

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 758

**Sustav za detekciju i  
raspoznavanje prometnih znakova**

Mislav Mandarić

Zagreb, lipanj 2014.

*Umjesto ove stranice umetnite izvornik Vašeg rada.*  
*Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.*

*Zahvaljujem se mentoru izv. prof. dr. sc. Zoranu Kalafatiću  
i asistentu mag. ing. comp. Ivanu Filkoviću na savjetima, vođenju i pomoći pri izradi  
ovoga rada.*

# SADRŽAJ

<b>1. Uvod</b>	<b>1</b>
<b>2. Teorijski pregled područja</b>	<b>2</b>
2.1. Opis područja . . . . .	2
2.2. Problem detekcije objekata . . . . .	5
2.3. Problem klasifikacije . . . . .	7
2.4. Povezani radovi . . . . .	8
<b>3. Algoritam Virole i Jonesa</b>	<b>10</b>
3.1. Opis algoritma . . . . .	10
3.1.1. Značajke . . . . .	11
3.1.2. Klasifikacija i odabir značajki . . . . .	14
3.1.3. Kaskadna arhitektura . . . . .	15
3.2. Karakteristike algoritma . . . . .	17
<b>4. Algoritam slučajne šume</b>	<b>19</b>
4.1. Opis algoritma . . . . .	19
4.1.1. Stabla odluke . . . . .	20
4.1.2. Izgradnja slučajne šume . . . . .	22
4.1.3. Izdvajanje primjera za učenje . . . . .	23
4.2. Karakteristike algoritma . . . . .	24
<b>5. Implementacija</b>	<b>25</b>
5.1. Arhitektura izgrađenog sustava . . . . .	25
5.2. Opis implementacije . . . . .	26
5.2.1. Implementacija Viola-Jones detektora . . . . .	28
5.2.2. Implementacija klasifikatora slučajne šume . . . . .	30
5.3. Korištene tehnologije i biblioteke . . . . .	31
5.4. Korištenje sustava . . . . .	32

<b>6. Rezultati</b>	<b>36</b>
6.1. Skup podataka . . . . .	36
6.2. Metodologija obrade rezultata . . . . .	38
6.3. Rezultati detekcije . . . . .	39
6.4. Rezultati raspoznavanja . . . . .	40
6.5. Ukupni rezultati sustava . . . . .	41
6.6. Problemi sustava . . . . .	43
<b>7. Zaključak</b>	<b>44</b>
<b>Literatura</b>	<b>45</b>

# 1. Uvod

Razvoj računala donio je nove mogućnosti i nova područja koja mogu iskoristiti računala za pomoć u rješavanju nekih problema. Jedna od industrija koje su počele intenzivno koristiti računala je i auto industrija. Automobili ovog stoljeća dolaze sa mnogo elektroničke opreme i mnogo dodatne opreme koristi računalu za pomoć vozaču tijekom vožnje. ADAS (engl. *Advanced Driver Assistance Systems*) su sustavi predviđeni za pomoć vozaču tijekom vožnje. Jedan od sustava ADAS je i sustav za detekciju i raspoznavanje prometnih znakova.

Sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine svoju primjenu može naći kao jedan od sustava koje vozač koristi tijekom vožnje. Takav sustav mogao bi pomoći kod sigurnosti vozača tako što bi u slučaju prekoračenja brzine vozača mogao upozoriti na opasnost. Neka druga primjena mogla bi biti kao jedan od mnogih sustava iskorištenih u samovozećim automobilima. Sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine bio bi jedan od senzora pomoću kojeg bi automobil sam mogao odrediti optimalnu brzinu vožnje.

Opisani i izgrađeni sustav za detekciju koristi Viola-Jones algoritam, a za raspoznavanje algoritam slučajne šume. Oba algoritma su algoritmi opće namjene i koriste se uobičajene, jednostavne značajke bez kompleksne obrade. Viola-Jones algoritam koristi Haarove značajke za detekciju objekata, a algoritam slučajne šume za značajke koristi vrijednosti slikovnih elemenata slike sivih razina. Sustav je izgrađen kao vrlo fleksibilna biblioteka koja omogućuje jednostavnu implementaciju novih algoritama i jednostavno ju je koristiti u različitim aplikacijama.

U drugom poglavlju opisano je područje koje predstavlja teorijsku podlogu potrebnu za izradu sustava za detekciju i raspoznavanje. Treće i četvrto poglavlje sadrže detaljan pregled dvaju algoritama koji obavljaju centralnu funkcionalnost sustava — algoritam detekcije Viola-Jones i algoritam klasifikacije slučajne šume. U petom poglavlju detaljno je opisana implementacija sustava. U šestom poglavlju prikazani su rezultati izgrađenog sustava, dok sedmo poglavlje sadrži zaključak rada.

## 2. Teorijski pregled područja

### 2.1. Opis područja

Razvojem računala i povećanjem računalne moći, sposobnost računala da pomogne u rješavanju nekih teških problema mnogostruko se povećala. Dok su nekada računala uglavnom radila s tekstom, danas računala služe za prikaz i obradu teksta, zvuka, slike, videa. Slika i video, koji je samo sekvenca slika, posebno su kompleksni tipovi podataka, čija je analiza posebno komplicirana.

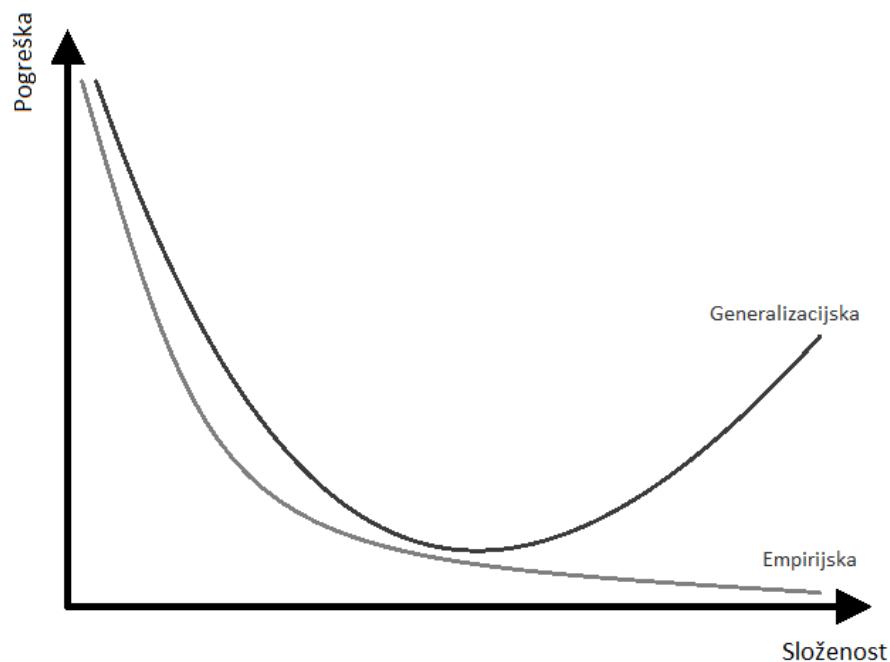
Računalni vid grana je računarstva koja se bavi pretvaranjem slike dobivene kroz senzor u računalu prihvatljiv oblik te analizom i izvlačenjem informacija iz slike. Slika pohranjena u računalu može se analizirati ili mijenjati. Razvijen je niz algoritama za obradu slika koji mogu izvući informacije o objektima na slici, mijenjati oblik zapisa slike ili pretvarati sliku u druge oblike zapisa. Ukoliko se iz slike želi izvući neko novo znanje, potrebno je postupke i algoritme računalnog vida kombinirati sa postupcima dviju grana umjetne inteligencije — stojnim učenjem i raspoznavanjem uzoraka. Strojno učenje bavi se razvojem algoritama i sustava koji mogu učiti iz nekog skupa podataka. Raspoznavanje uzoraka proučava uzorke i pravilnosti u podacima. Te dvije grane su povezane i gledajući iz perspektive računalnog vida, mogu se promatrati kao jedno područje.

Teorija svih metoda učenja bazira se na učenju iz prethodno skupljenog skupa podataka. Metode učenja koriste prethodno skupljeni skup podataka da bi našle poveznice između sličnih podataka i uočile koje značajke podatke razlikuju. Općenito je pravilo da broj prikupljenih primjera izravno utječe na rad sustava strojnog učenja te se povećanjem prikupljenog skupa mogu dobiti bolji rezultati.

Prikupljeni skup podataka dijeli se na dva podskupa, skup za učenje i skup za testiranje. Ta dva podskupa moraju biti disjunktna. Skup za učenje u pravilu je veći i to je skup koji sustav koristi u procesu učenja. Iz njega se izvlače poveznice među značajkama primjera i određuje što povezuje slične, a što odvaja različite primjere. Skup za testiranje je skup na kojem se radi testiranje rada izgrađenog sustava.

Testiranje rada sustava radi se na odvojenom skupu zbog problema generalizacije. Sposobnost generalizacije nekog sustava je sposobnost sustava da raspozna je ili grupira dosad neviđene primjere. To je stvarni problem koji se želi riješiti, da izgrađeni sustav zna pravilno kategorizirati ili grupirati primjere koje do tada nije vidio.

Pogrešku sustava određuje broj pogrešnih izlaza sustava. Ukoliko se pogreška mjeri na skupu za učenje, to predstavlja empirijsku pogrešku. Ta pogreška tipično pada sa složenošću odabranog modela sustava jer se sustav prilagodi podacima i dolazi do prenaučenosti sustava. Pogreška mjerena na skupu za testiranje predstavlja pogrešku generalizacije. To je pogreška koja ovisi o krivom izlazu sustava na skupu primjera koje sustav do tada nije vidio i ona određuje koliko sustav ima dobru sposobnost generalizacije. Ova pogreška tipično sa složenošću pada do neke točke koja predstavlja optimalnu složenost sustava obzirom na podatke nad kojima sustav uči. Nakon te točke pogreška generalizacije tipično raste jer je došlo do prenaučenosti sustava. Sustav gotovo savršeno raspoznaje primjere iz skupa za učenje, ali za nepoznate primjere daje loše rezultate. Prikaz odnosa empirijske pogreške i pogreške generalizacije o složenosti sustava može se vidjeti na slici 2.1 [1].



**Slika 2.1:** Prikaz ovisnosti pogreške o složenosti modela sustava

Postoje dva pristupa učenju iz podataka [1] — nadzirano i nenadzirano učenje. Nadzirano učenje je tip učenja u kojem su podaci nad kojima se uči unaprijed označeni. To znači da se za svaki podatak iz skupa za učenje unaprijed zna koja se vrijednost



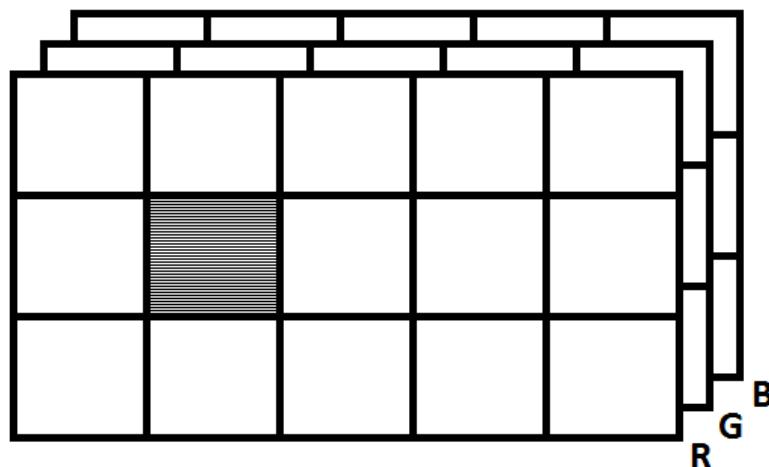
želi dobiti učenjem. Nenadzirano učenje radi sa neoznačenim podacima. Postupak nenadziranog učenja na ulaz dobije skup podataka koji treba na neki način povezati i to tako da slični podaci budu povezani i odvojeni od podataka koji se od njih razlikuju.

Dva tipa problema rješavaju se nadziranim učenjem — klasifikacija i regresija. Klasifikacija je podjela podataka u razrede. Svaki podatak iz skupa s kojim se radi pripada jednom od unaprijed poznatih razreda i cilj je dobiti sustav koji može automatski klasificirati nove podatke. Regresijom se pokušava riješiti problem predviđanja vrijednosti koja nije iz nekog unaprijed poznatog skupa već može biti bilo koja vrijednost. Cilj regresijskog sustava je dobiti što bolje predviđanje izlazne vrijednosti za nove, neviđene podatke.

Nenadziranim učenjem rješava se problem grupiranja. Pošto ulazni skup nije označen, podaci se pokušavaju grupirati u unaprijed zadani broj grupa na način da su podaci unutar grupe slični i razlikuju se od podataka ostalih grupa. Karakteristika po kojoj se određuje sličnost nije eksplicitno zadana već se izvlači iz podataka.

Povezivanjem računalnog vida, strojnog učenja i raspoznavanja uzoraka, dobivaju se sustavi za učenje i traženje uzoraka u slici ili video sekvenci. Skup podataka tada čini skup slika, gdje je svaka slika pohranjena kao višedimenzionalno polje. Postoje razni načini za pohranu slike, ovisno o tipu slike i načinu pohrane, no najčešće se pohranjuje kao niz dvodimenzionalnih matrica. Ukoliko je riječ o slici sivih razina tada se slika može pohraniti kao jedna dvodimenzionalna matrica gdje svaki element matrice ima vrijednost razine sive boje u određenoj točki slike. Ukoliko je riječ o slici u boji, potrebno je više dvodimenzionalnih matrica za pohranu informacije. Najčešći način zapisa je RGB (engl. *Red Green Blue*) zapis slike. Zapis se sastoji od tri dvodimenzionalne matrice, gdje svaka matrica predstavlja zapis vrijednosti razine crvene, zelene ili plave boje. Kombinacijom tih boja moguće je u slici prikazati gotovo cijeli spektar boja, ovisno o veličini zapisa jednog elementa jedne matrice. Najčešće se za zapis jednog elementa koristi 8 bita, čime jedan zapis može poprimiti 256 različitih vrijednosti. Ukoliko je riječ o slici sivih razina tada jedna točka slike može poprimiti jednu od 256 različitih vrijednosti, a ukoliko je riječ o slici u boji u RGB obliku, svaka točka slike može poprimiti jednu od 16.777.216 vrijednosti. To je broj različitih boja koje jedna točka slike može poprimiti. Prikaz slike u boji koristeći RGB zapis može se vidjeti na slici 2.2.

Dvije su kategorije problema koji su posebno zanimljivi koje sustavi temeljeni na računalnom vidu i strojnom učenju rješavaju. To su problemi detekcije nekog objekta na slici i problem klasifikacije objekta prikazanog slikom. Posebno su zanimljivi sustavi koji rješavaju kombinaciju ta dva problema na video sekvenci.



**Slika 2.2:** Slika u boji veličine  $5 \times 3$  zapisana u obliku RGB zapisa

Kombiniranje detekcije i raspoznavanja vrlo je često zbog velikog spektra primjena koje takav sustav može imati. Nekad je potrebno iz velikog skupa slika samo prepoznati objekt koji se na slici nalazi dok je nekad potrebno neki objekt detektirati na nekoj video sekvenci. No često je potrebno detektirati neki skup objekata koji dijele neke značajke i nakon detekcije je potrebno taj skup objekata razvrstati u posebne razrede čime se javlja potreba za traženjem rješenja za kombinaciju ta dva problema. Svaki od problema posebno je opisan u nastavku.

## 2.2. Problem detekcije objekata

Problem detekcije objekata na slici jedan je od najzanimljivijih problema u području računalnog vida. Problem je zanimljiv zbog svoje kompleksnosti, ali i zbog toga što bi rješenje tog problema imalo jako široku primjenu u raznim industrijama. Primjeri primjene sustava za detekciju objekata mogu biti sigurnosni sustavi koji detektiraju osobu na video snimci u realnom vremenu ili detektor lica koji pri korištenju kamere označi lice na zaslonu pametnog telefona ili digitalne kamere.

Detekcija objekata na slici zanimljiva je sama po sebi, no posebno je zanimljiv problem detekcije objekata na video sekvenci. Taj problem donosi nekoliko dodatnih zahtjeva koji se tiču brzine rada detektora. Poželjna osobina detektora objekata na video sekvenci je rad u realnom vremenu. To ograničava sustav detekcije da radi prezahtjevne operacije nad slikom te zahtjeva da obrada bude jednostavna i brza, dok

samo rješenje problema zahtjeva veliku preciznost. Pošto su to često dva međusobno proturječna zahtjeva, potrebno je prilagoditi sustav da ta dva zahtjeva budu uravnotežena.

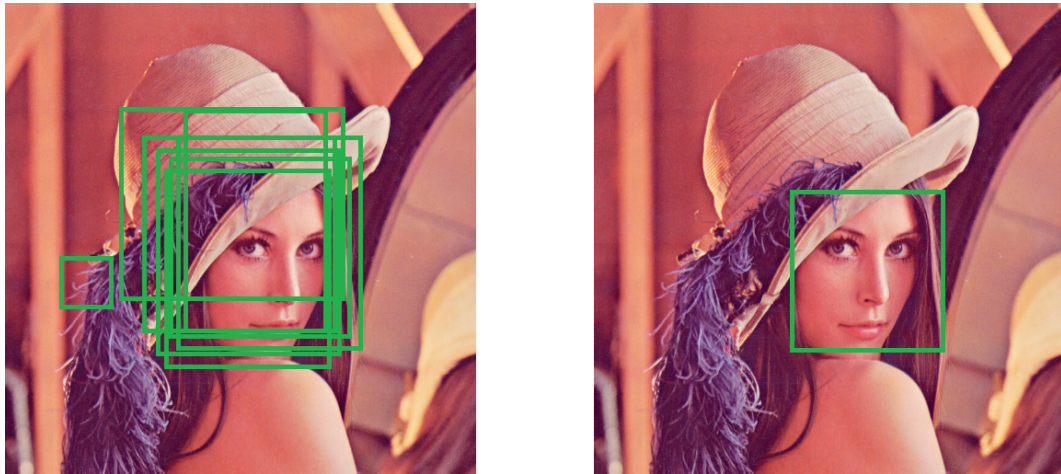
Detekcija objekta na slici podrazumijeva da se na jednoj velikoj slici nalazi objekt (ili više njih) koji je potrebno detektirati. Detekcija se tada obavlja na način da se cijela slika skenira kliznim prozorom. Klizni prozor predstavlja podsliku određene veličine unutar koje će se pokušati detektirati objekt. Ukoliko se objekt detektira u tom podprozoru slike, cijeli se podprozor označuje kao pozitivan. Neovisno o rezultatu detekcije, nakon završetka obrade trenutnog podprozora, prelazi se na sljedeći podprozor tako da se cijeli podprozor pomakne za jedan slikovni element udesno. Na taj se način skenira cijela slika, pomicanjem podprozora udesno i prema dolje, dok se cijela slika ne obradi. To se ponavlja za razne veličine podprozora i slike ovisno o algoritmu i parametrima odabranog algoritma. Slika 2.3 prikazuje kretanje kliznog prozora po slici pri radu detektora lica.



**Slika 2.3:** Prikaz kretanja kliznog prozora

Zbog načina na koji se vrši detekcija objekata, gotovo je neophodno da će se jedan objekt detektirati u više različitih podprozora što bi dovelo do mnogo pozitivnih odziva sustava za jedan objekt. Zbog toga je potrebno imati nekakav mehanizam stapanja svih pozitivnih odziva jednog objekta u jedan. Najčešće se uz taj postupak stapanja provodi odbacivanje podprozora koji bi mogli predstavljati lažno pozitivne detekcije. Pretpostavka je da će se u okolini stvarnog objekta kojeg se želi detektirati nalaziti mnogo pozitivnih odziva sustava. Ukoliko se negdje na slici nalazi podprozor koji ima pozitivan izlaz, a u okolini se ne nalaze drugi podprozori s pozitivnim izlazom, pretpostavlja se da je riječ o lažno pozitivnom odzivu za taj podprozor i on se eliminira prije konačnog kraja rada detektora. Primjer stapanja odziva podprozora slike kod

detektora lica dan je na slici 2.4.



**Slika 2.4:** Prikaz rada mehanizma za stapanje više odziva

### 2.3. Problem klasifikacije

Klasifikacija je jedan od osnovnih problema strojnog učenja. Klasifikacijski problemi rješavaju se izradom sustava koji koristi neki od algoritama učenja za izradu modela koji bi trebao klasificirati nove, neviđene primjere. Primjer primjene klasifikatora slika može biti predlaganje poznanika za označavanje na slikama na društvenim mrežama.

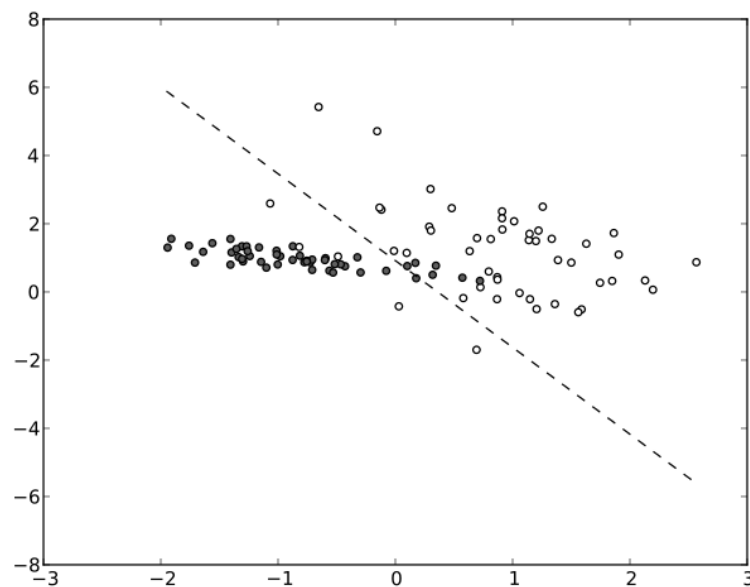
Skup podataka korišten u procesu učenja je označeni skup slika. Svaki podatak predstavljen je parom  $(\mathbf{x}, y)$  gdje  $\mathbf{x}$  predstavlja ulazni vektor značajki dobivenih iz vrijednosti slikovnih elemenata slike, a  $y$  predstavlja izlaznu oznaku razreda kojem taj primjer pripada. Postupak klasifikacije proces je pronalaska funkcije koja preslikava ulazne primjere u razrede tako da broj pogrešnih klasifikacija bude što manji. Funkcija koja obavlja preslikavanje naziva se hipoteza i općenita formula za funkciju koja preslikava ulazni skup značajki  $\mathbf{x}$  u neki od razreda  $y$  je [1]:

$$h(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ pripada razredu } y \\ 0 & \text{inače} \end{cases}$$

Značajke koje se koriste kod klasifikacije slika dobiju se transformacijom slikovnih elemenata slike. Značajke mogu biti različitih razina kompleksnosti. Nekada se kao značajke koriste same vrijednosti slikovnih elemenata bez transformacije, dok se ponekad koriste vrijednosti slikovnih elemenata, ali ne čiste ulazne slike, već na neki način obrađene slike. Često se obrada svodi na pojednostavljenje slike na način da se pretvori iz slike u boji u sliku sivih razina ili

binarnu sliku. Postoje mnogi kompleksniji pristupi dobivanja značajki iz slikovnih elemenata poput značajki histograma orijentiranih gradijenata ili Haarovih značajki. Svaki od pristupa ima svojih prednosti i odabir optimalnih značajki ovisi o problemu koji se rješava.

Broj značajki određuje dimenzionalnost ulaznih primjera. Povećanjem dimenzionalnosti povećava se složenost modela i broj parametara koje je potrebno naučiti da bi se dobila hipoteza. Na slici 2.5 dan je prikaz dvodimenzionalnih ulaznih podataka i funkcije koja razdvaja primjere u dva razreda. Pronađena hipoteza koja dovoljno dobro odvaja primjere za učenje zapravo je pravac.



**Slika 2.5:** Funkcija koja razdvaja primjere u dva razreda

Kao što se iz slike 2.5 može vidjeti, cilj klasifikatora nije savršeno dobro razdvojiti skup primjera za učenje. Skup primjera za učenje predstavlja mali podskup svih mogućih primjera s kojima bi klasifikator trebao moći raditi i ukoliko se model nauči savršeno dobro prilagoditi podacima za učenje vjerojatno će doći do prenaučenosti modela i sposobnost generalizacije tog sustava neće biti dobra.

## 2.4. Povezani radovi

Izgrađeni sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine svoj rad temelji na detektoru Viole i Jonesa [2] i klasifikatoru slučajne šume

[3]. Detektor za značajke koristi Haarove značajke i spada u skupinu multifunkcionalnih metoda detekcije gdje je detektor moguće naučiti da prepoznaje bilo kakav objekt koji se može opisati Haarovim značajkama. Detektor je naučen da prepoznaje sve okrugle znakove za ograničenje brzine. Za klasifikator se kao značajke koriste vrijednosti slikovnih elemenata slike sivih razina. Klasifikator znak raspoređuje u jedan od pet razreda koji odgovaraju ograničenjima brzine od 30 km/h do 70 km/h.

Mnogo je različitih pristupa u rješavanju problema detekcije i raspoznavanja prometnih znakova. Razlike se mogu nalaziti u značajkama koje se koriste u fazama detekcije ili raspoznavanja ili se mogu nalaziti u metodama koje se koriste za detekciju ili raspoznavanje. Neki sustavi osim detekcije i raspoznavanja implementiraju i mogućnost praćenja prometnog znaka čime se može poboljšati rad cjelokupnog sustava.

Jedan pristup može biti prepoznavanje o kojem se ograničenju brzine radi korištenjem metoda za digitalno prepoznavanje znamenki, kao što je to napravljeno u [4]. U tom radu za detekciju svih znakova koristi se Viola-Jones detektor. Za prepoznavanje se koristi umjetna neuronska mreža, a ako je izlaz mreže znak za ograničenje brzine, onda se određivanje točnog ograničenja brzine vrši nad segmentiranom slikom znamenki.

Primjeri za pristup koji koristi praćenje znakova kroz više sličica video sekvence mogu se naći u [5], [6] i [7]. Najčešći razlog za praćenje objekta kroz vrijeme je smanjenje broja lažno pozitivnih izlaza detektora. Kod rješavanja problema detekcije prometnih znakova, broj lažno pozitivno detektiranih znakova je obično vrlo velik. Većina tih lažno pozitivnih detekcija se ne pojavljuje uzastopno kroz više sličica videa te ih se može odbaciti korištenjem neke od metoda praćenja.

Različite metode koriste se u koraku klasifikacije. Često rješenje je korištenje umjetne neuronske mreže, poput rješenja u [4], [5] ili [8]. Primjer korištenja slučajnih šuma može se vidjeti u [9] uz razliku što se za značajke koriste mapa udaljenosti i histogrami orijentiranih gradijenata.

Za detektor prometnih znakova najčešće se koristi Viola-Jones detektor. Primjeri se mogu vidjeti u [4] i [10], dok se primjer jednog drugog pristupa detekciji može vidjeti u [5]. Koristi se Houghova transformacija i informacija o rubovima na slici. Zbog specifičnog oblika znakova, informacija o rubovima može se iskoristiti za izradu dobrog i robustnog detektora.

## 3. Algoritam Virole i Jonesa

### 3.1. Opis algoritma

Algoritam Virole i Jonesa [2], predstavljen 2001. godine, brzo je postao jedan od najpopularnijih pristupa kod rješavanja problema detekcije objekata na slici. Koncepti koje su uveli Paul Viola i Michael Jones doveli su do višestruko bržih detekcija od dotadašnjih pristupa i omogućili rad detektora u stvarnom vremenu, pritom ne umanjujući kvalitetu i preciznost detektora.

Postupak detekcije Viola-Jones algoritma temelji se na tri vrlo bitna koncepta. Svaki od koncepata poznati su od ranije i koristili su se za rješavanje raznih problema detekcije i klasifikacije. Međutim, Viola-Jones detektor povezuje poznate koncepte na novi, inovativan način, čime se dobio jedan od najrobusnijih i najkorištenijih algoritama detekcije.

Prvi bitan koncept na kojem se temelji rad Viola-Jones algoritma su značajke s kojima algoritam radi. Najbitnija karakteristika koju odabrane značajke moraju imati je jednostavnost. Ideja je da se kombinacijom velikog broja jednostavnih, lako izračunljivih značajki dobiju dobre značajke koje mogu opisati gotovo bilo koji objekt. Izvorno odabrane značajke koje posjeduju tu karakteristiku su Haarove značajke, dok se danas često koriste LBP značajke (engl. *Local Binary Pattern*). Razlog tomu je činjenica da su LBP značajke cjelobrojne, za razliku od realnih Haarovih značajki, te su zbog toga još jednostavnije i brže za izračunati. Iako jednostavnije, bitno ne umanjuju kvalitetu detekcije te se zbog toga sve češće koriste umjesto Haarovih značajki.

Uz odabir značajki, Viola i Jones uveli su novi, brzi način izračuna navedenih značajki. Prolaskom kroz sliku, za svaki se slikovni element izračuna njegova integralna vrijednost, čime se dobije integralna slika (engl. *Integral Image*). Iz integralne slike moguće je vrlo brzo, osnovnim operacijama, dobiti vrijednost značajke.

Drugi bitan koncept Viola-Jones algoritma je postupak učenja klasifikatora. Za

metodu učenja odabran je algoritam AdaBoost (engl. *Adaptive Boosting*) koji koristi postupak ojačavanja (engl. *Boosting*) za odabir značajki i učenje klasifikatora. Zbog izuzetno velikog broja značajki u jednom podprozoru slike, potreban je mehanizam odabira najboljih značajki koje će se iskoristiti za detekciju. Algoritam učenja AdaBoost koristi slabe klasifikatore te ih ojačava, čime se dobije jaki klasifikator. U procesu ojačavanja, samo najbolje značajke se koriste za ojačavanje te se velik dio značajki odbacuje, a sve u cilju ubrzavanja rada klasifikatora.

Treći koncept, a ujedno i najzanimljiviji, je kaskadna arhitektura detektora. U procesu učenja odabire se više klasifikatora sa različitim brojem značajki i različitim pragovima prihvatljivosti. Ti se klasifikatori spajaju u kaskadu na način da su najjednostavniji klasifikatori postavljeni na početak kaskade, a svaka iduća razina kaskade sadrži kompleksniji klasifikator od prethodne razine. Ako se u bilo kojem koraku kaskade podprozor odbaci kao negativan, odziv cijelog detektora je negativan. Time se postiže da se najveći dio podprozora odbaci već u ranoj fazi kaskade kada je izračun, zbog jednostavnosti prvih klasifikatora, vrlo jednostavan i računalno nezahtjevan.

Kombinacijom ovih komponenti, postiže se visoka razina točnosti detekcije uz vrlo veliku brzinu rada. Svaka od tih komponenti detaljnije je obrađena u narednim potpoglavljima.

### **3.1.1. Značajke**

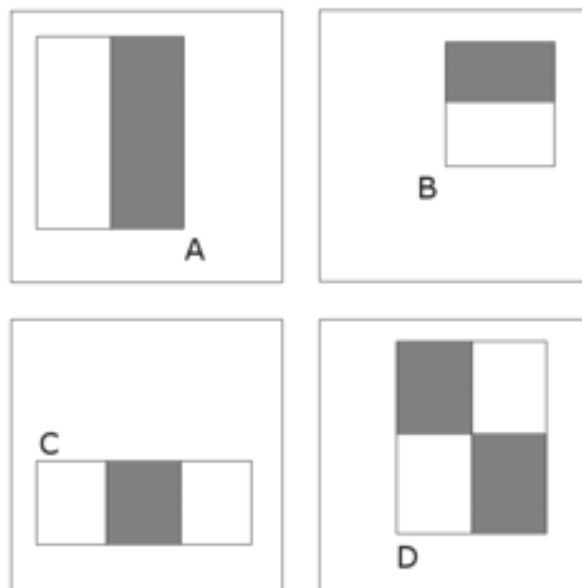
Značajke koje se koriste kod detektora Viole i Jonesa trebale bi biti vrlo jednostavne i lako izračunljive. Uz takve karakteristike, proces detekcije se uveliko ubrzava zbog velike brzine izračunavanja vrijednosti značajki, a kombinacijom većeg broja značajki moguće je i dalje imati vrlo visoku razinu točnosti detekcije.

Značajke koje imaju te karakteristike su značajke koje podsjećaju na Haarove bazne funkcije — Haarove značajke [2]. Riječ je o značajkama koje se dobiju izračunom površina pravokutnog oblika, a koristi se tri tipa značajki. Značajke se mogu vidjeti na slici 3.1.

Prvi tip su značajke koje čine dva pravokutnika. Pravokutnici mogu biti postavljeni horizontalno ili vertikalno. Vrijednost značajke dobije se na način da se vrijednost sume slikovnih elemenata koji se nalaze u bijelom području oduzme od vrijednosti sume elemenata koji se nalaze u crnom području.

Drugi tip značajki su značajke koje čine tri horizontalno postavljena pravokutnika. Vrijednost te značajke je broj koji se dobije kada se od vrijednosti sume slikovnih





**Slika 3.1:** Haarove značajke korištene kod detekcije Viola-Jones algoritmom

elemenata unutrašnjeg, crnog pravokutnika oduzme zbroj suma vrijednosti slikovnih elemenata dvaju vanjskih, bijelih pravokutnika.

Treći tip značajki su značajke sa četiri pravokutnika. Pravokutnici su postavljeni tako da se dijagonalni pravokutnici zbrajaju. Vrijednost značajke je tada ponovno razlika sume vrijednosti slikovnih elemenata bijelih i crnih pravokutnika.

Ove značajke vrlo su jednostavne i lako izračunljive. Zbog toga što su jednostavne, ne sadrže mnogo informacije te je potrebno kombinirati mnogo značajki za opis nekog objekta. Izračun ovih značajki relativno je jednostavan, a dodatno je ubrzan novim načinom zapisa slike — integralnom slikom.

### Integralna slika

Integralna slika [2] novi je način zapisa slike koji su uveli Viola i Jones za potrebe detekcije objekata. On nudi izuzetno brz način za izračun površine pravokutnika, poput onih koji opisuju Haarove značajke.

Integralna slika predstavlja sliku opisanu tako da se na poziciji svakog slikovnog elementa nalazi vrijednost sume svih slikovnih elemenata koji se nalaze lijevo i gore od trenutnog elementa. Drugim riječima, to je suma svih slikovnih elemenata koji čine pravokutnik čija je gornja, lijeva točka početak slike s koordinatama  $(0, 0)$ , a donja, desna točka trenutni element. Koristeći formulu 3.1 dobije se vrijednost integralne slike za točku  $(x, y)$  [2].

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (3.1)$$

Koristeći formule 3.2 i 3.3, integralnu sliku moguće je izračunati i rekurzivno [2].

$$s(x, y) = s(x, y - 1) + i(x, y) \quad (3.2)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \quad (3.3)$$

Vrijednost  $s(x, y)$  predstavlja sumu vrijednosti slikovnih elemenata retka, od njegovog početka do trenutne točke  $(x, y)$ . Integralna slika  $ii(x, y)$  je tada suma vrijednosti integralne slike za točku koja se nalazi iznad trenutne  $ii(x - 1, y)$  i vrijednosti sume retka trenutne točke  $s(x, y)$ . Ove rekurzivne formule omogućavaju izračun integralne slike uz samo jedan prolazak kroz originalnu sliku, što uvelike pridonosi brzini rada detektora.

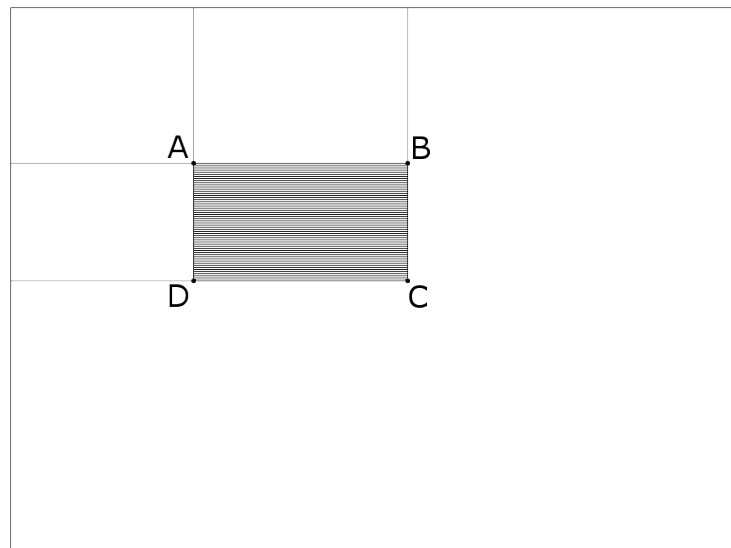
Nakon dobivene integralne slike, izračun vrijednosti Haarovih značajki vrlo je jednostavan i računalno vrlo nezahtjevan. Dobivanje vrijednosti Haarove značajke ne ovisi o veličini slike i složenost izračuna vrijednosti značajke je konstantna.

Na slici 3.2 dan je primjer jednog pravokutnika koji može predstavljati jedan od pravokutnika neke Haarove značajke. Korištenjem integralne slike, sumu slikovnih elemenata koji se nalaze unutar označenog pravokutnika moguće je dobiti korištenjem formule 3.4. Formula se sastoji od jednostavnog zbrajanja i oduzimanja vrijednosti integralne slike za granične točke pravokutnika [2].

$$a = ii(C) + ii(A) - ii(B) - ii(D) \quad (3.4)$$

Za svaki od tri tipa Haarovih značajki potreban je različit broj graničnih točaka koje su potrebne za izračun vrijednosti značajke. Značajke koje čine dva pravokutnika zahtjevaju šest graničnih točaka, značajke koje čine tri pravokutnika zahtjevaju osam točaka, a značajke koje čine četiri pravokutnika zahtjevaju devet graničnih točaka za izračun vrijednosti značajke.

Jednom kada se izračuna integralna slika, izračun vrijednosti jedne značajke zahtjeva samo nekoliko operacija zbrajanja ili oduzimanja nad vrijednostima integralne slike graničnih točaka. Obzirom na željeni cilj detekcije objekata u realnom vremenu, to je jako dobra karakteristika.



**Slika 3.2:** Izračun sume označenog pravokutnika korištenjem integralne slike

### 3.1.2. Klasifikacija i odabir značajki

Izračun vrijednosti jedne odabrane Haarove značajke vrlo je brz. No zbog činjenice da se u jednom podprozoru slike nalazi velik broj značajki, njih više od 180.000 u podprozoru veličine  $24 \times 24$  slikovna elementa, izračun svake od njih za svaki podprozor slike trajao bi predugo. Zbog toga je potreban nekakav postupak redukcije broja značajki koje će se koristiti u procesu detekcije.

Rješenje tog problema pronađeno je u odabiru klasifikacijskog algoritma. Za klasifikaciju je odabran algoritam AdaBoost, a način implementacije je takav da se taj algoritam ujedno koristi i za odabir značajki. AdaBoost [11] je meta-algoritam koji kombinira izlaze drugih slabijih klasifikatora u jedan jaki klasifikator, čime postiže vrlo dobre rezultate klasifikacije. Taj postupak kombiniranja više slabih klasifikatora u jedan jaki klasifikator naziva se ojačavanje.

Postupak ojačavanja započinje učenjem slabog klasifikatora i dohvatom vrijednosti hipoteze tog klasifikatora. Ovisno o izlazu slabog klasifikatora i njegovoj pogrešci, ažuriraju se vrijednosti težina uzoraka za učenje i to na način da se pogrešno klasificiranim primjerima dodijeli veća važnost, čime se idući slabi klasifikator koncentrira na točnu klasifikaciju tih primjera. Taj postupak se ponavlja za više slabih klasifikatora te konačni jaki klasifikator predstavlja težinsku sumu izlaza slabih klasifikatora. Pseudokod algoritma AdaBoost opisuje algoritam 1 [2].

Kod Viola-Jones algoritma, slabi klasifikator je klasifikator koji koristi samo jednu Haarovu značajku i pokušava što bolje razvrstati primjere koristeći samo tu

---

**Algorithm 1** AdaBoost

---

**Ulaz:**  $(x_1, y_1), \dots, (x_n, y_n)$  — Skup primjera za učenje.

**Izlaz:**  $h(x)$  — Hipoteza jakog klasifikatora.

Inicijaliziraj težine  $w_{1,i} = \frac{1}{2m}$  za  $y_i = 0$  i  $w_{1,i} = \frac{1}{2l}$  za  $y_i = 1$

**for**  $t = 1, \dots, T$  **do**

- Normaliziraj težine

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

- Nauči slabi klasifikator  $h_k$  za svaku od značajki  $k$  s pogreškom

$$\epsilon_k = \sum_{j=1}^n w_{t,j} |h_k(x_j) - y_j|$$

- Izaberi slabi klasifikator  $h_t$  s najmanjom pogreškom  $\epsilon_t$
- Ažuriraj težine

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}, \quad \beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$$

**end for**

**return**

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t, \quad \alpha_t = \log \frac{1}{\beta_t} \\ 0 & \text{inače} \end{cases}$$

---

jednu značajku. U svakom krugu ojačavanja odabire se najbolji slabi klasifikator i dobiva se pogreška tog jednog klasifikatora koji koristi tu jednu odabranu značajku. Konačni jaki klasifikator predstavlja težinsku sumu slabih klasifikatora.

Pošto svaki slabi klasifikator koristi samo jednu značajku, postupak učenja klasifikatora ujedno je i postupak odabira značajki koje će se koristiti za rad algoritma. Odabrane značajke najbolje razdvajaju zadani skup primjera za učenje i razina točnosti klasifikatora s podskupom najboljih značajki nije mnogo lošija od onoga koji bi koristio cijeli skup značajki. Razlog tomu je činjenica da je manji broj dobrih značajki dovoljno dobar da dobro opiše objekt detekcije.

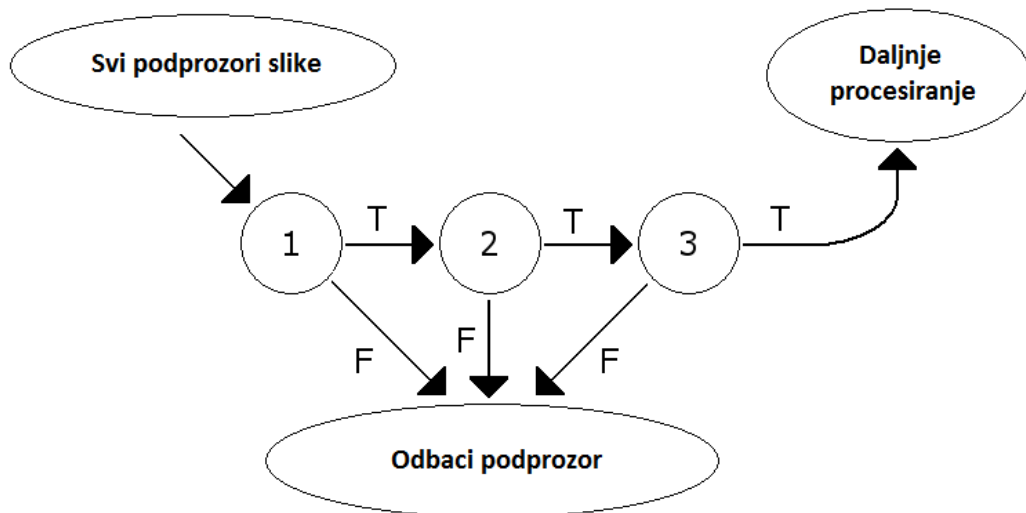
### 3.1.3. Kaskadna arhitektura

Korištenje malog podskupa jednostavnih, brzo izračunljivih značajki nije dovoljno da bi se dobio dobar, visoko učinkovit i dovoljno brz algoritam detekcije objekata na slici.

Uz pretpostavku da su jednostavne značajke dovoljne da bi se dobro opisao objekt te pretpostavku da je za to dovoljan samo mali podskup takvih značajki, treća bitna pretpostavka koja doprinosi kvaliteti i brzini detektora je da se na cjelokupnoj slici objekt nalazi u samo malom broju podprozora slike.

Ta pretpostavka dovela je do korištenja kaskadne arhitekture pri detekciji objekata [2]. Kaskadna arhitektura je arhitektura u kojoj se koristi više različitih klasifikatora postavljenih slijedno u kaskadu, na način da se na jednoj razini kaskade nalazi jedan klasifikator.

Svaki od klasifikatora koristi druge značajke i ima različite vrijednosti praga. Cilj je svakog od klasifikatora što ispravnije klasificirati one primjere koje je klasifikator prethodne razine označio ispravnima. Na taj se način postiže da za pozitivnu klasifikaciju objekta svi klasifikatori moraju ulazni podprozor slike označiti kao pozitivan, a ukoliko samo jedan od klasifikatora, na bilo kojoj razini kaskade, označi ulaz kao negativan, ulaz se odbacuje i izlaz cijelokupne kaskade se računa kao negativan. Prikaz rada takve kaskade može se vidjeti na slici 3.3.



**Slika 3.3:** Kaskadna arhitektura

Klasifikatori kaskade složeni su tako da su jednostavniji klasifikatori koji koriste manje značajki i imaju niži prag detekcije postavljeni na početak. Pošto se podprozori slike koje čak i ti jednostavni klasifikatori označavaju negativnima odmah odbacuju, to dovodi do toga da se većina podprozora odbacuje već na prvoj ili drugoj razini kaskade. Pošto je izračun tih prvih razina kaskade vrlo brz, odbacivanje većine podprozora događa se vrlo brzo i efikasno. Klasifikator zato može posvetiti većinu

vremena pokušavajući klasificirati sumnjiva područja koja nalikuju traženom objektu.

Zbog takve arhitekture gdje je za odbacivanje podprozora potrebna samo jedna negativna klasifikacija, bitno je da svaki klasifikator ima što je moguće manji broj lažno negativnih izlaza, pa makar i po cijenu velikog broja lažno pozitivnih. To je posebno bitno u ranim razinama kaskade, kada bi broj ispravno detektiranih objekata trebao biti približno 100%. Kako ulazni podprozor slike prolazi kroz razine kaskade, povećava se složenost klasifikatora i broj lažno pozitivnih primjera pada. Povećanjem broja razina kaskade smanjuje se broj lažno pozitivnih primjera, ali i pada broj točnih detekcija te je potrebno u fazi učenja parametre sustava podesiti tako da se pronade dobar balans između ta dva, protivna zahtjeva.

## **3.2. Karakteristike algoritma**

Algoritam Viole i Jonesa jedan je od prvih algoritama sposoban za detekciju objekata na slici koji može raditi u realnom vremenu, pritom zadržavajući vrhunsku razinu točnosti detekcije. Kompletna arhitektura sustava dizajnirana je s ciljem rješavanja problema rada u realnom vremenu i taj problem je riješen vrlo dobro. Izvođenje Viola-Jones detektora može raditi u realnom vremenu na svim modernim uređajima, čak i na ugradbenim ili mobilnim uređajima relativno male procesorske snage.

Koliko je algoritmu prednost brzina rada, toliko mu je mana brzina učenja. Zbog izuzetno kompleksnog problema koji se rješava u procesu učenja, ono čak i na suvremenim stolnim računalima može trajati danima. Razlog tomu je što algoritam radi na problemu optimizacije između broja razina kaskade, broja značajki koje će se upotrijebiti za detekciju u pojedinoj razini kaskade i praga detekcije pojedine razine.

Takav spor proces učenja može biti problematičan kada je potrebno fino podešavati parametre algoritma u procesu učenja. Promjenom parametara potrebno je ponovno proći cijeli proces učenja i ponovnog testiranja algoritma, a ukoliko taj proces traje danima, povratna vrijednost o kvaliteti rada algoritma nakon promjene parametara dobiva se vrlo sporo. Ovisno o problemu detekcije koji se rješava, proces učenja može zbog toga biti izuzetno dugotrajan.

Algoritam radi nad slikama sive razine te se ulazna slika prije izvršavanja algoritma pretvara u sliku sivih razina. Pošto algoritam koristi Haarove značajke za opis objekta detekcije, a one se računaju samo obzirom na vrijednost jakosti osvjetljenja slikovnog elementa, slika u boji nije potrebna za detekciju. To omogućava brži izračun značajki, a zbog činjenice da se ne koristi boja kao značajka, postiže se velika robustnost obzirom na osvjetljenje ulazne slike.

Kod rješavanja problema detekcije prometnih znakova u video sekvenci, algoritam daje velik broj lažno pozitivnih detekcija. Jedan od razloga za to je relativno malen broj znakova u odnosu na broj sličica u videu. Algoritam se može naučiti da koristi veću kaskadu čime bi se smanjio broj lažno pozitivnih primjera, no to bi dovelo do smanjenja broja ispravnih detekcija. Zbog toga se, ukoliko broj lažno pozitivnih primjera predstavlja veliki problem, često broj lažno pozitivnih primjera pokušava smanjiti u koraku nakon same detekcije, nekim klasifikacijskim postupkom. Ukoliko je smanjenje točnosti detekcija prihvatljivo malo, rješenje smanjenja broja lažno pozitivnih odziva većom kaskadom također je prihvatljivo.

## 4. Algoritam slučajne šume

### 4.1. Opis algoritma

Algoritam slučajne šume [3] jedan je od najpopularnijih algoritama današnjice. Predstavljen u radu Lea Breimana 2001. godine, kombinira nekoliko prije razvijenih koncepata u području strojnog učenja za rješavanje problema klasifikacije, regresije i grupiranja.

Osnovne značajke algoritma su otpornost na šum u podacima, velika brzina učenja i izvršavanja, jednostavnost izrade i vrlo velika razina točnosti. Algoritam je najjednostavnije opisati kao skup stabala odluke, gdje svako stablo izvršava klasifikaciju i konačna klasifikacija je ona čiji je rezultat vratila većina stabala odluke.

Postupak učenja slučajnih šuma svodi se na učenje velikog broja stabala odluke. Svako stablo odluke ima svoj, slučajno generirani skup primjera za učenje, dobiven iz početnog, zadanog skupa. Skup primjera za učenje generira se iz  $2/3$  početnog skupa korištenjem postupka generiranja novih podataka uzorkovanjem s ponavljanjem — *bagging* (engl. *Bootstrap aggregating*). *Bagging* je postupak kojim se iz zadanog skupa generira novi skup tako što se iz iste vjerojatnosne distribucije uzorkuju novi primjeri. Primjeri se mogu ponavljati jer se koristi uzorkovanje s ponavljanjem.

Izdvojena trećina početnog skupa primjera za učenje koristi se kao OOB (engl. *out-of-bag*) skup primjera. Za svako stablo zasebno se bira OOB skup. Pošto se ti primjeri ne koriste za izgrađivanje stabla u procesu učenja, OOB primjeri se mogu iskoristiti u procesu testiranja rada algoritma. Korištenjem mehanizma OOB skupa primjera moguće je tijekom samog procesa učenja dobiti povratnu informaciju o kvaliteti algoritma. Tako se OOB primjeri koriste za izračun nepristrane pogreške algoritma i izračun važnosti pojedine varijable za određivanje konačnog rezultata naučenog algoritma. Posebno je bitna mogućnost izračuna nepristrane pogreške tijekom učenja jer to omogućuje provjeru algoritma bez korištenja posebnog testnog skupa primjera za učenje.

Algoritam ovisi o nekoliko parametara, ali je daleko najvažniji od njih broj značajki



koje se koriste u procesu izgradnje stabala odluke. Pri izgradnji stabala se ne koriste sve značajke u procesu učenja, već se za svako stablo koristi slučajno odabrani podskup svih značajki. Broj značajki koji se odabire za svako stablo unaprijed se određuje i konstantan je za sva stabla u cijelom procesu učenja. Taj broj najviše utječe na kvalitetu rada slučajnih šuma zbog toga što utječe na dva svojstva o kojima ovisi kvaliteta rada algoritma [12].

Prvo bitno svojstvo je međuzavisnost između dva stabla odluke, a drugo je snaga svakog pojedinog stabla. Povećanjem međuzavisnosti između stabala, povećava se i pogreška ukupnog algoritma. S druge strane, povećanjem snage pojedinog stabla, smanjuje se pogreška cjelokupnog algoritma jer se povećanjem snage pojedinačnog stabla, povećava ukupna snaga skupa stabala.

Broj odabranih značajki povećava oba ta svojstva. Povećanjem broja značajki koje jedno stablo koristi, povećava se međuzavisnost svih stabala jer više stabala koristi iste značajke u procesu učenja. Također, povećanjem broja značajki pojedino stablo ima više informacija za izgradnju te mu se povećava snaga jer može bolje naučiti odvajati primjere za učenje. Ta dva svojstva potrebno je balansirati dobrim odabirom broja korištenih značajki, a postupak kojim se to može postići je validacija korištenjem OOB pogreške.

Drugi parametar koji se može odrediti i koji utječe na rad algoritma je broj stabala slučajne šume. Povećanjem broja stabala korištenih u radu algoritma povećava se kvaliteta rada algoritma i smanjuje se pogreška. Pogreška koja se smanjuje je pogreška generalizacije i zbog toga povećanjem broja stabala ne može doći do prenaučivosti algoritma. Zbog činjenice da povećanje broja stabala također povećava i složenost izračuna rezultata rada algoritma, potrebno je postaviti broj stabala na vrijednost koja daljnjim povećanjem neće bitno poboljšati rad algoritma, a omogućit će prihvatljivu brzinu izvođenja [12].

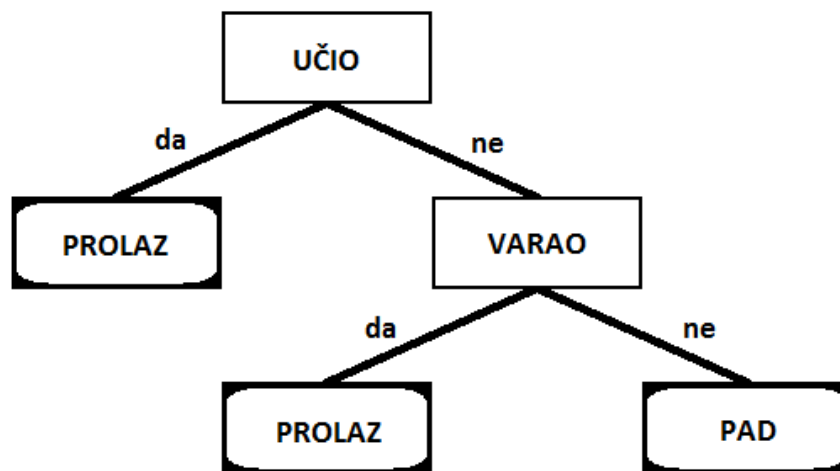
#### **4.1.1. Stabla odluke**

Slučajne šume svoj rad temelje na stablima odluke. Korištenjem više stabala odluke dobije se više rezultata klasifikacije, za svako stablo poseban rezultat. Konačna klasifikacija slučajne šume je ona klasifikacija koju je kao rezultat dao najveći broj stabala odluke.

Stabla odluke jedno su od najpopularnijih općenitih rješenja klasifikacijskih problema. Vrlo su jednostavna za izgradnju i interpretaciju, daju dobre rezultate i otporna su na šum u podacima. Postoji velik broj algoritama koji koriste stabla odluke

kao metodu učenja iz podataka.

Izgradnja stabla odluke ovisi o značajkama ulaznog skupa za učenje. Značajke predstavljaju attribute koji se testiraju i ovisno o vrijednosti atributa određuje se izgled stabla. Svako stablo odluke sastoji se od čvorova i grana. Čvorovi predstavljaju testiranje atributa, a grane predstavljaju vrijednosti atributa. Grananje za svaki atribut u čvoru odvija se obzirom na vrijednosti tog atributa. Na slici 4.1 može se vidjeti primjer jednog jednostavnog stabla koje obrađuje problem klasifikacije prolaska srednjoškolskog ispita.



**Slika 4.1:** Prikaz jednog jednostavnog stabla sa dvije značajke

Počevši od korijena stabla, na svakom čvorištu stablo se grana obzirom na odabranu značajku iz skupa značajki primjera za učenje. Nakon grananja stvaraju se novi čvorovi koji se ponovno granaju obzirom na neku drugu značajku, sve dok se ne potroše sve značajke za grananje. Listovi stabla predstavljaju klasifikaciju stabla za odabrani put stabla. Na primjeru stabla sa slike 4.1 mogu se vidjeti svi ti elementi stabla. Stablo je izgrađeno iz dvije značajke ili dva atributa. Značajka "Učio" je najznačajnija i ona predstavlja korijen stabla. Ukoliko je vrijednost atributa "da" tada se dolazi do lista stabla koje označava klasifikaciju kao "Prolaz". Ukoliko je vrijednost atributa "ne", obrađuje se desno podstablo. Prvi novi čvor tog podstabla predstavlja atribut "Varao". Vrijednost atributa "da" dovodi do klasifikacije "Prolaz", a vrijednost atributa "ne" dovodi do klasifikacije "Pad".

Odabir redoslijeda značajki kojim će se odvijati grananje stabla ovisi o konkretnom algoritmu kojim se obavlja učenje stabla odluke. Ideja je stablo graditi tako da se najprije grana koristeći značajke koje najbolje razdvajaju skup primjera za učenje, a da se značajke koje ga lošije razdvajaju u procesu grananja koriste kasnije.

Slučajne šume koriste algoritam CART (engl. *Classification and Regression Tree*). Kriterij za odabir značajki koji se koristi je informacijska dobit korištenjem Gini indeksa nečistoće. Vrijednost Gini indeksa je koliko često će slučajno odabrani primjer skupa biti pogrešno klasificiran, ako mu se slučajno dodijeli razred. Formula 4.1 koristi se za izračun Gini indeksa [13].

$$I_G(f) = \sum_{i=1}^m p_i(1 - p_i) = 1 - \sum_{i=1}^m p_i^2 \quad (4.1)$$

Za svaki od  $m$  razreda izračuna se udio primjera koji pripadaju razredu  $m$  u skupu primjera za učenje. To je vrijednost  $p_i$ . Gini indeks je tada suma kvadrata tih vrijednosti oduzeta od 1.

Odabir značajke se tada odvija prema formuli 4.2 [1].

$$I(f) = I_G(f) - \sum_{j=1}^k \frac{|f_j|}{|f|} I_G(f_j) \quad (4.2)$$

Značajka  $f$  može poprimiti jednu od  $k$  različitih vrijednosti. Vrijednost  $I(f)$  predstavlja informacijsku dobit i ona označava koliko se promijeni Gini indeks ukoliko se skup primjera za učenje podijeli obzirom na odabranu značajku  $f$ . Što veća informacijska dobit, to će se bolje podijeliti skup primjera za učenje te je bolje grananje stabla obaviti koristeći tu značajku.

#### 4.1.2. Izgradnja slučajne šume

Slučajne šume predstavljaju skup stabala odluke. Pošto imamo samo jedan skup primjera za učenje, potrebno je razviti mehanizam kojim će se iz tog skupa odabrati skup za svako pojedinačno stablo. Osnovica mehanizma je postupak koji se zove *bagging*. *Bagging* je postupak generiranja novog skupa iz postojećeg na način da se iz vjerojatnosne distribucije početnog skupa generira novi skup [3].

Iz početnog skupa primjera za učenje  $D$  odabere se  $2/3$  skupa. Zatim se iz vjerojatnosne distribucije tog skupa uzorkuje s ponavljanjem novi skup  $D'$ . Novogenerirani skup se koristi za izgradnju jednog stabla odluke te se za svako stablo postupak ponavlja. Time se postiže da svako stablo ima jedinstven skup primjera za učenje.

Postupak izgradnje jednog stabla odluke odvija se korištenjem uobičajenih algoritama za izgradnju stabala odluke. Razlika u izgradnji stabla za slučajnu šumu je u tome što se za izgradnju stabla ne koriste sve značajke primjera za učenje. Od svih

$M$  značajki, za svako se stablo odabire  $m \ll M$  slučajnih značajki. Parameter  $m$  određuje se unaprijed i konstantan je tijekom izgradnje svih stabala šume.

Ovakav se pristup koristi zbog smanjenja međuovisnosti između stabala. Pošto slučajne šume rade na način da se nakon završetka rada svakog stabla kao rezultat cijele šume uzima ona klasifikacija čiji je izlaz dao najveći broj stabala, potrebno je minimizirati međuovisnost između stabala da bi svaka od pojedinih klasifikacija bila što je moguće nezavisnija.

### 4.1.3. Izdvajanje primjera za učenje

Iz početnog skupa primjera za učenje  $2/3$  primjera je iskorišteno za postupak *bagginga*. Ostalih  $1/3$  podataka izdvojeno je u OOB skup primjera [12] [3].

Za svako se stablo stvara poseban OOB skup. Pošto se taj skup ne koristi u postupku učenja tog stabla, on se može iskoristiti za testiranje izgrađenog stabla. Testiranje se može odnositi na pogrešku rada algoritma i tada se govori o OOB procjeni pogreške, a može se odnositi i na testiranje značaja pojedinih značajki.

OOB procjena pogreške predstavlja nepristranu pogrešku dobivenu iz rezultata rada algoritma na OOB skupu primjera. Za svaki se primjer iz OOB skupa, nakon što je stablo izgrađeno, dobiva vrijednost klasifikacije na tom stablu. Nakon kraja tog postupka, za svaki se primjer iz originalnog skupa za učenje dobila klasifikacija na približno trećini stabala šume. Iz toga se dobije klasifikacija slučajne šume za sve primjere za učenje. OOB procjena pogreške je prosječan broj svih pogrešnih klasifikacija. Formula 4.3 služi za izračun OOB pogreške [12].  $D$  je početni skup primjera za učenje,  $h(\mathbf{x})$  izlaz algoritma slučajne šume za primjer  $\mathbf{x}$ , a  $y$  razred kojem  $\mathbf{x}$  zaista pripada.  $N$  je ukupan broj primjera za učenje.

$$E_{OOB} = \frac{1}{N} \sum_{\mathbf{x} \in D} \mathbf{1}\{h(\mathbf{x}) \neq y\} \quad (4.3)$$

Druga važna procjena koja se može dobiti korištenjem OOB primjera je procjena važnosti pojedinih značajki. Postupak se svodi na provjeru pogreške na OOB skupu te ponovnu provjeru nakon slučajne permutacije vrijednosti određene značajke. Ovisno o tome koliko su se rezultati promijenili, toliku važnost u određivanju rezultata ima određena značajka.

Taj postupak može se koristiti za odabir značajki. Ukoliko je početni broj značajki prevelik, postupkom procjene važnosti značajki mogu se odrediti najvažnije značajke. Za učenje i rad algoritma mogu se iskoristiti samo najvažnije značajke, ukoliko su

rezultati nad tim podskupom značajki dovoljno dobri. Time se može smanjiti dimenzionalnost problema i skratiti vrijeme izvršavanja algoritma.

## 4.2. Karakteristike algoritma

Algoritam slučajne šume vrlo je popularan algoritam i često se koristi za rješavanje mnogih problema strojnog učenja. Algoritam se može iskoristiti za rješavanje problema klasifikacije, ali i regresije i grupiranja. Svoju popularnost može zahvaliti mnogim dobrim svojstvima koje algoritam pruža.

Kvaliteta i točnost su najvažnija svojstva, a slučajne šume daju rezultate usporedive vrhunskim modernim algoritmima. Vrlo su robustne i omogućavaju jednostavan rad s velikim skupovima podataka i sa njihovom velikom dimenzionalnošću.

Slučajne šume u samom procesu učenja mogu korisniku dati povratnu informaciju. Korištenjem OOB skupa podataka može se dati nepristrana procjena pogreške bez korištenja posebnog, testnog skupa. Moguće je također odrediti koje značajke su bitne u procesu učenja tako što se odredi utjecaj svih pojedinih značajki na ukupnu pogrešku sustava. Algoritam se dobro prilagođava šumu u podacima, a u slučaju nedostajućih podataka, ima metode za procjenu vrijednosti tih podataka.

Algoritam ima vrlo malo parametara i oni imaju vrlo jasna svojstva. Njihovo podešavanje je relativno jednostavno, a još je jednostavnije jer se u kombinaciji sa OOB procjenom pogreške može raditi validacija nad OOB skupom, bez potrebe za generiranjem novog skupa iz skupa primjera za testiranje.

U slučaju da su podaci dobro raspoređeni uz malu razinu šuma, teško dolazi do prenaučivosti. No ako podaci podliježu šumu, moguće je doći do prenaučivosti algoritma, što se može smatrati jednom od mana algoritma. Druga bitna mana je što, za razliku od stabala odluke, algoritam nema jasnu interpretaciju. Nije ga teško shvatiti ili implementirati, ali se kombinacijom velikog broja stabala odluke uz slučajan odabir značajki na slučajno generiranom skupu gubi na jednostavnoj i jasnoj interpretaciji kakvu nudi jedno stablo odluke.

## 5. Implementacija

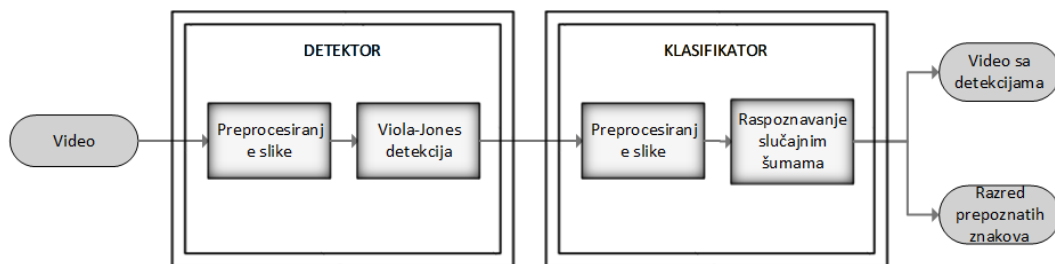
### 5.1. Arhitektura izgrađenog sustava

Izgrađeni sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine sastoji se od tri odvojena modula, vezana uz tri stadija rada sustava. Za svaki od modula razvijena je posebna konzolna aplikacija.

Prvi modul koristi se u fazi razvoja sustava. To je modul koji služi za učenje modela detektora i klasifikatora. Ulaz čine parametri potrebni za učenje koji se razlikuju ovisno o odabranom algoritmu čiji se model želi naučiti. Primjeri ulaznih parametara koje bi svaki algoritam trebao imati su putanja do skupa primjera za učenje i putanja do mjesta na disku gdje bi se naučeni model trebao spremati. Izlaz modula je naučeni model koji se sprema na mjesto zadano ulaznim parametrom.

Drugi modul vezan je uz fazu testiranja rada detektora, klasifikatora ili cjelokupnog sustava. Kao ulaz prima putanju do skupa primjera za testiranje i putanju do datoteke s naučenim modelom. Za svaki od primjera vrši se provjera točnosti i izračunava se konačna pogreška generalizacije zadanog podsustava ili kompletnog sustava. Izlaz ovog modula je datoteka sa spremljenim rezultatima rada.

Treći modul je modul za demonstraciju rada kompletnog sustava i to je finalni proizvod koji je rezultat ovoga rada. Na ulaz prima video sekvencu. Svaka se slička videa dohvati i obrađuje. Dohvaćena slička predaje se podsustavu za detekciju. Slika se preprocesira, normalizira i pretvara u sliku sivih razina. Na tako obrađenoj slici obavlja se proces detekcije koji vraća niz prometnih znakova za ograničenje brzine. Iz tog se niza jedan po jedan izrezani prometni znak za ograničenje brzine predaje podsustavu za raspoznavanje. Svaki se znak ponovno preprocesira, normalizira, pretvara u sliku sivih razina i smanjuje na veličinu  $24 \times 24$  slikovna elemenata. Klasifikator kao izlaz vraća razred prometnog znaka. Sustav zatim na ekran ispisuje razred svih detektiranih prometnih znakova i iscertava sličicu sa svim detektiranim znakovima. Proces se zatim ponavlja za sljedeću sličicu, sve do kraja video sekvence. Prikaz ove arhitekture može se vidjeti na slici 5.1.



**Slika 5.1:** Arhitektura sustava za detekciju i raspoznavanje

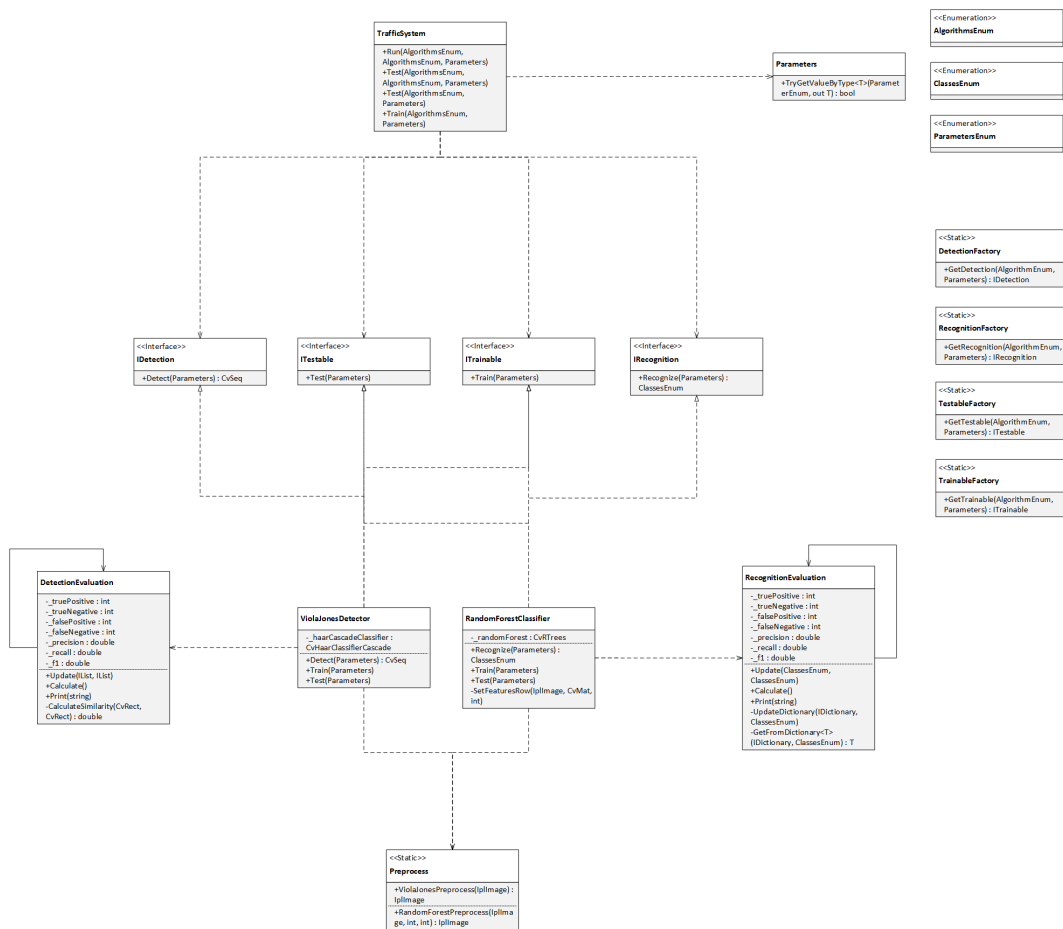
Kao što se može vidjeti iz slike 5.1, sustav kao cjelina se sastoji od dvije velike komponente — detektora i klasifikatora. Podsustav za detekciju na svoj ulaz prima sliku, koju obrađuje i priprema za proces detekcije. Izlaz podsustava je niz pravokutnika koji označavaju niz detektiranih prometnih znakova. Ulaz u podsustav za raspoznavanje je izrezana slika prometnog znaka. Podsustav za raspoznavanje sliku obrađuje i obavlja proces klasifikacije. Izlaz ovog podsustava je razred kojem taj prometni znak pripada. Kombinacijom tih dviju komponenti dobije se cijeli sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine.

## 5.2. Opis implementacije

Sustav je implementiran na način da omogući korisnicima fleksibilnost u odabiru algoritama detekcije i raspoznavanja ili odabiru metoda za preprocesiranje slike. Cijeli sustav sastoji se od tri konzolne aplikacije i jedne biblioteke. Cjelokupna funkcionalnost sustava sadržana je u biblioteci čime se postiže fleksibilnost u odabiru aplikacije koja će koristiti funkcionalnost sustava.

Biblioteka je osmišljena i implementirana kao centralna jedinica koja obavlja kompletnu funkcionalnost sustava i omogućuje korisnicima da je koriste za više različitih aplikacija. Biblioteka je fleksibilna i omogućuje jednostavne promjene algoritama ili metoda, a da se pri tome ne mora mijenjati logika sustava. Dijagram razreda biblioteke može se vidjeti na slici 5.2, dok je sama biblioteka detaljnije objašnjena u nastavku.

Tri su osnovne funkcionalnosti koje biblioteka omogućava — učenje, testiranje i pokretanje. Učenje omogućava korisniku biblioteke pokretanje procesa učenja modela željenog algoritma detekcije ili raspoznavanja. Naučeni model se sprema na disk računala i omogućuje se njegovo korištenje u daljnim koracima rada sustava. Testiranje daje mogućnost provjere rada naučenih algoritama. Testirati se mogu pojedinačni algoritmi ili cijeli sustav kao cjelina. Rezultati testiranja spremaju se u



Slika 5.2: Dijagram razreda biblioteke

datoteku na mjesto na disku koje odabere korisnik pri pokretanju procesa testiranja. Treća, osnovna funkcionalnost biblioteke je pokretanje rada sustava. Pokretanjem se prezentira rad izgrađenog sustava. Kao i kod procesa testiranja sustava, za pokretanje je potrebno sustavu predati putanje do naučenih modela detektora i klasifikatora.

Centralni razred biblioteke koji omogućava pokretanje sve tri funkcionalnosti je TrafficSystem. On predstavlja javno sučelje biblioteke. Sve što klijentske aplikacije trebaju napraviti da bi dobile funkcionalnost sustava je pozvati odgovarajuće metode tog razreda. Metode koje je moguće izvršiti su Train, Test i Run. Sve tri metode primaju iste tipove parametara. To su AlgorithmsEnum i Parameters. Enumeracijom AlgorithmsEnum odabire se koji algoritam se želi koristiti u metodi, dok parametar Parameters predstavlja riječnik parametara koji su potrebni za izvođenje odabrane metode.

Razred Parameters se koristi u gotovo svakom razredu kao jedan od parametara metoda. Ključ tog riječnika je enumeracija ParametersEnum koja predstavlja ograničeni skup parametara koji se mogu koristiti kroz sustav. Ovakvo



rješenje je odabrano zbog fleksibilnosti sustava. Gotovo svaki algoritam ima svoje posebne parametre potrebne za izvršavanje. Da bi se omogućilo korištenje raznih algoritama bez promjena postojećeg koda sustava, metode koriste taj razred za prenošenje svih parametara.

Struktura koja omogućava fleksibilnost sustava u odabiru i implementaciji algoritama je struktura `ITrainable`, `ITestable`, `IDetection` i `IRecognition` sučelja. Svaki algoritam koji implementira ova četiri sučelja može se koristiti u sustavu bez većih promjena u kodu ostatka sustava.

Sučelje `ITrainable` implementiraju algoritmi koji se mogu koristiti u fazi učenja. Sučelje ima jednu metodu `Train` u kojoj razred algoritma koji je implementira mora imati napisanu logiku procesa učenja algoritma. Ako razred algoritma implementira tu metodu, moguće je pokrenuti proces učenja za taj algoritam koristeći infrastrukturu ove biblioteke.

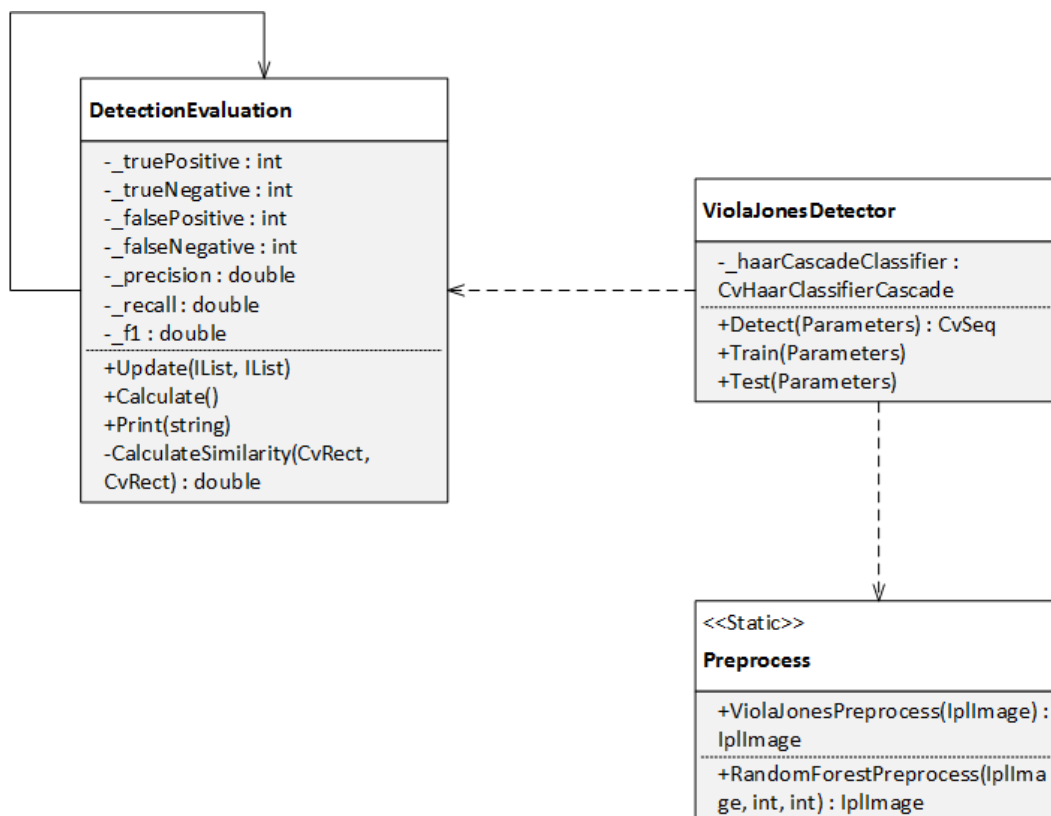
Sučelje `ITestable` omogućava razredima algoritama da implementiraju metodu `Test` koja služi za provođenje procesa testiranja algoritma. Implementacijom te metode testiranja algoritma može se obaviti na razini sistema koristeći istu infrastrukturu biblioteke kao i kod učenja algoritama.

Sučelja `IDetection` i `IRecognition` omogućavaju razredima algoritama detektora odnosno klasifikatora da implementiraju metode koje se koriste kod rada sustava za detekciju i raspoznavanje. Algoritam detekcije koji implementira sučelje za detekciju može se bez većih promjena koristiti kroz cijeli sustav kao odabrani algoritam detekcije. Također, algoritam raspoznavanja može implementirati sučelje za raspoznavanje i koristiti se u sustavu bez većih promjena na ostatku koda. Ovakva arhitektura omogućava testiranje raznih rješenja i pomaže u odabiru korištenih algoritama jer omogućava relativno brzo i jednostavno testiranje novih algoritama.

Implementirani algoritmi su Viola-Jones detektor i klasifikator slučajne šume. Oba algoritma implementiraju prije navedena sučelja. Implementacija tih dviju komponenti sustava detaljnije je opisana u nastavku.

### **5.2.1. Implementacija Viola-Jones detektora**

Podsustav za detekciju Viola-Jones algoritmom prikazan je na slici 5.3. Centralni razred koji implementira sučelja `IDetection`, `ITestable` i `ITrainable` je razred `ViolaJonesDetector`. Implementacija same funkcionalnosti Viola-Jones detektora nalazi se u biblioteci `OpenCV`. Razred `CvHaarClassifierCascade` pruža C# omotač oko `OpenCV` implementacije.



**Slika 5.3:** Dijagram razreda podsustava za detekciju

Metoda `Detect` je implementacija sučelja za detekciju. Ona kao ulaz prima sliku na kojoj se odvija detekcija, a izlaz joj je sekvenca pravokutnika koji obilježavaju mjesto na slici gdje se prometni znakovi za ograničenje brzine nalaze. Metode `Train` i `Test` implementacije su sučelja za učenje i testiranje algoritma. Metoda za učenje koristi postojeće metode OpenCV biblioteke za učenje Viola-Jones detektora. Parametri potrebni za metodu učenja su putanje do pozitivnih primjera slika na kojima se nalaze znakovi za ograničenje brzine, putanje do negativnih primjera slika, putanje do direktorija u koji će se spremiti naučena kaskada te ukupnog broja pozitivnih i negativnih primjera za učenje. Metoda za testiranje na ulaz prima putanju do naučene kaskade, putanju do skupa za testiranje i putanju do datoteke gdje će se spremiti rezultati testiranja.

U procesu testiranja, za evaluaciju i generiranje rezultata koristi se pomoćni razred `DetectionEvaluation`. Nakon svake detekcije znakova na slici, poziva se metoda `Update` koja osvježava trenutno stanje brojača rezultata. Ona prima na ulaz dvije liste, listu znakova koje je sustav detektirao i listu znakova koji su trebali biti detektirani. Nakon što testiranje završi i izvrši se brojanje točno i netočno detektiranih znakova, poziva se metoda `Calculate` koja izračuna sve potrebne

vrijednosti koje se zatim korištenjem metode `Print` ispisuju u datoteku.

Za određivanje točnosti detekcije koristi se Jaccardov koeficijent sličnosti. Uspoređuje se preklapanje dvaju pravokutnika, pravokutnika detekcije sustava i stvarne pozicije znaka. Mjera koja se dobije kao kvocijent presjeka dvaju pravokutnika i njihove unije predstavlja mjeru sličnosti dva pravokutnika. Formula 5.1 koristi se za izračun koeficijenta sličnosti [14]. Ukoliko je taj koeficijent veći od 0.5, detekcija se računa kao ispravna.

$$J(S, G) = \frac{|S \cap G|}{|S \cup G|} \quad (5.1)$$

Razred `Preprocess` je statični razred koji sadrži metode za postupak preprocesiranja slika. Metoda `ViolaJonesPreprocess` služi za preprocesiranje slike prije izvršavanja detekcije. Preprocesiranje u procesu detekcije Viola-Jones algoritmom se odnosi na normalizaciju histograma i pretvaranje ulazne slike u boji u sliku sivih razina.

### 5.2.2. Implementacija klasifikatora slučajne šume

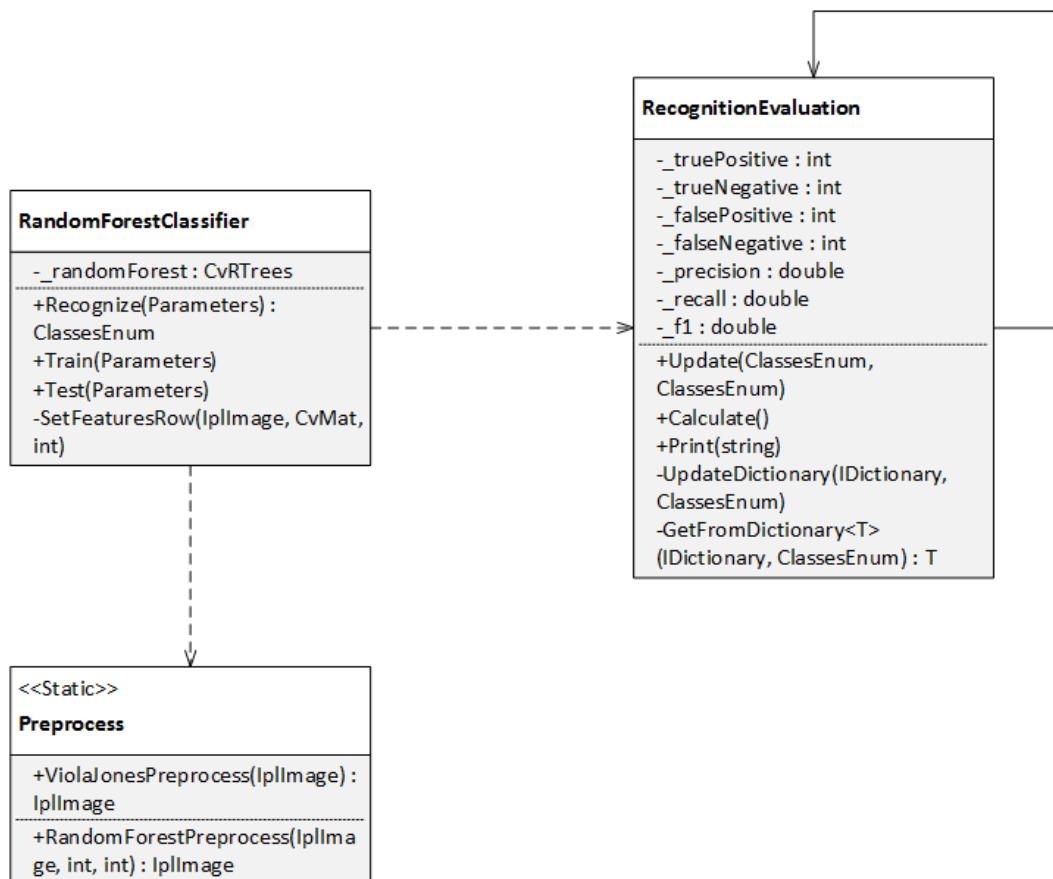
Na slici 5.4 prikazan je podsustav za raspoznavanje korištenjem algoritma slučajne šume. Razred `RandomForestClassifier` je razred koji sadrži implementaciju sučelja `IRecognition`, `ITestable` i `ITrainable`. Kao i u slučaju detektora, implementacija samog algoritma slučajne šume nalazi se u biblioteci `OpenCV` i ona se poziva preko razreda omotača `CvRTrees`.

Implementacija metode sučelja za raspoznavanje `Recognize` kao ulaz prima sliku koju treba klasificirati. Ulazna slika je jedan od znakova za ograničenje brzine i ova metoda vraća o kojem se konkretno znaku za ograničenje brzine radi.

Slika dobivena na ulaz metode se najprije obrađuje. Obrada slike obavlja se uz pomoć statičnog razreda `Preprocess`. Metoda `RandomForestPreprocess` prima ulaznu sliku i veličinu na koju se ulaznu sliku želi svesti. Ulazna slika se zatim smanjuje, normalizira joj se histogram i pretvara se u sliku sivih razina. Veličina na koju se slika smanjuje je  $24 \times 24$  slikovna elementa.

Slikovni elementi takve obrađene slike predstavljaju značajke klasifikatora. U procesu raspoznavanja ne koriste se neke napredne značajke, već se koriste obične vrijednosti slikovnih elemenata u rasponu od 0 do 255. Pošto je veličina slike  $24 \times 24$  to ukupno daje 576 značajki.

Metode `Test` i `Train`, kao i kod detektora, koriste se za proces učenja i testiranja. Metoda za učenje na ulaz prima putanju do skupa podataka za učenje i putanju do



**Slika 5.4:** Dijagram razreda podsustava za raspoznavanje

datoteke gdje će se nakon završetka procesa učenja spremiti naučeni model. Metoda za testiranje uz putanje do skupa podataka i modela prima i putanju do datoteke gdje će biti spremljeni rezultati testiranja klasifikatora.

Također jednako kao kod detektora, za evaluaciju i račun pogreške kod testiranja koristi se poseban razred, *RecognitionEvaluation*. Za svaki od razreda se spremaju vrijednosti lažnih i ispravnih, pozitivnih i negativnih klasifikacija. Iz tih se vrijednosti izračunavaju sve potrebne mjere poput preciznosti i odziva te se svi rezultati spremaju nakon izvršavanja procesa testiranja u zadanu datoteku.

### 5.3. Korištene tehnologije i biblioteke

Sustav je izrađen koristeći Microsoft tehnologije. Izrada se odvijala na Windows platformi koristeći programski jezik C# i razvojni okvir .NET verzije 4.5.1 [15]. Programski kod napisan je koristeći razvojno okruženje Visual Studio 2013 [16]. Za prilagodbu slika i anotacija koristio se programski jezik Python verzije 2.7.6 [17].

Korišteni algoritmi detekcije, raspoznavanja i obrade slika dio su biblioteke OpenCV [18]. Korištena je verzija 2.4.8 biblioteke. Biblioteka OpenCV pisana je u programskim jezicima C++ i C te je za potrebe korištenja biblioteke bio potreban omotač za pružanje funkcionalnosti jezicima .NET platforme. Korištena je biblioteka OpenCvSharp verzije 2.4.8 [19]. Pošto biblioteka omogućuje jednostavan pristup samo C sučelju OpenCV biblioteke, to sučelje se koristilo pri izradi ovoga sustava.

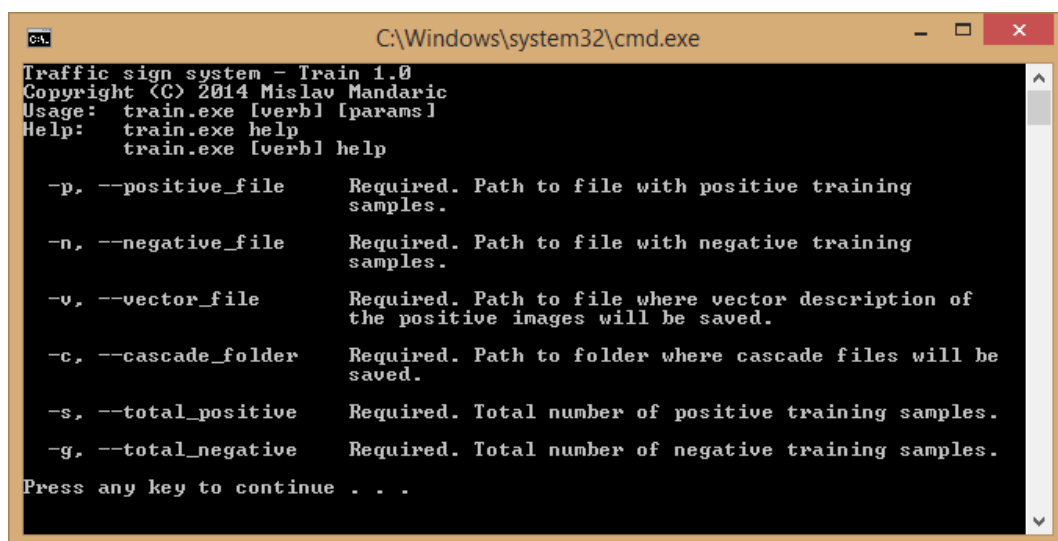
Pri izradi konzolnih aplikacija korištena je biblioteka CommandLine verzije 1.9.71 [20] za jednostavno parsiranje argumenata predanih na ulaz konzolne aplikacije.

Zbog odabira .NET tehnologije, sustav je ograničen na Windows platformu i to na verzije Windows Vista i novije. Odabrana je ta tehnologija zbog jednostavnosti razvoja aplikacije i prijašnjeg iskustva u korištenju te tehnologije.

## 5.4. Korištenje sustava

U sklopu rada izrađene su tri konzolne aplikacije, svaka za jednu od faza razvoja sustava. Aplikacije su vrlo jednostavne jer je sustav implementiran kao biblioteka i aplikacije služe za pojednostavljeno unošenje parametara sustava.

Prva konzolna aplikacija koristi se u fazi učenja algoritama. Odabirom algoritma kojeg se želi učiti odabire se način rada aplikacije i kao naredba se šalje naziv algoritma. Implementirana aplikacija ima dvije naredbe, jednu za učenje Viola-Jones algoritma i jednu za učenje algoritma slučajne šume. Svaka naredba ima posebnu listu obaveznih parametara koji se moraju proslijediti aplikaciji pri pokretanju.



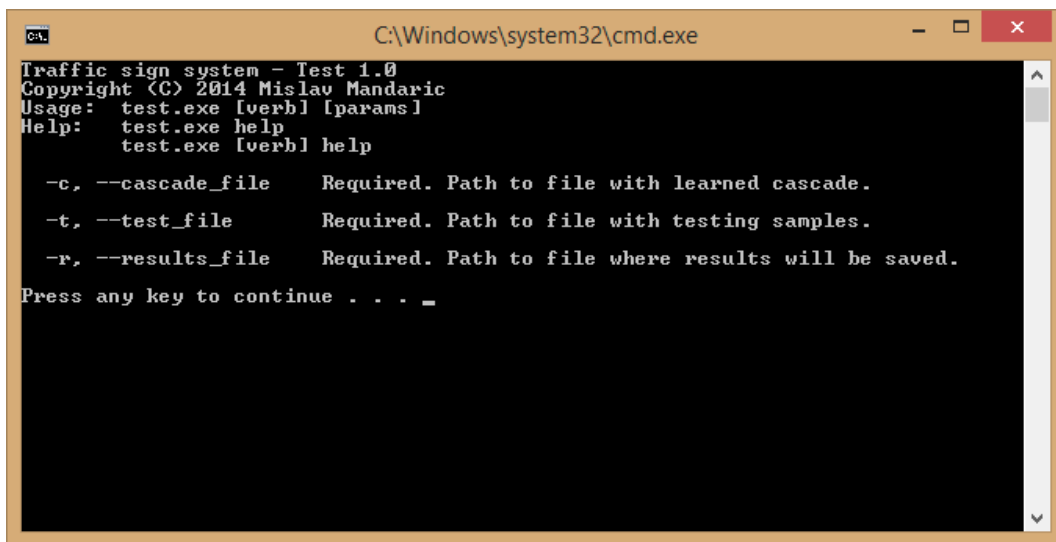
```
C:\Windows\system32\cmd.exe
Traffic sign system - Train 1.0
Copyright (C) 2014 Mislav Mandaric
Usage: train.exe [verb] [params]
Help: train.exe help
      train.exe [verb] help

-p, --positive_file      Required. Path to file with positive training
                           samples.
-n, --negative_file      Required. Path to file with negative training
                           samples.
-v, --vector_file        Required. Path to file where vector description of
                           the positive images will be saved.
-c, --cascade_folder     Required. Path to folder where cascade files will be
                           saved.
-s, --total_positive     Required. Total number of positive training samples.
-g, --total_negative     Required. Total number of negative training samples.
Press any key to continue . . .
```

Slika 5.5: Početni ekran aplikacije za učenje Viola-Jones detektora

Na slici 5.5 prikazan je početni ekran aplikacije za učenje Viola-Jones detektora. Parametri koje se mora unijeti u sustav kao parametre aplikacije su putanja do pozitivnih primjera, putanja do negativnih primjera, putanja do vektorskog zapisa pozitivnih primjera, putanja do direktorija gdje će se spremati naučena kaskada te ukupan broj pozitivnih i negativnih primjera. Aplikacija za učenje klasifikatora slučajne šume kao parametre ima putanju do primjera za učenje, putanju do modela i ukupan broj primjera za učenje.

Druga aplikacija služi za testiranje rada implementiranih algoritama. Aplikacija, kao i aplikacija za učenje, ima onoliko naredbi koliko ima implementiranih algoritama za testiranje. Implementirani su algoritmi Viola-Jones i slučajne šume te i ova aplikacija ima dvije naredbe, svaka sa svojom listom parametara.



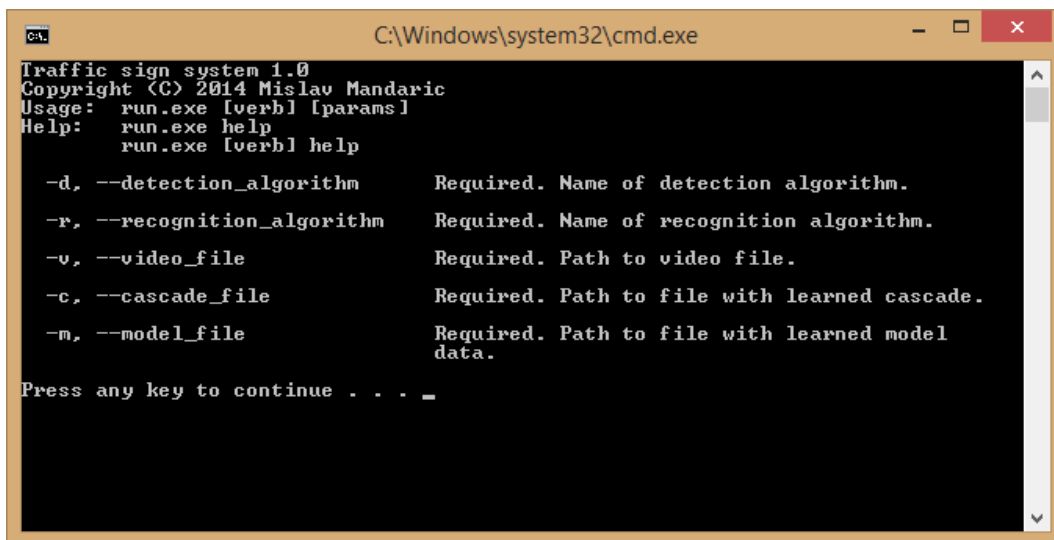
```
C:\Windows\system32\cmd.exe
Traffic sign system - Test 1.0
Copyright (C) 2014 Mislav Mandaric
Usage: test.exe [verb] [params]
Help: test.exe help
      test.exe [verb] help

-c, --cascade_file    Required. Path to file with learned cascade.
-t, --test_file       Required. Path to file with testing samples.
-r, --results_file    Required. Path to file where results will be saved.
Press any key to continue . . . _
```

**Slika 5.6:** Početni ekran aplikacije za testiranje Viola-Jones detektora

Slika 5.6 prikazuje početni ekran aplikacije za testiranje Viola-Jones detektora. Parametri potrebni za rad su putanja do skupa primjera za testiranje, putanja do naučene kaskade i putanja do datoteke gdje će biti spremljeni rezultati testiranja. Parametri naredbe za učenje algoritma slučajne šume su isti: putanja do skupa za testiranje, datoteke s rezultatima i datoteke sa naučenim modelom.

Treća aplikacija koristi se za testiranje rada cijelog sustava. Sustav se može testirati na dva načina. Prvi je test sličan onom koji se provodi korištenjem druge aplikacije, ali se ovaj test obavlja nad cjelovitim sustavom. Drugi test je zapravo demonstracija rada sustava nad video sekvencom. Obzirom na ta dva načina testiranja, aplikacija ima dvije naredbe. Prva naredba je `Video` kojom se pokreće demonstracija na videu, a druga naredba je `Images` kojom se pokreće testiranje na zadanom skupu slika.



```
C:\Windows\system32\cmd.exe
Traffic sign system 1.0
Copyright (C) 2014 Mislav Mandaric
Usage: run.exe [verb] [params]
Help: run.exe help
      run.exe [verb] help

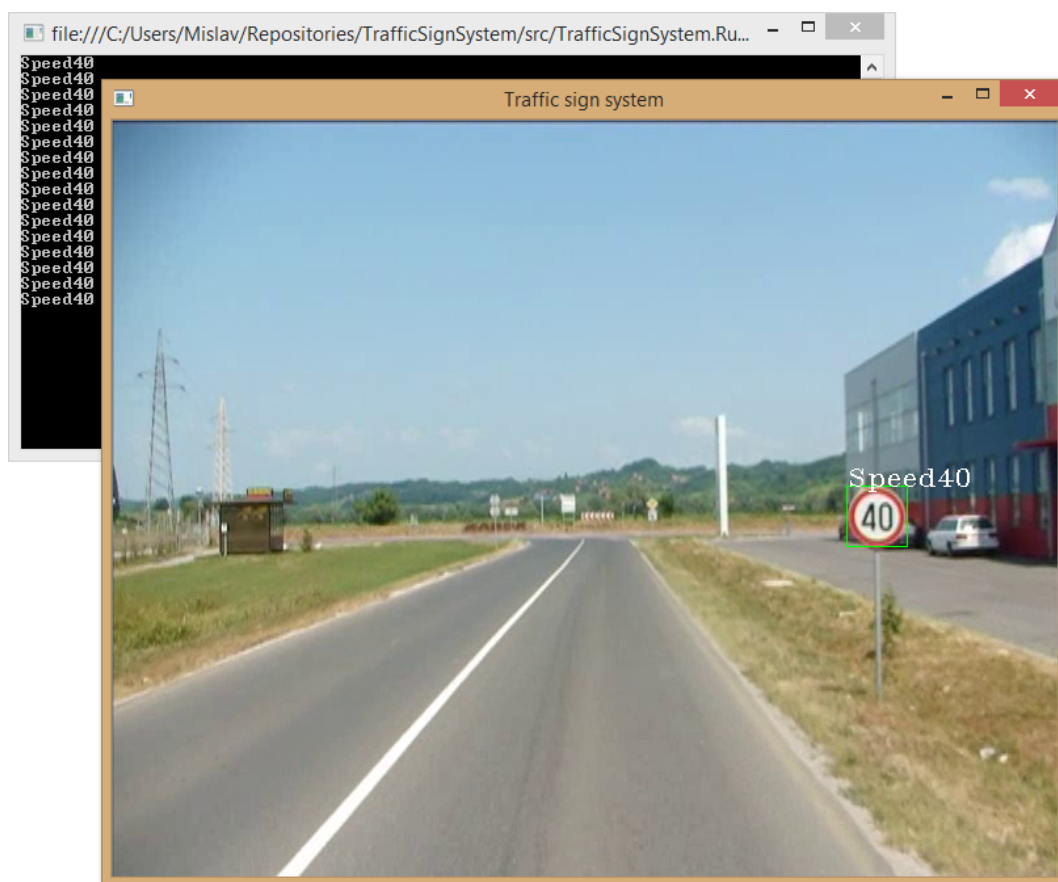
-d, --detection_algorithm    Required. Name of detection algorithm.
-r, --recognition_algorithm  Required. Name of recognition algorithm.
-v, --video_file             Required. Path to video file.
-c, --cascade_file           Required. Path to file with learned cascade.
-m, --model_file             Required. Path to file with learned model
                             data.

Press any key to continue . . . _
```

**Slika 5.7:** Početni ekran aplikacije za demonstraciju rada sustava

Početni ekran aplikacije za demonstraciju rada sustava na video sekvenci prikazan je na slici 5.7. Prije početka rada aplikacije potrebno je putem parametara odrediti koji će se algoritmi za detekciju i raspoznavanje koristiti u radu sustava, kao i putanje do naučene kaskade i modela koje ti algoritmi koriste. Osim ta četiri parametra, potrebno je predati putanju do željene video sekvence. Ukoliko se odabere naredba za testiranje rada sustava na skupu slika, osim prva četiri parametra kao i kod pokretanja na videu, potrebno je kao parametre predati putanju do testnog skupa slika i putanju do datoteke gdje će biti spremljeni rezultati testiranja.

Demonstracija rada sustava na videu može se vidjeti na slici 5.8. Pokretanjem aplikacije za demonstraciju pokreće se testni video. Svaka se slička videa obrađuje i rezultat se ispisuje na samoj video sekvenci, a rezultati raspoznavanja ispisuju se i na konzoli. Kao što se na slici može vidjeti, ispis rezultata podrazumjeva iscrtavanje zelenog pravokutnika oko detektiranih znakova za ograničenje brzine, ispis o kojem se znaku radi iznad pravokutnika te ispis o kojem se znaku radi u konzoli.



**Slika 5.8:** Prikaz rada aplikacije za demonstraciju rada sustava



## 6. Rezultati

### 6.1. Skup podataka

Korišteni skup podataka pripada bazi podataka MASTIF (engl. *Mapping and Assessing the State of Traffic control InFrastructure*) [21]. Baza MASTIF sastoji se od tri odvojena skupa podataka. Skup TS2009 pripada skupu video sekvenci prikupljenih 2009. godine, skup TS2010 je skup prikupljen 2010., a skup TS2011 je skup prikupljen 2011. godine. Svaki skup sastoji se od više anotiranih video sekvenci hrvatskih cesta. Svaki video sadrži veći broj prometnih znakova, uključujući i znakove za ograničenje brzine. Znakovi su se anotirali na način da je jedan znak anotiran u više sličica videa, čime su se dobili različiti pogledi na jedan znak u više trenutaka.

Cijeli skup MASTIF dodatno je obrađen i prilagođen problemu detekcije i raspoznavanja prometnih znakova za ograničenje brzine. Iz cijelog skupa MASTIF iskorištene su spremljene cijele sličice videa. Iz skupa su izdvojene sličice koje sadrže znakove za ograničenje brzine. Sve te sličice su podijeljene u razrede, po znakovima za ograničenje brzine. Dobilo se pet razreda: 30 km/h, 40 km/h, 50 km/h, 60 km/h i 70 km/h. Svaki od razreda je podijeljen u dva skupa, skup za učenje i skup za testiranje. Skupovi su podijeljeni tako da 70% skupa čini skup za učenje, a ostalih 30% je skup za testiranje.

Takve cijele i anotirane slike predstavljaju skup pozitivnih primjera za postupak detekcije. Negativan skup primjera slika bez znakova za ograničenje brzine je također prikupljen i podijeljen u skup za učenje i testiranje. Broj slika negativnog skupa je dvostruko veći od broja slika pozitivnog skupa. Primjer jedne slike iz skupa pozitivnih slika za detekciju je slika 6.1.

Skup za raspoznavanje izgrađen je na način da su cijele slike koje su prikupljene, podijeljene u razrede i podijeljene u skupove za učenje i testiranje, jednostavno izrezane. Tako se dobio stratificirani skup primjera za učenje i testiranje podijeljen u 70:30 omjeru. Primjer slika koje se koriste u procesu raspoznavanja može se vidjeti



**Slika 6.1:** Primjer slike iz pozitivnog skupa za detekciju

na slici 6.2. Za svaki od razreda prikazana je jedna slika. Pošto su anotirane slike znakova iz različitih trenutaka i udaljenosti od kamere, skup čine slike raznih veličina.



**Slika 6.2:** Primjer jedne slike za svaki razred iz skupa za raspoznavanje

Konačni skup za učenje detektora sastoji se od 516 pozitivnih slika za učenje na kojima se nalazi 523 znaka i 1032 negativne slike za učenje. Konačni skup primjera za učenje klasifikatora sastoji se od 523 znaka podijeljenih u 5 razreda.

Konačni skup za testiranje detekora sastoji se od 219 pozitivnih slika sa 224 znaka i 438 slika bez znakova. Skup za testiranje klasifikatora sastoji se od 224 znaka također podijeljenih u 5 razreda.

Iz skupa MASTIF je izdvojen i jedan video koji se koristi za testiranje i

demonstraciju rada sustava. Taj video sadrži samo jedan znak i traje približno tri minute. U svrhu testiranja, koristi se prije svega za provjeru broja lažno pozitivnih odziva sustava.

## 6.2. Metodologija obrade rezultata

Podsustavi za detekciju i raspoznavanje, kao i cijeli sustav, testiraju se nad posebnim skupom podataka. Skup podataka za testiranje potpuno je nezavisan od skupa za učenje i pogreška dobivena nad tim skupom je generalizacijska pogreška koja govori koliko dobro sustav radi na stvarnim, neviđenim primjerima.

Testiranje se odvija na način da se prethodno anotirane slike klasificiraju ili detektiraju i dobiveni rezultati se uspoređuju sa stvarnim podacima koji se nalaze u anotaciji. Ukoliko je klasifikacija ili detekcija sustava ista kao ona koja se nalazi u anotacijskoj datoteci, tada je riječ o ispravnoj detekciji ili klasifikaciji. Ukoliko se one razlikuju, detekcija ili klasifikacija nije ispravna. Osim podjele po ispravnosti, postoji podjela u ovisnosti o tome što je izlaz sustava. Iz tih podjela, skup primjera za testiranje može se podijeliti u četiri kategorije: ispravno pozitivni ( $TP$ ), ispravno negativni ( $TN$ ), lažno pozitivni ( $FP$ ) i lažno negativni primjeri ( $FN$ ).

Jednom kada se dobiju vrijednosti za te skupove, moguće je izraziti kvalitetu nekog sustava. Mjere koje se koriste su preciznost, odziv i f-mjera [1].

Preciznost predstavlja udio točnih pozitivnih izlaza u skupu svih pozitivnih izlaza sustava. Formula za preciznost je formula 6.1.

$$P = \frac{TP}{TP + FP} \quad (6.1)$$

Odziv je udio točnih pozitivnih detekcija u skupu pozitivnih primjera za testiranje. Pošto je skup primjera za testiranje unija skupa točno pozitivnih i lažno negativnih izlaza, formula za odziv odgovara formuli 6.2.

$$R = \frac{TP}{TP + FN} \quad (6.2)$$

Treća i najvažnija mjera je kombinacija prethodnih dviju mjera. F-mjera je harmonijska sredina preciznosti i odziva. Formula za f-mjeru je formula 6.3.

$$F_1 = \frac{2PR}{P + R} \quad (6.3)$$

Razlikujemo makro i mikro uprosječenu f-mjeru. Makro f-mjera dobije se kao prosječna vrijednost f-mjera koje se dobiju za svaki razred posebno, dok je mikro

obična f-mjera dobivena iz preciznosti i odziva svih razreda zajedno. Makro i mikro f-mjera koriste se kod prikaza rezultata višeklasnih sustava.

### 6.3. Rezultati detekcije

Testiranje detekcije obavljeno je na slikama iz skupa za testiranje. Naučene su i testirane četiri kaskade Viola-Jones algoritma. Svaka kaskada ima tri parametra. To su: broj razina kaskade, stopa točnosti detekcija u jednoj razini kaskade i stopa lažno pozitivnih detekcija u jednoj razini kaskade. Predviđena stopa točnih detekcija i stopa lažno pozitivnih detekcija cijele kaskade predstavlja potenciju vrijednosti tih parametara za jednu razinu na broj razina kaskade, što se može vidjeti iz formula 6.4 i 6.5.

$$rate_{TP} = minHitRate^{stages} \quad (6.4)$$

$$rate_{FP} = maxFalseRate^{stages} \quad (6.5)$$

Tablica 6.1 prikazuje sve parametre svih naučenih kaskada, a tablica 6.2 prikazuje rezultate rada Viola-Jones detekora na skupu slika za testiranje.

**Tablica 6.1:** Parametri Viola-Jones algoritma za sve četiri kaskade

Naziv	Vrijednost			
	Prva k.	Druga k.	Treća k.	Četvrta k.
Broj razina kaskade	12	12	15	15
Stopa točnosti detekcija	0.996	0.996	0.996	0.996
Stopa lažno pozitivnih detekcija	0.4	0.3	0.4	0.33

Rezultati su uglavnom očekivani. Prve dvije kaskade odbacuju manji broj znakova kao lažno negativne i imaju bolji odziv. Povećanjem broja razina kaskade odziv pada, ali raste preciznost jer se smanjuje broj lažno pozitivnih odziva detektora. Druga i treća kaskada, iako vrlo različite, imaju usporedive rezultate. Tu se može vidjeti utjecaj parametra za određivanje stope lažno pozitivnih primjera. Iako kraća za tri razine, druga kaskada ima bolji odziv i jednaku preciznost kao i treća na skupu primjera za testiranje.

Četvrta kaskada ima najmanju razinu lažno pozitivnih znakova. Zbog dugačke kaskade i relativno niskog parametra stope lažno pozitivnih znakova, problem lažno

**Tablica 6.2:** Rezultati detektora na skupu za testiranje

	Prva k.	Druga k.	Treća k.	Četvrta k.
$TP$	187	174	172	161
$FP$	188	9	10	1
$FN$	37	50	52	63
$P$	0.5	0.95	0.95	0.99
$R$	0.83	0.78	0.77	0.72
$F_1$	<b>0.62</b>	<b>0.86</b>	<b>0.85</b>	<b>0.83</b>

pozitivnih znakova je minimiziran. Iako odziv izgleda dosta nizak, on je i dalje dovoljno dobar. U odnosu na ostale kaskade koje imaju veći odziv, većina znakova koji na ovoj kaskadi nisu detektirani su znakovi koji tek ulaze u kadar. Pošto nije krucijalno svaki znak prepoznati u svakoj slicici u kojoj se nađe, kaskada s većim brojem lažno negativnih, ukoliko dobro rješava problem lažno pozitivnih znakova, može biti dovoljno dobro rješenje.

## 6.4. Rezultati raspoznavanja

Testiranje klasifikatora odvija se na skupu primjera za testiranje. Algoritam slučajnih šuma ima četiri parametra koja je potrebno podesiti. Prvi parametar je maksimalna dozvoljena dubina stabala odluke, drugi parametar je minimalan broj primjera koji je potrebno imati u podstablu, treći parametar je broj stabala slučajne šume, a posljednji, četvrti parametar je broj značajki koje će se koristiti kod grananja svakog čvora svakog stabla šume. Najznačajniji od ta četiri parametra je posljednji parametar kojim se određuje međuovisnost stabala u šumi i snaga svakog pojedinog stabla.

Svi parametri određeni su postupkom provjere na skupu za validaciju. Validacijski skup iskorišten za postupak provjere je OOB skup primjera. Konačne vrijednosti parametara dane su u tablici 6.3, a konačni rezultati rada algoritma mogu se vidjeti u tablici 6.4.

**Slika 6.3:** Primjeri slika znakova koji su pogrešno klasificirani

**Tablica 6.3:** Parametri algoritma slučajne šume

Naziv	Vrijednost
Maksimalna dubina	16
Minimalan broj primjera	8
Broj stabala	200
Broj značajki	34

**Tablica 6.4:** Rezultati klasifikatora na skupu za testiranje

	30 km/h	40 km/h	50 km/h	60 km/h	70 km/h	Ukupno
$TP$	59	61	25	36	26	207
$TN$	154	153	194	182	196	879
$FP$	8	6	2	1	0	17
$FN$	3	4	3	5	2	17
$P$	0.88	0.91	0.93	0.97	1.0	0.92
$R$	0.95	0.94	0.89	0.88	0.93	0.92
$MF_1$	0.91	0.92	0.91	0.92	0.96	<b>0.93</b>
$mF_1$						0.92

Konačni rezultati klasifikatora slučajne šume vrlo su dobri. Klasifikator se pokazao vrlo dobrim i robustnim rješenjem problema koji ima jako dobro svojstvo generalizacije. Neki od primjera koji se nisu dobro klasificirali nalaze se na slici 6.3. Kao što se može vidjeti, neke od znakova i čovjek teško raspoznaje. Uglavnom su loše klasificirane male slike, loše izrezane slike ili slike koje sadrže jako puno šuma.

## 6.5. Ukupni rezultati sustava

Testiranje sustava kao cjeline obavljeno je na istom skupu slika za testiranje kao i detektor, ali i na jednoj testnoj video sekvenci. Modeli korišteni za rad sustava su četvrta naučena kaskada Viola-Jones algoritma sa 15 razina i naučeni model slučajnih šuma uz grananje sa 34 značajke. Tablica 6.5 prikazuje konačne rezultate sustava na skupu za testiranje. Rezultati po razredima su rezultati samo skupa ispravno detektiranih znakova, a ukupni rezultati su kombinacija rezultata detektora i rezultata klasifikatora na skupu ispravno detektiranih znakova.

**Tablica 6.5:** Rezultati sustava na skupu za testiranje

	30 km/h	40 km/h	50 km/h	60 km/h	70 km/h	Ukupno
$TP$	49	47	21	25	11	153
$TN$	107	110	136	135	148	636
$FP$	4	3	1	0	0	9
$FN$	1	1	3	1	2	71
$P$	0.92	0.94	0.96	1.0	1.0	0.94
$R$	0.98	0.98	0.88	0.96	0.85	0.68
$MF_1$	0.95	0.96	0.91	0.98	0.92	0.94
$mF_1$						<b>0.79</b>

Zbog činjenice da se makro f-mjera računa samo na skupu dobro detektiranih znakova, ta vrijednost prikazuje kvalitetu rada klasifikatora na skupu dobro detektiranih znakova, a ne kvalitetu rada sustava. Najbolja mjera za kvalitetu sustava je mikro f-mjera, koja uzima u obzir kvalitetu klasifikatora i detektora.

Rezultati testiranja sustava na video sekvenci mogu se vidjeti na tablici 6.6. Duljina trajanja videa je oko tri minute. Pošto se na videu nalazi samo jedan znak, video se koristi primarno za provjeru broja lažno pozitivnih vrijednosti. Također, zbog postojanja samo jednog znaka nije potrebno računati makro i mikro f-mjere.

**Tablica 6.6:** Rezultati sustava na testnom videu

	Ukupno
$TP$	18
$TN$	0
$FP$	2
$FN$	2
$P$	0.9
$R$	0.9
$F_1$	<b>0.9</b>

Test na video sekvenci jedan je od osnovnih razloga odabira dulje i strože četvrte kaskade detektora. Video sekvenca koja traje samo tri minute, sadrži 4.300 sličica. Ako se na videu nalazi samo jedan znak, on se pojavljuje u približno 20 sličica. To je jako mali broj sličica koje sadrže znak i pretpostavka je da će detektor teško naučiti

odvojiti znak iz svih pozadina te da će broj lažno pozitivnih odziva uz kratku kaskadu biti prevelik.

## 6.6. Problemi sustava

Izgrađeni sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine koristi algoritme opće namjene za rješavanje tog problema. Ne koriste se napredne tehnike obrade slike ili izvlačenja značajki, već se problem pokušava riješiti na što jednostavniji način. Tako izgrađen sustav ima dobre rezultate i daljnje istraživanje u ovom smjeru moglo bi dovesti do dodatnih poboljšanja i još boljih rezultata.

Kao i kod svakog problema detekcije, velik problem stvaraju lažno pozitivni odzivi detektora. Ovaj sustav je taj problem riješio korištenjem malo dulje kaskade i niskim parametrom stope lažno pozitivnih detekcija, što se očitovalo povećanjem broja lažno negativnih znakova. Jedno od drugih mogućih rješenja ovog problema moglo bi biti dodavanje dodatnog koraka između kraja rada detektora i početka procesa klasificiranja. Izlaz iz kaskade može se prije procesa detekcije provesti kroz dodatni filter koji bi služio isključivo za odbacivanje lažno pozitivnih detekcija.

Drugo moguće rješenje moglo bi biti prepuštanje klasifikatoru rješavanje problema lažno pozitivnih detekcija. U procesu učenja klasifikatora može se koristiti skup koji sadrži primjere za svaki od razreda znakova za ograničenje brzine, uz dodatne primjere iz razreda lažno pozitivnih detekcija. Ti primjeri mogli bi biti iz skupa lažno pozitivnih odziva detektora skupa primjera za učenje detektora. Klasifikator naučen da raspoznaje lažno pozitivne znakove mogao bi takve izlaze detektora ispravno klasificirati i tijekom rada sustava.

Iako ima sposobnost rada u realnom vremenu, tijekom rada sustava zna se dogoditi smanjen broj obrađenih sličica u sekundi. Ukoliko se na slici nalazi više znakova ili više objekata koji sliče znakovima, to uzrokuje dulji proces detekcije jer mnogo podprozora slike tijekom obrade prolazi kroz veći broj razina kaskade. Rješenje ovog problema moglo bi biti korištenje kraće kaskade, no to bi dovelo do problema sa lažno pozitivnim odzivima detektora.

U procesu klasifikacije, kao značajke sustav koristi vrijednosti slikovnih elemenata slike sivih razina. Te značajke vrlo su jednostavne i korištenjem nekih naprednijih značajki sustav bi mogao dobiti bolje rezultate klasifikatora. Značajke koje bi mogle dati bolje rezultate su značajke histograma orijentiranih gradijenata, koje bi u kombinaciji sa slučajnim šumama mogle dati dobre rezultate.



## 7. Zaključak

Izgrađeni sustav za detekciju i raspoznavanje prometnih znakova za ograničenje brzine dobar je pokazatelj općenitog pristupa rješavanja problema detekcije i raspoznavanja objekata.

Iskorišteni algoritam detekcije Viola-Jones pokazao je dobra svojstva. Detektor je naučen da dobro prepoznaje znakove za ograničenje brzine i ostali tipovi znakova ne stvaraju veće probleme. No problem lažno pozitivnih odziva detektora ipak postoji. Povećanjem broja razina kaskade i smanjenjem parametra stope lažno pozitivnih detekcija u procesu učenja problem je riješen, ali uz posljedicu smanjenja točnih detekcija. Takvo kompromisno rješenje dovoljno je dobro iz razloga što smanjenje točnosti detekcije na skupu za testiranje ne znači da je manji broj fizičkih znakova detektiran, već se znakovi koji tek ulaze u kadar i male su veličine teže detektiraju.

Algoritam slučajne šume pokazao se kao jako dobro rješenje zadanog problema raspoznavanja prometnih znakova za ograničenje brzine. Algoritam je dovoljno robustan da rezultat klasifikacije detektiranih znakova koji nisu savršeno dobro izrezani bude istovjetan rezultatu prethodno izrezanih znakova. Algoritam bi možda mogao imati i bolje rezultate da se koriste naprednije značajke od vrijednosti slikovnih elemenata slike sive boje.

Sustav kao cjelina postigao je dobre rezultate i na skupu za testiranje i na testnoj video sekvenci. Uz rezultat od 0.79 za  $mF_1$  na skupu za testiranje i 0.9 za  $F_1$  na testnoj video sekvenci, može se reći da je sustav dao vrlo dobre rezultate. Uz neka poboljšanja u vidu rješavanja problema lažno pozitivnih detekcija u koraku nakon detekcije, korištenju kraće kaskade za postizanje većeg broja ispravnih detekcija i korištenje naprednijih značajki u procesu klasifikacije, sustav bi se mogao unaprijediti i davati još bolje rezultate.

# LITERATURA

- [1] Jan Šnajder i Bojana Dalbelo Bašić. Strojno učenje. Skripta za predmet Strojno učenje na Fakultetu elektrotehnike i računarstva, Sveučilište u Zagrebu.
- [2] Paul Viola i Michael Jones. Rapid object detection using a boosted cascade of simple features. U *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, svezak 1, stranice 511–518. IEEE, 2001.
- [3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [4] Anđelo Martinović, Goran Glavaš, Matko Juribašić, Davor Sutić, i Zoran Kalafatić. Real-time detection and recognition of traffic signs. U *MIPRO, 2010 Proceedings of the 33rd International Convention*, stranice 760–765. IEEE, 2010.
- [5] Miguel Angel Garcia-Garrido, Miguel Angel Sotelo, i Ernesto Martm-Gorostiza. Fast traffic sign detection and recognition under changing lighting conditions. U *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, stranice 811–816. IEEE, 2006.
- [6] Karla Brkić, Siniša Šegvić, Zoran Kalafatić, Ivan Sikirić, i Axel Pinz. Generative modeling of spatio-temporal traffic sign trajectories. U *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, stranice 25–31. IEEE, 2010.
- [7] Siniša Šegvić, Karla Brkić, Zoran Kalafatić, i Axel Pinz. Exploiting temporal and spatial constraints in traffic sign detection from a moving vehicle. *Machine Vision and Applications*, stranice 1–17, 2011.
- [8] Marcin L. Eichner i Toby P. Breckon. Integrated speed limit detection and recognition from real-time video. U *Intelligent Vehicles Symposium, 2008 IEEE*, stranice 626–631. IEEE, 2008.

- [9] Fatin Zaklouta, Bogdan Stanculescu, i Omar Hamdoun. Traffic sign classification using kd trees and random forests. U *Neural Networks (IJCNN), The 2011 International Joint Conference on*, stranice 2151–2155. IEEE, 2011.
- [10] Igor Bonači, Ivan Kusalić, Ivan Kovaček, Zoran Kalafatić, i Siniša Šegvić. Addressing false alarms and localization inaccuracy in traffic sign detection and recognition. U *16th computer vision winter workshop*, stranice 1–8, 2011.
- [11] Yoav Freund, Robert Schapire, i Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.
- [12] Leo Breiman i Adele Cutler. Random forests, Travanj 2014. URL <http://www.stat.berkeley.edu/~breiman/RandomForests/>.
- [13] Kardi Teknomo. Decision tree learning, Lipanj 2014. URL <http://people.revoledu.com/kardi/tutorial/DecisionTree/index.html>.
- [14] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, i Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. U *Neural Networks (IJCNN), The 2013 International Joint Conference on*, stranice 1–8. IEEE, 2013.
- [15] Microsoft. Razvojni okvir .NET, Travanj 2014. URL <http://www.microsoft.com/net>.
- [16] Microsoft. Razvojno okruženje Visual Studio, Travanj 2014. URL <http://www.visualstudio.com/>.
- [17] Python Software Foundation. Programski jezik Python, Svibanj 2014. URL <http://www.python.org/>.
- [18] Itseez. Biblioteka OpenCV, Travanj 2014. URL <http://opencv.org/>.
- [19] shimat. Biblioteka OpenCvSharp, Svibanj 2014. URL <http://github.com/shimat/opencvsharp>.
- [20] Giacomo Stelluti Scala. Biblioteka Command Line Parser, Svibanj 2014. URL <http://github.com/gsscoder/commandline>.
- [21] Fakultet elektrotehnike i računarstva. Mapping and Assessing the State of Traffic control InFrastructure, Svibanj 2014. URL <http://zemris.fer.hr/~ssegvic/mastif/datasets.shtml>.

## **Sustav za detekciju i raspoznavanje prometnih znakova**

### **Sažetak**

Rješavanje problema detekcije i raspoznavanja prometnih znakova za ograničenje brzine može se iskoristiti u mnogim segmentima auto industrije. Rješenje tog problema uz zadovoljavajuće rezultate mogu biti općeniti algoritmi za detekciju i raspoznavanje bez kompleksne obrade slika kao što su Viola-Jones detektor i klasifikator slučajne šume. Iskorišteni Viola-Jones detektor ima dobre rezultate uz problem lažno pozitivnih odziva. Taj problem riješio se korištenjem dulje kaskade uz posljedicu smanjenja broja točnih detekcija. Klasifikacija algoritmom slučajne šume daje vrlo dobre rezultate čak i uz korištenje jednostavnih značajki poput vrijednosti slikovnih elemenata slike sivih razina.

**Ključne riječi:** detekcija prometnih znakova, raspoznavanje prometnih znakova, prometni znakovi za ograničenje brzine, Viola-Jones algoritam, algoritam slučajne šume

## **Traffic sign detection and recognition system**

### **Abstract**

Speed limit traffic sign detection and recognition system can be used in car industry. Satisfactorily solution to this problem can be achieved using general purpose detection and recognition algorithms like Viola-Jones and Random forests, all without complex image processing. Used Viola-Jones detector gave good results but problem was large number of false positives. That problem is solved using longer cascade that raised number of false negatives. Random forests classification algorithm gave very good results even though very simple features like grayscale pixel values are used.

**Keywords:** traffic sign detection, traffic sign recognition, speed limit traffic signs, Viola-Jones algorithm, Random forests algorithm