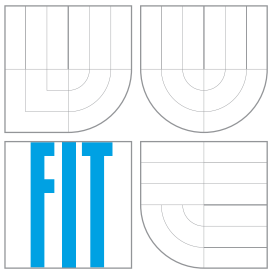
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ROZPOZNÁVANÍ DOPRAVNÍCH ZNAČEK V OBRAZE

DETECTION AND RECOGNITION OF TRAFFIC SIGNS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — RADOVAN VRÁNSKY
AUTHOR

VEDOUCÍ PRÁCE — Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá různými metodami detekce a rozpoznání dopravních značek v obraze. V úvodě jsou různé z těchto metod popsané a je ukázané jejich využití v praxi. V další části je podrobně popsaná implementace aplikace na detekci a rozpoznani dopravních značek v obraze s využitím Support Vector Machine. Také je tu popsaný způsob vytváření datové sady či různých modelů popisujících tuto sadu. V závěru je potom celá metoda zhodnocená.

Abstract

This bachelor thesis is about different methods of detection and recognition of traffic signs in pictures. The introduction several of these methods are described and their use is demonstrated. In the next part of the thesis, the implementation of the detection and recognition of traffic signs with the use of Support Vector Machine is described in detail. It also describes the method of creating of the dataset or different models describing this dataset. In the conclusion the method is evaluated.

Klíčová slova

detekce dopravních značek, OpenCV, Support Vector Machine, farebná segmentace, vektor príznaků

Keywords

traffic signs detection, OpenCV, Support Vector Machine, color segmentation, feature vector

Citace

Radovan Vránsky: Detekce a rozpoznávaní dopravních značek v obraze, bakalářská práce, Brno, FIT VUT v Brně, 2010

Detekce a rozpoznávaní dopravních značek v obraze

Prohlášení

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

.....

Radovan Vránsky

16. mája 2010

Poděkování

Na tomto mieste sa chcem poďakovať môjmu vedúcemu Ing. Adamovi Heroutovi, Ph.D za jeho ochotu vždy poradiť a pomôcť pri vytváraní tejto práce.

Obsah

Kapitola 1

Úvod

Dopravné značky sú dôležitou súčasťou premávky na pozemných komunikáciách. Patria medzi jedny z hlavných spôsobov, ako sprostredkovať dôležité informácie účastníkom cestnej premávky. V súčasnosti sa kladie veľký dôraz na bezpečnosť cestnej premávky a znižovanie počtu smrteľných dopravných nehôd. K dosiahnutiu tohoto cieľa môže dopomôcť aj systém schopný vyhľadávať a následne rozpoznávať dopravné značky v obraze. Takýto systém by sa mohol nachádzať v automobiloch a upozorňovať vodiča na dopravné značenie.

Ďalším možným využitím systému na detekciu a rozpoznávanie dopravných značiek je napríklad v autonómnych automobiloch. Problematika ohľadom rozpoznávania dopravných značiek v obraze je veľmi aktuálna, a preto sa ňou zaoberá niekoľko projektov a v rámci nich viacero výskumníkov.

V tejto bakalárskej práci sa pokúšam o klasifikáciu dopravných značiek v obraze. V prvej časti práce sú popísané rôzne metódy slúžiace na detekciu a klasifikáciu dopravných značiek. Taktiež je tu popísané aj ich konkrétne použitie v praxi. Väčšia časť prvej kapitoly je venovaná metóde, podľa ktorej bolo v tejto práci postupované. V kapitole s názvom Implementácia je potom podrobným spôsobom popísaná samotná implementácia aplikácie, spôsob vytvárania dátovej sady, či postup akým bol vytvorení model popisujúci túto dátovú sadu. Kapitola Testovanie sa zaoberá testovaním aplikácie za rôznych podmienok. A v kapitole Záver je stručne zhrnutá celá práca a zhodnotené výsledky, ktoré boli pri práci dosiahnuté.

Kapitola 2

Metódy rozpoznávania dopravných značiek v obraze

Detekcia a rozpoznávanie dopravných značiek v princípe pozostáva z dvoch hlavných krokov. Prvým krokom je určenie záujmovej oblasti, teda takej, kde sa dopravná značka nachádza. Tento krok sa vykonáva použitím rôznych techník na segmentáciu obrazu. A následne kroku, ktorý presne klasifikuje o aký druh značky v danom prípade ide. V tejto kapitole budú podrobnejšie popísané ako techniky na farebnú segmentáciu obrazu, tak aj metódy na klasifikovanie dopravných značiek. V druhej časti kapitoly bude podrobnejšie popísaná použitá metóda.

2.1 Farebná segmentácia obrazu

Farba je jedna z významných rysov dopravných značiek. Preto je farebná segmentácia často používaná pri detekcii dopravných značiek v obraze. Pomocou výsledkov segmentácie je možné určiť záujmové oblasti, kde by sa potencionálne mohla značka vyskytovať.

V súčasnosti sa pri rozpoznávaní dopravných značiek najčastejšie používajú RGB, HSV farebné modely. Farebná segmentácia obrazu nie je stopercentná, pretože môže byť ovplyvnená napríklad osvetlením, kvalitou farby značky či inými faktormi okolia.

2.1.1 RGB prahovanie

RGB farebný model je reprezentovaný tromi rôznymi farebnými zložkami a to *červenou* (R, red), *zelenou* (G, green) a *modrou*(B, blue). Každá z týchto zložiek býva uvádzaná v rozsahu 0 - 255, čo zodpovedá kódovaniu každej zložky do jedného bytu. Výslednú farbu je možné dostať kombináciou jednotlivých zložiek a ich rôznych intenzít. Ako je známe dopravné značky sú najčastejšie reprezentované červenou, modrou alebo zelenou, prípadne žltou farbou. Drvivá väčšina kamier či fotoaparátov poskytuje obraz reprezentovaný práve RGB modelom. Preto je vhodné použiť pri segmentácii dopravných značiek tento farebný model, čím je možné sa vyhnúť niekedy zdĺhavým transformáciám.

```
for all pixels i in image
    {
    if Ri > Gi and Ri - Gi≥ ΔRG ; Ri - Bi≥ ΔRB
       then pixel i is red
    else if Gi > Ri and Gi - Ri≥ ΔGR ; Gi - Bi≥ ΔGB
            then pixel i is green
    else if Bi > Gi and Bi - Gi≥ ΔBG ; Bi - Ri≥ ΔBR
            then pixel i is blue
    else pixel i is white(or black)
    end if
    }
end for
```

Obrázok 2.1: Algoritmus pre segmentáciu červených, modrých a zelených dopravných značiek

V prípade, že pracujeme s obrazom, ktorý je reprezentovaný RGB modelom, je napríklad dopravná značka STOP! reprezentovaná vysokými hodnotami zložky červenej farby (R, red). A naopak o poznanie nižšími hodnotami zložiek zelenej (G, green) a modrej (B, blue) farby. Takto je možné veľmi jednoduchým a výpočtovo nenáročným spôsobom vytipovať oblasti s potencionálnym výskytom dopravných značiek.

V tomto prípade môžeme naraziť na problém, že RGB model je náchylný na meniace sa svetelné podmienky. Pre predstavu to môže znamenať, že za normálnych podmienok môže byť objekt červenej farby reprezentovaný hodnotami RGB(255,0,0), ale pri zmene svetelných podmienok môžu tento objekt reprezentovať hodnoty RGB(248,30,29). Tento nedostatok sa snažia riešiť napríklad autori článku [2]. Pri tejto metóde autori sledovali meniace sa hodnoty zložiek RGB počas rôznych úsekov dňa a to v čase od 6^{30} do 21^{00}. Pomocou takto nazbieraných údajov zistili, že rozdiel ΔRG a ΔRB je výrazný aj pri meniacich sa podmienkach osvetlenia počas dňa. A preto je tieto hodnoty vhodné použiť ako prah k farebnej segmentácii. Výhodou metódy je fakt, že spomínané prahy nie je nutné nastaviť s veľkou presnosťou, pričom sú dosahované postačujúce výsledky. Samotný algoritmus je podrobne popísaný v 2.1. Spomínaný algoritmus využíva tri prahové hodnoty ΔRG, ΔRB a ΔGB. Je možné ho jednoducho doplniť o ďalšie prahové hodnoty a rozšíriť ho tak o detekciu napríklad žltej farby. Na obrázku 2.2 je potom vidieť použitie algoritmu na samotnú segmentáciu dopravnej značky.

2.1.2 Vytvorenie binárnej mapy pomocou RGB

Metóda podobná tej predchádzajúcej je napríklad vytvorenie binárnej mapy z obrazu reprezentovaného RGB modelom. Táto metóda je s úspechom použitá v článku [9]. Výsledkom

Obrázok 2.2: Použitie farebnej segmentácie v obraze

popisovanej metódy je binárna mapa, kde každý bod je reprezentovaný buď hodnotou 1 (biela) pre miesta, kde sa značka nachádza, alebo hodnotou 0 (čierna) pre miesta, kde sa značka nenachádza. Samotné rozhodovanie, či daný bod bude označený 0 alebo 1 je vykonávané pomocou vzťahu 2.1. V tejto metóde autori riešia vplyv zmeny osvetlenia či iných vonkajších činiteľov pomerom jednotlivých R,G,B zložiek modelu. Napríklad pri detekcii dopravných značiek s prevládajúcou červenou farbou by sme použili vzťah 2.2. Pre obidva vzťahy však platí, že premenné r,g a b predstavujú hodnoty farebných zložiek v danom bode. A α, β, γ sú konštanty určujúce prah úrovní farebných zložiek pre jednotlivé typy značiek. Na obrázku 2.3 je opäť možné vidieť, použite daného algoritmu k segmentácii dopravných značiek.

$$\frac{r}{g} > \alpha_{warn} \quad and \quad \frac{r}{b} > \beta_{warn} \quad and \quad \frac{g}{b} > \gamma_{warn} \tag{2.1}$$

$$\frac{r}{g} > \alpha_{red} \quad and \quad \frac{r}{b} > \beta_{red} \quad and \quad \frac{g}{b} > \gamma_{red} \tag{2.2}$$

2.1.3 Segmentácia obrazu pomocou CIECAM'97

Problém s meniacimi sa sveteľnými podmienkami pri súčasnom použití RGB farebného modelu, sa snažia riešiť aj autori článku [7]. Tu autori k farebnej segmentácii dopravných značiek využívajú model CIECAM. CIECAM alebo CIECAM'97 je model, určený na hodnotenie farebného vzhľadu. Tento model je odporúčaný komisiou CIE(Commission Internationale de l'Eclairage). Pôvodne bol skúmaný vo Veľkej Británii v rokoch 1980 až 1990. Model sa snaží čo najdôveryhodnejšie popísať vzhľad farby za rôznych pozorovacích podmienok. Farbu dokáže určovať s presnosťou priemerného pozorovateľa. Samotný model bol zostavený pomocou dlhodobého skúmania ľudských pozorovateľov a zbierky farebných dát na rôznych médiách, pre príklad spomeniem reflexný papier, textil či rôzne

Obrázok 2.3: Ukážka vytvorenej binárnej mapy v obraze

priesvitné materiály. V tabuľke 2.2 sú uvedené jednotlivé možné vstupy a výstupy modelu CIECAM'97.

V spomínanom článku autori vykonali rôzne merania na kuse bieleho papiera. Tieto merania prebiehali za rôznych svetelných podmienok a v rôznych častiach dňa po dobu dvoch týždňov. Merania vykonávali pomocou prístroja určeného na meranie farieb.Z takto získaných výsledkov vytvorili priemer a zostavili tabuľku 2.1 ktorú použili k ďalším výpočtom.

V prvom kroku samotnej farebnej segmentácie dopravných značiek bolo nutné obrázky z reálneho prostredia, reprezentované RGB farebným modelom, previezť do CIE XYZ hodnôt, s ktorými pracuje CIECAM. Tato transformácia bola vykonaná pomocou vzťahu 2.3. Následne bolo možné vygenerovať CIECAM model. Pre rôzne poveternostné podmienky boli použité rôzne nastavenia modelu použitím tabuľky 2.1. Následne pomocou hodnôt v tabuľke 2.3 bolo možné určiť vhodné prahy pre určovanie miest výskytu dopravných značiek.V tomto prípade autori uvádzajú 94% úspešnosť za slnečného počasia a $85-95\%$ úspešnosť počas dní so zvýšenou oblačnosťou a počas daždivých dní.

$$\begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \end{bmatrix} = \begin{bmatrix} 0.2169 & 0.1068 & 0.048 \\ 0.1671 & 0.2068 & 0.0183 \\ 0.1319 & -0.0249 & 0.3209 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} \tag{2.3}$$

2.1.4 CIELUV metóda

Veľa autorov nahrádza RGB farebný priestor za model H(hue, odtieň) S(saturation, nasýtenie) I(intensity, intenzita). Ako je možné vidieť v [7], kde autori transformovali údaje z RGB priestoru do priestoru HSI pomocou vzťahu 2.4. A k samotnej farebnej segmentácii použili model CIELUV. CIELUV je taktiež model odporúčaný komisiou CIE. Postup segmentácie je obdobný ako v prípade metódy popísanej v 2.1.3. V prvom rade sa určia

Poveternostné podmienky	Referenčná biela		Parametre okolia				
	X	y	C	F_{LL}	F	N_c	Y_b
Slnečný deň	0.3214	0.3228					
Zamračený deň	0.3213	0.3386	0.69	1	1	1	20
Daždivý deň	0.3216	0.3386					

Tabuľka 2.1: Parametre použité pre model CIECAM

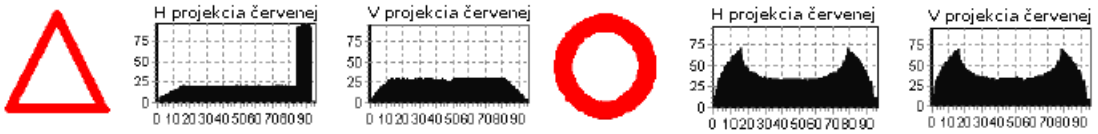
Vstupné hodnoty	Výstupné hodnoty
XYZ - relatívne trichromatické hodnoty farebného podnetu	svetlosť(J)
$X_W Y_W Z_W$ - relatívne trichromatické hodnoty bielej	farebnosť(M)
La - jas adaptačnej oblasti	sýtosť farby(C)
Y_b - relatívny jas pozadia	odtieň(h)
c, Nc, F_{LL}, F - parametre popisujúce okolie	jas(Q)
	sýtosť(S)

Tabuľka 2.2: Vstupné a výstupné informácie *CIECAM*

rozsahy pre rôzne podmienky osvetlenia a následne sa podľa týchto rozsahov určuje prah segmentácie. Takýmto postupom autori článku dosiahli úspešnosť 71% za slnečných dní a 61 až 63% úspešnosť počas zamračených i počas daždivých dní. V porovnaní s použitím modelu CIECAM je možné povedať že CIELUV má nižšiu úspešnosť.

Poveternostné podmienky	Odtieň		Sýtosť farby	
	Červená	Modrá	Červená	Modrá
Slnečný deň	$375 - 411$	$287 - 305$	$31 - 43$	$37 - 59$
Zamračený deň	$370 - 413$	$275 - 290$	$25 - 45$	$30 - 65$
Daždivý deň	$345 - 405$	$280 - 305$	$30 - 50$	$35 - 60$

Tabuľka 2.3: Rozsahy hodnôt farieb použité pri segmentácii značiek



Obrázok 2.4: Vertikálne a horizontálne zobrazenie histogramov pre niektoré dopravné značky.(Zdroj:[5])

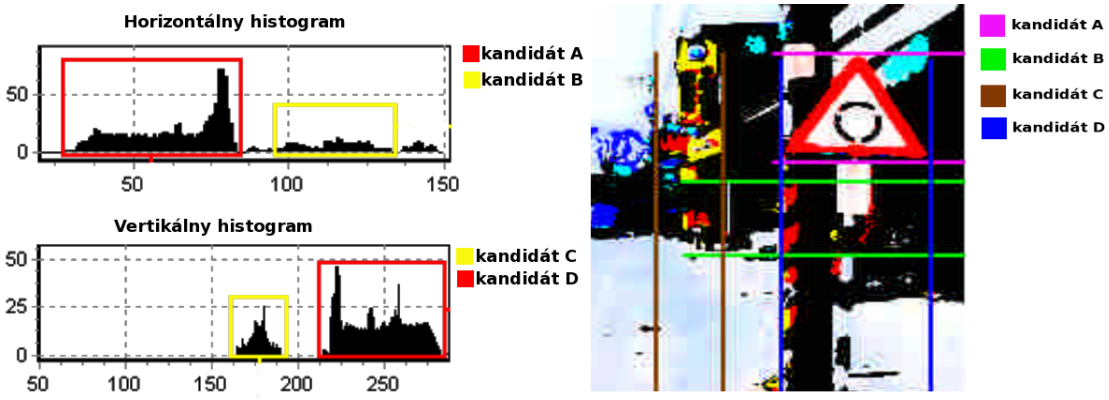
$$H = \cos^{-1}\left\{\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}, \quad R \neq G \; or \; R \neq B \tag{2.4}$$

$$S = Max(R,G,B) - Min(R,G,B)$$

$$I = \frac{(R+G+B)}{3}$$

2.1.5 Segmentácia obrazu pomocou HSV

HSV farebný model je podobne ako RGB model človeku blízky, ale jeho hlavnou výhodou je jeho menšia citlivosť na zmenu osvetlenia. Preto bol tento farebný model s úspechom využitý autormi v článku [5] na detekciu dopravných značiek. Postupovali tak, že farebný priestor rozdelili do ôsmich farebných oblastí. Tieto oblasti zahŕňali napríklad farby ako je červená, žltá, zelená, modrá či čierna a biela. Následne experimentálne došli k záveru, že značky ktoré sa rozlišujú svojim tvarom, sa rozlišujú aj v tvare histogramu a to dosť význačne. Príklady rôznych tvarov značiek a ich odpovedajúcich histogramov je možné vidieť na obrázku 2.4. Na základe týchto rozdielov mohli autori jednoducho určovať pozíciu dopravnej značky v obraze. Samotná detekcia je založená na vytváraní a analýze histogramov v horizontálnom a vertikálnom smere. Funguje tak, že pomocou prahov H_{min} a H_{max} je obrázok prevedený do binárneho obrázku. Ako jednotka je označený každý bod, ktorý patrí do intervalu medzi H_{min} a H_{max} a hodnotou nula sú označené všetky ostatné body. Následne sú vytvorené histogramy bodov reprezentovaných jednotkou. Pomocou týchto sú vybratí kandidáti na dopravnú značku. Môže nastať situácia, že sa v obraze nachádza či už v horizontálnom alebo vertikálnom smere viacero takýchto kandidátov. Tento problém rieši tak, že sa v každom smere vyberie oblasť, ktorá je reprezentovaná najväčším počtom jednotiek. Ako je možné vidieť na obrázku 2.5 vznikli v oboch smeroch dvaja kandidáti. Nakoniec však boli vybraní kandidát A a D. Výsledná oblasť je teda reprezentovaná prienikom týchto oblastí, čo je možné vidieť na obrázku v pravo.



Obrázok 2.5: Vľavo histogramy jednotlivých kandidátov. Vpravo oblasti vybrané na základe týchto histogramov. Oblasť výskytu dopravnej značky tvoria kandidáti A a D.(Zdroj:[5])

2.2 Metódy klasifikujúce dopravné značky

Pomocou farebnej segmentácie je vo vstupnom obraze možné určiť oblasti, v ktorých sa potencionálne môžu dopravné značky nachádzať. V tomto bode sa dostávajú k slovu metódy na presné určenie či sa jedná o dopravnú značku a prípadne aký druh dopravnej značky to je.

2.2.1 RBF siete

V prípade RBF (Radial Basis Function) sietí môžeme hovoriť o neurónových sieťach, ktoré majú len jednu skrytú vrstvu. Skladajú sa teda zo vstupnej, skrytej a výstupnej vrstvy. Učenie takýchto sietí sa dá v jednoduchosti prirovnať k hľadaniu takej mapovacej funkcie viacerých premenných, ktorej výstup sa najlepšie zhoduje so sadou trénovacích dát. Samotné učenie je možné popísať v nasledujúcich troch krokoch:

1. určenie vstupných váh
2. nastavovanie pomocných premenných RBF siete
3. nastavenie výstupných váh

Získavanie údajov z takto natrénovanej RBF sieti prebieha pomocou neurónov v skrytej vrstve. Každý neurón pritom môže mať napríklad nasledujúce správanie: *Ak príznak1 = 1 a príznak2 = 1 a $\cdots$ príznakN = 1 tak potom patrí do skupiny X.*

K rozpoznávaniu dopravných značiek použili RBF siete v článku [11]. V tomto prípade vytvorili autori trénovaciu sadu z fotiek dopravných značiek vytvorených za rôznych svetelných podmienok, z rôznych uhlov a vzdialeností. Z týchto fotiek potom vyrezali oblasti záujmu a teda dopravné značky a všetky normalizovali na rovnaké rozmery. Takto vytvorili šablónu. Následne z tejto šablóny vybrali dopravné značky obsahujúce červenú farbu a pomocou nasledujúceho vzťahu vypočítali pre každý pixel v obrázku hodnotu I:

$$I = \frac{1}{(1 + V_r^T \cdot \sum_r^{-1} \cdot V_r)} \tag{2.5}$$

pričom súčasne platí:

$$V_r = \begin{pmatrix} R - R_r \\ G - G_r \\ B - B_r \end{pmatrix} \tag{2.6}$$

Pre celý obrázok takto získali vektor príznakov. Keďže takto získané vektory môžu a často aj obsahujú redundantné dáta, aplikovali na tieto vektory LVQ (Learning Vector Quantization) výstupom tejto procedúry sú optimalizované údaje, ktoré sú vhodné na priame použitie k natrénovaniu RBF siete. Pomocou takto natrénovanej siete autori článku s úspechom rozpoznávali dopravné značky v obraze získaného pomocou kamery.

2.2.2 Support vector machine - SVM

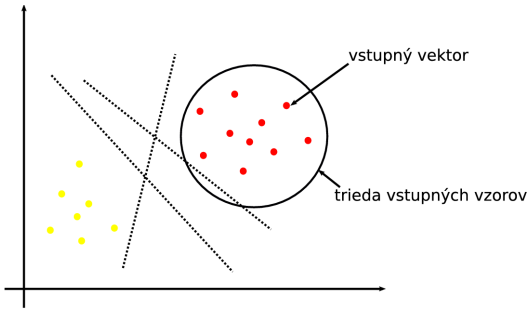
Support vector machine(SVM) je podľa [10] metóda používaná ku klasifikácii objektov. V jednoduchosti je možné povedať, že metóda pracuje so sadou trénovacích dát, z ktorých jednotlivé patria do jednej z dvoch tried - pozitívne(+1) alebo negatívne(-1). Trénovací algoritmus SVM potom vytvára model, za pomoci ktorého je možné určiť, či nový príklad patrí do jednej alebo druhej triedy. Support vector machines teda:

- definujú optimálnu deliacu rovinu - snaha maximalizovať hranice medzi triedami
- dokážu fungovať aj na lineárne nerozdelitelných problémoch - pomocou penalizácie za zlú klasifikáciu
- transformujú vstupné dáta do "feature space"

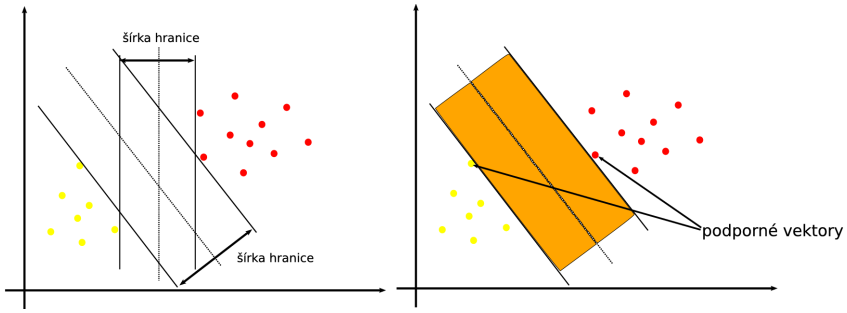
Problematiku SVM je možné vysvetliť na jednoduchom príklade. Ako je možné vidieť na obrázku 2.6 máme vstupnú množinu vektorov rozdelenú do dvoch tried. Túto množinu je nutné čo najvhodnejšie rozdeliť tak, aby bola maximalizovaná hranica medzi kladnou a zápornou množinou. Môže existovať nekonečné množstvo deliacich rovín. Preto je našou úlohou nájsť tú najvhodnejšiu, ako je možné vidieť na obrázku 2.7. V prípade formalizácie problému by sme vstupný vektor klasifikovali na základe vektora váh w a prahu b pomocou vzťahu 2.7. Potom pre body na okraji hranice platí 2.8. Aby sa ani jeden trénovací vektor nenachádzal medzi týmito dvomi hranicami, musí pre všetky trénovacie vektory platiť vzťah 2.9. Následne je optimalizačný problém možné definovať ako 2.10. Z hľadiska prístupu separácie dvoch tried vstupných vektorov je možné SVM rozdeliť na:

- *Hard-Margin* SVM - ktoré v 2.10 hľadajú riešenia parametrov w a b. Ak sú však vstupné vektory neseparovatelné, táto metóda riešenie nenájde. A je nutné pristúpiť k Soft-Margin SVM.

- *Soft-Margin* SVM - je nutné použiť v prípade, ak sa niektoré vstupné vektory nachádzajú za klasifikačnými hranicami. V tomto prípade sa vychádza zo vzťahu 2.10, ktorý je doplnený o premenné C a ξ_i a je upravený do výsledného tvaru 2.11. Algoritmus sa snaží ponechať tolerančné premenné ξ_i na nulovej hodnote pri súčasnej maximalizácii oddeľujúcej hranice. Väčšia hodnota $C \longrightarrow \infty$ spôsobuje, že riešenie sa približuje Hard-Margin. V prípade Soft-Margin SVM vždy existuje riešenie a sú robustnejšie v prípade zašumených vstupov.



Obrázok 2.6: Ukážka rôznych deliacich rovín



Obrázok 2.7: Hľadanie vhodnej deliacej roviny, s maximálnou šírkou

$$w \cdot x + b = 0 \tag{2.7}$$

$$w \cdot x + b = 1, \qquad w \cdot x + b = -1 \tag{2.8}$$

$$\omega_i(w \cdot x_i + b) \geq 1 \tag{2.9}$$

$$min\frac{1}{2} \parallel w \parallel^2 \quad \wedge \quad \omega_i(w \cdot x_i + b) \geq 1, \forall x_i \tag{2.10}$$

$$min\frac{1}{2} \parallel w \parallel^2 + C\sum_i \xi_i \quad \wedge \quad \omega_i(w \cdot x_i + b) \geq 1 - \xi_i, \forall x_i \tag{2.11}$$

K detekcii dopravných značiek použili SVM napríklad v [8]. V tomto prípade pozostával support vector z 20 prvkov. Každý z nich definovali ako vzdialenosť určitého bodu v skúmanej oblasti od hranice tejto oblasti. V prípade štyroch hraníc použili štyri rôzne SVM. Pri

použití trénovacej sady s 30 signálmi pre každý druh značky (kruhová, trojuholníková,...), dosiahli autori úspešnosť 96.5% pre kruhové značky a 94.8% pre trojuholníkové značky.

2.2.3 AdaBoost

Názov tohoto algoritmu je odvodený od slov Adaptive(prispôsobivé) a Boosting(posilnenie). Tento názov vystihuje základnú vlastnosť tohoto algoritmu - prispôsobivosť. Algoritmus pri učení využíva zle klasifikované objekty na natrénovanie nasledujúcich klasifikátorov, čím sa vlastne prispôsobuje. Podstatou samotného algoritmu je ukladanie si váh jednotlivých klasifikátorov. V prípade, že bol objekt klasifikovaný nesprávne, je váha daného klasifikátora posilnená. Toto spôsobí, že pri nasledujúcom trénovaní dokáže AdaBoost svoje rozhodnutie upraviť. Veľkou výhodou tohoto algoritmu je jeho odolnosť voči pretrénovaniu. To v podstate znamená, že pri veľkej sade trénovacích údajov nedosahuje algoritmus horšie výsledky, ako boli získané pri malej sade takýchto údajov. Konkrétne použitie algoritmu k rozpoznávaniu dopravných značiek je možné vidieť napríklad v článku [1]. Tu autori upravili pre svoje potreby algoritmus použitý Viola a Jonesom k rozpoznávaniu tvárí. Nevýhodou je potreba veľkého množstva trénovacích údajov a taktiež veľký počet stupňov kaskády klasifikátora pri dosiahnutí postačujúcej úspešnosti. Autori pre dosiahnutie spracovania dát v reálnom čase využili koncept Integrálneho obrazu:

$$ii(x, y) = \sum I(x', y') \tag{2.12}$$

Pomocou tohoto konceptu vypočítali príznaky a ku každému príznaku priradili jeho váhu. Takto pripravené dáta slúžili k natrénovaniu algoritmu AdaBoost. V tomto prípade každý stupeň kaskády znižoval označenie objektu za značku o 50% a vyradenie značky o 0.05% . Celá kaskáda klasifikátorov bola natrénovaná na sade 7000 pozitívnych a 12000 negatívnych príkladov. Vo výsledku bola pomocou tejto metódy dosiahnutá 98% úspešnosť detekcie dopravných značiek.

2.3 Použitá metóda

V nasledujúcej kapitole bude podrobne popísaná metóda [3] použitá na detekciu zákazových dopravných značiek. Popisovaná metóda bola východiskom pre túto prácu.

2.3.1 Prehľad systému

Ako je možné vidieť na obrázku 2.8 na vstup metódy je privedený obrázok nad ktorým chceme vykonať detekciu. V prvom kroku sa obrázok upraví na vhodnú mierku a vykoná sa farebná segmentácia a určenie oblastí, ktoré budú ďalej analyzované. Táto analýza prebieha formou postupného posúvania bloku, v tomto prípade o rozmeroch 24x24 pixelov. Pre každý blok sú extrahované príznaky. Tieto príznaky sú postupne vyhodnocované kaskádou klasifikátorov. V prípade, že je blok označený ľubovoľným stupňom kaskády ako negatívny,

nie je nutné vykonávať ďalšie testovanie. V opačnom prípade, ak daný blok prejde celou kaskádou, je označený ako pozitívny.

Obrázok 2.8: Diagram znázorňujúci postup metódy

2.3.2 Príznaky na základe histogramov orientovaných gradientov

V prípade tejto metódy existujú dve hlavné výhody, prečo použiť histogramy orientovaných gradientov(Histogram of Oriented Gradients,HOG). Prvou je, že dopravné značky majú výrazné tvary, ktoré v rôznych vrstvách môžu tvoriť silné hrany, gradientom je možné tieto príznaky zachytiť. Druhou výhodou je to, že HOG môže pomôcť k detekcii značiek v rôznych mierkach.

2.3.3 Definovanie príznakov

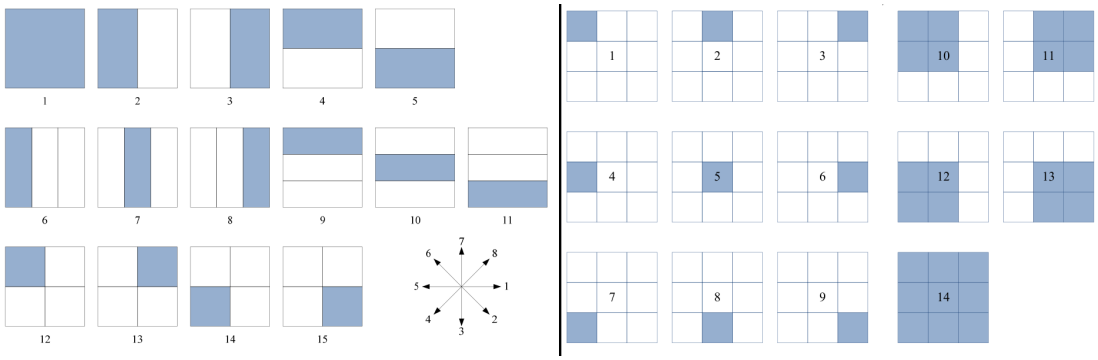
V prvom rade rozdelíme obrázok na 3x3 štvorcové plochy a definujeme 14 sub-blokov ukázaných na obrázku 2.9 vpravo . Potom si definujeme 15 šablón a rozsah orientácie $[0; 2\pi]$ rozdelíme na osem smerov ako znázorňuje obrázok 2.9 vľavo. Jednotlivé príznaky sú definované ako $F(i,j,k) = [f_r(i,j,k), f_g(i,j,k), f_b(i,j,k)]^T$. Kde každá s troch zložiek predstavuje jednu RGB vrstvu. Za predpokladu, že i-ty sub-blok nazveme B_i a j-tu šablónu zase C_j. Potom pre $f_r(i,j,k)$ v prvom rade vypočítame gradient pre každý pixel (x,y) vrstvy R ako:

$$G_{rx}(x,y) = [-1\ 0\ 1] * I_r(x,y) \tag{2.13}$$

$$G_{rx}(x,y) = [-1\ 0\ 1]^T * I_r(x,y) \tag{2.14}$$

sila gradientu v bode (x,y) je:

$$G_r(x,y) = \sqrt{G_{rx}(x,y)^2 + G_{ry}(x,y)^2} \tag{2.15}$$



Obrázok 2.9: Rozdelenie oblasti na jednotlivé podoblasti, šablóny a smery.(Zdroj:[3])

a jeho orientácia je:

$$\theta_r(x,y) = \tan^{-1}\left(\frac{G_{ry}(x,y)}{G_{rx}(x,y)}\right) \tag{2.16}$$

Pripomeňme, že sme smer orientácie $[0;2\pi]$ rozdelili do ôsmich smerov. Ak priradenie k-teho smeru urobíme pomocou:

$$\psi_{rk}(x,y)\begin{cases} G_r(x,y), & \text{if } \theta_r(x,y) \in \text{smer}_k \\ 0, & \text{inde} \end{cases} \tag{2.17}$$

Hodnota príznaku je pre $f_r(i,j,k)$ definovaná ako:

$$f_r(i,j,k) = \frac{\sum_{(x,y)\in C_j} \psi_{rk}(x,y)}{\sum_{(x,y)\in B_i} (G_r(x,y) + G_g(x,y) + G_b(x,y))} \tag{2.18}$$

kde $G_g(x,y)$ a $G_b(x,y)$ sú veľkosti gradientov pre vrstvy G a B. Obdobným spôsobom sa príznaky rátajú aj pre zvyšné zložky RGB modelu. Keďže máme 14 sub-blokov, 15 šablón a 8 smerov vo výsledku dostaneme $16 \cdot 15 \cdot 8 = 1680$ príznakov.

2.3.4 Metódy učenia

Z množiny 1680 príznakov je nutné vybrať tie, ktoré najlepšie popisujú danú dopravnú značku. Tento proces je možné popísať v nasledujúcich bodoch.

1. Vytvoriť trénovaciu sadu zloženú z pozitívnych a negatívnych vzoriek. Pre každú vzorku vypočítať príznaky.

2. Použiť SVM na klasifikáciu vzoriek. Prispôsobiť parametre tak, aby čo najmenej značiek bolo označených negatívne.

3. Po získaní výsledkov vyberieme príznaky s najlepšou schopnosťou detekovať dopravné značky. Urobíme rôzne kombinácie týchto príznakov a zostavíme väčší vektor príznakov, aplikujeme ho na trénovaciu sadu. Dostávame vektor príznakov s vyššou rozlišovacou schopnosťou $\Longrightarrow$ prvý "Single Classifier".

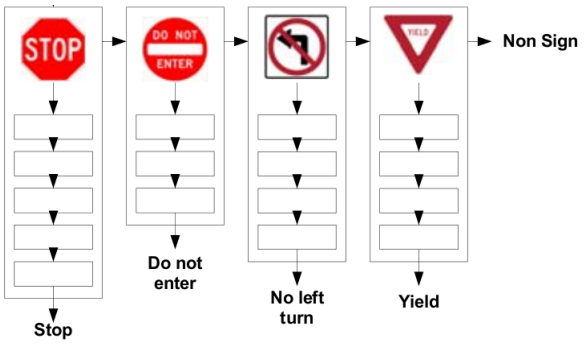
4. Výsledný klasifikátor overíme na úplne inej sade vzoriek. Zle detekované vzorky zaradíme do trénovacej sady a opakujeme kroky 2 až 4.

2.3.5 Trénovacia sada

V tomto prípade pozostávala trénovacia sada z 1921 príkladov. Tieto príklady obsahovali obrázky dopravných značiek, ale aj obrázky, kde sa dopravné značky nenachádzali. Trénovacia sada bola zostavená zhruba z 800 fotografií vytvorených autormi. V prípad obrázkov obsahujúcich dopravné značky tieto boli manuálne označené a upravené na mierku 24x24 pixelov. V druhom prípade, keď sa v obrázkoch dopravné značky nevyskytovali, boli tieto v rôznych rozmeroch vyrezané z tých istých fotiek a následne tak ako v prípade dopravných značiek pomocou bilineárnej interpolácie zmenšené na mierku 24x24 pixelov. Finálna trénovacia sada pozostávala zo 161 obrázkov značiek STOP!, 72 Daj prednosť v jazde!, 72 Zákazov odbočenia v ľavo, 83 Zákazov vjazdu a 1532 príkladov kde sa nevyskytovala žiadna dopravná značka. Na tejto trénovacej sade boli vytvorené klasifikátory a to tak, že pre danú značku boli pozitívne brané obrázky tejto značky a ostatné značky a pozadia boli chápané ako negatívne. Postup trénovania bol popísaný v kapitole 2.3.4.

2.3.6 Kaskádový detektor značiek

Po úspešnom trénovaní máme teda pre každú značku vytvorenú samostatnú kaskádu klasifikátorov. Výsledný detektor značiek je zostavený z týchto klasifikátorov tak, že sú postupne radené za seba, ako je možné vydieť na obrázku 2.10. Ak na vstup detektora privedieme obraz. V prvom kroku je testovaný na výskyt dopravnej značky STOP!. Ak všetky stupne klasifikátora pre túto značku úspešne prekoná, vo výsledku je vyhodnotený ako dopravná značka. A detekcia pre daný vstup je ukončená. V opačnom prípade obraz pokračuje k nasledujúcim kasifikátorom pri ktorých je proces obdobný. Ak však ani jeden klasifikátor neoznačí obraz ako značku, je klasifikácia ukončená s negatívnym výsledkom.



Obrázok 2.10: Kaskádový detektor značiek.(Zdroj:[3])

Kapitola 3

Implementácia

V nasledujúcej kapitole bude podrobne popísaná implementácia samotnej aplikácie. Aplikácia je primárne implementovaná v programovacom jazyku C++. Nektoré jej časti sú v jazyku C. Pri vývoji aplikácie boli použité knižnice OpenCV[4] verzie 2.0.0 a SVM-Light[6] verzie 6.02. Samotná aplikácia bola vyvíjaná a testovaná na operačnom systéme Ubuntu 8.04 LTS.

3.1 OpenCV

Je knižnica v počiatku vyvíjaná z prevažnej časti firmou Intel. Postavená je na programovacom jazyku C a C++ a je platformovo nezávislá. Bola navrhovaná s dôrazom na efektivitu výpočtov a spracovanie obrazu v reálnom čase. Všetky zdrojové kódy knižnice sú voľne dostupné a šíriteľné pod BSD licenciou. Od roku 1999, kedy bola knižnica zverejnená, sa celosvetovo rozšírila a v dobe písania tejto práce bola dostupná verzia 2.1. V súčasnosti knižnica OpenCV v sebe zahŕňa viac ako 500 optimalizovaných algoritmov, ktorých použitie je možné uplatniť v rôznych oblastiach, počínajúc medicínou cez bezpečnosť a končiac v priemyselnej výrobe. Súčasťou knižnice sú tiež prostriedky pre tvorbu jednoduchých grafických užívateľských rozhraní. A keďže v mnohých prípadoch je neoddeliteľnou súčasťou počítačového videnia aj počítačové učenie, tak knižnica obsahuje aj Machine Learning Library (MLL), ktorá ponúka rôzne metódy počítačového učenia. V samotnej práci bola využitá knižnica OpenCV verzie 2.0.0.

3.2 SVM-Light

SVM-Light je knižnica ktorá implementuje Support Vector Machine pomocou programovacieho jazyka C. Slúži k riešeniam problému rozpoznávania. Medzi hlavné výhody knižnice patrí možnosť efektívneho spracovávania problémov, ktoré môžu mať až niekoľko tisíc vektorov príznakov. Taktiež poskytuje rôzne metódy na hodnotenie výkonnosti či chybovosti algoritmu učenia. Samotná knižnica nachádza široké uplatnenie pri riešení rôznych proble-

mov napríklad rozoznávanie textu, obrazu, v medicíne či bioinformatike. Autorom knižnice je Thorsten Joachims z Cornell University. V súčasnosti sú knižnica a prípadné aplikácie dostupné vo verzii 6.02 vydanej v roku 2008. Táto verzia knižnice je použitá aj pri vývoji tejto práce.

3.3 Predspracovanie vstupného obrazu

Aplikácia preberá názov vstupného súboru, ktorý chceme spracovať ako parameter príkazového riadku. Samotný obrázok je načítaný pomocou funkcie `cvLoadImage()` z knižnice OpenCV. Táto funkcia pri otváraní súboru zároveň testuje jeho existenciu. V prípade, že funkcia vráti hodnotu 0 je program ukončený a užívateľovi je oznámené, že súbor pravdepodobne neexistuje výpisom na štandardný výstup. Po úspešnom načítaní obrázku je tento upravovaný funkciou `cvResize()` na veľkosť 640 x 480 bodov. V tomto rozlíšení je obrázok po spracovaní zobrazený a uložený. Pôvodný vstupný obrázok však ostáva zachovaný, pracuje sa len s jeho kópiou. Obrázok je následne programom v cykle zmenšovaní do rôznych mierok pomerom 1:0.9 pokiaľ jeden s jeho rozmerov nedosiahne veľkosť menšiu ako 26 bodov. V tomto cykle sa vykonáva prístup k jednotlivým bodom obrázku, určovanie a vyhodnocovanie farby týchto bodov a samotná klasifikácia.

Prístup k bodom obrázku

Vstupný obrázok je načítaný pomocou funkcie `cvLoadImage()` do štruktúry `IplImage`. Táto štruktúra uchováva napríklad informáciu o rozmere obrázku, jeho farebnú hĺbku, ale taktiež vo forme poľa `imageData` aj jednotlivé hodnoty bodov obrázku. V prípade, že si obrázok predstavíme ako dvojrozmernú maticu, tak k jednotlivým bodom môžeme pristupovať po riadkoch a následne po stĺpcoch. Pričom v prípade obrázku reprezentovanom tromi farebnými kanálmi R, G, B platí `3x + c`, kde `x` je daný stĺpec a `c` v našom prípade 0,1 alebo 2 reprezentuje jednotlivé farebné kanály. Postup prístupu k farebným zložkám obrázku je možné vidieť na nasledujúcom kóde kde `img` predstavuje spracovávaný obrázok :

```
for(int i = 0 ; i < img->height ; i++){
    uchar *ptr = (uchar*)(img->imageData + i*img->widthStep);
    for(int j = 0 ; j < img->width ; j++){
        R = ptr[3*j+2];   //červená zložka
        G = ptr[3*j+1];   //zelená zložka
        B = ptr[3*j];     //modrá zložka
    }
}
```

Určovanie farby bodu

Počas prechodu obrázkom a prístupu k hodnotám jednotlivých bodov je v aplikácii jednoduchým spôsobom vyhodnocovaná farba daného bodu. Toto vyhodnocovanie je vykonávané na základe prahu. V prípade zisťovania červenej farby je použitý vzťah 3.1.

$$D = \sqrt{(Red - 255)^2 + (Green - 0)^2 + (Blue - 0)^2} \tag{3.1}$$

Výsledok tohoto vzťahu je následne porovnaný s nami zvoleným prahom, v prípade aplikácie definovaným konštantou `TRESH`. Hodnotu prahu je možne ľubovolne meniť a ovplyvňovať tak množstvo bodov označených ako červených. Ďalšou možnosťou je tento prah určovať pre každý obrázok individuálne ako aritmetický priemer hodnôt získaných vzťahom 3.1 pre každý bod obrázku. V tomto prípade by však bol nevyhnutný jeden prechod spracovávaným obrázkom naviac.

3.4 Klasifikácia obrazu

V priebehu spracovávania obrazu je tento postupne zmenšovaní na rôzne mierky. V každej z týchto mierok je celá plocha obrázku prechádzaná okienkom o rozmeroch 24 x 24 bodov. A následne je každé toto okienko spracované triedou `Klasifikator`. Konštruktor tejto triedy v parametroch preberá vstupný obrázok v aktuálnej mierke, polohu okienka, nad ktorým sa vykonáva klasifikácia, rozmer tohoto okienka, v našom prípade 24 bodov a nepovinný parameter využívaní pri tvorbe modelu. Tento parameter nadobúda dve hodnoty 1 alebo -1, hodnota 1 určuje, že sa v danom okienku nachádza značka a naopak hodnota -1 zase že sa v okienku značka nenachádza.

Nad danou oblasťou táto trieda vykoná sériu výpočtov počínajúc volaním metódy `Gradient()`. Táto metóda pre každý bod v oblasti pomocou vzťahu 2.15 vypočíta silu gradientu a to pre každú z farebných zložiek R, G, B. Metóda `Gradient()` taktiež vyvolá metódu `Tangens()` pre každú farebnú zložku. Táto metóda pomocou funkcie `atan_2()` vypočíta uhol gradientu a následne mu priradí jeden z ôsmich smerov ako bolo popisované v 2.3.3 na obrázku 2.9. Nakoniec sú volané metódy `CalSubBlocks()` a `CalTemplates` ktoré spočítajú hodnoty jednotlivých podoblastí a šablón ktoré sú taktiež znázornené na obrázku 2.9 v kapitole 2.3.3. Trieda `Klasifikator` taktiež obsahuje metódy `ResultPrint()`, ktoré vo forme reťazcov navracajú výstupný vektor príznakov o rôznej dĺžke v tvare `č_príznaku:hodnota`.

V aplikácii je ako parameter referencia na túto triedu predaná funkcii `SvmDecide()`. Toto je členská funkcia triedy `SVM`. Konštruktor tejto triedy po jeho zavolaní alokuje miesto pre modely a načíta ich pomocou funkcie `read_model()`. Následne je v konštruktore model upravený do potrebného tvaru funkciou `add_weight_vector_to_linear_model()`. Trieda `SVM` tiež obsahuje metódu `SvmCompare()`. Táto metóda vo vstupných parametroch preberá vektor príznakov vytvorený funkciou `Klasifikator::ResultPrint()` a model, s ktorým chceme tento vektor príznakov porovnávať. Vo vnútri metódy je volaná funkcia `create_example()`, ktorá vytvára z vektoru príznakov model a následne je tento model

a model, s ktorým chceme porovnávať spracované funkciou `classify_example_linear()`. Táto funkcia nám vráti odchýlku *dist* týchto dvoch modelov. Táto hodnota je potom výstupom metódy `SvmCompare()` a môže nadobúdať záporné a kladné hodnoty. Čím viac je hodnota dist, kladná tým viac je model zhodný s vektorom príznakov a naopak, čím viac je číslo záporné tým sa viac od seba odlišujú. Táto metóda je využívaná hlavne metódou `SvmDecide()`, ktorá predstavuje kaskádu klasifikátorov a jej výstupom je v prípade neúspechu návratová hodnota 0 a v prípade úspechu odchýlka dvoch modelov, teda hodnota dist. Ak je vrátená táto odchýlka, tak je jej hodnota uložená v štruktúre `Hit`. Štruktúra Hit uchováva aj údaje o polohe nálezu a mierke, v ktorej bol nájdený.

Takto naplnená štruktúra je uložená v zozname nálezov. Takýchto nálezov sa vo vstupnom obraze môže vyskytovať viacero, a preto je potrebné ich ďalšie spracovanie. To spočíva v tom, že nálezy, ktoré sa vzájomne prekrývajú sú spojené do zhlukov. A následne sú z týchto zhlukov vybrané nálezy s najväčšou kladnou odchýlkou. Nakoniec sú tieto nálezy vyznačené vo výstupnom obrázku. Každý výstupný obrázok má pre lepšie odlíšenie prefix üpraveny_".

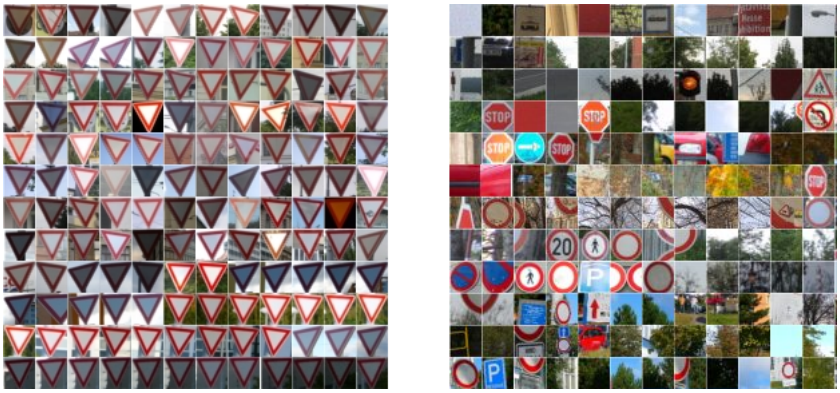
3.5 Vytváranie modelu

Ako bolo spomínané aplikácia k svojej správnej činnosti využíva rôzne modely. Každý model popisuje určitým spôsobom sadu trénovacích dát. Tvorba takéhoto modelu pozostáva z dvoch hlavných krokov. Zostavenia trénovacej sady dát a vytvorení samotného modelu.

Sada dopravných značiek a prvkov okolia

Pri tvore aplikácie boli použité tri sady dopravných značiek a prvkov z okolia. Prvou bola trénovacia sada, potom bola zostavená sada testovacia a nakoniec overovacia sada, slúžiaca na overenie výsledkov.

Trénovacia sada, ktorá v tejto práci bola použitá k vytvoreniu rôznych modelov, bola zostavená z rôznych fotiek. Tieto fotky obsahovali dopravné značky zachytené pri rôznych svetelných podmienkach, rôznych uhloch pohľadu a rozličných vzdialenostiach. Tiež tu boli zahrnuté fotky, ktoré obsahovali zábery budov, stromov a iného okolitého prostredia. Fotky s dopravnými značkami boli upravené tak, že z nich boli vyrezané len dopravné značky bez okolitého prostredia. Takto vyrezané dopravné značky boli následne upravené na rozmer 24 x 24 bodov. Rovnakým spôsobom boli upravené aj fotky okolitého prostredia. Tieto obrázky s veľkosťou 24 x 24 bodov boli následne spojené do dvojrozmernej matice a uložené ako jeden obrázok. Takto vo výsledku vznikli dva obrázky, jeden obsahoval dopravné značky a druhý rôzne prvky z okolitého prostredia. Príklad takto vytvorenej sady trénovacích údajov je možné vidieť na obrázku 3.1. Podobným spôsobom bola zostavená aj trénovacia a overovacia sada.



Obrázok 3.1: Ukážka trénovacej sady. Vľavo matica 12 x 12 obrázkov značky Daj prednosť v jazde!. Vpravo matica 12 x 12 obrázkov prvkov z okolitého prostredia.

Proces vytvárania modelu

Obidva takto pripravené obrázky tvoriace trénovaciu sadu boli spracované programom `trenuj`. Tento program na vstupe očakáva za prepínačom `--picture` vstupný obrázok, za prepínačom `--size` rozmer malého obrázku v našom prípade 24 a za prepínačom `--flag` hodnotu 1 ak išlo o obrázok s dopravnými značkami a -1 ak sa jednalo o obrázok s okolitými prvkami . Výstupom programu je potom 1680 súborov. Každý súbor obsahuje niekoľko riadkov, pričom každý riadok predstavuje jeden príznak patriaci obrázku s rozmerom 24 x 24 bodov. Príznak s prefixom 1 reprezentuje dopravnú značku a s prefixom -1 reprezentuje okolité prostredie. Každý súbor teda predstavuje akýsi slabý klasifikátor.

Všetky takto vytvorené súbory boli spracované skriptom, ktorý v prvom kroku odstránil príliš malé súbory, takto sme dostali 1050 použiteľných súborov. Následne skript pomocou programu `svm_perf_learn` vytvoril pre každý tento súbor model. Každý tento model bol potom pomocou programu `svm_perf_classify` otestovaný na súboroch získaných pomocou programu `trenuj` nad testovacou sadou. Výstupom tohoto testovania bol súbor, ktorý na každom riadku obsahoval hodnoty pre každý slabý klasifikátor. Všetky riadky mali nasledujúci tvar:

```
147-i_1-j_14-k_1 : Accuracy: 96.23 Precision: 94.74  Recall: 94.74
```

Pričom `147-i_1-j_14-k_1` predstavuje označenie daného klasifikátora. Hodnota `Accuracy` určuje koľko dopravných značiek a prvkov okolia bolo označených správne. `Precision` určuje pomer pozitívnych prípadov (v našom prípade dopravných značiek) ku všetkým správne určeným prípadom (správne určené dopravné značky + správne určené prvky okolia). A hodnota `Recall` popisuje koľko dopravných značiek bolo označených ako okolité prostredie. Z týchto výsledkov boli vybrané tie slabé klasifikátory, ktoré mali najvyššie hodnoty Accuracy a Recall. Takto vybrané klasifikátory boli spájané, a vytvorili tak silnejší klasifikátor. Tento klasifikátor bol opäť aplikovaný na trénovaciu sadu, a následne pomocou

programu `svm_learn` bol vygenerovaný model patriaci k tomuto klasifikátoru. Kvalita takto vytvoreného modelu a klasifikátora bola otestovaná na overovacej sade dopravných značiek a prvkov z okolia.

Aplikácia pri činnosti využíva niekoľko klasifikátorov rôznej veľkosti a pre rôzne druhy dopravných značiek. Preto je ku každému takémuto klasifikátoru a pre každý typ dopravnej značky potrebné vytvoriť samostatný model.

Kapitola 4

Testovanie

V nasledujúcej kapitole budú popísané rôzne testy ktorím bola vytvorená aplikácia podrobená. Testy názorne ukazujú, ako je možné ovplyvniť rýchlosť, prípadne kvalitu vyhľadávania a klasifikácie dopravných značiek v obraze.

Pre testovanie som zvolil dopravné značky typu Daj prednosť v jazde a Hlavná cesta. Tieto dve značky sa odlišujú farebne a aj tvarovo. Metóda pomocou ktorej bola aplikácia implementovaná však umožňuje rozpoznávať celé spektrum dopravných značiek. V tomto prípade by bolo vytvorenie dátovej sady a testovanie podstatne časovo náročnejšie, preto som pracoval len s týmito dvomi značkami.

V prípade testov Test II. až Test V. boli z množstva obrázkov vybrané tie, ktoré boli vždy správne klasifikované, preto sa v tabuľkách môžu vyskytnúť hodnoty pri úspešnosti až 100%. Tento výber bol účelový aby bolo možné ukázať vplyv rôznych parametrov na beh aplikácie.

Test I.

Prvý a najzákladnejší test bol vykonaný v aplikácii v základnom nastavení. Test mal za cieľ zistiť s akou úspešnosťou je aplikácia schopná správne vyhľadávať a klasifikovať dopravné značky vo vstupnom obraze. Výstup a výsledky testovania je možné nájsť na CD v súbore:

Typ značky	celkom	správne určené	nesprávne určené	úspešnosť v %
Daj prednosť v jazde!	110	107	3	97.27
Hlavná cesta	34	32	2	94.12
Celková úspešnosť	144	139	5	96.53

Tabuľka 4.1: Úspešnosť vyhľadávania rôznych typov dopravných značiek v obraze.

testovanie\test1. Testovacie obrázky, ktoré boli použité ako vstup aplikácie, sa nachádzajú v súbore *testovanie\vstup*.

Test II.

Cieľom nasledujúceho testu bolo zistiť správanie sa aplikácie pri rôznych nastaveniach prahu pri farebnej segmentácii. Na vstupe aplikácie bolo 10 obrázkov, ktoré je možné nájsť v súbore *testovanie\test2\vstup*. Bolo testovaných niekoľko prahov v rozsahu 165 až 235 a krok medzi jednotlivými hodnotami bol 10. Okrem toho bol testovaný prah(IP) individuálne počítaný pre každý obrázok. Výsledky tohoto testovania je možné nájsť v súbore *testovanie\test2*. Pri teste bola meraná úspešnosť klasifikácie a doba trvania. Z testu vy-

Prah	165	175	185	195	205	215	225	235	IP
Čas[s]	31,9	35,8	46,7	75,3	132,3	309,9	489,7	592,9	675,3
Úspešnosť[%]	63,6	81,8	81,8	90,9	100	100(1)	100(2)	100(2)	100(2)

Tabuľka 4.2: Úspešnosť vyhľadávania dopravných značiek v obraze a doba trvania pri rôznych nastaveniach hodnoty prahu.

plýva možnosť meniť kvalitu vyhľadávania a klasifikácie dopravných značiek. Tiež je možné vidieť narastajúcu dobu trvania celého procesu. Ako vidno na grafoch A.3 a A.4 nachádzajúcich sa v prílohe, po určitú hranicu prahu narastá kvalita a aj čas vyhľadávania. Od istej hodnoty prahu, v tomto prípade 205 sa úspešnosť nezvyšuje, pričom čas spracovania aj ďalej narastá. Označenie (1) v tabuľke znamená, že aplikácia našla značku správne, ale tiež označila prvok z okolia ako značku.

Test III.

V tomto teste bol menený skok pohybu okienka klasifikátora po obraze v rozsahu 1 bod až 10 bodov, v normálnom prípade je táto hodnota nastavená na 1 bod. Tento test opäť znázorňuje jeden spôsob, ktorým je možné ovplyvniť rýchlosť aplikácie, v tomto prípade na úkor kvality vyhľadávania. Ako je vidieť z tabuľky a z grafov A.7,A.8 časovo najnáročnejšie

Bod	1	2	3	4	5	6	7	8	9	10
Čas	132,3	45,5	29,3	23,8	18,8	17,6	16,6	16,3	15,7	15,4
Úspešnosť[%]	100	90	80	80	40	50	20	20	30	0

Tabuľka 4.3: Úspešnosť vyhľadávania dopravných značiek v obraze a doba trvania pri rôznych nastaveniach hodnoty skoku pohybu klasifikátora.

ale najkvalitnejšie vyhľadávanie je pri posune okienka klasifikátora po 1 bode. Rozumná rýchlosť pri 80% úspešnosti sa dá dosiahnuť pri posune okienka po 4 bodoch. Od tejto hranice však už prudko klesá úspešnosť vyhľadávania pod 50%. Výstup tohoto testu je v súbore *testovanie\test3*.

Test IV.

V teste číslo štyri sa snažím ukázať ako farebná segmentácia urýchľuje proces vyhľadávania dopravných značiek v obraze. Počas tohoto testu boli vykonané tri merania času pri spracovávaní 10 obrázkov. V prvom prípade sa meral čas pri použití klasifikátora spolu s farebnou segmentáciou. Následne bolo vykonané meranie doby trvania pri použití klasifikátora samostatného a nakoniec bola meraná doba trvania samotnej farebnej segmentácie. Výsledky testu sú v nasledujúcej tabuľke. Z výsledkov je na prvý pohľad vidieť, že samotný

	Klasifikátor + farebná segmentácia	Klasifikátor	Farebná segmentácia
Čas[m]	2,195	23,741	0,275

Tabuľka 4.4: Časová náročnosť jednotlivých operácií.

klasifikátor je nesmierne časovo náročný, pričom farebná segmentácia prebehla za necelých 17 sekúnd. Z testu vyplýva, že segmentácia podstatným dielom skracuje dobu vyhľadávania dopravných značiek v obraze.

Test V.

Pri poslednom piatom teste bol meraný vplyv pomeru, ktorým sú v cykle postupne zmenené obrázky na menšie, na kvalitu vyhľadávania a klasifikácie dopravných značiek. Tiež bola meraná doba trvania pri spracovaní 10 obrázkov, na ktorých sa nachádzalo 11 dopravných značiek. Z testu vyplýva, že najkvalitnejšie spracovanie je pri pomere 1:0.9, v tomto prípade však kvalita narastá na úkor doby trvania spracovania ktorá, je v tomto prípade značná. Naopak čím menšou konštantou sa obrázok násobil, tým čas spracovania klesal, v tomto prípade však neboli dosahované rozumné výsledky vyhľadávania. Výsledky tohoto testu sa nachádzajú v súbore *testovanie\test5*. Graf časovej závislosti a tiež závislosti úspe-

Pomer	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Čas[s]	7,6	10,6	14,8	21,7	33,8	57,3	133,6
Úspešnosť[%]	0	18,2	27,3	36,4	36,4	72,7	100

Tabuľka 4.5: Úspešnosť vyhľadávania dopravných značiek v obraze a doba trvania pri rôznych nastaveniach pomeru zmenšovania obrazu.

šnosti vyhľadávania na hodnote pomeru zmenšovania vstupného obrazu, je možné vidieť na grafoch A.10 a A.11.

Kapitola 5

Záver

Počas tvorby bakalárskej práce som sa oboznámil s viacerými metódami na detekciu a rozpoznávanie dopravných značiek. Taktiež som sa oboznámil so základnými funkciami knižnice OpenCV. Z rôznych postupov slúžiacich k detekcii a rozpoznávaniu dopravných značiek som si zvolil jeden, podľa ktorého som postupoval pri implementácii.

Ako je možné vidieť v testoch z predchádzajúcej kapitoly, aplikácia pri vyhľadávaní dopravných značiek dosahuje celkom vysokú úspešnosť. Tento fakt môže byť spôsobený tým, že testy boli vykonávanú nad malou sadou testovacích dát. V súčasnom stave sú vytvorené klasifikátory a k nim prislúchajúce modely pre dopravné značky typu Daj prednosť v jazde! a Hlavná cesta. Tieto značky dokáže aplikácia vyhľadávať a klasifikovať. Aplikáciu je však možné jednoducho rozšíriť aj o ďalšie typy dopravných značiek.

Z testov tiež vyplynulo, že aplikácia pre svoju činnosť potrebuje dostatok času. Tento fakt zatiaľ vylučuje jej nasadenie v real-time systéme. Najväčšiu časť svojho času aplikácia spotrebováva v samotnom procese klasifikácie dopravnej značky. V budúcnosti by preto bolo zaujímavé vyskúšať iný spôsob počítačového učenia, napríklad s využitím algoritmu AdaBoost, či pri vytváraní vektoru príznakov použiť napríklad koncept integrálneho obrazu.

V prípade ďalšieho vývoja aplikácie by som sa zameral na dosiahnutie jej činnosti v reálnom čase a spracovávaní obrazu z kamery pripevnenej na pohybujúcom sa vozidle.

Literatúra

[1] Baro, X.; Vitria, J.: Fast Traffic Sign Detection on greyscale images. [online]. `http://upisun4.uab.es/~jordi/BaroCCIA2004.pdf`, [cit. 2010-04-07].

[2] Bénallal, M.; J.Meunier: Real-time color segmentation of road signs [online]. `http://benallal.free.fr/articles/CCGEI2003/CCGEI2003BenallalMeunier.pdf`, 2003-05-07 [cit. 2010-04-07].

[3] Chen, C.; Chen, M.; Gao, T.: Detection and Recognition of Alert Traffic Signs [online]. `http://ai.stanford.edu/~kosecka/final_report_final_traffic.pdf`, [cit. 2010-04-07].

[4] Gary, B.; Adrian, K.: *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly, 2008 [cit. 2010-04-07], ISBN 978-0-596-51613-0.

[5] Hsien, J.; Liou, Y.; Chen, S.: Road Sign Detection and Recognition Using Hidden Markov Model [online]. `http://www.asia.edu.tw/ajhis/no1/07-his06018.pdf`, 2006 [cit. 2010-04-07].

[6] Joachims, T.: SVMlight Support Vector Machine. [online]. `http://svmlight.joachims.org`, [cit. 2010-04-07].

[7] Kolektív autorov: *Colour Vision Model-Based Approach for Segmentation of Traffic Signs.* EURASIP Journal on Image and Video Processing, 2008 [cit. 2010-04-07].

[8] Kolektív autorov: Traffic Sign Classification Invariant to Rotations Using Support Vector Vachines [online]. `http://agamenon.tsc.uah.es/Investigacion/gram/publications/acivs04-lafuente.pdf`, [cit. 2010-04-07].

[9] Shneier, M.: *Road Sign Detection and Recognition.* IEEE Computer Society International Conference on Computer Vision and Pattern Recognition, 2005 [cit. 2010-04-07].

[10] Sonka, M.; Hlavac, V.; Boyle, R.: *Image Processing, Analysis and Machine Vision.* Thomson Learning, 2008 [cit. 2010-04-07], ISBN 10: 0-495-08252-X.

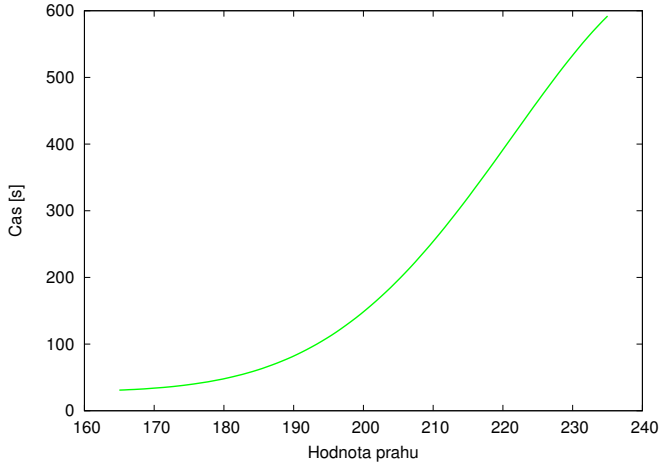
[11] Zheng, Y.-J.; Ritter, W.; Janssen, R.: An adaptive system for traffic sign recognition [online]. http://dx.doi.org/10.1109/IVS.1994.639496, 1994 [cit. 2010-04-07].

Dodatok A

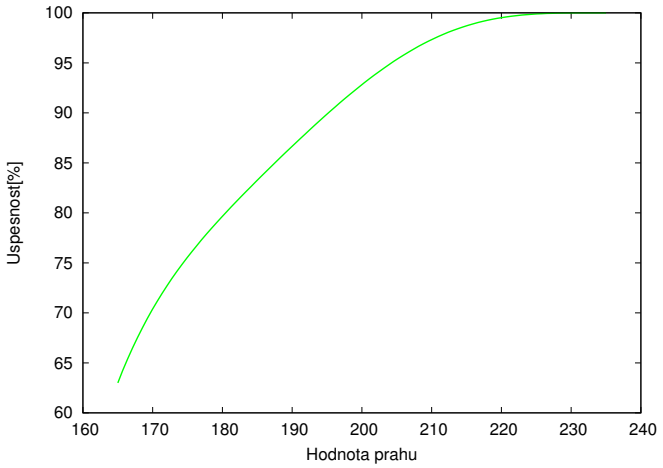
Príloha A - Výstupy a Grafy k testom

Obrázok A.1: *Test 1*:Ukážka funkčnosti aplikácie na obrázku s dopravnou značkou Daj Prednosť v jazde!

Obrázok A.2: *Test 1*:Ukážka funkčnosti aplikácie na obrázku s dopravnou značkou Hlavná cesta.

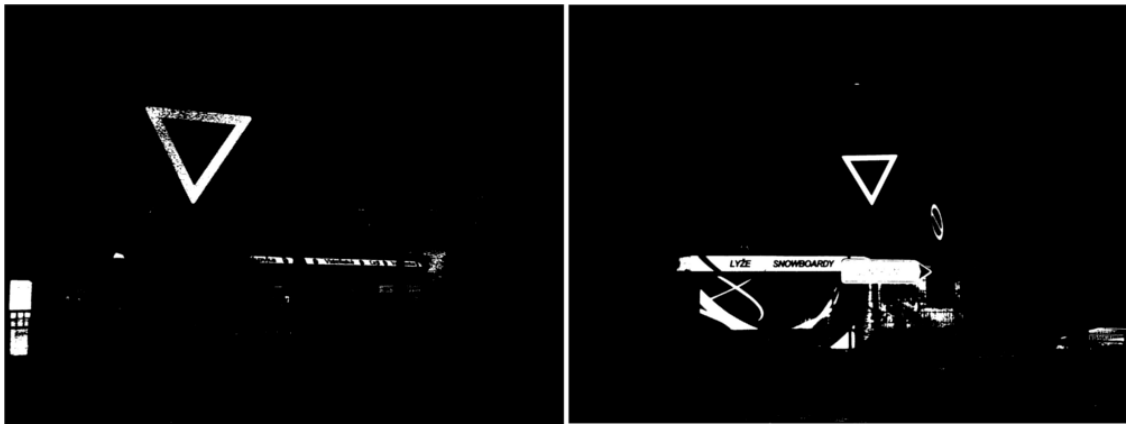


Obrázok A.3: *Test 2*: Graf závislosti času, na zmene hodnoty prahu pri farebnej segmentácii.

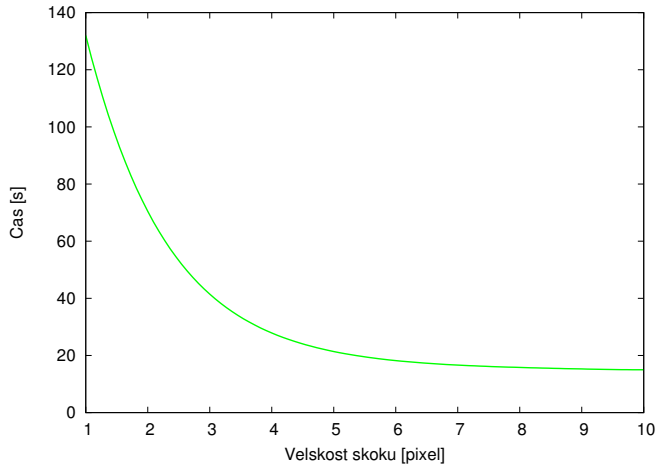


Obrázok A.4: *Test 2*: Graf závislosti úspešnosti nálezu, na zmene hodnoty prahu pri farebnej segmentácii.

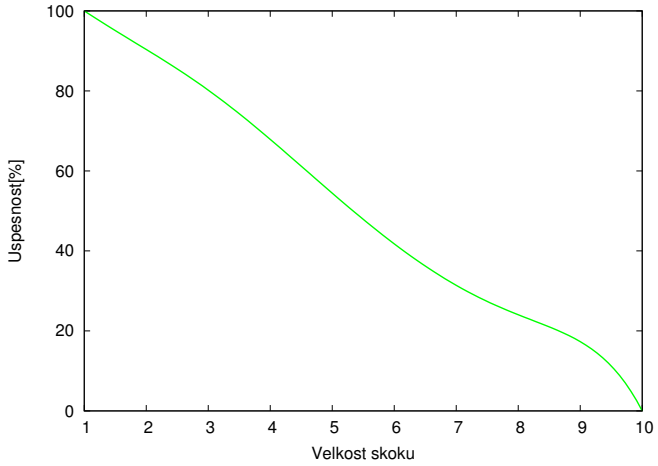
Obrázok A.5: *Test 2*: Ukážka funkčnosti aplikácie s nastaveným prahom farebnej segmentácie na hodnotu 185.



Obrázok A.6: *Test 2*: Ukážka samotnej farebnej segmentácie s prahom nastaveným na hodnotu 185.

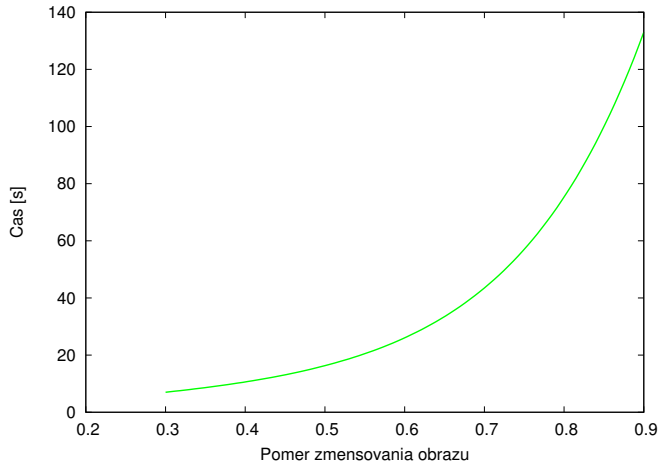


Obrázok A.7: *Test 3*: Graf závislosti času, na zmene hodnoty skoku pri pohybe okienka klasifikátora.

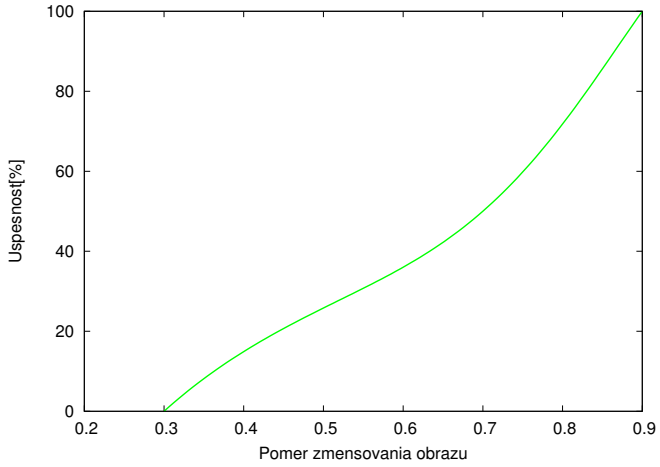


Obrázok A.8: *Test 3*: Graf závislosti úspešnosti nálezu, na zmene hodnoty skoku pri pohybe okienka klasifikátora.

Obrázok A.9: *Test 3*: Ukážka nálezu vľavo hodnota pohybu nastavená na 1 bod a vľavo na hodnotu 3 body.



Obrázok A.10: *Test 5*: Graf závislosti času, na zmene hodnoty ktorou bol zmenšovaní obraz.



Obrázok A.11: *Test 5*: Graf závislosti úspešnosti nálezu, na zmene hodnoty ktorou bol zmenšovaní obraz.

Obrázok A.12: *Test 5*: Ukážka aplikácie s rôzne nastavenými hodnotami pomeru ktorým bol zmenšovaní obraz. Vpravo s hodnotou 0.5 a vľavo s hodnotou 0.8.