve středu sloužil jako prahovací hodnota a výsledkem bylo binární číslo. LBP operátor byl později změnen a umožnoval používat rozdílnou velikost okolí.

2.4 Klasifikátory

Úkolem klasifikátoru je přiřadit daný objekt k odpovídající třídě. Protože nelze získat často dokonalý klasifikátor a tak obecnějším úkolem je určení pravděpodobnosti pro každou třídu. Výstupem klasifikátoru v úloze detekce dopravních značek je tedy informace udávající jestli vstupní výřez obrazu je dopravní značka či okolí. Nebo může představovat konkrétní typ dopravní značky.

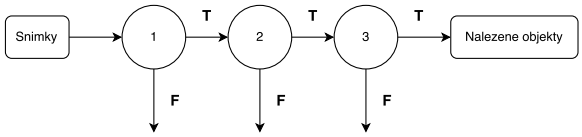
Boosting

Metoda je založena na principu kombinování slabých klasifikátorů s horší úšpěšností do jednoho výsledného kvalitního klasifikátoru. Slabý klasifikátor může být reprezentován rozhodovacím stromem, perceptronem a další. Omezujícím kritériem výběru je aby byla chybovost klasifikátoru menší než 0,5. Výsledný klasifikátor $H(x)$ 2.3 lze matematicky zapsat jako lineární kombinací slabých klasifikátorů $h_t(x)$. Výsledný klasifikátor není lineární.

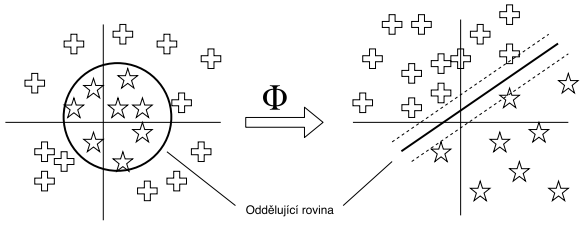
$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_i())$$

V součastnosti nejpoužívanější variantou metody boosting je AdaBoost (**ada**ptive **boost**ing). Metoda umožnuje při trénování přidávat slabé žáky tak dlouho, dokud není splněna optimální hodnota klasifikační chyby. Každý trénovací vzorek dostane přidělenou váhu, která určí pravděpodobnost zařazení do trénovací množiny pro jednotlivé klasifikátory. Je-li vzorek klasifikován přesně, jeho opětovné zařazení do klasifikace klesá a v opačném případě roste. AdaBoost se zaměřuje na obtížné vzorky. Jeho funkce je definována následovně:

- Při inicializaci dostanou všechny vzorky stejné hodnoty vah. V každém iteračním kroku k se natrénuje klasifikátor C_k
- V následujícím kroku je pro klasifikátor C_k proveden výpočet váhy klasifikátoru α_k. Hodota váhy je závislá na trénovací chybě klasifikátoru.
- Pro chybně klasifikované vzorky se hodnota váhy zvýší a při správné klasifikaci se hodnota sníží. Z_k je normalizační konstanta.



Obrázek 2.1: Kaskáda klasifikátorů



Obrázek 2.2: SVM- transformace prostoru. Zdroj [6].

Kaskáda klasifikátorů

Myšlenka kaskádních klasifikátorů je mít sekvenčně seřezené slabé klasifikátory do jedinného silného klasifikátoru. Např. práce Viola & Jones [20] na detekci obličejů je založena na skenování výřezů pomocí posuvného okna. Tím bude k detekci velká část vzorků, které nebudou patřit do hledaných tříd. Viola & Jones si uvědomili, že rychlost detektoru závisí na rychlosti klasifikace pozadí. Pomocí algoritmu AdaBoost je sestavena soustava slabých klasifikátorů, kterým byla nastavena nejvhodnější váha a výsledkem je pak kaskádový klasifikátor. Velká část výřezů je tak v každé úrovni kaskády zahozena a část výřezů, která je klasifikována kladně, projde do další úrovně zpracování. Výřez, který projde až na konec kaskády je s velkou pravděpodobností hledaný objekt.

Na obrázku 2.1 je zobrazena architektura kaskády. Počáteční klasifikátory vyřadí velkou část negativních vzorků a pozitivní vzorky postupují k dalšímu zpracování. Většina negativních vzorků je vyřazena v počátečních úrovních kaskády a tím může v následujících fázích stoupnout výpočetní složitost klasifikace. Do poslední úrovně kaskády se dostane už jenom málo vzorků a celká klasifikace je velmi rychlá. Výstupem posledního klasifikátoru v kaskádě je pak hledaný objekt.

Support Vector Machines

Support Vector Machines (SVM) se často používá pro klasifikaci kvůli jeho velké přesnosti a schopnosti pracovat s vícedimensionálními daty. Tuto metodu používám ve své práci. SVM patří do kategorie tzv. jádrových algoritmů (kernel machines) [10], které poskytují efektivní algoritmy pro nalezení lineární hranice a zároveň jsou schopny zpracovávat vysoce složité nelineární funkce. Jedním se základních principů je transformace nelineárního vstupního prostoru do jiného, vícedimensionálního, kde již lze od sebe lineárlně oddělit třídy. Tato myšlenka je v podstatě jednoduchá. Na obrázku 2.2 v levo ve dvourozměrném prostoru jsou dvě třídy, oddělené nelineární kružnící, které jsou transformovány do jiného prostoru (future space) pomocí nelineární funkce Φ. Takto vzniký prostor na obrázku 2.2 v pravo je vícedimensionální, který už umožnuje oddělit obě třídy lineární rovinou.

Obrázek 2.3: Datová sada GTSRB

Základní funkcí SVM je nalezení oddělující roviny mezi dvěma třídami dat . K nalezení roviny pomáhají podpůrné vektory (support vectors) , které leží na okraji prázdné oblasti a přímo ovlivnují řešení. Čím budou mít větší vzdálenost od oddělující roviny tím zlepší výsledný klasifikátor. Kdyby se ostatní data vypustila z trénovacího setu, výsledek by se nezměnil.

Více informací o SVM, lze získat v článku [12].

2.5 Datové sady

V této sekci budou popsány datové sady, které byly použity nebo vytvořeny pro testování, trénování detektoru a klasifikátoru. Nejprve bude popsaná datová sada GTSRB, kterou používám ve své práci, jelikož součástí byly vypočítané vektory descriptů HOG. Následně bude popsána datová sada Belgických dopravních značek, ze které jsem si vytvořil vlastní negativní datovou sadu.

Mogelmose [14] provedl srovnání veřejných dostupných datových pro aplikaci ADAS.

Datová sada GTSRB

Datová sada GTSRB (**G**erman **T**raffic **S**ign **D**etection **B**enchmark) [18] je tvořena německými dopravními značkami z různých regionů Německa pořízených z autokamery. Snímky byly pořízeny v různých podnebních podmínkách. Jednotlivé snímky značek jsou realizovány pomocí výřezů o velikosti mezi 15x15 až 250x250 pixelů. Dopravní značka není přesně umístěna ve středu snímku a okolí značky je tvořeno minimálně z 10% z celkového obsahu plochy obrázku. Obrázky jsou uloženy ve formátu *ppm*.

Sada obsahuje celkem 39 209 snímků pro trénování rozdělených do 43 tříd podle typu značky. Snímky lze rozdělit na čtyři skupiny podle typu značky: zákazové, příkazové, výstražné a upravující přednost. Na obrázku 2.3 jsou zobrazeny snímky, které jsou rozmazané, odlišné velikostí a pořízené v různou denní dobu.

K dostupným výřezům jsou k dispozici také vypočítané vektory deskriptorů pro histogram orientovaných gradientů. Snímky byly zmenšeny na velikost 40x40 pixelů. Velikost buňky byla nastavena na 5x5 pixelů, posunutí 5x pixelů v obou směrech a velikost výsledného vektoru descriptoru nastavena na 1568.

Součástí datové sady je anotace uložená ve formátu *csv*, která obsahuje tzn. *groundtruth*. Informace jsou odděleny pomocí ";"a jsou složeny z jména souboru, šířky snímku, výšky snímky, souřadnicemi dopravní značky a číslem zařazení do třídy 0 do 42.

Společně s trénovací datovou sadou je dodána i testovací sada, která obsahuje 12 569 snímků. Testovací sada dodržuje stejné rozvržení jako trénovací sada.

Datová sada BelgiumTSC

Datová sada BelgiumTSC (**B**elgium **T**raffic **S**ign **C**lassification **B**enchmark) [19] obsahuje 4 591 výřezů pro trénování a 2 534 výřezů pro testování. Snímky jsou rozděleny do 62 tříd podle typu značek. Datová sada dodržuje stejnou strukturu jako datová sada GTSRB, akorát neobsahuje vypočítané vektory descriptorů.

Součástí datové sady je i sadu obrázků, které tvoří pozadí respektive, kde není dopravní značka. Sada obsahuje 20 550 snímků, které byly pořízeny pomocí kamery z auta. Snímky zachycují běžný silniční provoz a ulice. Dále jsou obsaženy snímky stromů, lidí, domů, reklamních ploch a další.

Kapitola 3

Návrh

V této kapitole budou popsány problémy při detekci a klasifikaci. Následně bude popsán návrh řešení a návrh validace existujících řešení.

3.1 Analýza problému

Nalezení kvalitního řešení detekce a klasifikace dopravních značek není jednoduché. Řešení je ovlivňováno různými faktory, které budou následně představeny.

Cílem detekce dopravních značek je určení oblasti zájmu s potenciálně hledaným objektem ve snímku z videa. Tento problém lze považovat za segmentaci obrazu, kdy existují dvě třídy: dopravní značka a pozadí. A úkolem je identifikovat hledaný objekt od pozadí zachycené v snímku.

Klasifikace dopravních značek klasifikuje potenciální kandidáty značek do předem stanovených tříd nebo do nadtřídy.

Klasifikace

Při procesu klasifikace existuje mnoho problémů. Prvním problémem je existence velkého počtu tříd dopravních značek např. v Chorvatsku existuje 332 konkrétních tříd značek. A vzniká tak obtížný problém klasifikace to více tříd. Možným řešením problému je zjednodušení počtu tříd, kdy se podobné třídy značek spojí do jedné nadtřídy. Další problém vzniká, aby algoritmy strojového učení fungovaly správně vyžadají velké množství dat v dobré kvalitě už v předpřipravených třídách. Toho lze pro určité třídy težko dosáhnout vzhledem k nerovnoměrné přítomnosti dopravních značek v reálném světě, což má za následek nevyvážené rozložení třídy v datové sadě. Třetím problémem jsou podobné značky ve stejné podmnožině a činí je to tak více obtížeji rozeznatelné. Na obrázku 3.1 je zobrazena podmožina výstražných značek, které jsou charakterizovány stejným tvarem, barvou a podobným piktogramem. Čtvrtým problémem jsou rozdíly dopravních značek mezi jednotlivými státy. Na obrázku 3.2 je zobrazena dopravní značka typu konec hlavní silnice, která je mírně odlišná mezi jednotlivými státy. A mnoho dalších problemů je tvořeno změnou povětrnostních podmínek, různým natočením dopravní značky, zpracovaní v reálném čase, pozorování značky z různé perspektivy. A tvoří tak úkol mnohem težší. Na 3.2 jsou zobrazeny dopravní značky, které jsou různě osvětleny, pootočeny a zakryty překážkou.

Obrázek 3.1: Výstražné dopravní značky s podobným piktogramem ve stejném řádku.

Obrázek 3.2: V horním řádku jsou zobrazeny dopravní značky, které jsou různě osvětleny, pootočeny a zakryty překážkou. V druhém řádku je zobrazena značka konec hlavní silnice v různých státech. Z leva Rakousko, Belgie, Česká republika, Německo a Estonsko.

Detekce

Detekce doravních značek má podobné problémy jako proces klasifikace. Ale má navíc problémy v lokalizační nepřesnosti a nalezení falešně pozitivních objektů. Je-li dopravní značka nalezená je dál předána klasifikační fázi. Je-li lokalizace dopravní značky nepřestnost [3] snižuje se tak přestnost klasifikační metody. Neboť většinou klasifikátor nebývá natrénovaný na datech, které jsou posunuty vlivem nepřesné lokalizace značky. S touto skutečností je třeba počítat a natrénovat tak model pro klasifikaci. Falešně pozitvní objekty mohou být detekovány, když se objekty podobají dopravní značce v barvě, vzhledu nebo tvaru. Cílem je minimalizovat počet falešně pozitivních poplachů, tak že řešení lze použít v asistečních systémech. V ADAS není žádoucí a by se FP často objevovaly a narušovaly tak pozornost řidiče vozidla. A systém by byl tak nepoužitelný. Pro lokalizaci dopravní značky je nutné prohledávat celý snímek pomocí posuvného okna v různém měřítku snímku což je výpočetně náročné.

Dopravní značky jsou jednoduché rigidní objekty, které jsou omezeny tvarem, intenzitou barvy a jsou navrženy tak, aby byly čitelné pro člověka. Detekce dopravních značek je lokalicaze rigidních objektů.

3.2 Návrh řešení

Při návrhu řešení jsem vycházel z myšlenky práce Viola & Jones [20], kde autoři vytvořili kaskádu slabých klasifikátorů. A v každém kroku procesu se snažili, vždy eliminovat nejvíc negativních snímků a až ve výsledku jim zůstanou jenom pozitivní objekty, které se mohou klasifikovat. Na základě toho jsem rozdělil úlohu na dvě části detekci a klasifikaci.

Do návrhu řešení jsem zahrnul představené problémy v kapitole 3.1 zabývající se analýzou problému detekcí a klasifikací dopravních značek.

Detekce

Detektor je založený na kaskádovém klasifikátoru 2.4, který je znázorněn na obrázku 2.1 a jeho účelem je eliminovat negativní vzorky a předat dál nalezené objekty klasifikátoru do více tříd. Pro nalezení dopravní značky je nutné prohledat celý snímek a proto bude použita metoda posuvného okna, která skenuje snímek v různém měřítku. Výstup jsou nalezené oblasti objektu. Na základě zvoleného prahu pomocí experimentování, lze rozhodnout jestli daný objekt je dopravní značka či nikoliv. Tímto krokem lze snížit případné FP objekty.

Pro natrénovaní detektoru bude použita pozitivní a negativní datová sada. Pro vyhodnocení správně natrénovaného modelu bude použita metodu křížová validace, která je popsána v kapitole 3.4 společně s dalšími metodamy, které se často používají. Na základě vyhodnocení a analýzy problému budou FP snímky přidány k negativní datové sadě a znovu bude natrénovaný detektor.

Na základě dostupných řešení jsem zvolil pro impementaci detektoru dopravních značek kaskádový klasifikátor knihovny OpenCV [1], která je představená v kapitole 4.7. Kaskádový klasifikátor je implemetovaný pomocí AdaBoost 2.4.

Podle dostupných existujících řešení, kde se často a úspěšně používal pro detekci WaldBoost, jsem se rozhodl v rámci experimentu **??** nahradit kaskádový klasifikátor WaldBoostem. Výhodou je rychlé trénování a vyhodnocování v reálném čase. Pro implementaci jsem zvolil knihovnu OpenCV.

Klasifikace

Klasifikace je založená na SVM 2.4 a úkolem je nalezené objekty pomocí detektoru klasifikovat do tříd. Nalezené objekty bude pak následně nutné převést na HOG příznaky 2.3 a provedení klasifikace do více tříd. Na základě zvoleného prahu pomocí experimentování, lze rozhodnout jestli daný objekt je dopravní značka či nikoliv. Tímto krokem lze snížit případné FP objekty.

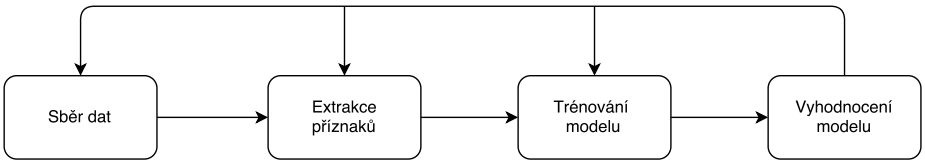
Pro natrénovaní bude použita jenom pozitivní datová sada, která je rozšířená a nepřesně lokalizované dopravní značky pro zvýšení přestnosti klasifikace. Pro správné vyhodnocení natrénovaného modelu bude použita metoda křížové validace.

Na základě výsledků v existujících [13] řešení jsem rozhodl použít knihovnu LIBSVM, která je více popsaná v kapitole 4.7.

Návrh validace existujicích řešení

Základní podmínkou při validaci existujích řešení bude veřejná dostupnost použité datové sady. Jestli je dostatečně velká, kvalitní a anotovaná či nikoliv. Dalším důležitým bodem je vyhodnocení natrénovaných modelů. Je vhodné použít běžných metod pro vyhodnocení detektoru např. ROC křivku, která je popsaná v kapitole 5.1 nebo klasifikaci pomocí matice záměn, která je popsaná v kapitole 5.1. Dalším bodem je uvedení konktétních parametrů použitých při trénování a extrakci příznaků. V případě řešení zpracovávající video v reálném čase uvedení konfigurace zařízení při testování.

[1] http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html



Obrázek 3.3: Blokové schéma procedury trénování a vyhodnocení

3.3 Procedura trénování a testování

Postup trénování a testování je znázorněn na obrázku 3.3. Řešení se skládá ze čtyř bloků - sběr dat, extrakce příznaků, trénování modelu a vyhodnocení. Na základě vyhodnocení natrénovaného modelu bude zdřejmé jak je model kvalitní či nikoliv. V případě neuspokojivých výsledků je vhodné se v procesu vrátit, na základě vyhodnocení upravit určitý krok. A následně zopakovat celý postup až po vyhodnocení. Z návrhu je patrné, že nalezení kvalitního modelu je časově náročné.

Prvním krokem pro detekci objektů v obraze je sběr dat. Příprava dat je časově nejnáročnější a nejobtížnější krok celého procesu. Současně je to ale krok, který má klíčový význam pro úspěch výsledné aplikace. Je vhodné mít dostatečně velkou datovou sadu pozitivních ale i negativních objektů. Bude-li datová sada tvořena např. z 95% pozitivních objektů a zbylých 5% negativních objektů. Při vyhodnocení natrénovaného modelu bude mít natrénovaný model velmi dobré výsledky pro hledané objekty. Ale v případě negativních dat může klasifikátor nagativní data označit jako FP a výrazně se to neprojeví pří vyhodnocení. A při tak malé negativní datové sadě, se ve výsledku natrénovaný model může chybně považovat za kvalitní. K zabránění tomuto nežádoucímu stavu se snaží předejít pomocí různých metod vyhodnocení modelu, které jsou popsány v kapitole 5.1.

Extrakce příznaků je důležitým krokem a dá se sní ovlivnit rychlost a přestnost detekce a klasifikace. Velikost descriptoru příznaku má určitou velikost, která ovlivňuje rychlost zpracování. Není vhodné mít velký descriptor příznaku, který obsahuje hodně informací pro zpracování, ale také není v hodné co nejmenší, při kterém se zrychluje rychlost zpracování, ale klesá přestnost.

Dalším krokem je vytvoření modelu pro trénovaní. Pro správné vyhodnocení se používají určité postupy, které jsou popsány v následující kapitole 3.4.

Posledním krokem je vyhodnocení natrénovaného modelu. Na základě vyhodnocení lze vyhodnotit jestli natrénovaný model je kvalitní či nikoliv. V případě neuspokojivých výsledků je vhodné rozšířit datovou sadu. V případě detektoru rozšířit negativní datovou sadu o FP snímky a při klasifikaci rozšířit sadu a nepřesně lokalizované dopravní značky.

3.4 Vytvoření modelu

Při trénování modelu se většinou používá metoda učení s učitelem, kterou využívám ve své práci. Výchází se tedy z toho, že se učícímu algoritmu předají trénovací data, která jsou zařazena do odpovídajících tříd. Na testovacích datech se pak porovnává úspěšnost klasifikovaných hodnot z trénovacích dat. Metod testování je více, záleží jaká data se použijí pro testování a jaká pro trénování:

- testování v celých trénovacích datech
- křížová validace (cross-validation)

- bootstrap
- leave-one-out
- testování na testovacích datech.

Testování na datech, které byly použity pro učení klasifikátoru mají nejmenší vypovídající schopnost o tom, jak budou hodnotné znalosti získané z klasifikování nových zkoumaných dat. Při trénování na datech může často dojít k přeučení (overfitting) a může se tento stav snadno přehlédnout.

Testovací metoda křížová validace zamezuje překrývání testovacích dat. Data se rozdělí na K částí, kde část $\frac{1}{K}$ se použije pro testování a zbylé data $1-\frac{1}{K}$ se použijí pro trénování. To se opakuje K-krát. Každá testovací část je použita právě jednou pro testování modelu vzniklého ze zbylých částí dat. Nejčastěji volená hodnota K bývá 10 nebo 30. Ve své práci využívám tu to metodu a kvůli časové náročnosti testovaní jsem zvolil hodnotu $K = 3$.

U metody bootstrap, která je podobná křížové validaci se mohou vybrané data pro učení použít vícekrát. Trénovací data se rozdělí v poměru 63,2% pro trénování a zbylých 36.8% pro testování

Metoda leave-one-out je taky podobná metodě křížové validace, ale narozdíl z dat o N vzorcích se vybere jenom jeden pro testování a zbytek $N-1$ se použije na trénování. Celý postup se provede N-krát. Tato metoda se obvykle používá na medicínských datech.

Testování na testovacích datech je nejvhodnějším způsobem ověření modelu. Problém může vzniknout při nedostatku dat.

Kapitola 4

Implementace

V této kapitole budou popsány implementované programy a scripty pro realizaci práce. Na závěr kapitoly budou představeny použité knihovny.

4.1 Vytvoření trénovacího modelu

Pro vyhodnocení natrénovaného modelu využívám metodu křížové validace, která je popsaná v kapitole 3.4. Z časových důvodů vyhodnocení natrénovaných modelů jsem zvolil postup, který pomocí křížové validace s hodnotou $K = 3$ rozdělí testovací a trénovací soubory na tři části. Pro vytvoření trénovacího modelu pro detektor je vytvořen script *prepare_detector.py* a pro vytvoření modelu klasifikátoru do více tříd je vytvořen script *prepare_class.py*. Oba scripty očekávají adresářovou strukturu jak je použita v datové sadě GTSRB 5.4.

Script pro detektor očekává adresář s pozitivními snímky obrázků s příslušným anotačním souborem a adresář s negativní datovou sadou. Script pracuje a načte zvlášť absolutní cesty k souborům v datových sadám. A podle křížové validace jsou vytvořeny soubory pro trénovaní a testovaní, které obsahují cesty k souborům. V případě zpracovávaní pozitivní datové sady je k cestě souboru přidána velikost obrázku a pozice levého horního bodu označující začátek dopravní značky. Výstupem scriptu jsou také soubory pro testování a vyhodnocení pomocí ROC křivky.

V případě scriptu pro klasifikaci do více tříd je očekáván adresář, ve kterém bude pozitivní datová sada už převedená na descriptory příznaků HOG. Script načte absolutní cesty k souborům a pomocí metody křížové validace, která je implementovaná v metodě *split_cross* rozdělí soubory na trénovací a testovací. Protože hodnoty descriptorů nejsou v požadované formě knihovnou LIBSVM 4.7 je proveden převod na požadovaný tvar. Převod do požadovaného formátu obstarává metoda *prepare_data*, které jsou předány hodnoty decriptoru příznaků jednoho snímku. Metoda postupně iteruje předané hodnoty a přidává k hodnotě její načtené pořadí oddělené ":". Tak to vzniklý řetězec, využívá metoda *format_file*, která před takto vzniklý řetezec přidává třídu zařazení daného vzorku. Pro vytvoření modelu je použita metoda *one-vs.-rest (OvR)*, která spočívá v tom, že zvolená třída bude označena kladně a zbylé třídy záporně. Třídy jsou označeny podle počtu tříd v daném adresáři. Výstupem scriptu jsou modely pro trénování a testování, které se uloží na pevný disk. Součástí scriptu jsou i metody, které umožnují dodatečné přidaní negativních nebo pozitivních příznaků do výsledných modelů. Výstupem scriptu jsou také soubory pro vyhodnocení natrénovaného modelu pomocí ROC křivky.

4.2 Trénování modelu

V této sekci bude popsán popstup trénování detektoru a klasifikátoru a volba vhodných parametrů.

Z počátku jsem používal pro natrénovaní klasifikátoru jsem používal binární soubory dostupné pomocí knihovny LIBSVM a to pro trénování konkrétně *svm-train* a pro klasifikaci *svm-predic*. Binární soubory očekávaly na vstupu vytvořené modely v požadovaném formátu. Pro trénováí detektoru používám binární soubory *opencv_createsamples* a *opencv_traincascade*.

Později jsem si implementoval vlastní program *train* pro trénování klasifikátoru. Program je implementovaný v jazyce C++. Program očekává trénovací soubor a výstupem je natrénovaný model.

V případě detektoru je očekáván vytvořený soubor pozitivních snímků popsaný v kapitole 4.1 společně s počtem pozitivních snímků a s údajem o výšce a šířce snímku. Výsledkem je kolekce snímků ve stupních šedi v jednom souboru. V dalším kroku je vyžadován takto vzniklý vektorový soubor pozitivních snímků společně s počtem snímků vynásobený poměrem 0.9. Dále je očekáván soubor s negativními snímky vytvořený v kapitole 4.1 a počet negativních snímků. Rozměry negativních snímků musí být větší než pozitivních, neboť při trénování budou pozitivní snímky vkládány na negativní a vyhodnocovány v každé trénovací etapě. Dále můžou být předány parametry o použití příznaků LBP nebo Haarových vlnek. Po skončení trénování kaskády klasifikátorů bude natrénovaný model uložený v cascade.xml.

V případě trénování klasifikátoru je vyžadován trénovací soubor společně s parametry. Pro trénování využívám funkce z *svm.cpp* a *svm.h* dostupné z knihovny LIBSVM. Program načte trénovací soubor zjistí se délka příznakových descriptorů a počet vzorků. Podle získaných hodot se alokuje pamět ve struktuře *svm_model* a pomocí automatu jsou do ní ukládany hodnoty tříd a hodnoty descritorů. Po načtení souboru je struktura předaná funkci *train* z knihovny, která vrací natrénovany model a pro uložení modelu na pevný disk je volaná funkce *save_to_file*.

Vyzkoušel jsem přesnost klasifikace natrénovaných modelů s využitím lineárního kernelu a kernelu RBF. Ukázalo se, že klasifikace pomocí RBF kernelu je přesnější, ale proces trénovaní a klasifikování je pomalejší.

Na základě testování jsem zjistil, že klasifikace s základními parametry při trénovaní není ideální. Velká část vzorků byla špatně klasifikována do třídy. Po prostudování manuálu ke knihovně LIBSVM jsem zjistil, že zlepšit výsledky klasifikace mohu pomocí paramatrů *-c* a *-g*. Parametr *-c* může být měněn od hodnoty 2^{-5} až po hodnotu 2^{15}. Používá se k posunutí oddělovací roviny mezi třídami dat. Čím víc bude hodnota parametru c větší, tak tím víc se rozhodovací rovina přikloní k menší skupině třídy. V případě malé hodnoty se bude přiklánět k větší skupině. Defaultní hodnota je na stavena na 1. Parametr *-g* může být měněn od hodnoty 2^{-15} až po hodnotu 2^3. Je to konstanta pro funkci kernelu. Defaultně bývá nastavena jako:

$$g = \frac{1}{delka\ vektoru\ descriptoru} \tag{4.1}$$

Podle článku [13], ve kterém se používá stejná datová sada jako v mé práci, jsem volil parametr $c = 10$. Výsledky byly mnohem lepší. Experimentoval jsem s různými hodnotami c a g a na základě základě experimentování jsem zvolil hodnoty $c = 10$ a $g = 0.01$.

Zkoušel jsem použít pro natrénování modelu i SVM z knihovny OpenCV 4.7, které je založené na LIBSVM. Výsledky z testování byly nepatrně horší se stejnýma parametrama. Důvodem je, že OpenCV používá starší verzi LIBSVM. A natrénované modely nejsou vzájemně kompatibilní tak jsem to to řešení ve své práci nepoužil.

4.3 Výsledná aplikace

Výsledná aplikace je implementována v *classifySign* a je napsaná v jazyce C++. Jsou realizovány dvě třídy pro práci s detektorem a s klasifikátorem do tříd. Program očekává soubor obsahující absolutní cesty obrázků nebo videí a adresář do kterého se budou ukládat výsledky. Testovací soubor může obsahovat i jedinný snímek. Při spuštění jsou načteny natrénované modely pro detekci a klasifikaci. Z předaného souboru jako argument, je vždy vybírán jedinný řádek s cestou souboru, který je načten a zmenšen na velikost 640x480. Následně je snímek převeden do stupnů šedi a předán funkci *DetectMultiScale* z OpenCV, která detekuje dopravní značku. Funkce *DetectMultiScale* je založená na principu posuvného okna a principu pyramidy obrázku,který se zmenšuje na základě předaného parametru *scaleFactor* a na daném snímku pomocí posuvného okna se prohledává snímek. Funkce společně s nalezeným objektem vrací hodnotu s pravděpodobností nalezeného objektu, která je ukládána do souboru pro vyhodnocení pomocí ROC křivky společně s anotací. Na základě prahové hodnoty je rozhodováno jestli nalezený objekt je dopravní značkou nebo FP.

Při nalezení objektu, jsou získány souřadnice a podle nich je výříznutý výřez, který se extrahuje na descriptor histogramu orientovaných gradientů jako vektor typu float. Podle délky vektoru se alokuje pamět ve struktuře *svm_node* a jsou do ní uloženy hodnoty z vektoru a pro klasifikování se volá funkce *svm_predic_probhility*, které se předá struktura s příznakem a struktura načteného modelu. Výstupní hodnoty jsou ukládány do souboru pro vyhodnocení pomocí ROC křivky. Na základě prahové hodnoty je ještě rozhodováno jestli daný snímek je dopravní značka či nikoliv. Je-li nalezený snímek dopravní značkou k nalezenému objektu do obrázku je vložená ikonka klasifikované třídy. Ikona se zobrazuje na levo od nalezeného objektu a má stejnou velikost jako nalezený objekt. A navíc nalezený objekt je ohraničen zeleným čtvercem o velikosti objektu. Výsledné obrázky nebo video soubor jsou uložené v cílovém adresáři.

Při experimentování jsem narazil na problém, že u snímků zmenšených na $640x480$ dochází k mírné deformaci a z horšení kvality snímku a to vede k horší přestnosti klasifikace nalazených objektů. A proto vždy je klasifikace provedena nad nezmenšeným snímkem a přepočítána lokalizace objektu z menšeného snímku na původní.

Aplikace také umožnuje do adresáře ukládat FP objekty pro opětovné natrénovaní detektoru. A také umožnuje uložit TP objekty pro natrénovaní klasifikátoru pro zvýšení přestnosti klasifikace na lokalizačně posunutých nalezených dopravních značek.

4.4 Vyhodnocení

V této sekci bude popsány nástroje pro vyhdnocení. Nejprve anotační nástroj, vyhodnucující křivky a confusion matrix tabulka. A na závěr metoda F-measure.

Anotační program

Ve své práci umožnuji přidaní dalších snímků k pozitivní datové sadě a pro anotování přidaných snímků jsem vytvořil program v C++ využívající funkce z OpenCV. Program

očekává na vstupu soubor s cestami snímků a soubor do kterého se budou ukládat anotační údaje. Program čte jeden snímek po jednom. Program okamžitě ohraničuje objekt pomocí stavu *EVENT_LBUTTONDOWN* kliknutím levého tlačítka myši a vykresluje dokud není tlačítko uvolněno. Aplikace je navrhnutá, že vždy výsledné souřadnice jsou začínající v levém horním rohu.

ROC a Precision-recall křivka

Pro vyhodnocení natrénovaných modelů jsem implementoval ROC křivku 5.2 a Precision-recall křivku. Pro realizaci používám knihovnu scikit-learn[1]. Pro vykreslení průběhu grafu využívám vykreslovací framework matplotlib.pyplot [2]

Pro detektor je implementován script *roc_detector.py* a klasifikátor do více tříd *roc_multiclass.py*. Vstupem jsou vyhodnocovací soubory a výstupem je graf ROC a Precision-recall křivky.

Confusion matrix

Realice tabulky confusim matrix je implementovaná ve scriptu *confusion_matrix.py*, který očekává soubor, který je výstem klasifikátoru do více tříd a testovací soubor. Program nejprve načte soubor do pole a následně je rozdělen na skupinu tříd. Kde jednotlivé třídy jsou sečteny a způměrovány. Výsledkem je tabulka o počtu tříd, ze které lze vyčíst přestnost klasifikace vůči jiným třídam. Hodnoty v nejlepším případě na hlavní diagonále dávají 1.

4.5 Pomocné scripty

Extrakce HOG příznaků

Ve své práci se zabývám implementací extrakce příznaků HOG pro klasifikátor založený na SVM. Pro výpočet descriptou využívám funkci *HOGDescriptor* z knihovny OpenCV. Program očekává adresář se snímky a cílový adresář, kde se budou ukládat výsledné descriptory. Každý snímek je zmenšen na velikost 40x40 a následně pomocí funkce vypočítán descriptor s nasatvením velikost bloku 10x10 pixelů , velikost buňky 5x5 pixelů, s posunutím bloku při normalizaci o 5 pixelů v obou směrech, spočtem 8 kanálů v buňce a normalizací L2Hys s hranicí 0,5.

Generování negativní sady

Ve své práci si vytvářím vlastní negativní datovou sadu. Jako zdroj snímků s okolím používám sadu snímků z datové sady 2.5, které jsou pořízeny z kamery ve vozidle. Snímky jsou pořízeny v Belgii za deního světla a dobrých světelných podmínek. A dále používám vlastní záznamy pořízené v okolí Zlínského kraje v různou denní dobu s vlivem podnební činnosti. Výsledná negativní datová sada je navíc rozšířená o falešně pozitivní snímky s přibližně 20h záznamu.

Script očekává adresář s negativnímy snímky a výstupní adresář, ve kterém budou uloženy výřezy. Script načte vždy jeden snímek, který následně zmenší na velikost 640x480 a pomocí posuvného okna jsou generovány výřezy o velikosti 75x75. Celkem negativní datová sada obsahuje 56 378 výřezů, která je navíc rozšířená o FP objekty při detekci.

[1] http://scikit-learn.org/stable/
[2] http://matplotlib.org/api/pyplot_api.html

Sloučení datové sady

Pro natrénovaní klasifikátoru nad snímky, které jsou lokalizačně posunuté oproti anotaci. Jsem vytvořil script *concat_set.py*. Který očekává na vstupu dva adresáře s datovou sadou, kterou sloučí a ukládá do jiného adresáře.

4.6 WaldBoost

V rámci experimentu jsem se rozhodl nahradit detektor jako kaskádový klasifikátor za WaldBoost. Pro implementaci používám WaldBoost z experimentální větve OpenCV. Program *waldboost* je implementovaný v jazyce C++. Na základě vstupních parametrů program umožnuje trénovaní a detekovaní.

Při trénovaní program vyžaduje soubor absolutních cest k pozitivní datové sadě a soubor cest negativní datové sady. Následně program si vygeneruje z negativní datové sady $5x$ víc negativní vzorků na základě počtu pozitivních snímků. Negativní vzorky jsou generovány o velikosti snímku $24x24$. Snímky jsou převedeny na LBP příznaky a pro trénování je volaná funkce *fit* z knihovny. Výsledkem je natrénovaný model, který se uloží na pevný disk.

Při detekovaní je očekáván seznam snímků a natrénovaný model. Načtený snímek se prochází pomocí plovoucího okna $24x24$ a pyramidy obrázku. Takto vzniklé výřezy se vyhodnocují pomocí knihovní funkce *predic* a výřezy jsou dál předány funkci *groupRectangles*.

Na základě experimtování jsem zjistil, že řešení není vhodné z časových nároků pro detekovaní nad plnými snímky. Ale pro detekovaní výřezů je dostatečné.

4.7 Použité knihovny

Pro implementaci mé práce jsem využil některé dostupné existující knihovny.

Knihovna OpenCV 3.1

Knihovna OpenCV[1] je volně dostupná a multiplatformní knihovna pro práci s obrazem nebo s videem v reálném čase zameřená na algoritmy počítačového vidění. Dále poskytuje nástroje pro extrakci příznaků, rozpoznávání a klasifikaci nebo také podporu pro strojového učení - SVM, AdaBoost, neuronové sítě a další. Knihovna je implementovaná v jazyce C/C++ a poskytuje rozhraní pro jazyky C, C++, C#, Python, Java. Je distribuována pod BSD licencí.

Z možností knihovny jsem využil funkce pro načítaní a ukládání obrázku nebo videa. Vytvoření výřezu z obrázku, extrakci příznaků, trénování kaskádového klasifikátoru a detekci objektů.

Knihovna LIBSVM 3.21

Další knihovou, kterou jsem používal je knihovna LIBSVM [5] verze 3.21, která je napsaná v jazyce C/C++ a je multiplatformní. Je vyvíjena na Národní Taiwanské univerzitě. Jedná se o knihovnu pro prácí se support vector machines. Knihovna implementuje kernel funkce: lineární, polynomiální, RBF a sigmoidální. Ve své práci jsem využíval RBF kernelu.

[1] http://opencv.org/

Ve své práci jsem využíval funkce pro trénování a klasifikaci ze souborů *svm.cpp* a *svm.h*. Na základě návodu[1] jsem upravil funkce v *svm.cpp* pro podporu OpenMP[2], která umožnuje paralerizaci pro rychlejší trénovaní modelu a klasifikaci.

Více informací k LIBSVM, lze získat v manuálu ke knihovně [4]

Knihovna Boost 1.51.0

Poslední knihovnou, kterou používám ve své práci je Boost[3] ve verzi 1.51.0, která je napsaná v jazyce C++ a je multiplatformní. Jedná se rozsáhlou knihovnu usnadňující programování některých obecných úloh v jazyce C++ např. datové struktury, iterátory, správa paměti, genetické programovaní a další.

Ve své práci jsem využíval funkce pro práci se soubory.

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html#f432
[2] http://openmp.org/wp/
[3] http://www.boost.org/

Kapitola 5

Experimenty a vyhodnocení

V této kapitole se budu věnovat vyhodnocení detektoru a klasifikátoru dopravních značek na natrénovaných modelech. Pro vyhodnocení je použita charakteristika $F-measure$, která je popsána v kapitole 5.1. V první části kapitoly budou popsány metriky pro vyhodnocení modelu, následně vyhodnocení a závěr kapitoly bude věnovaný experimentům.

5.1 Metriky pro vyhodnocení kvality modelu

V této kapitole budou popsány metriky, které se používají k vyhodnocení natrénovaného modelu. Při psané této kapitoly jsem čerpal informace z [9] a [2].

Matice záměn

Matice záměn (confusion matrix) je užitečný nástroj k určení jak dobře nebo s jakou přesností klasifikátor rozpoznal testovací data a zařadil do tříd. Matice záměn viz tabulka 5.1, je to čtvercová matice o velikosti daných počtem tříd, do kterých se provádí klasifikace. V matici řádky představují třídy ve kterých je správná klasifikace a sloupce značí klasifikované třídy pomocí natrénovaného modelu. Tabulka 5.1 zobrazuje případ, kdy klasifikátor data zařazuje do dvou tříd, pozitivní a negativní.

Vysvětlení pojmů, které se vyskytují v tabulce:

- True Positive (TP) - vzorek patří do pozitivní třídy a je klasifikátorem označený správně pozitivní
- False Negative (FN) - vzorek patří do pozitivní třídy a je klasifikátorem označený chybně negativní
- True Negative (TN) - vzorek patří do negativní třídy a je klasifikátorem označený správně negativní
- False Positive (FP) - vzorek patří do negativní třídy a je klasifikátorem označený chybně pozitivní

Hodnoty TP a TN udávají jak klasifikátor správně klasifikoval data a zatímco hodnoty FP a FN udávají chybnou klasifikaci. V tabulce mohou být navíc řádky a sloupce, pro celkové součty. V tabulce mohou být ještě uvedeny řádky a sloupce pro celkový počet klasifikovatelných tříd. Kde P a N udávají celkový počet správně klasifikovaných vzorků a

to pozitivně nebo negativně. P' a N' opět udávají počet, ale klasifikované vzorky modelem a to pozitivně nebo negativně.

	Klasifikace modelem		
Správná klasifikace	Pozitivní	Negativní	Celkem
Pozitivní	TP	FN	P
Negativní	FP	TN	N
Celkem	P'	N'	P + N

Tabulka 5.1: Matice záměn

Na základě matice záměn lze spočítat různé charakteristiky k určení vyhodnocení modelu:

Úspěšnost (accuracy)

$$ACC = \frac{TP + TN}{P + N} \tag{5.1}$$

Úspěšnost vrací hodnotu v procentech jak klasifikátor správně klasifikoval testovací data. Metoda je závislá na složení testovacích datech, které snadno můžou vést k chybnému vyhodnocení.

Chyba (error)

$$ERR = \frac{FP + FN}{P + N} \tag{5.2}$$

Celková chyba vyjadřuje relativní počet chybných rozhodnotí systému. Společně s úspěšností slouží jako nejjednodušší charakteristika vyhodnocení.

Úplnost (Senzivita)

$$SENS = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{5.3}$$

Úplnost je charakteristika, která říká kolik z hledaných objektů bylo úspěšně nalezeno.

Specificita (Specificity)

$$SPEC = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{5.4}$$

Specificita je chrakteristikou, která určuje jak byly správně klasifikované nehledané objekty. Není doporučeno používat ji jako jedinnou charakteritiku. Mohlo by to vést k chybným závěrům o kvalitě systému.Je to tedy poměr správně negativních vzorků oproti všech negativně označených vzorků.

Přestnost (Precision)

$$PREC = \frac{TP}{TP + FP} \tag{5.5}$$

Přestnost je charakteristika určující jak klasifikátor správně klasifikoval hledané objekty. Není doporučeno používat ji jako jedinnou charakteritiku. Mohlo by to vést k chybným závěrům o kvalitě systému. Je to tedy poměr správně pozitivních vzorků oproti všech pozitivně označených vzorků.

Efektivita (Efficiency)

$$EFF = \frac{SENS + SPEC}{2} \tag{5.6}$$

Efektivita reprezentuje průměr mezi senzivitou a specifitou a výsledek vrací v procentech. Optimální hodnota je 100%, ale toho se zřítka dosáhne.

F-míra (F-measure)

$$F = \frac{2 * presnost * uplnost}{2 * presnost + uplnost} = \frac{2\,TP}{2\,TP + FP + FN} \tag{5.7}$$

F-míra je sourhná charakteristika přestnosti klasifikace. Nejlepší hodnota vyhodnocení dosahuje 1 a nejhorší 0.

5.2 ROC křivka

Pracovní charakteristika přijímače (**R**eceiver **O**perating **C**haracteristic) je užitečný visuální nástroj pro porovnání dvou modelů. ROC křivka vznikla během druhé světové války pro analyzování radarových snímků. Tato křivka dává do souvislosti vztah mezi specificitou 5.4 a senzitivitou 5.3 pro všechny hodnoty prahu θ . Tedy křivka dává do poměru správně klasifikované pozitivní vzorky a chybně klasifikované negativní vzorky jako pozitivní.

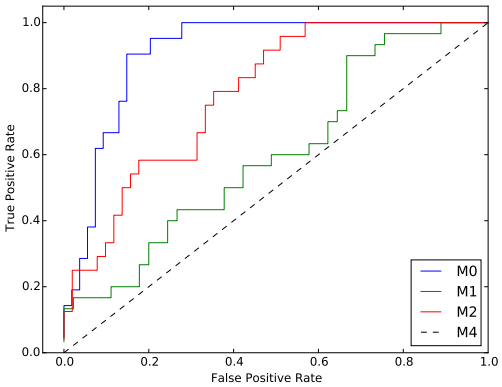
Graf ROC křivky

Grafem ROC křivky je dvourozměrný graf, kde na ose x je zobrazena pravděpodobnost chybné klasifikace negativních vzorků a na ose y pravděpodobnost správné klasifikace pozitivních vzorků. Pro každou hodnotu prahu θ, lze získat jeden bod na ROC křivce. Pro realizaci ROC křivky jsem výcházel z [8].

Na obrázku 5.1 je znázorněna ROC křivka se čtyřmi průběhy. Kde $M4$ zobrazuje průběh náhodného prediktoru. Bude-li průběh pod touto diagonálou tak je to nevyhovující stav a řešením je převrátit výsledky klasifikátoru. Z grafu je patrné, že ROC křivka vždy prochází počátečním bodem (0,0) a koncovým bodem (1,1). Průběh křivky je neklesající. Průběh $M2$ je považován za normální a bude-li se průběh křivky blížit průběhu $M0$, jde o ideální stav. Kdy klasifikátor zařadil většinu vzorků správně do třídy. Ideální křivka začíná v bode (0,0), pokračuje do bodu (0,1) a následně do bodu (1,1).

velikost AUC	hodnocení
0.5 - 0.6	nedostatečně
0.6 - 0.7	dostatečně
0.7 - 0.8	dobře
0.8 - 0.9	velmi dobře
0,9 - 1.0	výborně

Tabulka 5.2: Tabulka hodnot AUC



Obrázek 5.1: Znázorněná ROC křivka s průběhy. Kde $M0$ průběh je nejlepší.

Plocha pod ROC křivkou - AUC

Plocha pod ROC křivkou (**A**rea **U**nder the ROC **C**urve) slouží k porovnání ROC křivek a to buť vizuálně a nebo početně. Hodnota bude vždy mezi $0-1$. Z poznatku o náhodném prediktoru, který nebude klesat pod 0,5, bude výsledná hodnota mezi $0.5-1$. AUC vyjadřuje kromě plochy pod křivkou také míru uspořádanosti vzorků.

Na tabulce 5.2 je zobrazení ohodnocení testu podle plochy pod křivkou:

Jedným z možných použití AUC je pravděpodobnost jak vybraný vzorek byl klasifikovaný do tříd.

5.3 Vyhodnocení

Tato část kapitoly se bude zabývat vyhodnocením natrénovaných modelů detektoru a klasifikátoru.

Trénování a vyhodnocení bylo provedeno na notebooku Lenovo E530 s procesorem Intel Pentium B970 @ 2.30GHz a pamětí RAM 8GB.

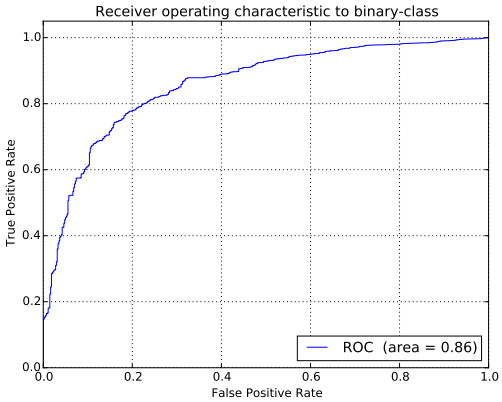
Vyhodnocení modelu německých dopravních značek

Pro vyhodnocení natrénovaného modelu byla použita metoda křížové validace, která rozdělí datovou sadu na testovací a trénovací části. Trénovací proces byl proveden třikrát a pokaždé bylo provedeno vyhodnocení na odpovídající testovací sadě. Na základě vyhodnocení byl vždy zvolen model s nejlepšímy výsledky.

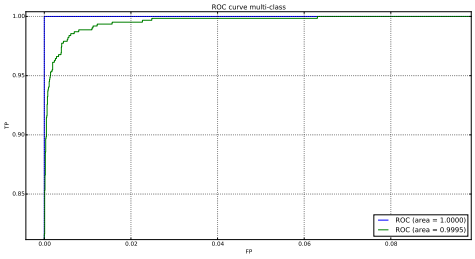
Vyhodnocení úspěšnosti probíhalo pomocí ROC křivky na testovací sadě určené zvlášť detektoru a klasifikátoru. A také pomocí metody F-Measure.

Pro natrénování modelu neměckých dopravních značek byla použita datová sada 5.4, která obsahuje přibližně 40 000 snímků značek. A negativní datová sada přibližně o počtu o 56 000 obrázků.

Testovací sada pro detektor obsahovala přibližně 30 000 snímků a detektor dosahoval úspěšnosti 74%. Testovací sada pro klasifikátor obsahovala přibližně 15 000 snímků a úspěšnost klasifikace 99,7%. Při společném vyhodnocení detektoru a klasifikátoru na testovací sadě úspěšnost dosahovala 42%.



(a) Detektor



(b) Klasifikátor

Obrázek 5.2: ROC křivka trénovacího souboru vytvořeného křížovou validací

(a) Výstup z aplikace

(b) Výstup z aplikace

Obrázek 5.3: Vyhodnocení na dat68ové sadě GTSDB.

Na obrázku 5.2a je znázorněná ROC křivka detektoru s využitím příznaků LBP a na obrázku 5.2b je ROC křivka klasifikátoru do více tříd s využitím příznaků HOG. Která je přiblížená a jsou zobrazeny nejlepší a nejhorší průběhy.

Z grafu křivky 5.2a je patrné, že detektor nachází hodně FP snímků. Řešením je znovu natrénovat nový model s FP snímky. A z vyhodnocení celého systému je patrné, že klasifikátor hůře klasifikuje lokalizačně posunuté dopravní značky z výstupu z detektoru.

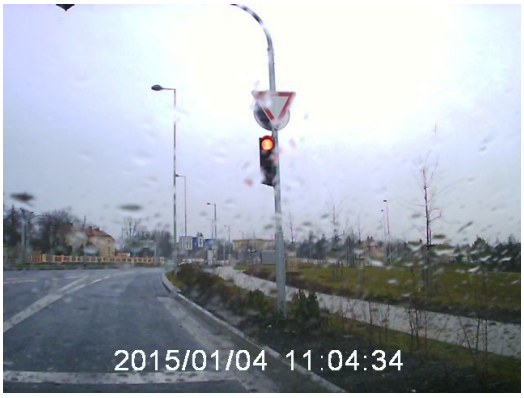
5.4 Experimenty

V této sekci budou popsány provedené experimenty.

Testovací sada

Cílem toho experimentu je vyhodnotit detektor a klasifikátor nad celými snímky. Testovací sada GTSDB obsahuje 900 anotovaných snímků pořízených z autokamery. Snímky jsou pořízeny v různou denní dobu a různých podnebních podmínek. Snímky zachycují dopravní situaci v mlze a v děšti. Přestnost detekce dosahuje 67% a přestnost celého systému 58%.

(a) Výstup z aplikace.



(b) Výstup z aplikace.

Obrázek 5.4: Vyhodnocení na datové sadě v dešti.

(a) Výstup z aplikace.

(b) Výstup z aplikace.

Obrázek 5.5: Výstup z aplikace.

Různé podnební podmínky

Cílem experimentu je vyhodnotit systém nad celými snímky za deště. Snímky byly pořízeny v okolí Prahy a obsahují celkem 42 snímků. Snímky jsou zmenšeny $640x480$. Na snímcích se vyskutují i dvě dopravní značky najednou. Převážně je sada tvořena dopravní značkou hlavní silnice, která je horší pro detekci.

Detekce na datové sadě dosahovala úspěšnosti 54% a celý systém 50%. Z obrázku 5.3 a je patrné, že na systém nemá vliv déšt, ale přestnost detektoru.

5.5 Pomocník do vozidla

Natrénované modely a připravené řešení6868 jsem integroval do aplikace, která detekuje a klasifikuje dopravní značky přímo z kamery. Pro videa v rozlišení 68 $640x480$ je rychlost zpracování 12 FPS.

Na obrázku 5.4 je zobrazen výstup z aplikace. V zelených rámečcích jsou zobrazeny objekty, které detektor považuje za značku. A na levo je zobrazená klasifikovaná třída dopravní značky.

Kapitola 6

Závěr

Cílem bakalářské práce je navrhout a implementovat systém pro detekování a klasifikaci dopravních značek. Tento úkol se mě podařilo splňit. A na testovací sadě vytvořené pomocí křížové validace a vyhodnocené pomocí metody F-measure dosahovala detekce přesnost 74%, klasifikace do více tříd 99% a výsledná přesnost aplikace 42%. Nízká přestnost výsledné aplikace dána nepřestnou lokalizací dopravní značky v obraze.

Aplikace umožnuje zpracovávat video nebo přímo vstup z kamery. Při rozlišení $640x480$ je rychlost zpracování 11 FPS.

Součástí řešení je sada scriptů, která zpracovává datovou sadu, vytváří trénovací a vyhodnovací sady. A scripty pro vyhodnocení natrenévaných modelů.

Během implementace jsem prostudoval dostupné řešení zabývající se detekcí a klasfikací značek a na základě získaných informací jsem navrh a implementoval řešení. Ve své práci jsem si vytvořil vlastní negativní datovou sadu pro trénovaní a vyhodnocení.

Možným rozšířením práce by mohlo být znovu natrénovaní modelů s FP a lokalizačně posunutými dopravními značkami. Dalším rozšířením může být rozšíření datových sad. Dalším možným rozšířením může být použité CUDA pro paralerní výpočty, pomocí kterých by bylo urychleno trénovaní a detekovaní.

Jsou práci jsem přihlásil na Exel@Fit 2016.

Literatura

[1] Benenson, R.; Omran, M.; Hosang, J.; aj.: Ten years of pedestrian detection, what have we learned? In *ECCV, CVRSUAD workshop*, 2014.

[2] Berka, P.: *Dobývání znalostí z databází*. Praha: Academia, vyd. 1. vydání, 2003, ISBN 8020010629.

[3] Bonači, I.; Kusalic, I.; Kovaček, I.: Addressing false alarms and localization inaccuracy in traffic sign detection and recognition. 2011.

[4] Chang, C.-C.; Lin, C.-J.: A Library for Support Vector Machines. 2001, dostupné na http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] Chang, C.-C.; Lin, C.-J.: LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, ročník 2, 2011: s. 27:1–27:27, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[6] Duda, R. O.; Stork, D. G.; Hart, P. E.: *Pattern classification*. New York, N.Y: John Wiley & Sons, druhé vydání, c2001, ISBN 0471056693.

[7] de la Escalera, A.; Armingol, J.; Mata, M.: Traffic sign recognition and analysis for intelligent vehicles. *Universidad Carlos III de Madrid*.
URL <http://orff.uc3m.es/bitstream/handle/10016/7089/traffic_escalera_IVC_2003_ps.pdf?sequence=1>

[8] Fawcett, T.: An introduction to ROC analysis. *Pattern Recognition Letters*, ročník vol. 27, č. issue 8, 2006: s. 861–874, ISSN 01678655, doi:10.1016/j.patrec.2005.10.010.
URL http://linkinghub.elsevier.com/retrieve/pii/S016786550500303X

[9] Han, J.; Kamber, M.; Pei, J.: *Data mining*. Boston: Elsevier, třetí vydání, c2012, ISBN 9780123814791.

[10] Hofmann, T.; Schölkopf, B.; Smola, A. J.: Kernel methods in machine learning: s. 1171–1220.

[11] 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.

[12] Ivanciuc, O.: Applications of Support Vector Machines in Chemistry. *Applications of Support Vector Machines in Chemistry*, ročník 23, 2007: s. 291–400.
URL
http://www.ivanciuc.org/Files/Reprint/Ivanciuc_SVM_CCR_2007_23_291.pdf

[13] Mathias, M.; Timofte, R.; Benenson, R.; aj.: Traffic Sign Recognition - How far are we from the solution? In *ICJNN*, 2013.

[14] Mogelmose, A.; Trivedi, M. M.; Moeslund, T. B.: Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems. *IEEE Transactions on Intelligent Transportation Systems*, ročník vol. 13, č. issue 4, 2012: s. 1484–1497, ISSN 1524-9050, doi:10.1109/TITS.2012.2209421.
URL
http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6335478

[15] Moutarde, F.; Bargeton, A.; Herbin, A.; aj.: Modular traffic signs recognition aplied to on-vehicle real-time visual detection of american and european speed limit signs.
URL <http://arxiv.org/pdf/0910.1295.pdf>

[16] Ruta, A.; Li, Y.; Liu, X.: Real-time traffic sign recognition from video by class-specific discriminative features. *Elsevier*, 2009: s. 416–430.
URL <http://people.brunel.ac.uk/~csstyyl/papers/pr2010.pdf>

[17] Shojania, H.: Real-time traffic sign detection. 2003, dostupné z http://hassan.shojania.com/pdf/TrafficSignDetection-Paper.pdf.
URL <http://hassan.shojania.com/pdf/TrafficSignDetection-Paper.pdf>

[18] 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.

[19] Timofte, R.; Zimmermann, K.; Van Gool, L.: Journal of Machine Vision and Applications (MVA 2011). In *Multi-view traffic sign detection, recognition and 3D localisation*, December 2011.

[20] Viola, P.; Jones, M. J.: Robust Real-Time Face Detection. 2004.
URL http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

Přílohy

Seznam příloh

Příloha A

Obsah CD

K práci je přiložené jedno DVD, které obsahuje zdrojové kódy, plakát, demostrační video, zprávu a datové sady. Součástí jsou i natrénované modely a výsledky.

- Aplikace
  - src - zdrojové soubory aplikace pro trénování, extrakci příznaků a detekování
  - tools - obsahuje scripty pro vytvoření modelů pro trénovaní, vyhodnocení natrénovaných modelů, práce s datovými sadami
- DataSet - obsahuje ziskané a nebo vrámci práce vytvořené datové sady
- Modely - natrénované modely
- Plákat - obsahuje plakát
- Video - obsahuje demostrační video
- Zpráva
- Výsledky

Příloha B

Plakat

Detekce dopravních značek z kamery ve vozidle

Michal Jurča, xjurca07@stud.fit.vutbr.cz
Vedoucí Ing. Vítězslav Beran, Ph.D.

Pozitivní datová sada

- Německé dopravní značky
- Celkem 39,102 snímků
- 43 tříd značek

Negativní datová sada

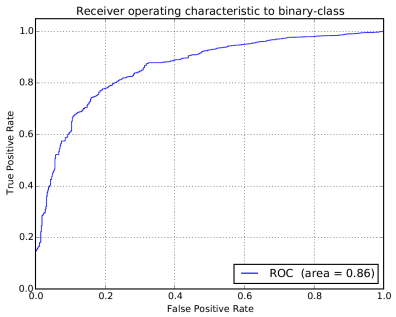
- Snímky s okolím
- Celkem 60,000 snímků

Pomocník do auta

- Při rozlišení 640x480 rychlost zpracování 12 FPS

Detekce značek

- Kaskádový klasifikátor
- LBP příznaky
- Křížová validace
- Testováno na testovací sadě s úspěšností 74%
- Vyhodnocení detektoru pomocí ROC křivky a F-Measure.



Klasifikace značek

- Support Vector Machine
- HOG příznaky
- Křížová validace
- Testováno na testovací sadě s úspěšností 99%
- Vyhodnocení klasifikátoru do více tříd pomocí ROC křivky a F-Measure.