SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1560

LOKALIZACIJA ODZIVA ALGORITMA VIOLE I JONESA UZ POMOĆ STROJA S POTPORNIM VEKTORIMA

Lucija Zadrija

Zagreb, srpanj 2010.

Sadržaj

Uvod

Razne županijske i državne službe vode brigu o održavanju prometnih znakova na kolnicima. Kako bi se posao pregledavanja stanja znakova automatizirao, koriste se razni algoritmi za detekciju objekata, pomoću kojih se na snimljenom materijalu detektiraju prometni znakovi.

Jedan od takvih algoritama je i algoritam Viole i Jonesa. Ovaj algoritam koristi kaskadu pojačanih Haarovih klasifikatora za detekciju objekata. Algoritam gotovo uvijek vraća višestruke odzive. Nedostatak heurističkog postupka grupiranja ovih odziva jest nepouzdana lokalizacija detektiranih objekata.

Kako bi se prometni znakovi uspješno probali lokalizirati, koristi se stroj s potpornim vektorima. Generiraju se binarni klasifikatori. Pomoću binarnih klasifikatora testiraju se odzivi algoritma u svrhu da se nađe najbolje klasificirani odziv. Takva detekcija uzima se za reprezentativni odziv.

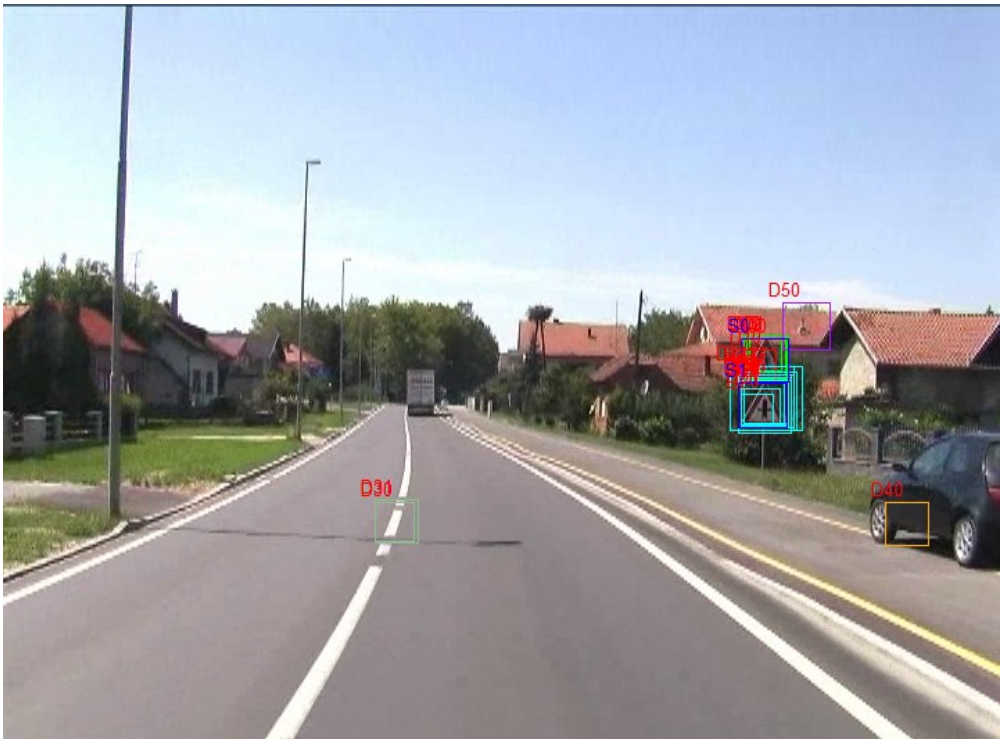
Rad je organiziran kako slijedi. Algoritam Viole i Jonesa za detekciju znakova i metoda grupiranja odziva, objašnjeni su u poglavlju 2, odjeljcima 2.1 i 2.2. U odjeljku 2.3 proučena je teorija stroja s potpornim vektorima. Nadalje, iznosi se na koji način je zadatak programski izveden (poglavlje 3). Dakle, objašnjen je postupak generiranja binarnog klasifikatora i kako su dobiveni testni podaci. U poglavlju 4 razmatraju se dobiveni eksperimentalni rezultati. Iznosi se učinkovitost primjene klasifikatora na odzive.

2. Korišten algoritmi, metode i izvorne slike

2.1. Algoritam Viole i Jonesa

Algoritam za detekciju objekata Viole i Jonesa je brza i robusna metoda koja koristi kaskadu jednostavnih Haarovih klasifikatora koji se pojačavaju (eng. *boost Haar classifier*). Ovaj algoritam za detekciju objekata je 15 puta brži od bilo kojeg ranije napravljenog algoritma za detekciju objekata [7]. Tipično se postiže točnost detekcije od 95%. Iako je algoritam primarno razvijen za detekciju ljudskih lica, zbog toga što koristi metode strojnog učenja može se koristiti i za detekciju bilo kakvih drugih objekata.

Ovdje se metoda koristi za detekciju trokutaskith prometnih znakova (Znakovi kategorije A) kao što se vidi na *Slici 2.1.1.*

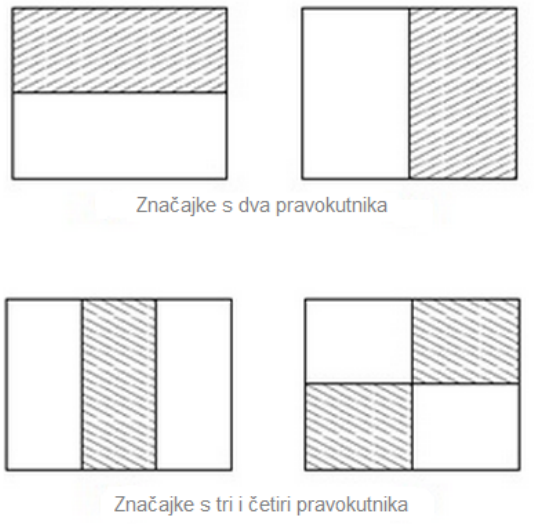


Slika 2.1.1: Primjer korištenja algoritma Viole i Jonesa za detekciju prometnih znakova

Postupak detekcije temelji se na prolasku detekcijskog okna kroz sliku. Veličina detekcijskog okna se mijenja i ono prolazi kroz sliku više puta. Prilikom prolaska okna kroz sliku na svakom položaju okna primjenjuje se klasifikator. Ako se u prozoru nalazi traženi objekt, klasifikator će javiti detekciju, u suprotnom ne će.

2.1.1. Osnovne Haarove značajke i integralna slika

Za detekciju objekata algoritam Viole i Jonesa koristi osnovne Haarove značajke. U osnovnoj varijanti postupka koriste se tri osnovna tipa značajki.



Slika 2.1.2: Osnovni tipovi Haarovih značaki

Značajke su pravokutne i sastoje se od više manjih pravokutnika. Tako postoje dvije značajke s dva pravokutnika, jedna s tri i jedna s četiri. Vrijednost značajki računa se kao razlika između sume piksela bijelih pravokutnika i sume piksela crnih pravokutnika [8]. Osnovni tipovi značajki prikazani su na *Slici 2.1.2.*

Izračun pravokutnih značajki je jednostavan, ali spor proces. Za postizanje velike brzine, algoritam Viole i Jonesa koristi posredni prikaz slike, odnosno, tzv. integralnu sliku (eng. *integral image*). Integralna slika na lokaciji x, y sadrži zbroj vrijednosti piksela slike lijevo i iznad promatranog elementa x, y (uključivo).

2.1.2. Generiranje pojačanih Haarovih klasifikatora

Dosad opisanim postupcima generira se tzv. slabi klasifikator (eng. *week classifier*) [7]. Kako bi se ubrzao proces detekcije, koristi se manji skup značajki za formiranje klasifikatora. Za odabir tog podskupa značajki koristi se koncept pojačanja. Koncept pojačanja (eng. *boosting*) je metoda koja iz skupa slabih klasifikatora gradi jedan snažan klasifikator (eng. *strong classifier*) [7]. *AdaBoost* [11] algoritam je jedna od metoda koja iz slabih klasifikatora gradi snažne. Zadatak *AdaBoost* algoritma je odabrati nekoliko stotina značajki i pridružiti im težine korištenjem skupa slika za treniranje. Algoritam za stvaranje snažnog klasifikatora prikazan je na *Slici 2.1.3*:

1. Formira se velik skup jednostavnih značajki
2. Postavljaju se težine uzoraka za treniranje
3. Za *T* koraka:
    1. Normaliziraju se težine
    2. Za dostupne značajke iz skupa značajki, trenira se klasifikator korištenjem jedne značajke i evaluira se pogreška treniranja
    3. Odabire se klasifikator s minimalnom pogreškom
    4. Ažuriraju se težine slika za tereniranje: povećavaju se ako je klasifikator pogrešno klasificirao sliku, smanjuju se ako je klasifikacija ispravna
4. Snažni klasifikator se dobiva kao linearna kombinacija *T* klasifikatora (koeficijent je veći ako je pogreška treniranja manja)

Slika 2.1.3: Postupak stvaranja snažnog klasifikatora

2.1.3. Stvaranje kaskade korištenjem pojačanih klasifikatora

Snažni klasifikator koji bi kvalitetno detektirao znakove, trebao bi biti sastavljen od velikog broja značajki. Računanje velikog broja značajki vremenski je vrlo zahtjevno te se stoga konstruira kaskada klasifikatora (eng. *cascade of classifiers*). Korištenjem kaskade klasifikatora povećava se postotak detekcije objekata i radikalno se smanjuje vrijeme izračuna [7]. Ključ je u tome da se konstruira manji i efikasniji pojačani klasifikator koji može odbaciti velik broj negativnih rezultata i pronaći skoro sve tražene objekte. Jednostavni klasifikatori koriste se za odbacivanje većine

prozora koji bi dali negativne razultate. Ako bilo koji od klasifikatora u procesu evaluacije vrati negativan rezultat, odmah se odbacuje prozor koji se trenutno razmatra. Nakon njih se pozivaju složeni klasifikatori koji se koriste za smanjivanje broja lažnih pozitiva.

2.2. Metoda grupiranja bliskih odziva

Algoritam Viole i Jonesa u većini slučajeva vraća više detekcija istog objekta [10], što je i prikazano na *Slici 2.1.1.* Također se može vidjeti da su ponekad te detekcije i lažne (prepoznate krošnje drveća, središnja traka kolnika i sl.). Ako se detekcije koje vraća algoritam dalje obrađuju, npr., u svrhu prepoznavanja tipa znakova koji su detektirani, nema smisla obrađivati sve detekcije, jer velika većina njih ne prikazuje dobro traženi objekt. Kako bi se riješio ovaj problem, predstavljena je u [10] metoda grupiranja bliskih odziva u jedinstveni odziv. Za dobivanje što boljeg rezultata koriste se postprocesiranje i heuristike. Cijela metoda detaljno je opisana u [10], stoga će ovdje biti navedena ideja i rezultati koje metoda daje.

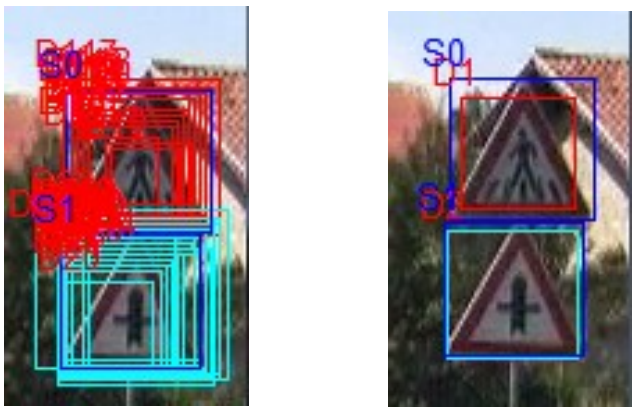
2.2.1. Osnovna ideja

Osnovna je ideja grupirati bliske odzive u zajednički skup i zatim izračunati odziv koji najbolje reprezentira odzive unutar skupa. Grupirani odzivi se međusobno preklapaju. Odziv koji najbolje reprezentira odzive računa se pomoću aritmetičke sredine svih odziva koji se nalaze u grupi.

Gornja lijeva i donja desna točka detekcije okružene su zamišljenim prozorom. Do preklapanja dolazi ako se gore navedene točke detekcije koja je površinom manja, nalaze unutar zamišljenog prozora površinom veće detekcije. Ti se odzivi grupiraju u jedan skup. Nakon što su svi odzivi obrađeni, oni koji su smješteni u jednu grupu, grupiraju se u jedan reprezentativni odziv.

Nakon grupiranja bliskih odziva još jednom se prolazi kroz grupirane odzive i grupiraju se dva po dva odziva. Ovo je implementirano kako bi se eliminirali slučajevi kada se nalaze grupirani odzivi jedan unutar drugog, gdje je ovaj prvi puno manji od drugog. Zbog toga se manji grupirani odziv koji se nalazi unutar većeg odziva

odbacuje jer je za očekivati da manji odziv ne predstavlja kvalitetnu detekciju označenog prometnog znaka. Grupirane detekcije i reprezentativni odzivi grupa mogu se vidjeti na *Slici 2.1.4.*



Slika 2.1.4. Prikaz grupa detekcija i reprezentativni odzivi prometnih znakova

2.2.2. Nedostaci

Kao što se vidi na *Slici 2.1.4.* grupiranje odziva u reprezentativne odzive ne daje uvijek najbolje rezultate. Na slici su oznake prikazane tamno-plavim pravokutnicima. Detekcije koje su grupirane u isti skup označene istom bojom (na *Slici 2.1.4,* grupe odziva prikazane su na desnoj slici crvenom i svijetlo-plavom bojom). Reprezentativni odziv donjeg znaka približno dobro prikazuje znak, iako je određeno odstupanje vidljivo. S druge strane, reprezentativni odziv gornjeg znaka ne prikazuje dobro znak i možemo reći da je detekcija nije zadovoljavajuća. Osim ovog primjera, u dosta slučajeva reprezentativni odziv ne opisuje dobro znak na način da se ne zaokruži desni ili lijevi dio znaka u slučaju da se znak nalazi pri kraju ekrana. U dosta primjera javljaju se lažni pozitivi, odnosno, zaokruži se i grupira nešto što nije znak.

Jedan način rješavanja ovog problema je primjena binarnog klasifikatora na sve odzive grupe i traženje najbolje klasificiranog odziva. Provjerava se koliko neka detekcija iz grupe bolje opisuje znak od reprezentativnog odziva grupe.

2.3. SVM – stroj s potpornim vektorima

Strojevi s potpornim vektorima (eng. *Support vector machines*) predstavljaju skup srodnih metoda nadziranog učenja (eng. *supervised learning*) koje se koriste za klasifikaciju i regresiju. Nadzirano učenje je tehnika strojnog učenja kojom se izračunava funkcija iz skupa podataka za treniranje. Dakle, svaki podatak ima oznaku koja označava pripadnost jednoj kategoriji, odnosno klasi. Za dani skup podataka za treniranje SVM algoritam za treniranje generira model na temelju kojega se određuje kojoj kategoriji pripada nezavisni testni podatak.

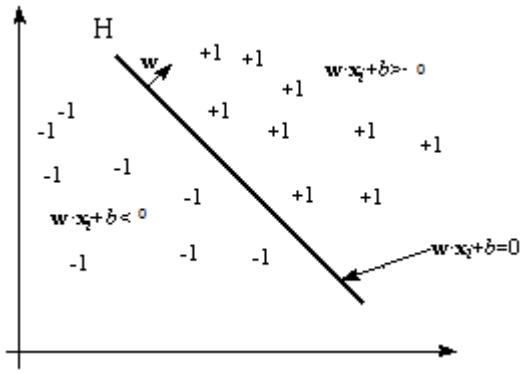
2.3.1. Opis problema i uvod u SVM

Na primjer, promotrimo klasifikaciju s dvije klase. Potrebno je dobiti preslikavanje $X \rightarrow Y$, gdje je $x \in X$ podatak, a $y \in Y$ oznaka kase. Dakle, $x \in R^n$, a $y \in \{\pm 1\}$. Primjerice, uzme se 50 slika krajolika u kakvom se inače nalaze prometni znakovi i 50 slika prometnog znaka npr. tipa A33 (znak opasnosti koji označava obilježen pješački prijelaz) kao što se vidi na *Slici 2.1.*

Slika 2.1: Primjeri znakova za treniranje

Slike se skaliraju na veličinu 24x24 piksela pa je $x \in R^n$ gdje je $n=576$. Ako se testira nekoliko drugih fotografija istog znaka A33, cilj je odgovoriti na pitanje radi li se o znaku ili slici krajolika? I ako se radi o znaku, koliko je pojedini znak „bolje klasificiran“ od drugog znaka? Drugim riječima, koliko je neki znak, bolje prepoznat kao znak, u usporedbi s ostalima?

Za dani skup podataka za treniranje u obliku vrijednost-oznaka (x_i,y_i), $i=1,\ldots,l$, gdje je $x_i \in R^n$ i $y \in \{\pm 1\}^l$, stroj s potpornim vektorima traži rješenje sljedećeg optimizirajućeg problema [3], što je prikazano na *Slici 2.2*:



Slika 2.2: Prikaz linearne odvajajuće ravnine

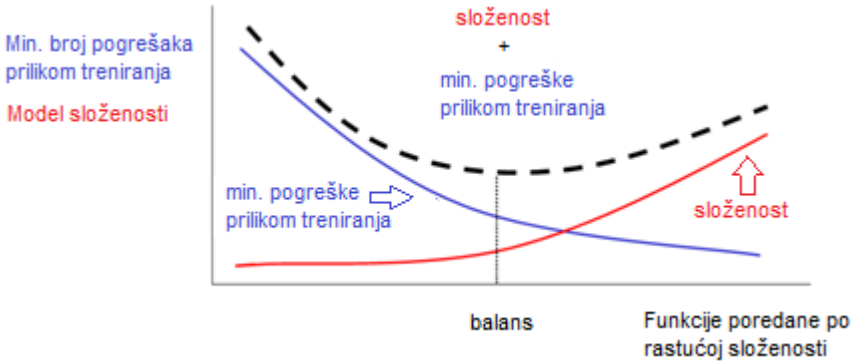
$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

$$\text{pod uvjetom:} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \qquad (1)$$
$$\xi_i \geq 0.$$

Ovdje se vektori x_i preslikavaju u viši (ponekad i beskonačan) dimenzionalni prostor pomoću funkcije ϕ. SVM pronalazi odvajajuću linearnu hiperravninu, s maksimalnom marginom u tom više-dimenzionalnom prostoru. Pritom je w normalizirani vektor okomit na hiperravninu, C faktor pogreške, ξ_i su tzv. *slack variables* koje mjere pogrešku u točki (x_i,y_i), a $\frac{b}{\|w\|}$ predstavlja pomak hiperravnine duž vektora w.

U idealnom slučaju, kada se podaci svrstavaju u npr. dvije klase, rezultat SVM analize trebala bi biti hiperravnina koja potpuno odvaja vektore značajki u dvije grupe koje se ne preklapaju [4]. Međutim, ponekad potpuno odvajanje nije moguće ili može rezultirati modelom s velikim brojem dimenzija vektora značajki. Ovo posljednje ima za posljedicu da model ne reagira dobro na nepoznate podatke, odnosno da klasifikacija nije zadovoljavajuća. Kako bi se dopustilo određeno odstupanje prilikom klasificiranja, postoji parametar pogreške C. Njime se osigurava da su, iako mora

postojati striktno definirana margina odvajanja, ipak dozvoljene određene pogreške prilikom treniranja, što se vidi na *Slici 2.3*.



Slika 2.3: Idealni omjer između dopuštenih pogrešaka i složenosti

Stvara se manje prodorna margina koja dopušta određene pogreške u klasificiranju. Međutim, povećavanjem vrijednosti parametra pogreške C, povećava se cijena pogrešnog klasificiranja podatka i forsira stvaranje što precizinijeg modela, što u konačnici može rezultirati modelom koji ne klasificira dobro nepoznate podatke. Kako bi se pronašao optimalan parametar pogreške, koriste se razne metode, jedna od metoda je i tzv. mrežno pretraživanje (eng. *grid search*). Ovom se metodom pretražuju vrijednosti traženog parametra unutar nekog raspona, tako da se povećavaju za neki određeni korak.

2.3.2. Odabir jezgre

U slučaju kad su podaci linearni, moguće ih je odvojiti odvojiti odgovarajućom hiperravninom. Međutim, u mnogim slučajevima, podaci nisu linearni i nemoguće ih je odvojiti. U tim slučajevim koriste se razne funkcije jezgri koje nelinearno preslikavaju podatke u višedimenzionalan prostor.

Postoje četiri osnovne jezgre koje koristi SVM:

i. Linearna $K(x_i, x_j) = x_i^T x_j$

ii. Polinomna $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0,$

iii. RBF (eng. *Radial Basis Function*) $K(x_i, x_j) = e^{-\gamma(\|x_i - x_j\|^2)}, \gamma > 0$

iv. Sigmoid $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Pri čemu su γ, r i d parametri jezgre.

Za potrebe ovog završnog rada, korištene su linearna i RBF jezgra. RBF jezgra je tzv. osnovna radijalna funkcija Gausovog tipa (eng. Gaussian *Radial Basis Function*). Ova funkcija ima jedan parametar γ koji kontrolira širinu „zvona“ Gausove funkcije. Ova jezgra nelinearno preslikava uzorke u više dimenzionalni prostor.

2.3.3. Unakrsna provjera valjanosti

Kao što je već spomenuto, kako bi se dobio optimalan parametar pogreške C, potrebno je provesti neku od metoda pretraživanja. Uz to, potrebno je pronaći i optimalnu širinu RBF funkcije, odnosno γ. Unakrsna provjera valjanosti (eng. *Cross-validation*) je metoda pomoću koje se određuju najbolje vrijednosti dotičnih parametara za dani zadatak.

v-unakrsna provjera valjanosti (eng. *v-fold cross-validation*) je jedan tip unakrsne provjere valjanosti. Dakle, prvo se skup podataka za treniranje podijeli na v podskupova jednake veličine. Zatim se redom svaki podskup testira korištenjem klasifikatora koji je treniran na preostalih v-1 podskupova. Tako je svaki dio (podskup) čitavog skupa za treniranje testiran jednom pa je točnost (eng. *accuracy*) unakrsne provjere valjanosti jednaka postotku podataka koji su točno klasificirani.

Kako bi se pronašli najbolje paramatre C i γ, isprobavaju se razne vrijednosti C i γ i odabiru se one za koje je točnost unakrsne provjere valjanosti bolja. Vrijednosti C i γ eksponencionalno se povećavaju ($C=2^{-5}, 2^{0}, 2^{5}, 2^{10}, 2^{15}$ i $\gamma=2^{-14}, 2^{-12}, \ldots, 2^{2}$) i provjerava se njihova točnost unakrsne provjere valjanosti. Ako se ustanovi da je točnost bolja za određen skup vrijednosti C i γ, preporuča se detaljnije ponoviti postupak unakrsne provjere valjanosti na tom skupu vrijednosti parametara kako bi se dobila bolja točnost.

2.4. Lokalizacija odziva algoritma Viole i Jonesa

Nakon odabira jezgre i pronalaska optimalnih parametara C i/ili γ, potrebno je izgenerirati klasifikator s tako odabranim parametrima. Takav klasifikator treba testirati na nezavisnim podacima da bi bio dobiven odgovor na pitanje s početka poglavlja, kojoj klasi pripada znak. Da bi se odgovorilo na pitanje koji je znak najbolje klasificiran potrebno je odrediti još nešto. Naime, prvo je potrebno provjeriti je li podatak ispravno klasificiran, znači, je li nakon testiranja uspješno određena oznaka klase podatka. Ako je ovaj uvjet zadovoljen na većini testnih podataka, hiperravina dobro odvaja podatke. Kako bi se odredilo koji podatak od onih koji su ispravno klasificirani je najbolje klasificiran potrebno, je odrediti vrijednost odluke (eng. *decision value*). Vrijednost odluke kod korištenja linearne jezge predstavlja udaljenost podataka od razdvajajuće hiperravnine.

Neka su testni podaci dani u obliku: $x_1,\ldots,x_l$. Vrijednosti odluke na temelju kojih su podaci klasificirani tada su $h(x_1),\ldots,h(x_l)$. Funkcija na temelju koje se dobivaju klase znakova je:

$$\mathrm{sgn}\big(h(x)\big) \tag{2}$$

Funkcija $h(x)$ vraća vrijednosti odluke na temelju kojih se određuje klasa podataka. Ako se koristi linearni SVM, funkcija $h(x)$ je:

$$h(x) = \sum_{i=1}^{l} w_i x_i + b \tag{3}$$

U slučaju da se koristi jezgra i podaci nisu linearno odvojivi, funkcija $h(x)$ je:

$$h(x) = w\phi(x) + b \tag{4}$$

Parametri jednadžba (7) i (8) objašnjeni su u potpoglavlju 2.1.3.

Znači, ako funkcija $h(x)$ vraća negativnu vrijednost, podatak će (zbog korištenja funkcije sgn) biti smješten u klasu –1. Način na koji se računaju ostali parametri i deteljnjije objašnjenje može se pronaći u [1].

Ako funkcija odluke vraća pozitivnu vrijednost, znači podatak je klasificiran kao znak. Onaj podatak koji je klasificiran kao znak i čija vrijednost odluke je najveća, najbolje je klasificiran. Potrebno je primjeniti utrenirani klasifikator na svaki testni podatak iz skupine te pogledati koji od njih je nakon testiranja ispravno klasificiran i ima najveću vrijednost odluke.

3. Programska izvedba

3.1. Priprema podataka za rad s bibliotekom LIBSVM

ImageMagick [2] je besplatni programski paket za obradu rasteriziranih slika. Izdaje se pod *Apache 2.0* licencom, odobren od udruge *OSI (Open Source Initiative).* Osim što služi za stvaranje i uređivanje slika, može i čitati, pretvarati i zapisivati slike u različitim formatima (preko sto različitih formata), među kojima su i *gif, jpeg, png, Postscript, svg i tiff*.

Postoje brojna sučelja iz drugih programskih jezika prema *ImageMagick-u.* Neka od njih su*: Magick++ (C++), JMagick (Java), PythonMagick (Python), PerlMagick (Perl)* itd. Za potrebe ovog završnog rada korišteno je sučelje *PerlMagick* za pretvorbu slika u vektore značajki, kao što će biti objašnjeno kasnije.

Kako bi bilo moguće izgraditi binarni klasifikator na temelju kojeg će se određivati najbolja detekcija u algoritmu Viole i Jonesa*,* potrebno je pripremiti ulazne podatke. Iz skupa slika za učenje potrebno je isjeći znakove koje želimo klasificirati. To se čini uz pomoć *.dat* datoteke (generiranje te datoteke opisano je u [10]) koja sadrži popis svih znakova na svim slikama, broj znakova na pojedinoj slici, koordinate i dimenzije znakova. Zatim se svi tako dobiveni znakovi skaliraju na 24x24 piksela. Skripta *positiveExamples.sh* koja izvodi prethodno opisani postupak nalazi se u Dodatku A.

Sličan postupak ponavlja se pri pribavljanju slika pozadine-negativa (slike koje ne prikazuju znakove). Iz slika pozadine se isječe slučajan broj slika sa slučajno odabranih koordinata i skalira se na 24x24 piksela. Manipulaciju pozadinama izvodi skripta *negativeExamples.sh* koja se nalazi u Dodatku A.

Tako dobivene slike pozadine i znakova pretvaraju se u vektore značajki i predaju binarnom klasifikatoru. Za manipulaciju slikama korištena je *ImageMagick* naredba `convert` te opcije `-extract` (izdvaja zadani dio iz slike) i `-resize` (mijenja veličinu slike na zadanu).

3.2. Rad s bibliotekom libSVM

LIBSVM [1] (A Library for Support Vector Machines) je besplatna biblioteka za rad s potpornim vektorima. To otvorena programska podrška za klasifikaciju potpornih vektora (C-klasifikacija potpornih vektora, *C-SVC,* v-klasifikacija potpornih vektora, *v-SVC*), regresiju (ε-regresija potpornih vektora*, ε-SVR, v*-regresija potpornih vektora*, v-SVR*) i procjenu raspodjele (*one-class SVM*). Podržana je klasifikacija potpornih vektora u više razreda. Detaljne upute za rad s ovom bibliotekom nalaze se u README datoteci bibiloteke.

LIBSVM bibilioteka je implementirana u nekoliko programskih jezika, kao što su *C++, Java, Python, Matlab* i drugi. Za izgradnju binarnog klasifikatora i testiranje korištena je *C++* implementacija. Postavljanje postavki za omogućavanje paralelizacije korištenjem *OpenMP*-a [12] opisano je u Dodatku B.

Jednostavan postupak klasifikacije potpornih vektora koji u ovom slučaju daje zadovoljavajuće rezultate, opisan je u sljedećih nekoliko koraka:

i. Pretvaranje ulazne podatke u format koji prihvaća *LIBSVM* biblioteka
ii. Skaliranje ulaznih podataka
iii. Izbor jezgre
iv. Pronalazak najboljih parametara γ i/ili C
v. Treniranje korištenjem najboljih parametara γ i/ii C
vi. Testiranje dobivenog klasifikatora

3.2.1. Priprema ulaznih podataka

LIBSVM zahtjeva da svaki podatak koji se koristi (npr. slika koju je potrebno klasificirati) bude predstavljen u obliku vektora realnih brojeva. Vektor realnih brojeva za svaki podatak treba izgledati ovako:

```
<oznaka> <indeks1>:<vrijednost1> <indeks2>:<vrijednost2> ...
  . . .
```

- `<oznaka>` je cijeli broj koji označava klasu kojoj pripada podatak ako se vrši klasifikacija. Koristi se izračun točnosti ili pogreške. Za slučaj kada su podaci razmješteni u deset različitih klasa i podatak pripada klasi s oznakom 4, umjesto `<oznaka>` stajat će broj 4. U ovom slučaju, slika koja se klasificira može imati oznaku 1 ili -1, ovisno o tome radi li se o pozadini ili znaku. Ako se radi o regresiji, `<oznaka>` može biti bilo koji realni broj, a u slučaju procjene raspodjele `<oznaka>` se ne koristi i može biti bilo koji broj.
- `<indeks1>` može biti bilo koji redni broj, počevši od jedan. U ovom slučaju predstavlja broj piksela u slici. Indeksi moraju biti postavljeni uzlaznim redosljedom.
- `<vrijednost1>` može biti bilo koji realni broj i predstavlja vrijednost značajki, odosno pojedninog piksela. U ovom slučaju, radi se o RGB (eng. *red, green* i *blue*) vrijednostima određenog piksela. Ako broj vrijednosti nije prevelik, istraživanja pokazuju da se bolji rezultati dobiju ako se svaki piksel prikazuje s tri vrijednosti, odnosno svaka je vrijednost jedna RBG komponenta.

Primjer vrijednosti prvih pet značajki za prvi znak:

```
1 1:0.183594 2:0.257813 3:0.238281 4:0.179688 5:0.238281 . . .
```

Za treniranje klasifikatora korišteno je 2048 slika znakova veličine 24x24 piksela i 3749 *pozadina* (slika na kojima se ne nalaze znakovi) također veličine 24x24 piksela. Generirana datoteka s vektorima značajki svih slika je veličine 130MB. Provjeru sadržaja generirane datoteke s ulaznim podacima vrši *LIBSVM* skripta `checkdata.py` koja se nalazi u direktoriju *tools* biblioteke *LIBSVM*. Skripta prima kao argument datoteku s ulaznim podacima i ispisuje pogrešku ukoliko podaci nisu u zadanom formatu. Za pretvorbu slike u vektore značajki može se koristiti funkcija `GetPixels` *ImageMagick* sučelja *PerlMagick* [2] pomoću kojih se mogu dobiti RBG vrijednosti pojedinog piksela slike. Osim ovog alata, za dani problem moguće je koristiti i program napisan u programskom jeziku *Java*, korištenjem metode `getRGB`. Na isti način generiraju se datoteke za testiranje.

3.2.2. Skaliranje ulaznih podataka

Kao što je već navedeno, za izgradnju binarnog klasifikatora koristi se program `svm-train`. Kada bi se programu `svm-train` predala prethodno generirana datoteka te se na tako izgenerirani klasifikator primjenilo testiranje, rezultati bi bili poprilično nezadovoljavajući, kao što se vidi u [3]. Razlog tome je činjenica da bi vrijednosti u većem numeričkom rasponu dominirale onima u manjem numeričkom rasponu. Zato se prije samog treniranja mora provesti skaliranje ulaznih podataka. Skalirani podaci najčešće su u rasponu [-1,1] ili [0,1]. Na isti način na koji su skalirani podaci za izgradnju klasifikatora, moraju biti skalirani i testni podaci. U ovom su slučaju skalirani podaci u rasponu [-1,1]. Prednost skaliranja je i u tome da se skaliranjem izbjegavaju poteškoće pri računanju s velikim brojevima. Vrijednosti funkcija jezgri dobivaju se, na primjer, kod polinomne ili linearne jezgre, kao produkt ulaznih vrijednosti, a izračuni sa skaliranim podacima jednostavniji su od računanja s velikim brojevima. Skaliranje se vrši na sljedeći način. Ako su `xmax` i `xmin` maksimalna i minimalna vrijednost i-tog piksela, skaliranje na [-1,1] se vrši ovako:

$$x' = 2\frac{(x-x_{min})}{(x_{max}-x_{min})} - 1 \tag{5}$$

a skaliranje na [0,1]:

$$x'' = \frac{(x-x_{min})}{(x_{max}-x_{min})} \tag{6}$$

U slučaju korištenja RBF jezgre, skaliranje se vrši na slijedeći način:

$$x' - y' = \frac{(x-y)}{(x_{max}-x_{min})}, x'' - y'' = 2\frac{(x-y)}{(x_{max}-x_{min})} \tag{7}$$

U slučaju brojnih nula u ulaznim podacima. vrijeme izračuna kod skaliranja na [0,1] razlikuje se od onoga kod skaliranja na [-1,1]. Naime, skaliranje na [0, 1] osigurava oskudnost ulaznih podataka i tako štedi vrijeme

Za skaliranje podataka koristi se *LIBSVM* program `svm-scale` biblioteke *LIBSVM*. Program vraća skalirane ulazne podatke.

Primjer poziva programa `svm-scale`:

```
./svm-scale -l -1 -u 1 -s train.dat.range train.dat > train.dat.scale
```

Znači, ulazne vrijednosti koje se nalaze u datoteci `train.dat` skaliraju se na [-1,1]. Parametri skaliranja spremaju se u datoteku `train.dat.range`, a skalirani podaci u `train.dat.scale`.

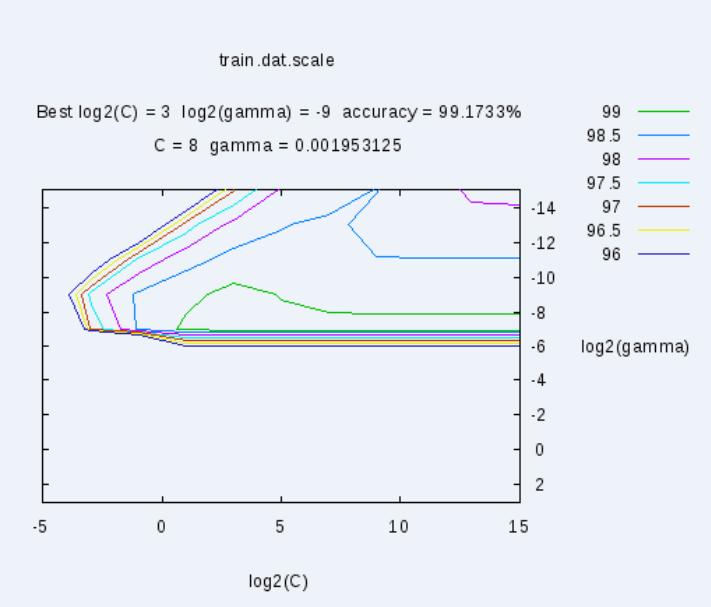
3.2.3. Odabir jezgre i izvođenje unakrsne provjere valjanosti

Autori biblioteke *LIBSVM* početnicima preporučuju korištenje RBF jezgre prilikom treniranja klasifikatora. Mnogo je razloga za to i detaljno su opisani u [1]. Jedan od razloga je i taj da linearna jezgra s parametrom pogreške C daje iste rezultate kao i RBF jezgra s odgovarajućim parametrima (C i γ). Također je bitno spomenuti da *LIBSVM* tretira linearne i ne-linearne SVM-ove na isti način. Iako je korištenjem linearne jezgre moguća ušteda vremena prilikom testiranja i treniranja, *LIBSVM* nije pretjerano učinkovit kada se koristi linearna jezgra. To je posebno izraženo kada je parametar pogreške velik i broj podataka za treniranje puno veći od broja pripadajućih atributa.

Međutim, u ovom i sličnim slučajevima kada je broj značajaka velik, preslikavanje ulaznih podataka u više-dimanzionalan prostor ne poboljšava uspješnost izvođenja. U tim sličajevima, korištenje linearne jezgre daje dovoljno dobre rezultate i samo je potrebno pronaći odgovarajući parametar pogreške C. Korištenje RBF jezgre i u ovom slučaju daje zadovoljavajuće rezultate, no tek nakon što se pronađu dovoljno dobri parametri C i γ. Kako bi se pokrenulo tretniranje korištenjem linearne jezgre, potrebno je programu `svm-train` predati kao parametar –t vrijednost 0. Ako se pak želi utrenirati klasifikator korištenjem RBF jezgre, isti parametar se postavlja na vrijednost 2.

Programu `svm-train` predaju se kao argumenti najbolje vrijednosti parametara C i/ili γ. Ako se koristi RBF jezgra, traže se najbolji parametri C i γ, u protivnom, ako se korisit linearna jezgra, traži se samo parametar C. Skripta `grid.py`

koja izvodi unakrsnu provjeru valjanosti na način opisan u odjeljku 2.3.3., nalazi se u *LIBSVM* direktoriju *tools*.



Slika 3.1: Graf koji se dobije kao rezultat unakrsne provjere valjanosti korištenjem skripte `grid.py` za RBF jezgru

Skipta kao rezultat vraća graf koji prikazuje točnost unakrsne provjere valjanosti za pojedine parametre C i γ i *.out* datoteku koja prikazuje točnost unakrsne provjere valjanosti za svaku točku na grafu s koordinatama ($\log_2 C$, $\log_2 \gamma$).

Kao što se vidi na grafu koji je prikazan na *Slici 3.1*, najbolji $C=2^3=8$ i $\gamma=2^{-9}=0.001953125$, a točnost za te parametre iznosi 99.1733%. Najbolje vrijednosti na grafu su iscrtane zelenom bojom. Ovi podaci dobiveni su kada se koristi RBF jezgra. U slučaju da se koristi linearna jezgra, $C=0.03125$, a točnost iznosi 98.3294%. Skripta `grid.py` prima prethodno skalirani skup ulaznih podataka. Primjer poziva skripte `grid.py` za slučaj da se traže najbolji C i γ :

```
python grid.py -log2c -5,15,1 -log2g -14,2,1 -v 5 train.dat.scale
```

Prvi argument je raspon parametra C i korak za koliko se povećava eksponent, drugi parametar prima iste podatke kao i prvi, samo za vrijednost γ. Parametar -v označava o kakvoj se unakrsnoj provjeri radi.

3.2.4. Treniranje

Nakon odabira najboljih parametara C i γ, te se vrijednosti predaju programu svm-train koji vraća binarni klasifikator SVM model na temelju ulazne datoteke i parametara. Kao ulazna datoteka predaje se datoteka sa skaliranim podacima, a izlazna datoteka *.model* predstavlja binarni klasifikator. U slučaju korištenja linearne jezgre, u programu svm-train potrebno je u programu parse_command_line promjeniti vrijednost varijable param.kernel_type iz RBF u LINEAR.

Primjer poziva programa svm-train ako se koristi RBF jezgra:

```
./svm-train -c 8 -g 0.001953125 train.dat.scale
```

Prva dva argumenta primaju vrijednosti parametara C i γ. Poziv je isti u slučaju korištenja linearne jezgre, samo se ne predaje vrijednost parametra -g.

Sve gore navedene postupke skaliranja ulaznih podataka, unakrsne provjere valjanosti, treniranja i testiranje moguće je napraviti odjednom, koristeći automatsku skriptu easy.py koja se nalazi u direktoriju *tools*. U ovom završnom radu korištena je ta skripta za treniranje binarnog klasifikatora, a testiranje je napravljeno odvojeno, programom svm-predict. Primjer poziva easy.py koja izvodi skaliranje, unakrsnu provjeru valjanosti i treniranje:

```
python easy.py train.dat
```

Skripta se izvodila na računalu s 8 jezgara te je prije pokretanja bilo potrebno postaviti sljedeću varijablu okoline: export OMP_NUM_THREADS=8. Na taj način se postavlja broj dretvi (u ovom slučaju 8) koje skripta koristi prilikom izvođenja. Specifikacije računala: HP ProLiant ML370 G5, 8GB RAM, 64bitni Ubuntu 10.04, a procesori su Xeon E5320 @ 1.86GHz (2x4-way). Izvođenje ovako pokrenute skripte s

ulaznom datotekom koja sadrži podatke o 5797 slika, s uključenom *OpenMP* [12] paralelizacijom i korištenjem RBF jezgre trajalo je oko 24 sata, a potrošnja je bila oko 750% jedne jezgre. U slučaju linearnog SVM-a, izvođenje skripte trajalo je oko tri sata. Rezultat izvođenja skripte je između ostalog i binarni klasifikator `train.dat.model`. Sadržaj prvih nekoliko redova datoteke (pri korištenju RBF jezgre) uz objašnjenja nalazi se u *Tablici 3.1.*

Tablica 3.1.: Sadržaj i objašnjenje prvih nekoliko redaka klasifikatora

Ispis	Objašnjenje ispisa
svm_type c_svc	radi se o C-klasifikaciji potpornih vektora
kernel_type rbf	tip jezgre je RBF
gamma 0.00195312	vrijednost prametra γ=0.00195312
nr_class 2	broj klasa je dva
total_sv 639	ukupan broj potpornih vektora 639
rho -0.180067	rho=-0.18006 i predstavlja izraz -b u funkciji odluke sgn(w^Tx + b)
label 1 -1	korištene oznake su 1 i -1
nr_sv 275 364	Broj potpornih vektora jedne klase je 275, a druge 364

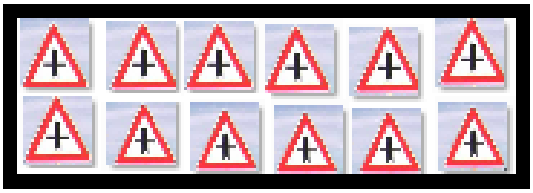
Ostatak `train.dat.model` datoteke prikazuje potporne vektore, točnije, svaki redak predstavlja jedan potporni vektor. Potporni vektori su sortirani s obzirom na oznake klasa zadane u ulaznoj datoteci za treniranje. Ako je `k` ukupan broj klasa (u našem slučaju 2), ispred potpornog vektora u klasi `j` nalazi se `k-1` koeficijenata `y*`α. Gdje su α dvostruka rješenja sljedećeg problema dviju klasa:

```
1 vs j, 2 vs j, ..., j-1 vs j, j vs j+1, j vs j+2, ..., j vs k
```

`Y=1` kod prvih `j-1` koeficijenata, a `y=-1` kod preostalih `k-j` koeficijenata.

3.2.5. Testiranje

Algoritam Viole i Jonesa primjenjuje se na slike s označenim znakovima. Kao rezultat se dobivaju grupe detekcija. Primjer grupe skaliranih detekcija može se vidjeti na *Slici 3.2*.



Slika 3.2. Primjer grupe znakova iz koje je potrebno izdvojiti najbolje klasificirani

Ona grupa kojoj pripada najmanja udaljenost značajke od oznake uzima se za testiranje klasifikatora. Testira se svaka detekcija u grupi. Nakon testiranja treba provjeriti jesu li sve detekcije ispravno klasificirane (znači klasificirane kao znak), koja od njih je najbolje klasificirana i koja je točnost (eng. *accuracy*) klasificiranja. Priprema ulaznih podataka za treniranje ista je kao i priprema ulaznih podataka za testiranje. Dakle, generira se po jedna ulazna datoteka za svaku detekciju u traženoj grupi. Dobivene ulazne datoteke potrebno je skalirati programom `svm-scale` istim parametrima kao i ulaznu datoteku za treniranje.

Nakon skaliranja potrebno je pokrenuti program `svm-predict` kojim se primjenjuje binarni klasifikator na skup testnih podataka. Kako bi se odredilo koja je detekcija najbolje klasificirana, potrebno je modificirati program `svm.cpp`. Tim modifikacijama dobila bi se vrijednost odluke (eng. *decision value*). Ona detekcija koja je ispravno klasificirana i ima najveću vrijednost odluke, uzima se kao najbolja u grupi. Da bi pronašli vrijednosti odluke, potrebno je modificirati program `svm.cpp`. Dio koda koji omogućuje ispis vrijednosti odluka nalazi se u Dodatku C.

Rezultat testiranja je (ako se vrši klasifikacija) datoteka `.predict` u kojoj se nalaze oznake klasa kojima pripadaju testni podaci. Kao izlaz se također dobiva i točnost klasifikacije (eng. *classification accuracy*).

$$\text{Točnost} = \frac{(\#\text{točno klasificirani podaci})}{(\#\text{svi podaci})} \times 100\% \qquad (8)$$

3.3. Programska izvedba lokalizacije odziva algoritma Viole i Jonesa

Algoritam Viole i Jonesa za detekciju objekata, u ovom slučaju prometnih znakova, integriran je unutar ljuske *cvsh2*. Instalacija, pokretanje i postavljanje postavki detaljno je opisano u [10].

Kao što je već navedeno u poglavlju 2.2. algoritam Viole i Jonesa gotovo uvijek vraća višestruke odazive. Kako se grupiranjem detekcija i računanjem reprezentativnog odziva ne dobivaju odgovarajući rezultati, potrebno je razmotriti drugi pristup. Kao što je već navedeno, potrebno je razmotriti primjenu binarnog klasifikatora na grupu detekcija znakova. Program koji se bavi rješavanjem navedenog programa zove se `lzadrija2` i može se pokrenuti kao što je to objašnjeno u [10] s tom izmjenom da se za opciju `-a` stavi `lzadrija2`.

Ideja je sljedeća. Potrebno je sve odzive grupirati na način na koji je to objašnjeno u [10]. Grupiranje odziva vrši se pozivom funkcije:

```
ext_vj_groupOnly(vecDetsRaw, vecDetGroups);
```

Pri čemu se u vektoru `vecDetsRaw` nalaze koordinate svih detekcija koje je za danu sliku vratio algoritam Viole i Jonesa. Ova funkcija sprema u vektor `vecDetGroups` grupirane detekcije.

Kako bi se odredila najmanja udaljenost neke značajke od označenog znaka, za svaku detekciju se računa udaljenost značajke (eng. *feature distance*) i vrijednosti se spremaju u matricu udaljenosti `fd2SeqDet`. Za svaku oznaku prometnog znaka traži se najmanja udaljenost značajke. Udaljenost značajke računa se pomoću funkcije `featureDistance`. Ova funkcija vraća rezultat funkcije:

$$(dx \cdot dx \cdot dy \cdot dy) \cdot e^{\left(\left(\frac{maxmagx}{minmagx}-1\right)+\left(\frac{maxmagy}{minmagy}-1\right)\right)} \quad (9)$$

Prvo je potrebno izračunati aritmetičke sredine x i y koordinata oznake i detekcije. Na taj se način dobiju koordinate slike koja je aritmetička sredina između oznake i

detekcije. Također je potrebno odrediti manju vrijednost između širina i visina oznake ili detekcije – `minmagx` i `minmagy`. Potrebno je odrediti i veću vrijednost širina i visina između oznake i detekcije – `maxmagx` i `maxmagy`. Da bi se dobila vrijednost `dx` računa se širina slike koja je dobivena kao aritmetička sredina i dijeli se s vrijednošću `minmagx`. Na isti na način se dačuna `dy`, samo što se računa razlika između visina i dijeli se s vrijednošću `minmagy`.

Ovaj program nudi dvije mogućnosti. Dakle, moguće je spremanje slika „najbolje“ grupe detekcija da disk kako bi se koristile kao testni podaci za *LIBSVM*. Pod „najbolja“ grupa misli se na onu grupu detekcija koja sadrži detekciju koja je minimalno udaljena od označenog prometnog znaka. Dakle, potrebno je pronaći koja grupa sadrži detekciju s minimalnom udaljenosti značajke i spremiti za svaki označeni znak tu grupu detekcija na disk. Uz detekcije se spremaju i označeni znakovi. Svaki označeni znak sprema se u zasebni direktorij, dok se u tom direktoriju stvara posebni direktorij u koji se spremaju „najbolje“ grupe detekcija. Detekcije se prije spremanja moraju izrezati korištenjem funkcije `cutImage` i skalirati korištenjem funkcije `rescale`.

Druga mogućnost koja se nudi jest prikaz najbolje klasificirane detekcije iz „najbolje“ grupe detekcija. Znači, nakon što se primjeni klasifikator na svaku detekciju iz grupe i dobije se vrijednost odluke, pronalazi se detekcija s najvećom vrijednosti odluke u grupi i ona se prikazuje kao reprezentativni odziv grupe. Prije provjere vrijednosti odluke, potrebno je provjeriti je li detekcija ispravno klasificirana. Ako je detekcija klasificirana u klasu -1, znači da je pogrešno klasificirana i odbacuje se. U slučaju da je detekcija smještena u klasu 1, znači da je ispravno klasificirana i provjerava se njena vrijednost odluke. Ona detekcija koja ima najveću vrijednost odluke, uzima se za reprezentativni odziv grupe. Također se prikazuje detekcija nastala metodom grupiranja [10] detekcija iz „najbolje“ grupe detekcija. Grupiranje se izvodi funkcijom `ext_vj_grouping`. Za svaku tako izračunatu detekciju računa se udaljenost te detekcije od detekcije koju je odabrao klasifikator. Udaljednost detekcije računa se na gore opisan način i može se dobiti funkcijom `featureDistance` .

4. Eksperimentalni rezultati

Eksperimentalni rezultati prikazani su u nastavku. Na *Slici 4.1.* prikazane su grupe odziva za znak A03 (lijeva slika) i konačne detekcije (desna slika).

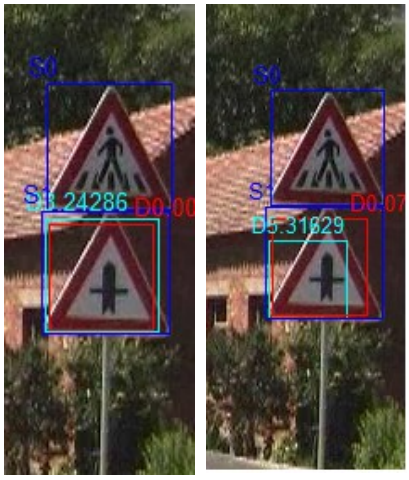
Slika 4.1: Prikaz grupe detekcija i reprezentativnih odziva odabranih primjenom binarnih klasifikatora s RBF i linearnom jezgrom i reprezentativnih odziva nastalih metodom grupiranja [10]

Ove detekcije nastale su kao rezultat primjene binarnih klasifikatora na „najbolju“ grupu odziva, kao što je opisano u poglavlju 3.1 i primjenom metode grupiranja, kao što je detaljno opisano u [10]. Reprezentativni odziv grupe nastao primjenom klasifikatora označen je svijetlo-plavom bojom i iznad okvira zapisana je vrijednost odluke. Uokvirena je ona detekcija iz grupe koja ima najveću vrijednost odluke (u ovom slučaju najveća vrijednost pri korištenju RBF jezgre je 3.54823, a pri korištenju linearne jezgre je 3.62766), kao najbolje klasificirana. Detekcija nastala metodom grupiranja označena je crvenom bojom. Iznad te detekcije zapisana je udaljenost detekcije nastale primjenom klasifikatora od te detekcije. Ako se koristi RBF jezgra ta vrijednost iznosi 0.00420464, a ako se koristi linearna jezgra iznos je 0.271832. Znak koji se traži označen je tamno-plavom bojom. U slučaju dva označena znaka na slici, traži se samo jedan. Drugi označeni znak traži se na drugoj slici. Detekcija nastala primjenom klasifikatora s RBF jezgrom označena na *Slici 4.1.* vrlo dobro predstavlja označeni znak, iako je vidljivo malo odstupanje od samog

znaka. S druge strane, detekcija nastala primjenom klasifikatora s linearnom jezgrom ne prikazuje dobro znak i odstupanje od oznake je više nego očito. Još jedan primjer prikazan je na *Slici 4.2.* Desna slika prikazuje detekciju nastalu primjenom klasifikatora s RBF jezgrom, a lijeva s linearnom jezgrom. Odabrana detekcija u oba slučaja ima najveću vrijednost odluke i ona iznosi za prvi slučaj 2.69021, a za drugi 3.9798. U oba slučaja korištenje klasifikatora bolje lokalizira znak od metode grupiranja. Odstupanje između detekcija u prvom slučaju iznosi 0.000958092, a u drugom 0.000191618.

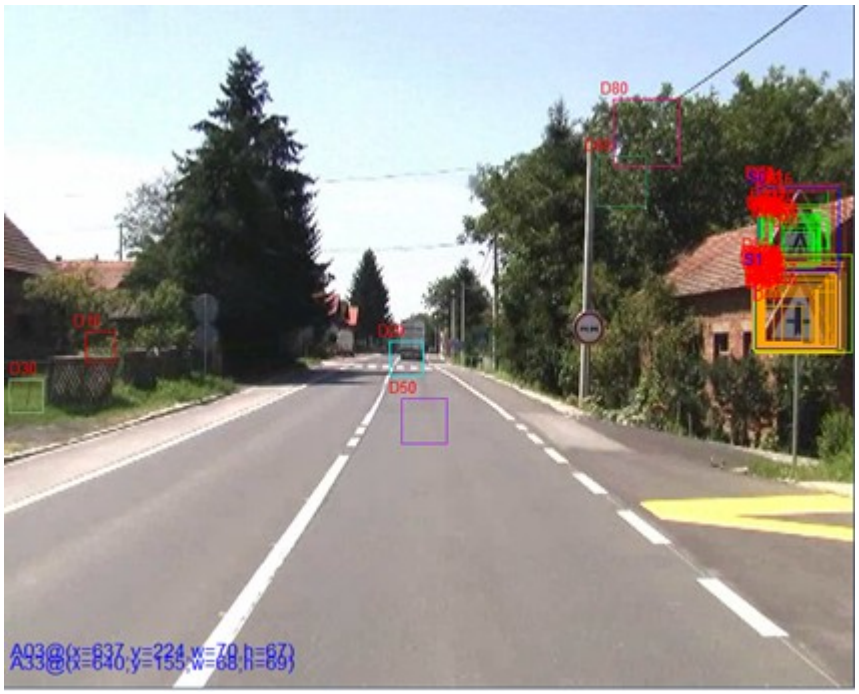
Slika 4.2: Prikaz lokalizacije prometnog znaka korištenjem klasifikatora s RBF i linearnom jezgrom

U večini slučajeva, kada se rezultati nalaze uz rub slike, rezultati nisu zadovoljavajući, bez obzira koji klasifikator se primjenjuje. Ovaj slučaj može se vidjeti na *Slici 4.3.* U večini slučajeva gdje se označeni znak nalazi uz rub ekrana, odabrana detekcija ne zaokružuje cijeli znak. Jedan dio površine na desnom djelu znaka nije zaokružen. Na *Slici 4.3.* detekcija odabrana klasifikatorom s RBF jezgrom bolje lokalizira znak od metode grupiranja, no još uvijek nedovoljno dobro. U slučaju korištenja klasifikatora s linearnom jezgrom rezultati su znatno lošiji i odziv nastao metodom grupiranja bolje opisuje znak.



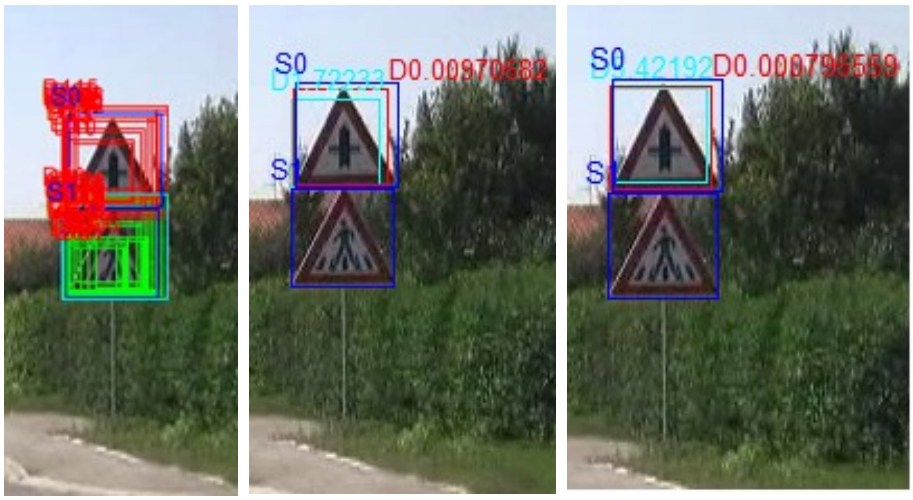
Slika 4.3: Prikaz neuspješne lokalizacije

Na *Slici 4.4.* može se vidjeti grupa odziva kojoj pripada detekcija sa *Slike 4.3.*, radi se o grupi gdje su oznake žute boje. Iz slike se vidi da većina oznaka iz grupe ne uokviruje dobro znak (oznake pretežno odsjecaju desni dio znaka). Kako u grupi nema odziva koji bi dovoljno dobro predstavio traženi znak, ni primjena klasifikatora ne može puno pomoći u potrazi za dobrom detekcijom. Unatoč tome, lokalizacija klasifikatorom s RBF jezgrom (desno) pokazala se boljom od metode grupiranja.



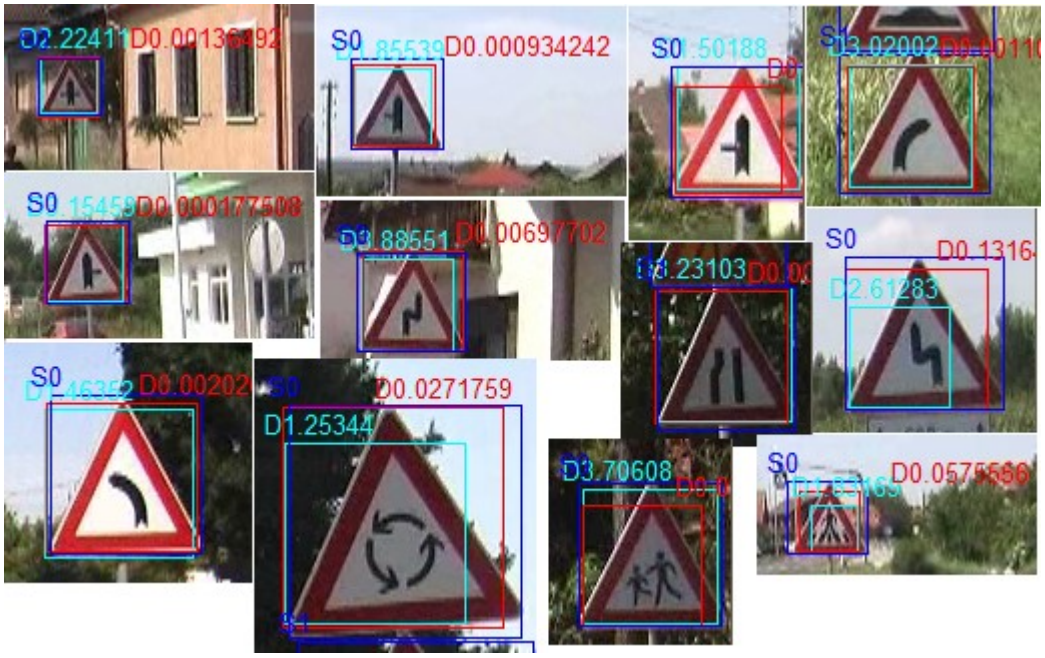
Slika 4.4: Prikaz grupe odziva iz koje je odabrana detekcija sa Slike 4.3.

Na *Slici 4.5* (sredina i lijeva strana) vidi se odstupanje reprezentativnog odziva od detekcije. U ovom slučaju, znak se ne nalazi uz rub ekrana. Na desnoj slici je prikazana grupa odziva iz koje su uzete detekcije koje se nalaze na desnoj slici i slici u sredini. Odzivi iz grupe prikazani su crvenom bojom na lijevoj slici. Sa slike se vidi da postoje detekcije koje bi možda bolje prikazale označeni znak. U slučaju korištenja RBF jezgre metoda grupiranja bolje lokalizira traženi znak, odnosno, zaokružuje se veća površina znaka. S druge strane, odziv nastao korištenjem klasifikatora s linearnom jezgrom bolje zaokružuje znak od odziva nastalog primjenom klasifikatora a RBF jezgrom.

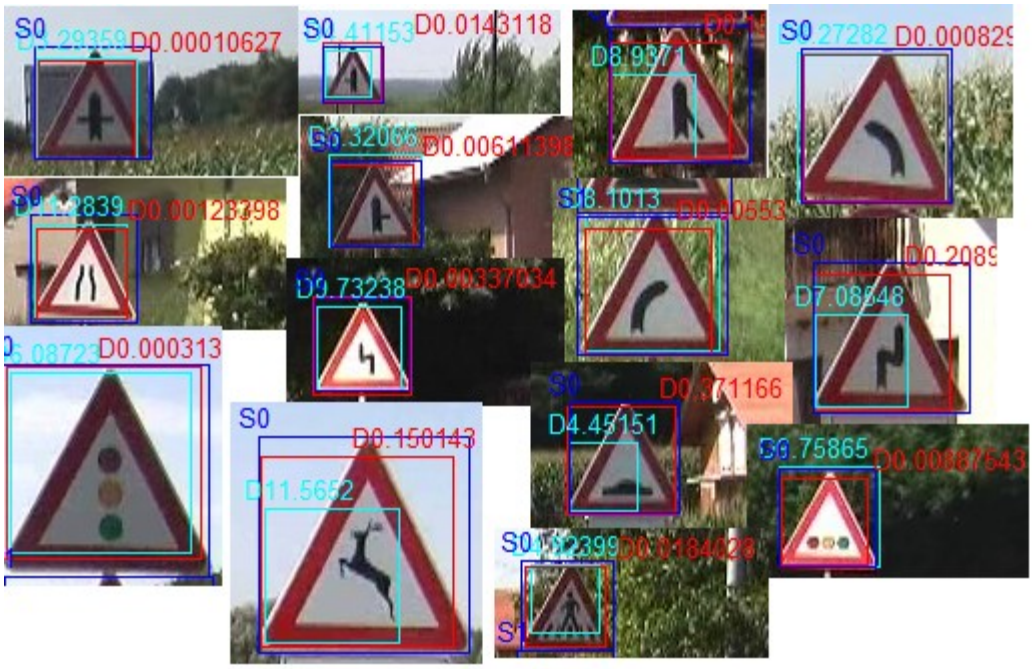


Slika 4.5: Prikaz slučaja gdje je lokalizacija znaka uspješnija korištenjem metode grupiranja

Na *Slici 4.6.* prikazani su razni odzivi, slučajevi gdje se metoda grupiranja pokazala kao bolji izbor za lokalizaciju prometnog znaka te primjeri gdje odziv odabran klasifikatorom bolje lokalizira znak od odziva nastalog metodom grupiranja. Ovi odzivi nastali su primjenom klasifikatora s RBF jezgrom, dok su odzivi nastali primjenom klasifikatora s linearnom jezgrom prikazani na *Slici 4.7.*



Slika 4.6: Prikaz raznih odziva korištenjem klasifikatora s RBF jezgrom



Slika 4.7: Prikaz raznih odziva korištenjem klasifikatora s linearnom jezgrom

5. Zaključak

Cilj ovog završnog rada bio je proučiti način rada stroja s potpornim vektorima te algoritam Viole i Jonesa za detekciju objekata. Osim toga, kako algoritam Viole i Jonesa vraća višestruke odzive, zadatak je također bio proučiti metodu grupiranja sličnih odziva te njezine nedostatke. Uz pomoć binarnog klasifikatora napravljenog u okviru biblioteke *LIBSVM*, potrebno je bilo pronaći najbolje klasificirani znak iz grupe odziva koju vraća algoritam za detekciju znakova.

Razvijeni su binarni klasifikatori koji s točnošću od oko 98% klasificiraju znakove. Također, klasifikatori iz grupe znakova vrlo dobro odabiru detekcije koje najbolje prikazuju označeni znak. Loše detekcije najčešće se pojavljuju na najbližim znakovima koji se nalaze uz rub slike. U tim slučajevima, kao i u onima gdje je prometni znak velik (veličine oko 70x70 piksela), metodom grupiranja dobiva se odziv koji bolje lokalizira znak od onog koji odabire klasifikator. Dakle, problem odsjecanja desne strane znakova prilikom detekcije u velikoj mjeri ostaje nerješen. Isto tako postoje slučajevi gdje odziv odabran klasifikatorima bolje lokalizira znak.

Kako bi se postotak detekcije znakova pobljšao predlaže se ponavljanje prethodnog eksperimenta s novim klasifikatorom. U dosadašnji skup negativnih primjera dodali bi se i *false-positive*-i dobiveni iz slika koje su u skupa za učenje. Za pozitivne primjere uzeo bi se ostatak primjera iz skupa za učenje. Iako je dani problem na ovaj način dobro rješen, još uvijek postoje problemi i potrebno je razmotriti alternativne metode za njihovo rješavanje.

6. Literatura

[1] Chang C., Lin C., LIBSVM: a Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf, 6. ožujak 2010

[2] Introduction to ImageMagick – ImageMagick, http://www.imagemagick.org/script/index.php#intro, 2010

[3] Hsu C., Chang C., Lin C., A Practical Guide to Support Vector Classifcation, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, 15. ožujak 2010

[4] SVM – support vector machines – DTREG, http://www.dtreg.com/index.htm, 2010

[5] Jakkula, V., Tutorial on Support Vector Machines, School of EECS, Washington State University

[6] Mercer's Conditions – Wikipedia, http://en.wikipedia.org/wiki/Mercer's_condition, 10. veljače, 2010

[7] Tarhini A., Efficient Face Detection Algorithm using Viola Jones method, http://www.codeproject.com/Articles/85113/Efficient-Face-Detection-Algorithm-using-Viola-Jon.aspx, 3. lipanj, 2010

[8] Viola-Jones object detection framework – Wikipedia, http://en.wikipedia.org/wiki/Viola-Jones_object_detection_framework, 11. listopad, 2010

[9] Babić, T., Programska implementacija pronalaženja objekata kaskadom boostanih Haarovih klasifikatora, Završni rad br. 806, Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[10] Benussi, T., Bosilj, P., Čuljak, M., Jurić, D., Miklenić, B., Morava, M., Trbojević, A., Zadrija, L., Pronalaženje i praćenje prometnih znakova Tehnička dokumentacija, Projekt, Fakultet elektrotehnike i računarstva, Zagreb, 12. siječanj, 2010

[11] AdaBoost – Wikipedia, http://en.wikipedia.org/wiki/AdaBoost, 6. svibanj, 2010

[12] OpenMP, http://openmp.org/wp/, svibanj, 2010

7. Dodatak A

1. Skripta za generiranje “pozitivnih” primjera

```
#!/bin/sh
while read line
do
        imgName=`echo "$line" | cut -d " " -f1`
        quantity=`echo "$line" | cut -f2 -d " "`
        pathToImg=`echo "$1" | sed -r 's/\/(.*).dat//'`
        if [ "$quantity" == "" ]
        then
                continue
        fi
        allDimensions=$(($quantity*4))
        dim=`echo "$line" | sed -r 's/(.*)\.bmp//' | sed -r 's/[0-9]+ //'`
        dimNum=1
        signCount=1
        while  [ "$dimNum" -le "$allDimensions" ]
        do
                x=`echo $dim | cut -f1 -d " "`
                y=`echo $dim | cut -f2 -d " "`
                hight=`echo $dim | cut -f3 -d " "`
                width=`echo $dim | cut -f4 -d " "`
                cutImg="$(echo $imgName | sed 's/.bmp//')_$signCount.bmp"
                echo $cutImg
                convert -extract "$width"x"$hight"+"$x"+"$y" -resize 24x24!
                        $pathToImg/$imgName $2/$cutImg
                dimNum=$(($dimNum+4))
                dim=`echo $dim | sed -r 's/[0-9]+ [0-9]+ [0-9]+ [0-9]+ //'`
                signCount=$(($signCount+1))
        done
done <$1
```

Primjer poziva skripte:

```
./positiveExmples.sh UcenjeTrokuti/trokuti.dat IzrezaniZnakovi
```

Prvi argument je *.dat* datoteka koja je smještena u direktoriju gdje su slike iz kojih se izrezuju znakovi, a drugi argument je direktorij u koji se spremaju izrezane slike znakova.

2. Skripta za generiranje “negativnih” primjera

```
#!/bin/bash
for i in `find $1 -regex '.*\.bmp'`
do
  imgName=$(echo $(basename $i) | sed 's/.bmp//');
  echo $imgName
  numNegExamplePerBackground=$(( ( $RANDOM % 30 )  + 1 ))
  for j in `seq 1 "$numNegExamplePerBackground"`
  do
    # choose a random dim from 24 to 100
    dim=$(( $RANDOM % 76 + 24 ))
    xlimit=$((720-$dim))
    ylimit=$((576-$dim))
    posx=$(( $RANDOM % $xlimit + 1 ))
    posy=$(( $RANDOM % $ylimit + 1 ))
    geom=${dim}x${dim}+${posx}+${posy}
    echo " $geom"
    outName=$(printf '%s/%s_%02d.bmp' $2 ${imgName} ${j} )
    echo "  $outName"
    convert -extract $geom -resize 24x24 $i $outName
  done
done
```

Primjer poziva skripte:

```
./negativeExmples.sh pozadine IzrezanePozadine
```

Prvi argument je direktorij s *.bmp* slikama iz kojih isjecamo pozadine, a drugi je direktorij u koji spremamo isječene i skalirane slike.

8. Dodatak B

Za paralelizaciju izvođenja korištenjem *OpenMP*-a [12], potrebno je izmjeniti *Makefile* datoteku na način da se naprave sljedeće izmjene:

```
CFLAGS = -Wall -Wconversion -O3 -fPIC -march=core2 -fopenmp -fomit-frame-pointer -msse -msse2 -msse3 -mfpmath=sse
```

`CFLAGS` opcije su osnovne opcije koje predajemo *gcc-u* kako bismo optimizirali kod. `-fopenmp` opcija omogućava korištenje funkcija *OpenMP*-a.

Također je potrebno modificirati razred `SVC_Q` u programu `svm.cpp` na sljedeći način:

```
int start, j;
if((start = cache->get_data(i,&data,len)) < len){
    #pragma omp parallel for private(j)
        for(j=start;j<len;j++)
            data[j] = (Qfloat)(y[i]*y[j]*(this->*kernel_function)(i,j));
 }
```

Uz to potrebno je modificirati funkciju `svm_predict_values` u programu `svm.cpp`:

```
int l = model->l;
double *kvalue = Malloc(double,l);

pragma omp parallel for private(i)
      for(i=0;i<l;i++)
            kvalue[i] = Kernel::k_function(x,model->SV[i],model->param);
```

9. Dodatak C

Kako bi se dobile vrijednosti odluke potrebno je modificirati program `svm.cpp` na način da se doda funkcija `svm_get_decval` koja vraća vrijednost odluke za najveću apsolutnu vrijednost za slučaj kada se koristi tip SVM-a, C-SVC.

```
double svm_get_decval(const svm_model *model, const svm_node *x)
{
        int nr_class = model->nr_class;
        double *dec_values = Malloc(double, nr_class * (nr_class - 1) / 2);
        svm_predict_values(model, x, dec_values);

        int pos2 = 0;
        double dv = 0;
        double decval = 0;
        double maxabsdecval = 0;

        for(int i = 0; i < nr_class; i++)
        {
                for(int j = i + 1; j < nr_class; j++)
                {
                        dv = dec_values[pos2++];
                        if (fabs(dv) > maxabsdecval)
                                maxabsdecval = fabs(dv);
                        decval = dv;
                }
        }

        free(dec_values);
        return decval;
}
```

Funkcija se poziva unutar metode `svm_predict` i rezultati se ispisuju na standardni ulaz na sljedeći način:

```
printf("%f\n", svm_get_decval(model, x));

free(vote);
free(dec_values);
return model->label[vote_max_idx];
```

Sažetak

U sklopu ovog rada razmatra se problem nepouzdane lokalizacije prometnih znakova detektiranih algoritmom za detekciju objekata Viole i Jonesa. Za rješavanje ovog problema korišen je stroj s potpornim vektorima. Opisan je algoritam Viole i Jonesa, način rada stroja s potpornim vektorima te generiranje binarnog klasifikatora pomoću biblioteke *LIBSVM.* Odzivi dobiveni algoritmom Viole i Jonesa klasificirani su pomoću izgrađenih klasifikatora. Najbolje klasificirani znak uzet je kao reprezentativni odziv grupe. Evaluirani su rezultati i predložen način za poboljšanje postotka detekcije znakova.

Ključne riječi:

Računalni vid, algoritam Viole i Jonesa, stroj s potpornim vektorima, prometni znakovi, lokalizacija, LIBSVM, Haarove značajke, grupiranje, binarna klasifikacija

Abstract

Localizing responses of the Viola Jones detector by a support vector machine classifier

This work addresses the problem of unreliable localization of the detections produced by the Viola-Jones algorithm. We propose solution based on support vector machines. The Viola-Jones algorithm, classification by support vector machines and usage of LIBSVM library are described. The results obtained by Viola-Jones algorithm are classified using the binary classifiers. The best classified sign is considered as the representative. The results are evaluated and method for detection improvement is proposed.

Key words:

Computer vision, Viola-Jones algorithm, support vector machines, traffic signs, localization, LIBSVM, Haar features, grouping, binary classification