Identifikacija objekata u slici dobivenoj s kamere u vozilu uz korištenje ROS-a

Jelić, Borna

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IDENTIFIKACIJA OBJEKATA U SLICI DOBIVENOJ S KAMERE U VOZILU UZ KORIŠTENJE ROS-A

Diplomski rad

Borna Jelić

Osijek, 2019.

SADRŽAJ

1. UVOD

Automobilska industrija u posljednjih nekoliko godina proživljava znatne promjene. Umjetna inteligencija omogućuje vozilima da neke funkcije u vožnji obavljaju samostalno, bez potrebe za interakcijom vozača. Razina autonomije vozila iz dana u dan raste te autonomna vozila polako postaju svakodnevica. Konačan cilj automobilske industrije je razvoj potpuno autonomnog vozila. Da bi se postigla razina 5 automatizacije vožnje, neophodna je fuzija informacija s velikog broja senzora, izvršavanje algoritama autonomne vožnje te aktiviranje aktuatora za autonomno obavljanje željene operacije. Za osiguranje autonomije vozila, računalo, odnosno ADAS (engl. *Advanced Driver Assistance System*) koji upravlja vozilom mora raspolagati ogromnom količinom informacija da bi znao raspoznati prostor oko vozila. Autonomna vozila moraju se ponašati i reagirati u najmanju ruku kao čovjek kojeg trebaju zamijeniti u vožnji. Pošto čovjek najviše informacija dobiva preko vida, tako je i računalu u vozilu najbitniji senzor upravo kamera. U svakom trenutku vozilo treba imati informaciju o objektima ispred njega. Navedenu funkcionalnost moguće je ostvariti korištenjem informacija dobivenih pomoću slike s kamere na prednjoj strani. Na temelju detektiranih objekata, računalo može pravovremeno donijeti odluku o ponašanju vozila.

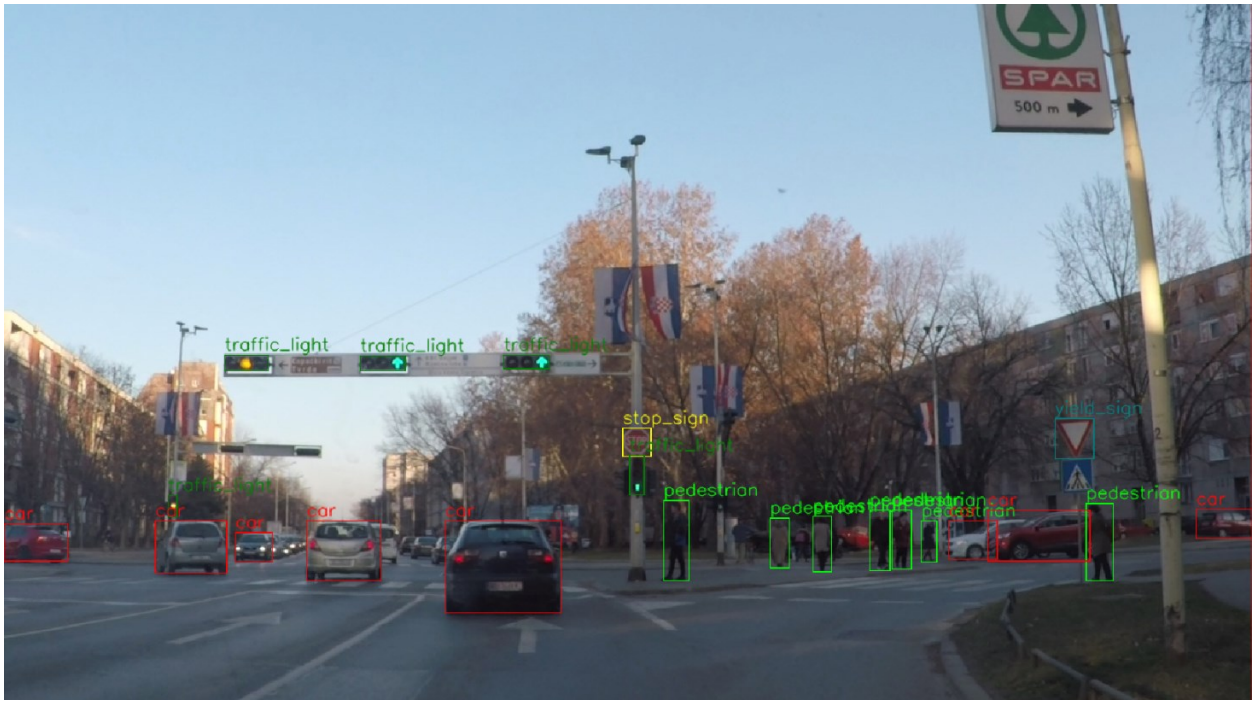
Zadatak diplomskog rada je razviti algoritam koji prima sliku s optičkog senzora vozila (kamere), identificira i izdvaja objekte u slici korištenjem grafičkog hardverskog ubrzanja, te objavljuje popis dobavljenih objekata posredstvom CAN (engl. *Controller Area Network*) sabirnice. U svrhu razvoja koristi se ROS (engl. *Robot Operating System*), dostupni skup algoritamskih elemenata i dostupna razvojna platforma. Algoritam treba detektirati objekte koji su tipični za promet (vozila, prometni znakovi, pješaci, biciklisti, semafori). U radu je bilo potrebno realizirati zadatak korištenjem odgovarajućih grafičkih sučelja i odgovarajućih signalizacijskih sučelja s CAN-om. Algoritam je izrađen u C++ programskom jeziku u ROS radnom okruženju (engl. *framework*) te je njegova implementacija prikazana na simuliranom ugradbenom (engl. *embedded*) sustavu.

Ostatak rada strukturiran je na sljedeći način. U drugom poglavlju opisani su određeni postojeći algoritmi detekcije objekata na slikama. Spomenute su neke od metoda detekcije te su detaljnije opisane dvije odabrane metode: Viola-Jones algoritam i YOLO algoritam za detekciju objekata. Navedene su prednosti i nedostaci pojedine metode. Također, navedena su postojeća rješenja koja koriste jednu od dviju odabrane metode. U trećem poglavlju opisani su novi vlastiti algoritmi za identifikaciju objekata korištenjem obaju pristupa navedenih u drugom poglavlju.

Detaljnije su opisani ROS radno okruženje i OpenCV biblioteka. Zatim je detaljno opisan proces izrade računalnih algoritama za identifikaciju objekata s kamere u C++ programskom jeziku u ROS radnom okruženju uz korištenje OpenCV biblioteke, što uključuje prikupljanje i obradu podataka za učenje. U četvrtom poglavlju opisana je izvršena evaluacija izrađenih algoritama nad validacijskim skupom slika i video sekvenci, njihova usporedba te su prikazani rezultati. Na kraju rada iznesen je zaključak.

2. METODE DETEKCIJE OBJEKATA U SLIKAMA

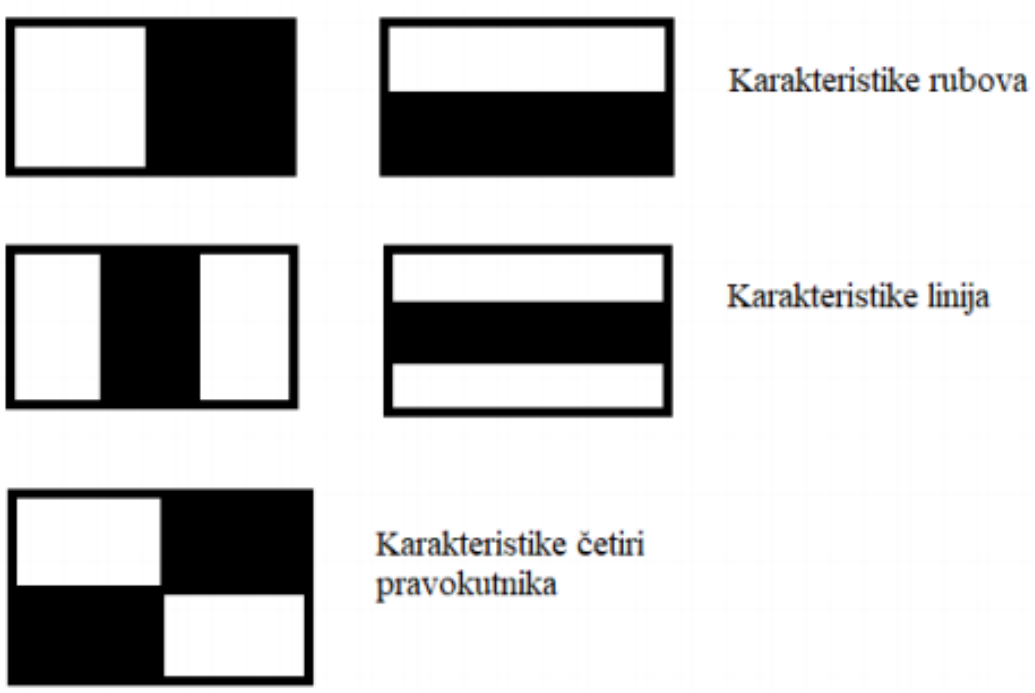
U ovom poglavlju navedene su i opisane neke od postojećih metoda detekcije objekata u slikama. Pravovremena i precizna detekcija objekata predstavlja jedan od glavnih problema kod ADAS. Na slici 2.1. prikazan je primjer idealne detekcije objekata u situaciji iz prometa. Svaki detektirani objekt od interesa ima pripadajući pravokutnik koji ga omeđuje i klasu kojoj pripada. Detekcija objekata je postupak detekcije područja na slici ili video zapisu koja imaju određena semantička značenja, a moguće ga je provesti pomoću različitih metoda: Viola-Jones algoritam za detekciju objekata [1], [2], HOG (engl. *Histograms of Oriented Gradients*) metoda [3], metode zasnovane na dubokom učenju (engl. *deep learning*) [4], metode segmentacije slike [5], detekcija objekata zasnovana na izdvajanju značajki (engl. *feature-based object detection*) [6] itd. U potpoglavlju 2.1. detaljnije je opisan Viola-Jones algoritam koji je zasnovan na strojnom učenju. U potpoglavlju 2.2. opisan je YOLO (engl. *You Only Look Once*) [4] algoritam za detekciju objekata zasnovan na dubokom učenju. Vlastita rješenja kreirana u sklopu ovog diplomskog rada i opisana u 3. poglavlju zasnivaju se na tim dvjema metodama i zbog toga su one detaljnije opisane. Također, duboko učenje je trenutno *state-of-the-art* tehnika koja se koristi za detekciju objekata u slici, a za primjenu u autonomnoj vožnji. U potpoglavlju 2.3. opisana su neka postojeća rješenja koja koriste Viola-Jones ili YOLO algoritam za detekciju objekata u autonomnoj vožnji.



Sl. 2.1. *Primjer idealne detekcije objekata u slici dobivenoj s prednje kamere u vozilu*

2.1. Viola-Jones algoritam za detekciju objekata u slici

2001. godine razvijen je Viola-Jones algoritam za detekciju objekata [1], [2]. U svojoj osnovi, to je metoda za brzo pronalaženje objekata na slici korištenjem kaskade pojačanih Haarovih klasifikatora. Prvenstveno je napravljen za otkrivanje ljudskih lica na slici, no kako koristi algoritme strojnog učenja, može se koristiti za pronalaženje bilo koje vrste objekata. Algoritam se zasniva na korištenju Haarovih karakteristika (slika 2.2.), AdaBoost algoritma te novog oblika prikaza slike – integralna slika [7]. Prilikom procesa učenja, koriste se pozitivne i negativne slike. Pozitivne slike predstavljaju sve slike na kojima se nalazi objekt od interesa, a negativne na kojima se ne nalazi.



Sl. 2.2. *Osnovni tipovi Haarovih karakteristika [8]*

Svaka karakteristika predstavlja vrijednost koja je dobivena oduzimanjem sume vrijednosti elemenata slike prekrivenih bijelim dijelom pravokutnika od sume vrijednosti elemenata slike prekrivenih crnim dijelom pravokutnika. Nakon što se odredi dimenzija prostora za detekciju, sve slike se skaliraju na tu dimenziju, a ona ujedno označava i najmanju dimenziju te omjer širine i visine objekta kojeg je moguće ubuduće detektirati na slici. Integralna slika koristi se radi bržeg računanja Haarovih karakteristika [8]. Zbog načina rada integralne slike, sve pozitivne i negativne slike su u skali sive boje. Na svakoj pozitivnoj slici pronađen je veliki skup Haarovih karakteristika, dok je na negativnima pronađen manji skup. Korištenjem AdaBoost algoritma

treniraju se slabi klasifikatori, koji predstavljaju karakteristike koje točno klasificiraju više od 50% uzoraka. Završni klasifikator se računa u zadnjem koraku te predstavlja jaki, binarni klasifikator sastavljen kao linearna kombinacija više slabih klasifikatora.

Viola-Jones uvodi kaskadu jakih klasifikatora čija je zadaća dodatno smanjenje vremena računanja i ubrzanje procesa detekcije. Kaskada klasifikatora sastavljena je od više razina, pri čemu svaka razina predstavlja skupinu slabih klasifikatora, tj. jedan jaki klasifikator. U prozoru za proračun značajki, tipično veličine 24x24 elemenata slike, računaju se Haarove karakteristike koje se pomiču po tom cijelom prozoru. Nakon toga prozor u kojem se računaju Haarove karakteristike se pomiče dok se ne prođe kroz cijelu sliku. Svaki dio slike također prolazi i kroz kaskadu klasifikatora koja označava trenutni dio slike kao pozitivan ili negativan. Ako je dio slike označen kao pozitivan, tada postoji mogućnost da je unutar toga dijela slike objekt od interesa, a ako je označen kao negativan, tada se smatra kako unutar toga dijela slike nije objekt od interesa. Također, ako je dio slike označen kao negativan, tada je klasifikacija trenutnog dijela slike završena, on biva odbačen te se prelazi na sljedeći dio slike. Većina dijelova slike ne sadrži traženi objekt, stoga odbacivanje ovih dijelova na prvim razinama kaskade ubrzava cijeli proces. Ako je dio slike označen kao pozitivan, tada taj dio prelazi na sljedeću razinu kaskade. Ukoliko određeni dio slike prođe kroz sve razine kaskade klasifikatora, tada je taj dio klasificiran kao objekt od interesa. Dijagram toka rada kaskade klasifikatora prikazan je na slici 2.3.

Sl. 2.3. *Dijagram toka rada kaskade*

Viola-Jones algoritam sadrži i određene nedostatke. Nedostatak algoritma je preveliko vrijeme trajanja procesa treniranja, što se može vidjeti u tablici 3.1. u potpoglavlju 3.1. Također, unaprijed određen fiksni omjer širine i visine prozora za detekciju također je i omjer širine i duljine detektiranog objekta, što predstavlja ograničenje rada algoritma. Algoritam nije otporan na rotaciju objekta, kao ni na djelomičnu okluziju istog. Više informacija o samom Viola-Jones algoritmu može se pronaći u [1], [2].

2.2. YOLO algoritam

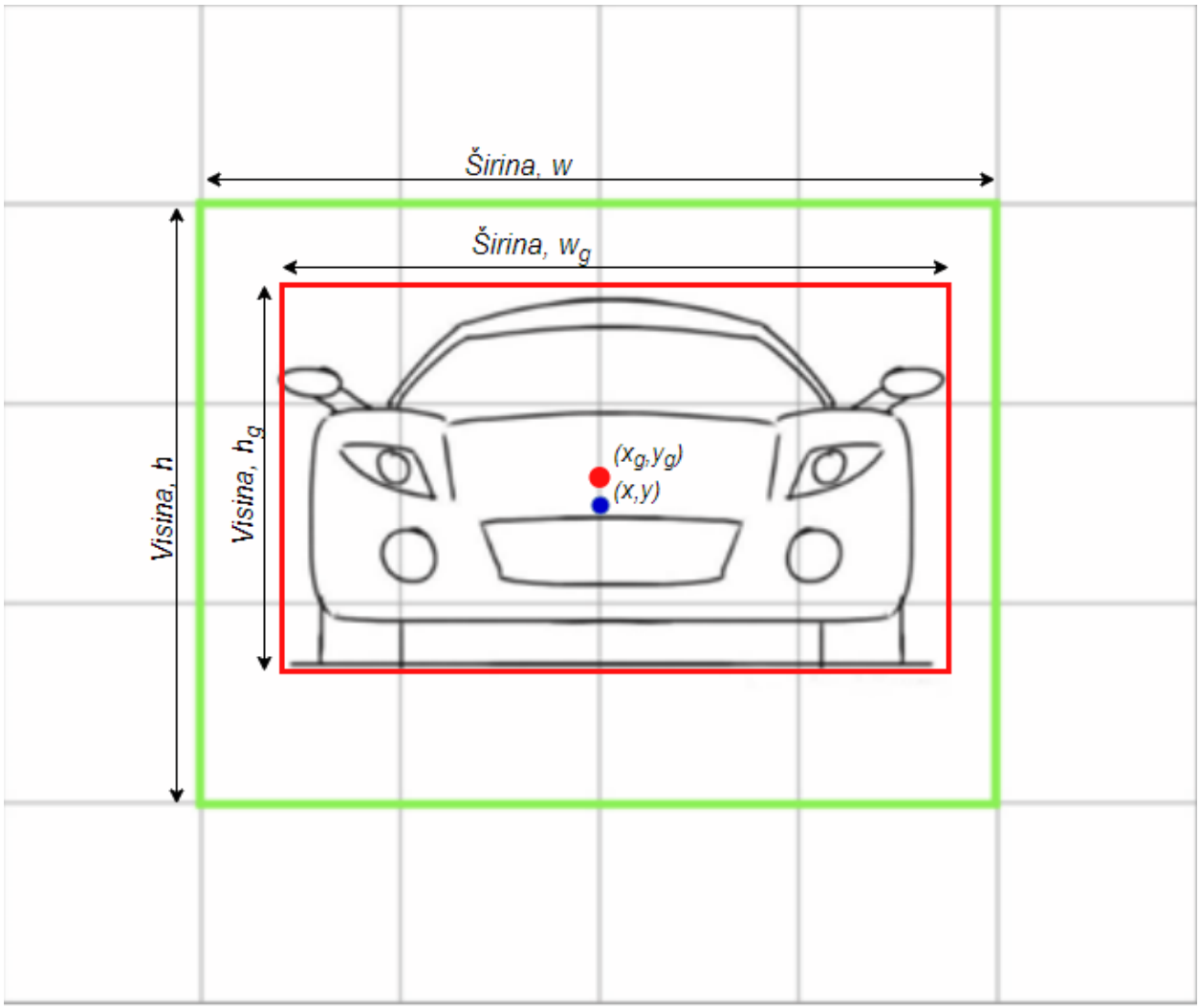
YOLO algoritam za detekciju objekata razvijen je 2016. godine [4]. On unificira komponente detekcije objekata u jednu jedinstvenu neuronsku mrežu te gleda detekciju objekata kao regresijski problem. Radi na principu da ulaznu sliku podijeli na pravokutne ćelije, odnosno matricu ćelija. U svakom dijelu matrice predviđa se određeni broj graničnih pravokutnika *B* (engl. *bounding box*) i daje postotak koji predstavlja vjerojatnost da je unutar tog graničnog pravokutnika objekt. Ova vjerojatnost označava koliko je model siguran da ova ćelija u matrici sadrži objekt. Idealno, želimo da vjerojatnost, ukoliko ćelija ne sadrži objekt, bude nula, a u suprotnom jedan. U ostalim slučajevima, vjerojatnost je jednaka omjeru površine presjeka detektiranog graničnog pravokutnika i stvarnog graničnog pravokutnika te sumirane površine detektiranog i stvarnog graničnog pravokutnika (engl. *Intersection over Union, IoU*) (2-1).

$$IoU = \frac{A \cap B}{A \cup B}, \tag{2-1}$$

gdje je:

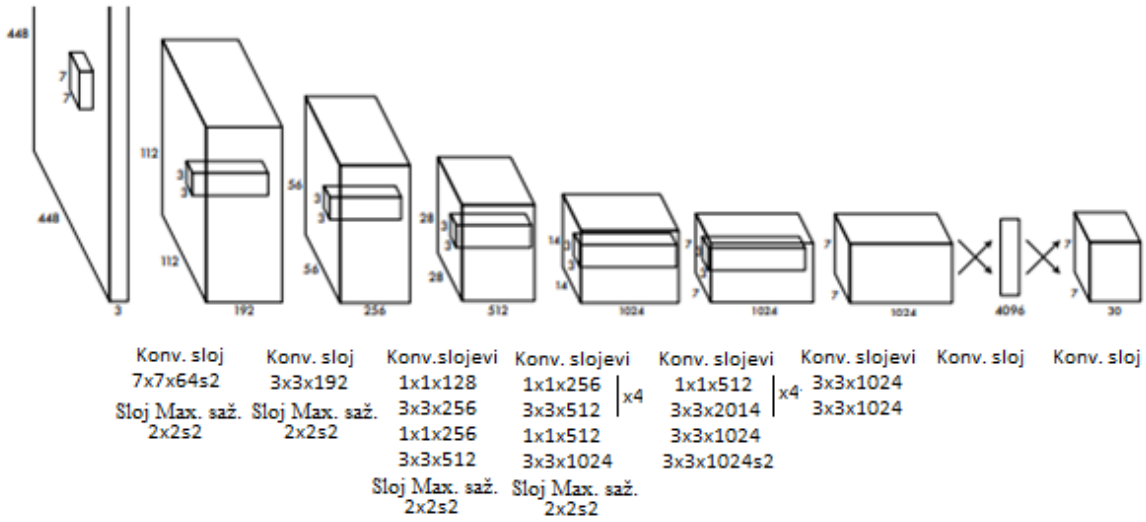
- IoU – *Intersection over Union,*
- A – površina detektiranog graničnog pravokutnika,
- B – površina stvarnog graničnog pravokutnika (engl. *ground truth*).

Za svaki granični pravokutnik daju se sljedeće predikcije: vjerojatnost predikcije i četiri predikcije označene kao *x, y, w, h.* Oznake *x* i *y* predstavljaju koordinate središta graničnog pravokutnika, dok oznake *w* i *h* predstavljaju dimenziju objekta, odnosno širinu i visinu (slika 2.4.). Vjerojatnost predikcije predstavlja *IoU* između predviđenog graničnog pravokutnika s parametrima (*x, y, w, h*) (na slici 2.4. označen zeleno) i stvarnog graničnog pravokutnika sa parametrima (x_g, y_g, w_g, h_g) (na slici 2.4. označen crveno). [4] Na primjeru sa slike 2.4. *IoU* iznosi približno 0,6.



Sl. 2.4. *Predikcija graničnog pravokutnika na slici* [9]

YOLO algoritam koristi konvolucijsku neuronsku mreža (engl. *convolutional neural network, CNN*). Konvolucijski slojevi mreže služe za izdvajanje značajki iz slike dok potpuno povezani slojevi predviđaju izlazne vjerojatnosti i koordinate. Originalni YOLO model ima 24 konvolucijska sloja nakon kojih slijede dva potpuno povezana sloja (slika 2.5.). Također, predstavljen je i ubrzani YOLO model s 9 konvolucijskih slojeva i manje filtera u tim slojevima, što rezultira manjim brojem parametara te jednostavnijim i bržim modelom.



Sl. 2.5. *Arhitektura originalnog YOLO modela* [4]

Ovaj model obavlja proces učenja na cijelim slikama. Prije početka učenja potrebno je podesiti niz parametara, poput stope učenja (engl. *learning rate*), veličine serije (engl. *batch size*), broj izlaznih klasa itd. Učenje modela se provodi na skupu označenih slika za učenje. Na početku učenja, sve dostupne slike u skupu skalirane su na 448 x 448 elemenata slike, nakon čega se slike propuštanju kroz CNN. Tijekom učenja modela, jedan granični pravokutnik je zadužen za točno jedan objekt. Proces treniranja originalnog YOLO modela izvršen je na Pascal VOC 2007 bazi podataka [10]. U tablici 2.1. prikazana je usporedba performansi originalnog YOLO modela i ubrzanog YOLO modela s R-CNN modelima (engl. *Regions with CNN features*). Može se vidjeti da dva YOLO modela predstavljena u [4] jedini rade u stvarnom vremenu, odnosno obradi se više od 30 okvira u sekundi (engl. *frames per second, FPS*) na NVIDIA GeForce Titan X grafičkoj kartici (engl. *graphics processing unit, GPU*), no uz manju prosječnu preciznost po klasama (engl. *mean average percision, mAP*).

mAP je mjera prosječne preciznosti detektora objekata i računa se na način da se izračuna prosječna preciznost detektora po svakoj klasi te se nakon toga izračuna prosječna vrijednosti tih iznosa. Uz mAP, promatrat će se svojstva preciznosti, odziva i točnosti te F1 mjera, svojstva koja služe za procjenu kvalitete rada algoritma. Preciznost (engl. *precision*) je omjer broja točno detektiranih objekata (engl. *true positive, TP*) u odnosu na broj svih detektiranih objekata (2-2). Odziv (engl. *recall*) je omjer broja točno detektiranih objekata u odnosu na broj svih objekata koji su trebali biti detektirani (2-3). Točnost je svojstvo koje prikazuje omjer broja detektiranih objekata u odnosu na broj svih objekata koji su detektirani (2-4). Mjera F1 predstavlja harmonijsku sredinu parametara odziva i preciznosti (2-5).

$$preciznost = \frac{TP}{TP+FP}, \tag{2-2}$$

$$odziv = \frac{TP}{TP+FN}, \tag{2-3}$$

$$točnost = \frac{TP+TN}{TP+FP+FN+TN}, \tag{2-4}$$

$$F1 = 2 * \frac{preciznost*odziv}{preciznost+odziv}, \tag{2-5}$$

gdje je:

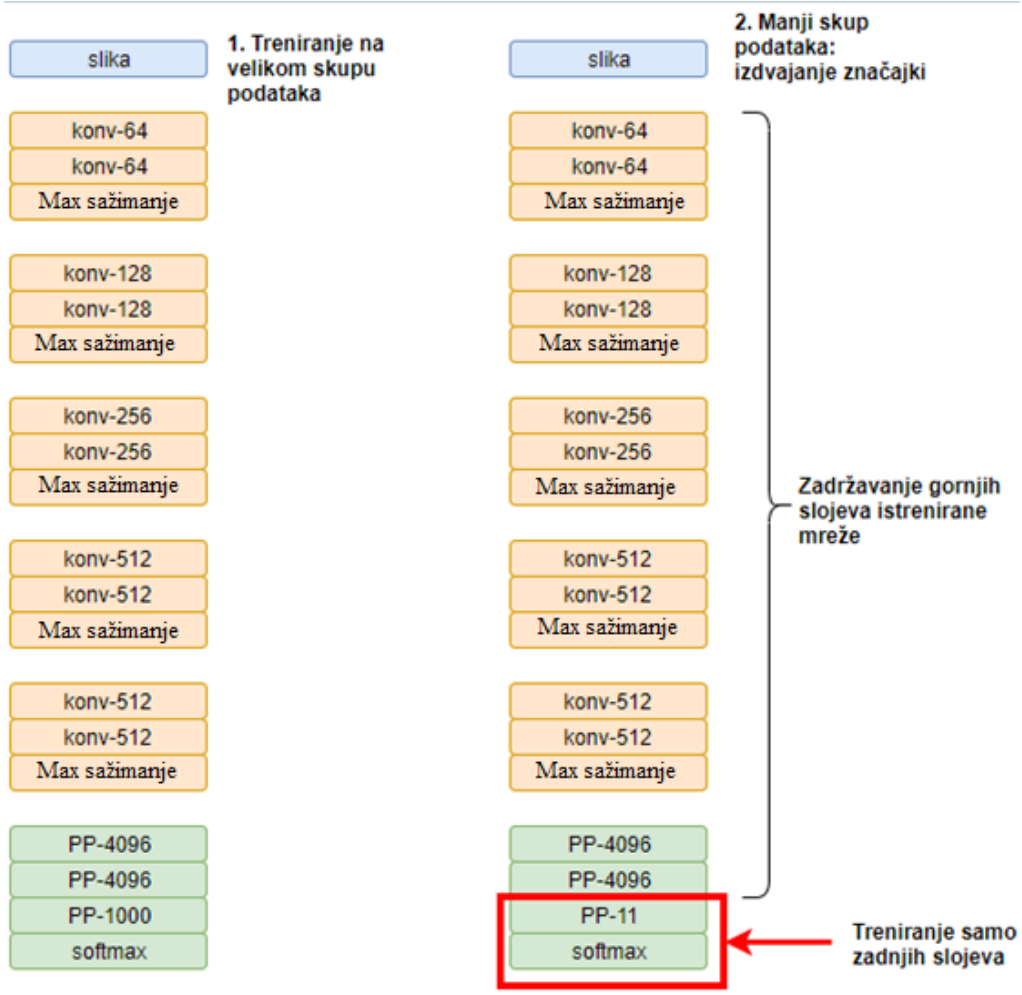
- TP – *true positive,* detekcije koje su proglašene objektom, a one to jesu,
- FP – *false positive,* detekcije koje su proglašene objektom, a one to nisu,
- FN – *false negative,* detekcije koje nisu proglašene objektom, a one to jesu,
- TN – *true negative,* detekcije koje nisu proglašene objektom, a one to nisu.

Tab. 2.1. *Usporedba YOLO modela sa R-CNN na Pascal VOC 2007 bazi* [11]

MODEL	***mAP***	**FPS**	**Rad u stvarnom vremenu**
Fast YOLO	52,7%	155	DA
YOLO	63,4%	45	DA
YOLO VGG-16	66,4%	21	NE
Fast R-CNN	70%	0.5	NE
Faster R-CNN VGG-16	73,2%	7	NE
Faster R-CNN ZF	62,1%	18	NE

Originalni YOLO model ima tri glavna nedostatka. Kako se model uči na skupu slika za treniranje, postoji problem s generalizacijom objekata kada se nađu u drugačijim okruženjima ili ako objekt nema sličan omjer širine i visine. Model koristi relativno grube značajke za predviđanje graničnih pravokutnika jer sadrži više slojeva poduzorkovanja (engl. *downsampling layers*). YOLO nameće prostorna ograničenja u predviđanju graničnih pravokutnika, tako što svaka ćelija predviđa samo dva granična pravokutnika i može imati samo jednu klasu. Ovo ograničava broj susjednih objekata koje model može predvidjeti, primjerice detekcija svih ptica u jatu. Također, kriterijska funkcija jednako tretira pogreške u predviđanju malih i velikih graničnih pravokutnika. Primjerice, mala greška u predviđanju velikog graničnog pravokutnika generalno ne znači mnogo, ali mala greška u predviđanju malog graničnog pravokutnika ima puno veći utjecaj na *IoU.*

Jedno od glavnih svojstava CNN je preneseno učenje (engl. *transfer learning*). Ovo svojstvo omogućuje učenje CNN na manjem skupu podataka. Neka su na raspolaganju dva skupa podataka, jedan koji sadrži veliki broj označenih slika i drugi koji je sličan, ali sadrži znatno manje slika. Na temelju većeg skupa izradi se prvi model neuronske mreže. Zatim se na temelju prvog modela i drugog skupa slika izradi konačni model. Konvolucijski slojevi služe za izdvajanje značajki sa slika te se na velikom podatkovnom skupu mogu jako dobro istreniraju. Zatim se ti početni (gornji) slojevi zadrže te se donji potpuno povezani slojevi nanovo istreniraju za vlastite potrebe, primjerice za klasifikaciju objekata u manje klasa. Na slici 2.6. prikazan je primjer prenesenog učenja gdje se završni potpuno povezani sloj (PP) nanovo istrenirao kako bi mogao raditi klasifikaciju u željenih 11 klasa, za razliku od početnog koji je klasifikaciju radio u 1000 klasa.



Sl. 2.6. *Primjer prenesenog učenja [11]*

Prenesenim učenjem postiže se kvalitetnije izdvajanje značajki nego korištenjem osnovnog YOLO modela, na način da se gornji dio mreže preuzme od neke dokazano dobre neuronske mreže te se treniraju samo potpuno povezani slojevi koji klasificiraju detektirane objekte.

Jedna od dodatnih funkcionalnosti koja se može implementirati radi bolje detekcije objekata je potiskivanje ne-maksimalne vrijednosti (engl. *non-maximal supression*). Ukoliko se nekakav veliki objekt nalazi na slici, on može biti detektiran u više ćelija. Potiskivanje ne-maksimalne vrijednosti rješava ovaj problem tako što pronađe detekciju s najvećom vjerojatnošću i proglasi ju objektom. Zatim izračunava *IoU* s ostalim detekcijama i odbacuje sve koje imaju preklapanje veće od nekog praga te ponavlja ova dva postupka dok ima detekcija (slika 2.7.). CNN detektira na istom području više automobila, sa različitim vjerojatnostima predikcije, no u stvarnosti je to jedan te isti automobil (lijeva slika). Provođenjem potiskivanja ne-maksimalne vrijednosti, zadržava se granični pravokutnik s najvećom vjerojatnosti predikcije (desna slika).

Sl. 2.7. *Primjer potiskivanja ne-maksimalne vrijednosti* [11]

YOLO također koristi i pristup predviđenih graničnih pravokutnika (engl. *anchor box*). Ovaj pristup predviđa lokacije graničnih pravokutnika, čime se smanjuje netočnost detekcija. Generira se tisuće predviđenih graničnih pravokutnika za svaki prediktor koji predstavljaju idealnu lokaciju, oblik i veličinu objekta kojeg trebaju detektirati. Za svaki predviđeni granični pravokutnik izračunava se *IoU* svakog objekta. Ako je najveći *IoU* iznad 50%, onda predviđeni granični pravokutnik treba detektirati objekt za koji je najveći *IoU.* U suprotnom, ako je *IoU* veći od 40%, onda je ispravna detekcija dvosmislena i neuronska mreža ne treba učiti iz ovog primjera. Ukoliko je najveći *IoU* manji od 40%, onda predviđeni granični pravokutnik treba predvidjeti da nema objekta [12]. U primjeru korištenja *RetinaNet* neuronske mreže [13] na *WiderFace* bazi podataka [14], na slici 2.8. može se vidjeti da su detektirana samo četiri lica jer se samo četiri granična pravokutnika, odnosno četiri lica preklapaju sa predviđenim graničnim pravokutnicima. Ukoliko se konfigurira veličina predviđenih graničnih pravokutnika i ponovno provuče ista slika kroz mrežu, rezultati su puno bolji jer ovaj puta se generiralo više predviđenih graničnih pravokutnika, zbog čega se i više lica poklapa sa njima (slika 2.9.).

Sl. 2.8. *Nepotpuna detekcija lica na slici* [12]

Sl. 2.9. *Idealna detekcija nakon rekonfiguriranja predviđenih graničnih pravokutnika* [12]

YOLO ima više verzija modela u kojima su poboljšane performanse originalnog algoritma YOLO9000 [15] i YOLOv3 [16]. U ovom diplomskom radu fokus će biti na analizi YOLOv3 modela, koji primjenjuje jednaki koncept kao originalni YOLO model uz dodatna poboljšanja.

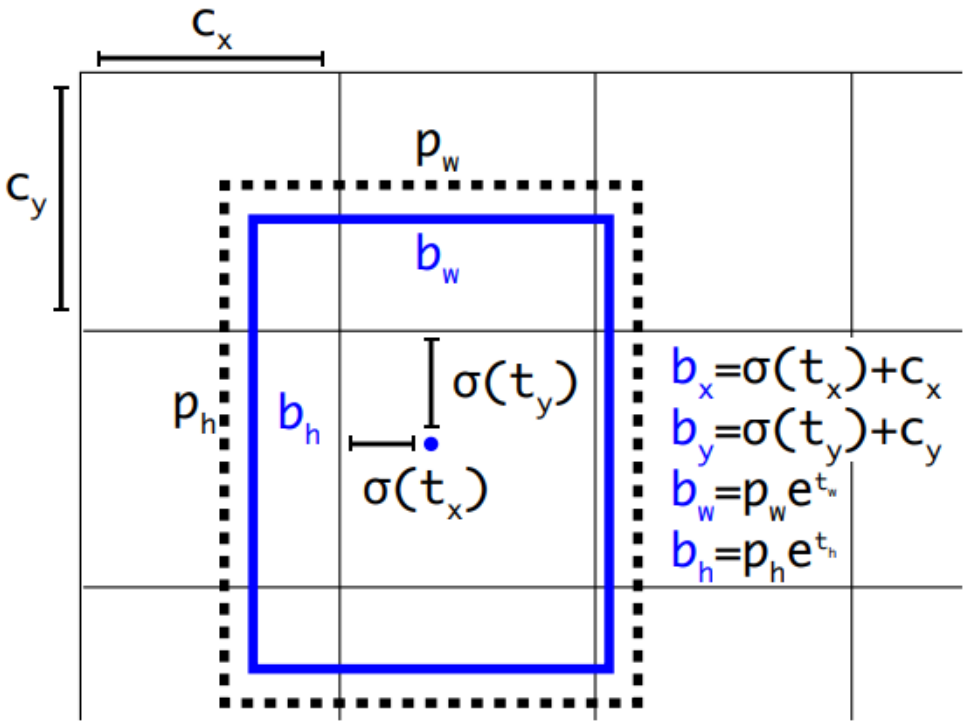
YOLOv3 predviđa granične pravokutnike koristeći klastere dimenzija kao i YOLO9000 [15]. Za dobivanje klastera dimenzija koristi se algoritam *K-Means Clustering* [17] na trening skupu podataka. Dobiveno je K klastera dimenzija predviđenih graničnih pravokutnika za trening skup podataka. Mreža predviđa četiri vrijednosti (t_x, t_y, t_w, t_h) za svaki predviđeni granični pravokutnik, koje odgovaraju vrijednostima (x, y, w, h) prikazanim na slici 2.4. Ukoliko ćelija ima pomak od gornjeg lijevog ugla slike za (c_x, c_y) onda predviđeni granični pravokutnik ima širinu i visinu p_w, p_h (slika 2.10.) te predikcije odgovaraju:

$$b_x = \sigma(t_x) + c_x \tag{2-1}$$

$$b_y = \sigma(t_y) + c_y \tag{2-2}$$

$$b_w = p_w e^{t_w} \tag{2-3}$$

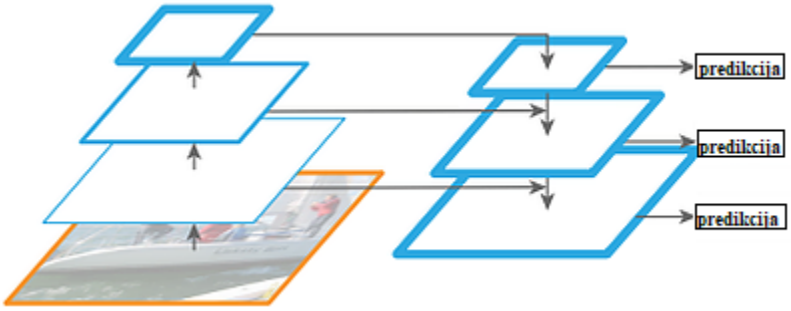
$$b_h = p_h e^{t_h} \tag{2-4}$$



Sl. 2.10. *Dimenzije predviđenog graničnog pravokutnika (iscrtano) i predikcije (plavo)* [15]

Model predviđa rezultat za svaki granični pravokutnika koristeći logističku regresiju. Rezultat treba biti 1 ukoliko se predviđeni granični pravokutnika preklapa sa stvarnim objektom

za više od bilo kojeg drugog predviđenog graničnog pravokutnika. Svaki granični pravokutnik predviđa klasu koristeći klasifikaciju višestrukih oznaka. Model ne koristi *softmax* sloj, jer je nepotreban za dobre performanse, već koristi nezavisne logističke klasifikatore. Ovakav način obilazi problem preklapanja oznaka, primjerice čovjek i žena. *Softmax* sloj nameće da svaki predviđeni granični pravokutnik ima samo jednu klasu, što često nije slučaj. Pristup višestrukih oznaka zato bolje označava podatke. YOLOv3 također predviđa predviđene granične pravokutnike na tri različite skale. Nakon prve skale, proširuje se mapa značajki dva puta. Također, pridodaje se mapa značajki otprije iz mreže. Ovom metodom dobiva se bolje semantičko značenje. Zatim se ovo provlači kroz nekoliko konvolucijskih slojeva te se predviđa matrica graničnih pravokutnika, ali dvostruko veća. Analogno ovome, radi se predikcija graničnih pravokutnika i na zadnjoj skali (slika 2.11.).



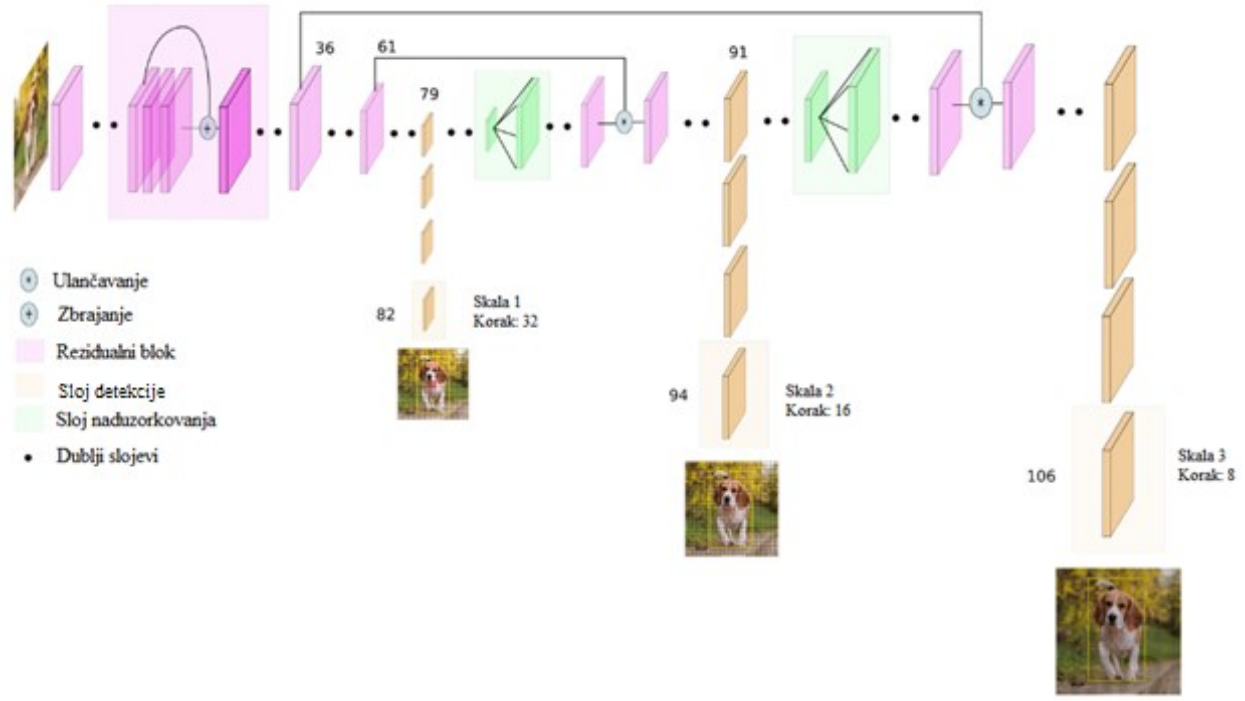
Sl. 2.11. *Piramida značajki kod YOLOv3 CNN na tri različite skale* [18]

Ovaj model koristi 53 konvolucijska sloja (slika 2.12.), a nastao je kao hibrid mreža YOLOv2, Darknet-19 [19] i rezidualnih slojeva.

	Vrsta	Filteri	Veličina	Izlaz
	Konvolucijski	32	3 × 3	256 × 256
	Konvolucijski	64	3 × 3 / 2	128 × 128
1×	Konvolucijski	32	1 × 1	
	Konvolucijski	64	3 × 3	
	Rezidualni			128 × 128
	Konvolucijski	128	3 × 3 / 2	64 × 64
2×	Konvolucijski	64	1 × 1	
	Konvolucijski	128	3 × 3	
	Rezidualni			64 × 64
	Konvolucijski	256	3 × 3 / 2	32 × 32
8×	Konvolucijski	128	1 × 1	
	Konvolucijski	256	3 × 3	
	Rezidualni			32 × 32
	Konvolucijski	512	3 × 3 / 2	16 × 16
8×	Konvolucijski	256	1 × 1	
	Konvolucijski	512	3 × 3	
	Rezidualni			16 × 16
	Konvolucijski	1024	3 × 3 / 2	8 × 8
4×	Konvolucijski	512	1 × 1	
	Konvolucijski	1024	3 × 3	
	Rezidualni			8 × 8
	Sloj sažimanja po prosjeku			Globalno
	Pot. povezani sloj			1000
	Softmax			

Sl. 2.12. *Darknet-53 mreža* [19]

Originalni YOLO model imao je problema s detekcijom malih objekata. S predviđanjem na tri različite skale, YOLOv3 preciznije i točnije detektira male objekte (slika 2.13.).



Sl. 2.13. *Arhitektura YOLOv3 CNN* [20]

U ADAS, računalni resursi su ograničeni. Stoga je napravljen manji YOLOv3 – *Tiny* YOLOv3*,* koji zauzima manje memorije te je potrebno manje parametara*.* Ovaj model nema 53, već 13 konvolucijskih slojeva. Ovakav model odgovara originalnom ubrzanom YOLO modelu, može obrađivati veći broj FPS, ali ima manji *mAP*. Prilikom izrade vlastitog algoritma za detekciju objekata opisanog u 3. poglavlju, koristit će se i YOLOv3 i *Tiny* YOLOv3.

2.3. Postojeća rješenja za detekciju objekata zasnovana na Viola-Jones i YOLOv3 algoritmu

Klasične metode detekcije objekata u računalnom vidu su često zasnovane na uparivanju značajki, poput Haarovih značajki. No, dokazalo se da se kombinacijom Haarovih ili HOG značajki i metoda poput metode potpornih vektora (engl. *support vector machines, SVM*) ili metoda ubrzavanja (engl. *Boosting methods*) postižu puno bolji rezultati u detekciji i klasifikaciji objekata [21], [22], [23]. Primjerice, Gawande [24] koristi kombinaciju Viola-Jones detektora sa SVM klasifikatorima zasnovanim na histogramu gradijenata u detekciji i prepoznavanju prometnih znakova. Predloženo rješenje ima visoku preciznosti s radom u stvarnom vremenu na ugradbenom sustavu. Međutim, metode dubokog učenja pokazale su se kao bolji pristup u

ovakvim situacijama, posebno CNN. U detekciji pješaka, Kuang i ostali [25] predlažu prošireni YOLO detektor. Predložena mreža sadrži *Inception* module koji su dokazano korisni u detekciji i lokalizaciji objekata, omogućujući povećanje dubine i širine mreže sa minimalnim računalnim zahtjevima. Valiati i ostali [26] predlažu metodu detekcije pješaka zasnovanu na YOLOv3 algoritmu sa semantičkom segmentacijom. Semantička segmentacija doprinosi preciznijoj lokalizaciji pješaka. Umjesto predviđanja samo graničnih pravokutnika, mreža predviđa i semantičke maske pješaka. Model time nauči „odrezati" regije koje ne sadrže pješake i naglašava dijelove mape značajki na kojima se nalaze, time povećavajući preciznost detekcije. Liu [27] naglašava da je najveći problem YOLO algoritma to što procesira svaki okvir kao zaseban. Predlaže novu tehniku memorijskog mapiranja koja u obzir uzima informacije između okvira, povećavajući sposobnost detekcije YOLO-a. Memorijska mapa veličine N pamti rezultate detekcije u N-1 okvira, čime mreža, kada predviđa detekciju na N-tom okviru, u obzir uzima prethodne detekcije. U radu [9] razvijen je sustav zasnovan na YOLO algoritmu sposoban za prepoznavanje tipičnih objekata prisutnih u prometu, poput automobila, kamiona, pješaka te životinja. Sustav se pokazao dobrim, no nije dostatan za autonomno vozilo jer ima poveći postotak pogreške te predviđa krive veličine graničnih pravokutnika. Softver za pokretanje svih spomenutih rješenja nažalost nije dostupan pa se ona nisu mogla dodatno testirati u sklopu ovog diplomskog rada.

3. IZRADA VLASTITIH ALGORITAMA ZA IDENTIFIKACIJU OBJEKATA U SLIKAMA DOBIVENIM S PREDNJE KAMERE NA VOZILU

U ovom poglavlju opisana je izrada vlastitog algoritma za identifikaciju objekata. Izrađena su dva algoritma, jedan na bazi Viola-Jones algoritma i drugi na bazi YOLOv3. Cilj je usporediti točnost i brzinu detekcije objekata od interesa na slikama i video zapisu. Oba algoritma izrađena su u C++ programskom jeziku u ROS radnom okruženju koristeći OpenCV biblioteku. U potpoglavlju 3.1. detaljno je opisan proces podešavanja OpenCV biblioteke i ROS radnog okruženja. U potpoglavlju 3.2. detaljno je opisan postupak kreiranja dviju baza slika za treniranje detektora i neuronske mreže te sami procesi treniranja. U potpoglavlju 3.3. opisana je izrada algoritma zasnovanog na Viola-Jones algoritma, dok je u potpoglavlju 3.4. opisana izrada algoritma zasnovanog na YOLOv3. Cilj ovih algoritama je identificirati objekte u prometu koji pripadaju jednoj od sljedećih klasa: automobil, autobus ili kamion, pješak, biciklist, semafor, znak obaveznog zaustavljanja, znak raskrižja s cestom s prednošću prolaska, znak opasnosti, znak ceste s prednošću prolaska, znakovi izričitih naredbi s crvenim rubom ili znak izričitih naredbi s plavom pozadinom (slika 3.1.).

(a) *(b)* *(c)* *(d)* *(e)*

(f) *(g)* *(h)*

(i) *(j)* *(k)*

Sl. 3.1. *Primjeri slika objekata koje je potrebno detektirati i klasificirati u sklopu diplomskog rada: (a) automobil, (b) kamion, (c) pješak, (d) biciklist, (e) semafor, (f) znak obaveznog zaustavljanja, (g) znak raskrižja s cestom s prednošću prolaska, (h) promet u oba smjera, (i) znak ceste s prednošću prolaska, (j) znak zabrane prometa u oba smjera, (k) znak izričite naredbe s plavom pozadinom – dopušteni smjerovi*

3.1. Podešavanje ROS radnog okruženja i OpenCV biblioteke

3.1.1. ROS radno okruženje

ROS radno okruženje je fleksibilno radno okruženje za pisanje programske podrške za robote. U principu, to je kolekcija alata, biblioteka i strukturnih konvencija s ciljem olakšavanja pisanja kompleksnog i robusnog robotskog ponašanja za široki spektar platformi. ROS je dizajniran da bude što više distributivan i modularan, tako da korisnici mogu koristiti koliko god ROS koda žele. Zbog svoje prirode, razvijena je velika zajednica koja je stvorila preko 3000 ROS paketa te poboljšala osnovnu ROS jezgru. Ovi paketi pokrivaju sve od *proof-of-concept* implementacija do upravljačkih programa i aplikacija industrijske kvalitete. [28]

Za izradu algoritama u sklopu diplomskog rada, korišten je ROS *Kinetic Kame* za Ubuntu 16.04 operacijski sustav. Za početak, potrebno je dodati ključ servera da bi se moglo pristupiti autentificiranim ROS paketima. Zatim se pokreće naredba za pristupanje i instalaciju:

sudo apt-get install ros-kinetic-desktop-full

koja uključuje ROS, 2D/3D simulatore, *rviz* alat za 3D vizualizaciju itd. Nadalje, potrebno je instalirati sve zavisne pakete sustava za kompajliranje i pokretanje osnovnih komponenti u ROS-u. Moguće je dodati sve ROS varijable okruženja (engl. *environment variables*) u konfiguraciju *Bash* upravljačkog prozora u Linuxu, tako da se prilikom pokretanja nove sesije upravljačkog prozora automatski učitaju sve ROS varijable, što omogućava kompajliranje i pokretanje ROS komponenata. ROS se sastoji od paketa, koji sadrže više čvorova. Čvor se može promatrati kao

izvršna datoteka koja koristi ROS za komunikaciju s drugim čvorovima. Ova komunikacija odvija se putem poruka i tema. Tema je komunikacijski kanal u koji dva ili više ROS čvora mogu slati poruke ili se pretplatiti na kanal te primati poruke koje su specifične za taj kanal. Sve se odvija ispod glavnog ROS čvora zvanog *Master*, koji se pokreće naredbom *roscore* u upravljačkom prozoru. Kompajliranje ROS paketa odvija se pomoću naredbe *catkin build* u upravljačkom prozoru.

3.1.2. OpenCV biblioteka

OpenCV (engl. *Open Source Computer Vision)* je biblioteka otvorenog koda koja sadrži funkcije korištene u računalnom vidu i strojnom učenju [29]. Napisana u C i C++ programskim jezicima, koristi se u aplikacijama koje zahtijevaju izvršavanje u stvarnom vremenu. Podržava korištenje na više operacijskih sustava, poput *Microsoft Windows, Linux, MacOS i Android.* Sadrži preko 2500 optimiziranih algoritama, među kojima su i algoritmi za detekciju objekata, praćenje detektiranih objekata i pokreta, stvaranje 3D oblaka točaka pomoću stereo kamera i sl.

Za izradu diplomskog rada korištena je OpenCV biblioteka 3.4.1. na Ubuntu 16.04 operacijskom sustavu. Za početak je instaliran *pkg-config* alat koji definira jedinstveno sučelje za korištenje instaliranih biblioteka u svrhu kompajliranja programske podrške koja ovisi o njima. Zatim se instaliraju sve potrebne biblioteke za rad s različitim video i slikovnim formatima, poput *libpng, ligjpeg, libavcodec, libavformat* i dr. Nakon toga, preuzet je OpenCV modul [30] i dodatni modul koji sadrži dodatne, ne-istestirane funkcije [31] s GitHub-a. Nadalje, pokrenut je proces konfiguriranja i instalacije OpenCV biblioteke pomoću *cmake* alata. Nakon što je završena instalacija, pokrenuta je naredba:

pkg-config --modversion opencv

koja vraća verziju instalirane OpenCV biblioteke, čime se potvrđuje uspješna instalacija OpenCV biblioteke. Pošto se sav programski kod algoritama pisao u C++ jeziku u C++11 standardu, korištena je naredba za kompajliranje aplikacija:

g++ naziv_aplikacije.cpp -o naziv_izvršne_datoteke 'pkg-config --cflags --libs opencv'
-std=c++11

3.2. Kreiranje baza slika i proces treniranja algoritama

Zbog prirode rada detektora u Viola-Jones algoritmu te CNN, bilo je potrebno napraviti dvije različite baze slika. Za kreiranje baza slika u svrhu treniranja korišten je vlastiti skup slika uz slike iz *online* dostupnih baza: German Traffic Sign Benchmarks [32], LISA Traffic Sign Dataset [33], KITTI Vision Benchmark Suite [34] i ImageNet [35]. Baza za treniranje detektora za algoritam zasnovan na Viola-Jones algoritmu sastoji se od 21115 pozitivnih slika koje sadrže točno jedan objekt jedne klase i 4000 negativnih slika koje ne sadrže objekte koje je potrebno detektirati u ovom diplomskom radu. Slike su različitih rezolucija, no nijedna slika nije manja od 5x15 elemenata slike (za semafore), 10x30 elemenata slike (za pješake) te 24x24 elemenata slike (za ostale klase i negativne slike). Broj slika po klasama za treniranje detektora za algoritam zasnovan na Viola-Jones algoritmu za detekciju objekata prikazan je u tablici 3.1. u odjeljku 3.2.1.

Za kreiranje baze slika u svrhu treniranja CNN snimane su video sekvence prometa u Osijeku pomoću *GoPro Hero 5* kamere postavljene s unutarnje strane prednjeg vjetrobranskog stakla. Snimane su video sekvence na rezoluciji 1280 x 720 elemenata slike uz 30 FPS-a s uskim vidnim poljem od 90° (engl. *field of view, FOV*). Ukupno je snimljeno 96,7 gigabajta zapisa, odnosno 5h i 11min zapisa vožnje po sunčanom, kišnom i maglovitom vremenu te pri noćnoj vožnji. Iz tih video zapisa izdvojeno je ukupno 12186 okvira, na način da se izdvajao svaki deseti okvir iz video zapisa dnevne vožnje te svaki peti okvir iz video zapisa noćnih vožnji te su daljnje filtrirani po svome sadržaju (sadrže li objekte koje je potrebno detektirati). 5770 njih korišteno je za treniranje YOLOv3 i *Tiny* YOLOv3 CNN, a preostalih 6416 ostavljeno je kao rezerva za potencijalno dodatno treniranje ukoliko bi bilo potrebno. Na kraju se pokazalo da za tim nema potrebe i ipak nije korišteno. Broj oznaka po klasama za treniranje CNN za algoritme zasnovane na YOLOv3 i *Tiny* YOLOv3 CNN prikazan je u tablici 3.3. u odjeljku 3.2.2.

3.2.1. Treniranje detektora za algoritam zasnovan na Viola-Jones algoritmu za detekciju objekata u slikama

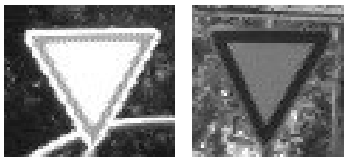
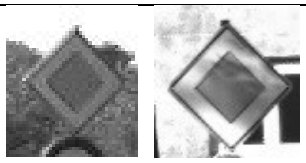
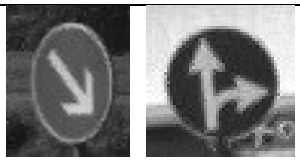
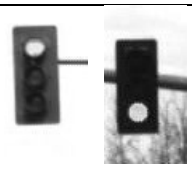
Budući da treba detektirati različite objekte ispred vozila, potrebno je napraviti bazu slika koja sadrži veliki broj slika automobila, busova, pješaka, biciklista, semafora te prometnih znakova podijeljenih u više kategorija: znak obaveznog zaustavljanja, znak raskrižja s cestom s prednošću prolaska, znakovi opasnosti, znak ceste s prednošću prolaska, znakovi izričitih naredbi sa crvenim rubom i znakovi izričitih naredbi sa plavom pozadinom. Zbog načina rada detektora, odnosno

ograničenja omjera visine i širine u Viola-Jones algoritmu, detektor automobila podijeljen je na detektor prednje/stražnje strane automobila i detektor bočne strane automobila.

U kreiranju baze za treniranje detektora potrebno je koristiti pozitivne i negativne slike. Pozitivne slike su sve slike koje sadrže objekte koje treba detektirati, dok negativne ne sadrže objekte od interesa. U poglavlju 3.2. nabrojane su *online* baze slika koje su korištene u pribavljanju pozitivnih i negativnih slika. Zbog velike raznolikosti, prikupljene su slike objekta od interesa u različitim dimenzijama, iz različitih kutova te specifičnih kriterija za neke klase. Primjerice, za pješake su pribavljane slike osoba oba spola, različitih visina itd. U tablici 3.1. prikazani su primjeri pribavljenih pozitivnih slika za svaki detektor, te količina pribavljenih pozitivnih i negativnih slika. Istih 4000 negativnih slika je korišteno pri treniranju detektora jer ne sadrže niti jedan objekt od interesa. Sve slike koje su se koristile za proces treniranja detektora algoritma zasnovanog na Viola-Jones algoritmu mogu se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.1.

Tab. 3.1. *Popis detektora s primjerima pozitivnih slika korištenih pri treniranju rješenja zasnovanog na Viola-Jones algoritmu*

DETEKTOR	**PRIMJERI POZITIVNIH SLIKA**	**BROJ POZITIVNIH SLIKA**	**BROJ NEGATIVNIH SLIKA**
PREDNJA/STRAŽNJA STRANA AUTOMOBILA		2118	4000
BOČNA STRANA AUTOMOBILA		1940	4000
PREDNJA/STRAŽNJA STRANA AUTOBUSA		1844	4000
PJEŠAK		2081	4000
BICIKLIST		2000	4000

ZNAK OBAVEZNOG ZAUSTAVLJANJA		879	4000
ZNAK RASKRIŽJA S CESTOM S PREDNOŠĆU PROLASKA		2000	4000
ZNAKOVI OPASNOSTI		1987	4000
ZNAK CESTE S PREDNOŠĆU PROLASKA		1876	4000
ZNAKOVI IZRIČITIH NAREDBI S CRVENIM RUBOM		1816	4000
ZNAKOVI IZRIČITIH NAREDBI S PLAVOM POZADINOM		1369	4000
SEMAFOR		1205	4000

Nakon kreiranja baze slika, bilo je potrebno obraditi slike da budu pogodne za treniranje detektora. Prvo, nijedna slika ne smije biti manjih dimenzija od dimenzija prozora za detekciju. Pomoću Python skripte su sve slike prebačene u nijanse sive boje te su pozitivne slike imenovane u obliku *imeKlase_redniBroj.jpeg* te negativne u obliku *neg_redniBroj.jpeg.* Također, napravljen je popis svih objekata koji se nalaze na pozitivnim slikama u obliku *naziv_slike broj_objekata x_koordinata_objekta y_koordinata_objekta širina_objekta visina_objekta.* Bitno je napomenuti da su slike već izdvojeni objekti te se na svakoj slici nalazi isključivo jedan objekt određene klase. Nakon pripreme baze, potrebno je izraditi uzorak pozitivnih slika te je izrađena VEC datoteka. Za to je korištena *opencv_createsamples* aplikacija [36], a za parametre se navodi putanja i naziv datoteka u kojoj su spremljene oznake objekata, putanja i naziv izlazne datoteke, broj pozitivnih uzoraka koje treba izraditi, širina i visina izlaznog uzorka u broju elemenata slike (slika 3.2.). Pri započinjanju procesa učenja korištena je aplikacija *opencv_traincascade* [37] uz navedene parametre: putanja izlaznog detektora, putanja i naziv VEC datoteke, putanja i naziv tekstualne datoteke s nazivima negativnih slika, broj pozitivnih i negativnih uzoraka korištenih u svakoj razini

kaskade, broj razina kaskada, način korištenja Haarovih karakteristika te širina i visina uzorka koji trebaju biti jednaki onima navedenima pri izradi VEC datoteke (slika 3.3.).

```
boki@bornaPC:~/Desktop/HAAR/trokut$ opencv_createsamples -info info/info.lst -num 2000 -w 24 -h 24
-vec positives.vec
```

Sl. 3.2. *Prikaz poziva opencv_createsamples aplikacije u upravljačkom prozoru*

```
boki@bornaPC:~/Desktop/HAAR/trokut$ opencv_traincascade -data data -info info/info.lst -bg bg.txt -
numPos 1850 -numNeg 4000 -numStages 17 -w 24 -h 24 -featureType HAAR -mode ALL
```

Sl. 3.3. *Prikaz poziva opencv_traincascade aplikacije u upravljačkom prozoru*

Kao rezultat, dobiva se XML datoteka koja predstavlja kaskadu klasifikatora, odnosno detektor. Učenje svih detektora izvršeno je korištenjem procesora Intel Core i5-7200U. Vrijeme trajanja učenja ovisi o broju razina kaskade i broju slika u procesu. Što je veći broj bilo kojeg parametra, to je duže vrijeme učenja. Na empirijski način je odabran broj razina kaskada pojedinog detektora kao i dimenzije izlaznih uzoraka. Kao referenca za broj kaskada i dimenzije izlaznih uzoraka uzet je rad [38]. U tablici 3.2. prikazana su vremena treniranja i dimenzije izlaznih uzoraka detektora. Dimenzije izlaznih uzoraka određene su promatranjem određenih objekata na slikama. Omjer širine i visine kod prometnih znakova, kao i kod prednje, odnosno stražnje strane automobila je otprilike 1, stoga su i dimenzije izlaznih uzoraka kvadratnog oblika. Visina pješaka i semafora je veća nego njihova širina, stoga su dimenzije izlaznih uzoraka kod ovih detektora drugačije.

Tab. 3.2. *Vremena treniranja, broj kaskada, dimenzije izlaznih uzoraka te broj pozitivnih i negativnih slika za svaki detektor*

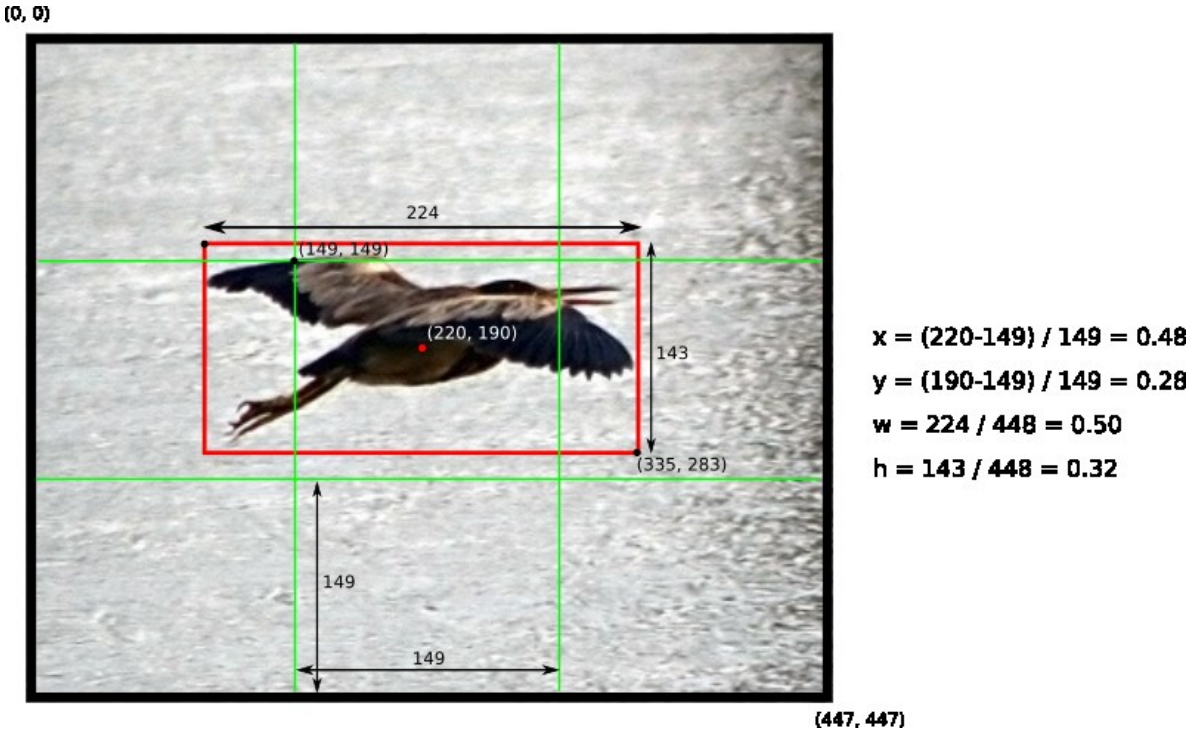
DETEKTOR	BROJ KASKADA	DIMENZIJE IZLAZNIH UZORAKA	BROJ POZITIVNIH SLIKA	BROJ NEGATIVNIH SLIKA	VRIJEME TRENIRANJA
PREDNJA/STRAŽNJA STRANA AUTOMOBILA	24	24x24	2118	4000	26h i 15min
BOČNA STRANA AUTOMOBILA	24	32x24	1940	4000	17h i 30min
PREDNJA/STRAŽNJA STRANA AUTOBUSA	15	24x24	1843	4000	26h i 10min
PJEŠAK	24	10x30	2081	4000	25h i 30min
BICIKLIST	25	24x24	2000	4000	21h i 20min

ZNAK OBAVEZNOG ZAUSTAVLJANJA	17	24x24	879	4000	11h i 20min
ZNAK RASKRIŽJA S CESTOM S PREDNOŠĆU PROLASKA	17	24x24	2000	4000	20h i 50min
ZNAKOVI OPASNOSTI	16	24x24	1987	4000	22h i 40min
ZNAK CESTE S PREDNOŠĆU PROLASKA	17	24x24	1876	4000	23h
ZNAKOVI IZRIČITIH NAREDBI S CRVENIM RUBOM	19	24x24	1816	4000	24h
ZNAKOVI IZRIČITIH NAREDBI S PLAVOM POZADINOM	19	24x24	1369	4000	23h
SEMAFOR	30	5x15	1205	4000	13h i 15min

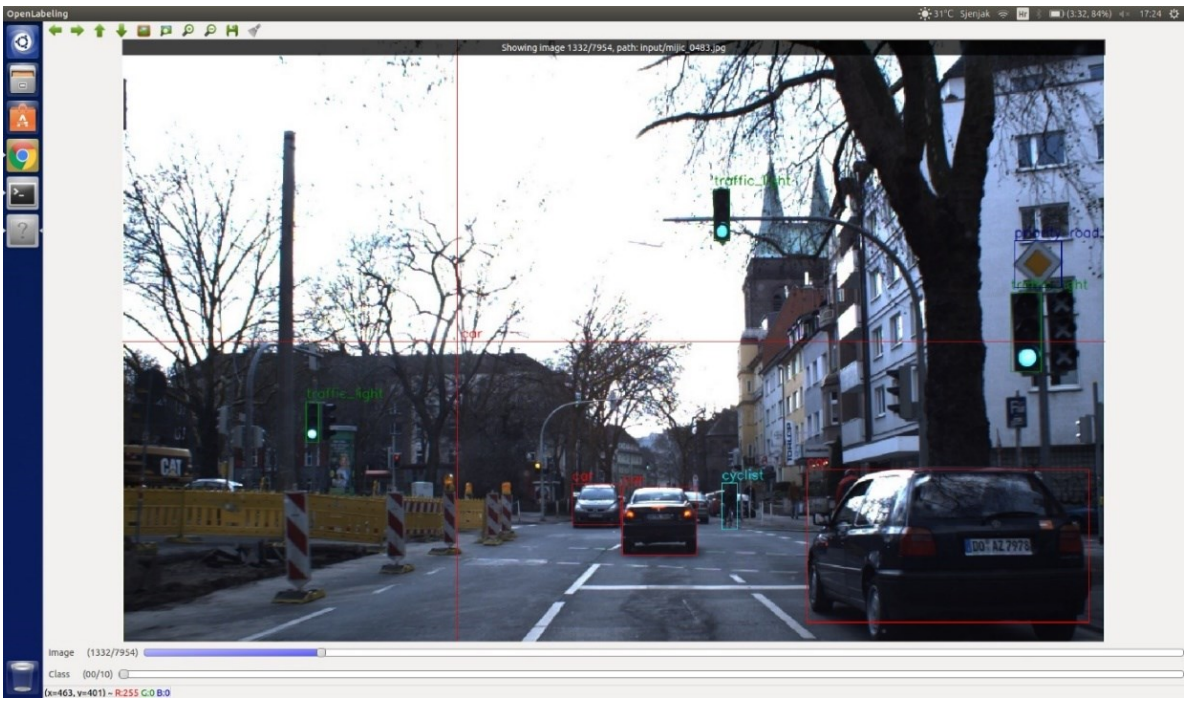
3.2.2. Treniranje CNN za algoritam zasnovan na YOLOv3 i *Tiny* YOLOv3 algoritmu za detekciju objekata na slikama

Nakon vožnje Osijekom i snimanja prometa, izdvojeni su okviri iz video sekvenci (proces opisan u potpoglavlju 3.2.). Baza slika za treniranje CNN sadrži ukupno 7360 slika, od čega je njih 670 iz KITTI baze, 920 iz video zapisa vožnje Münchenom te ostalih 5770 čine izdvojeni okviri iz video zapisa vožnje Osijekom. Preostalih 6416 okvira iz video zapisa vožnje Osijekom nisu korištene u procesu treniranja, već su ostavljene kao rezerva za potencijalno dodatno treniranje, no na kraju nije bilo potrebe za dodatnim treniranjem. Sve slike su pozitivne te sadrže barem jedan objekt neke od klasa koje je potrebno detektirati u ovom diplomskom radu. Sve slike koje su se koristile za proces treniranja detektora algoritma zasnovanog na YOLOv3, odnosno *Tiny* YOLOv3 algoritmu mogu se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.2. Zbog načina rada DarkNet CNN, potrebno je na slikama označiti sve objekte od interesa po klasama, a to je učinjeno koristeći alat OpenLabeling [39]. Ovaj alat kao izlaz daje tekstualni zapis svakog objekta na pojedinoj slici u obliku *klasa_objekta x_središta y_središta širina visina*, gdje su *x_središta,*

y_središta, širina i visina brojevi od 0,0 do 1,0 jer označavaju relativnu vrijednost širine i visine objekta u odnosu na širinu i visinu ulazne slike. Na slici 3.4. prikazan je primjer označenog objekta na slici rezolucije 447 x 447 elemenata slike. Primjer potpuno označene slike prikazan je na slici 3.5., dok je u tablici 3.3. prikazan konačan broj oznaka korištenih u treniranju CNN.



Sl. 3.4. *Primjer oznake objekta na slici rezolucije 447x447 elemenata slike s pripadajućim parametrima* [40]



Sl. 3.5. *Potpuno označena slika u OpenLabeling alatu*

Tab. 3.3. *Broj označenih objekata u bazi za treniranje CNN*

KLASA	BROJ OZNAKA
AUTOMOBIL	11727
AUTOBUS	1204
PJEŠAK	4254
BICIKLIST	930
ZNAK OBAVEZNOG ZAUSTAVLJANJA	1029
ZNAKOVI IZRIČITIH NAREDBI S PLAVOM POZADINOM	2254
ZNAKOVI IZRIČITIH NAREDBI S CRVENIM RUBOM	1867
ZNAKOVI OPASNOSTI	1044
ZNAK CESTE S PREDNOŠĆU PROLASKA	990
ZNAK RASKRIŽJA S CESTOM S PREDNOŠĆU PROLASKA	1373
SEMAFOR	3987
UKUPNO	**30659**

Slike su podijeljene na dva skupa, skup za treniranje i skup za validaciju u omjeru 85:15. Potrebno je kreirati tri datoteke: *naziv_mreže.data, naziv_mreže.names i naziv_mreže.cfg.* U datoteci *naziv_mreže.data* nalazi se broj klasa, putanja do tekstualnih datoteka sa popisom putanja slika za treniranje, odnosno validaciju mreže, zatim putanja do datoteke *naziv_mreže.names* i putanja za spremanje datoteka s težinama (slika 3.6. a)). U datoteci *naziv_mreže.names* navode se imena klasa, svaka klasa u poseban red (slika 3.6. b). Glavna konfiguracijska datoteka *naziv_mreže.cfg* sadrži sve potrebne parametre za treniranje i testiranje mreže te strukturu neuronske mreže. Na slici 3.7. prikazan je početak datoteke *naziv_mreže.cfg* u kojem su prikazani osnovni parametri konfiguracije mreže, poput širine (engl. *width*) i visine (engl. *height*) ulazne slike, stope učenja (engl. *learning rate*), moment (engl. *momentum*) itd. Sve tri datoteke mogu se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.3.3.

```
classes = 11
train = train.txt
valid = test.txt
names = jelic-yolov3.names
backup = backup/
```

a)

```
car
bus/truck
pedestrian
cyclist
stop_sign
mandatory_sign
prohibition_sign
warning_sign
priority_road_sign
yield_sign
traffic_light
```

b)

Sl. 3.6. a) *Sadržaj datoteke naziv_mreže.data* *b*) *Sadržaj datoteke naziv_mreže.names*

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=480
height=480
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
```

Sl. 3.7. *Isječak sadržaja datoteke naziv_mreže.cfg*

Unutar konfiguracijske datoteke potrebno je prilagoditi parametre vlastitim potrebama. U svakom sloju prije detekcije, potrebno je vrijednost filtera postaviti prema formuli:

$$filteri = (broj\_klasa + broj\_izlaza)\ x\ broj\_skala \tag{3-1}$$

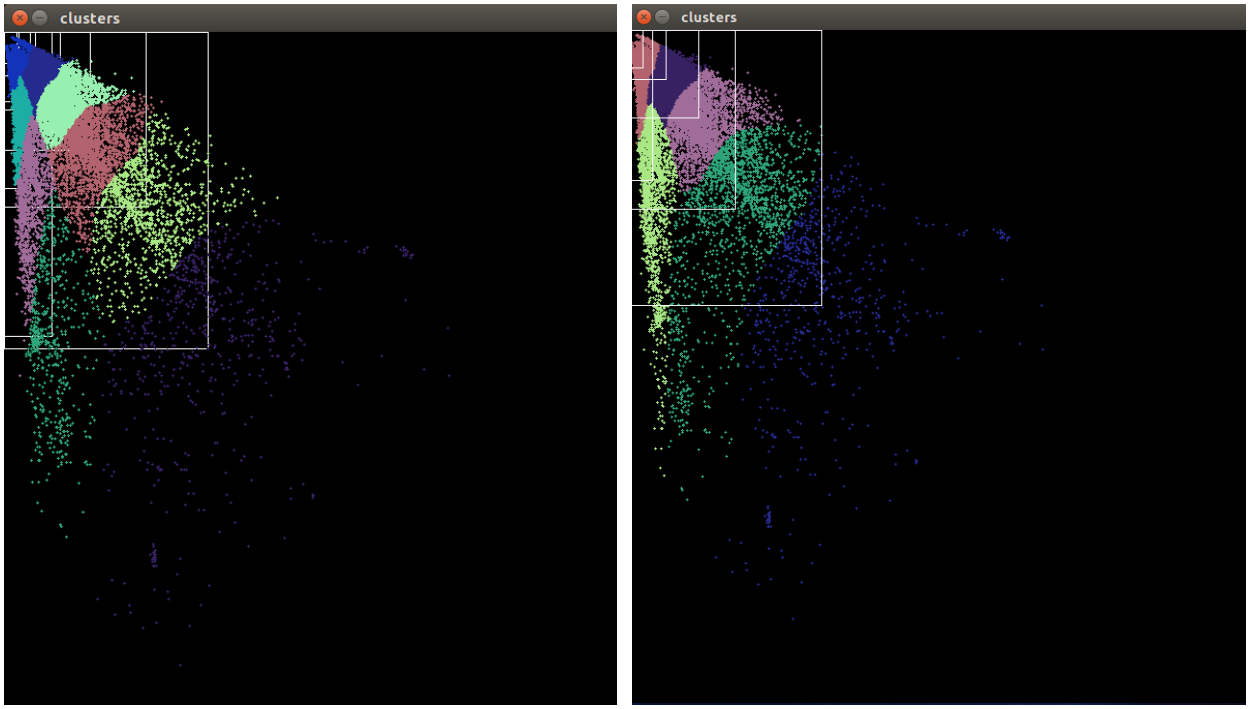
U formuli (3-1) koriste se sljedeće veličine:

- *filteri* – broj filtera u konvolucijskom sloju koji prethodi sloju detekcije,
- *broj_klasa* – broj klasa definiran u *naziv_mreže.data* datoteci,
- *broj_izlaza* – broj veličina u izlaznom vektoru, iznosi 5 (vjerojatnost predikcije, *x* i *y* koordinate lijevog gornjeg ugla graničnog pravokutnika, *x* i *y* koordinate donjeg desnog ugla graničnog pravokutnika),
- *broj_skala = 3,* jer YOLOv3 radi predikciju na tri skale.

Zatim je potrebno promijeniti vrijednosti predviđenih graničnih pravokutnika u svakom sloju detekcije. Pokreće se naredba za računanje:

./darknet detector calc_anchors putanja_do_data_datoteke -num_of_clusters [num_of_clusters] -width [width] -height [height] -show

Rezultati ove naredbe na vlastitom trening skupu podataka prikazani su na slikama 3.8. i 3.9.

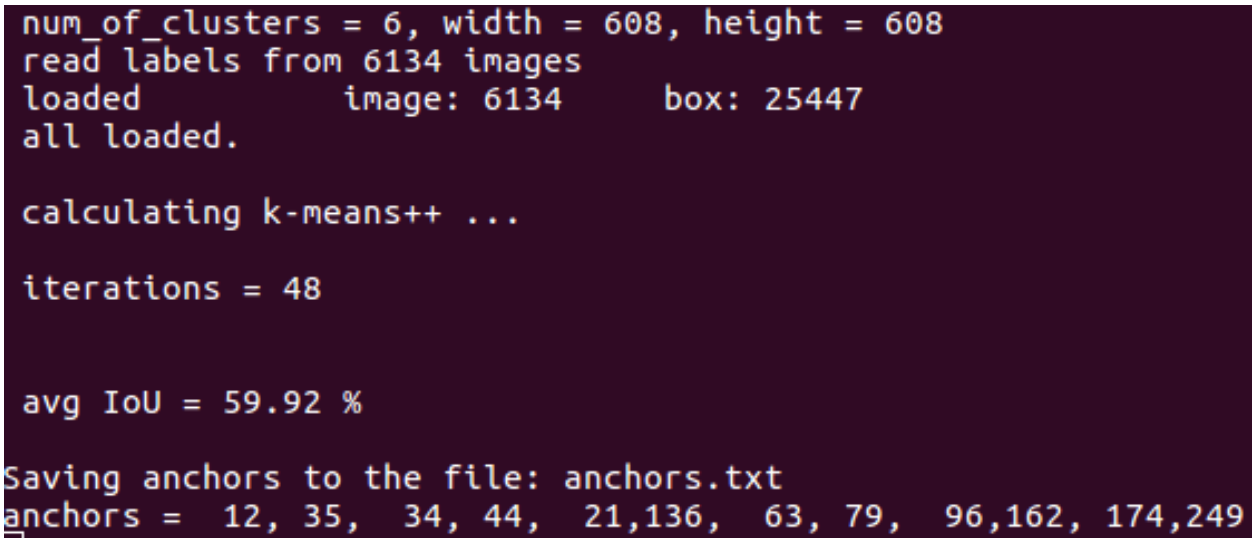


a) *b)*

Sl. 3.8. *Grafički prikaz klastera predviđenih graničnih pravokutnika za trening skup podataka za algoritme: a) YOLOv3 b) Tiny YOLOv3*

```
num_of_clusters = 9, width = 480, height = 480
read labels from 6135 images
loaded          image: 6135     box: 24689
all loaded.

calculating k-means++ ...

iterations = 96


avg IoU = 66.39 %

Saving anchors to the file: anchors.txt
anchors =   9, 21,  10, 52,  21, 29,  37, 46,  17,104,  57, 78,  32,202,  94,116, 135,210
```

a)



b)

Sl. 3.9. *Računski prikaz klastera predviđenih graničnih pravokutnika za trening skup podataka za algoritme: a) YOLOv3 b) Tiny YOLOv3*

Započeto je treniranje s osnovnom konfiguracijom koje je trajalo približno 72h na NVIDIA GeForce 1060 6GB GPU. Tekstualna datoteka s nazivima slika korištenim u procesu treniranja CNN može se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog 3.4. Treniranje se pokreće naredbom:

./darknet detector train putanja_do_data_datoteke putanja_do_konfiguracijske_datoteke putanja_do_datoteke_sa_inicijalnim_težinama -map

u upravljačkom prozoru. Određivanje minimalnog broja iteracija ovisi o broju klasa po kojima se vrši klasifikacija. Općenito pravilo je minimalno 1500 x *broj_klasa* [41] (u ovom slučaju 16500 iteracija) no kriterij zaustavljanja postavljen je na 25000 iteracija kako bi pokušali dobiti što bolje istreniranu mrežu. Kriterij zaustavljanja za algoritam *Tiny* YOLOv3 postavljen je na 30000 iz istog razloga, no za 5000 iteracija više jer je *Tiny* YOLOv3 manje precizan.

Cilj treniranja je smanjiti vrijednost kriterijske funkcije što više. Kriterijska funkcija predstavlja kriterij za evaluaciju rješenja, u ovom slučaju evaluaciju skupa težina za neuronsku mrežu. Kriterijska funkcija se sastoji od tri dijela: klasifikacijske greške (3-1), lokalizacijske greške (3-2) i detekcijske greške, ovisno je li objekt detektiran ili ne (3-3). Klasifikacijska greška (KG) računa pogrešku za svaku ćeliju, a računa se kao suma kvadrata razlike uvjetnih vjerojatnosti predikcije svake klase.

$$KG = \sum_{i=0}^{s^2} 1_i^{obj} \sum_{k \in klase} (p_i(k) - \hat{p}_i(k))^2, \quad (3\text{-}1)$$

gdje je:

- s^2 – broj ćelija na koji je slika podijeljena,
- $1_i^{obj} = 1$, ako se objekt pojavi u ćeliji *i,* inače 0,
- $p_i(k)$ – vjerojatnost predikcije objekta klase *k* u ćeliji *i,*
- $\hat{p}_i(k)$ – uvjetna vjerojatnost predikcije objekta klase *k* u ćeliji *i.*

Lokalizacijska greška (LG) mjeri pogrešku lokacije i veličine detektiranih graničnih pravokutnika. Gleda se samo granični pravokutnik odgovoran za detektirani objekt. Nije poželjno da pogreška u par elemenata slike nosi jednaku težinu prilikom predikcije velikog graničnog pravokutnika i predikcije malog graničnog pravokutnika. Stoga, YOLO kao predikciju daje drugi korijen širine i visine graničnog pravokutnika. Lokalizacijska greška računa se na slijedeći način:

$$LG = \lambda_{koord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] +$$
$$+ \lambda_{koord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right], \quad (3\text{-}2)$$

gdje je:

- λ_{koord} – koeficijent povećanja težine za grešku u koordinatama graničnog pravokutnika,
- *B* – broj predviđenih graničnih pravokutnika u ćeliji *i,*
- $1_{ij}^{obj} = 1$, ako je *j*-ti granični pravokutnik u ćeliji *i* odgovoran za detekciju objekta, inače 0,
- x_i – *x* koordinata središta predviđenog graničnog pravokutnika objekta u ćeliji *i*,
- $\hat{x}_i$ – *x* koordinata središta graničnog pravokutnika kojega je algoritam detektirao u ćeliji *i*,
- y_i – *y* koordinata središta predviđenog graničnog pravokutnika objekta u ćeliji *i*,
- $\hat{y}_i$ – y koordinata središta graničnog pravokutnika kojega je algoritam detektirao u ćeliji *i*,
- w_i – širina predviđenog graničnog pravokutnika objekta u ćeliji *i*,
- $\hat{w}_i$ – širina graničnog pravokutnika objekta kojega je algoritam detektirao u ćeliji *i*,
- h_i – visina predviđenog graničnog pravokutnika objekta u ćeliji *i*,
- $\hat{h}_i$ – visina graničnog pravokutnika objekta kojega je algoritam detektirao u ćeliji *i*.

Pošto većina ćelija ne sadrži nijedan objekt, to predstavlja neravnotežu u klasifikaciji. Stoga se težine ažuriraju uz faktor λ_{noobj}. Detekcijska greška (DG) računa se na sljedeći način:

$$DG = \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (c_i - \hat{c}_i)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (c_i - \hat{c}_i)^2, \quad (3\text{-}3)$$

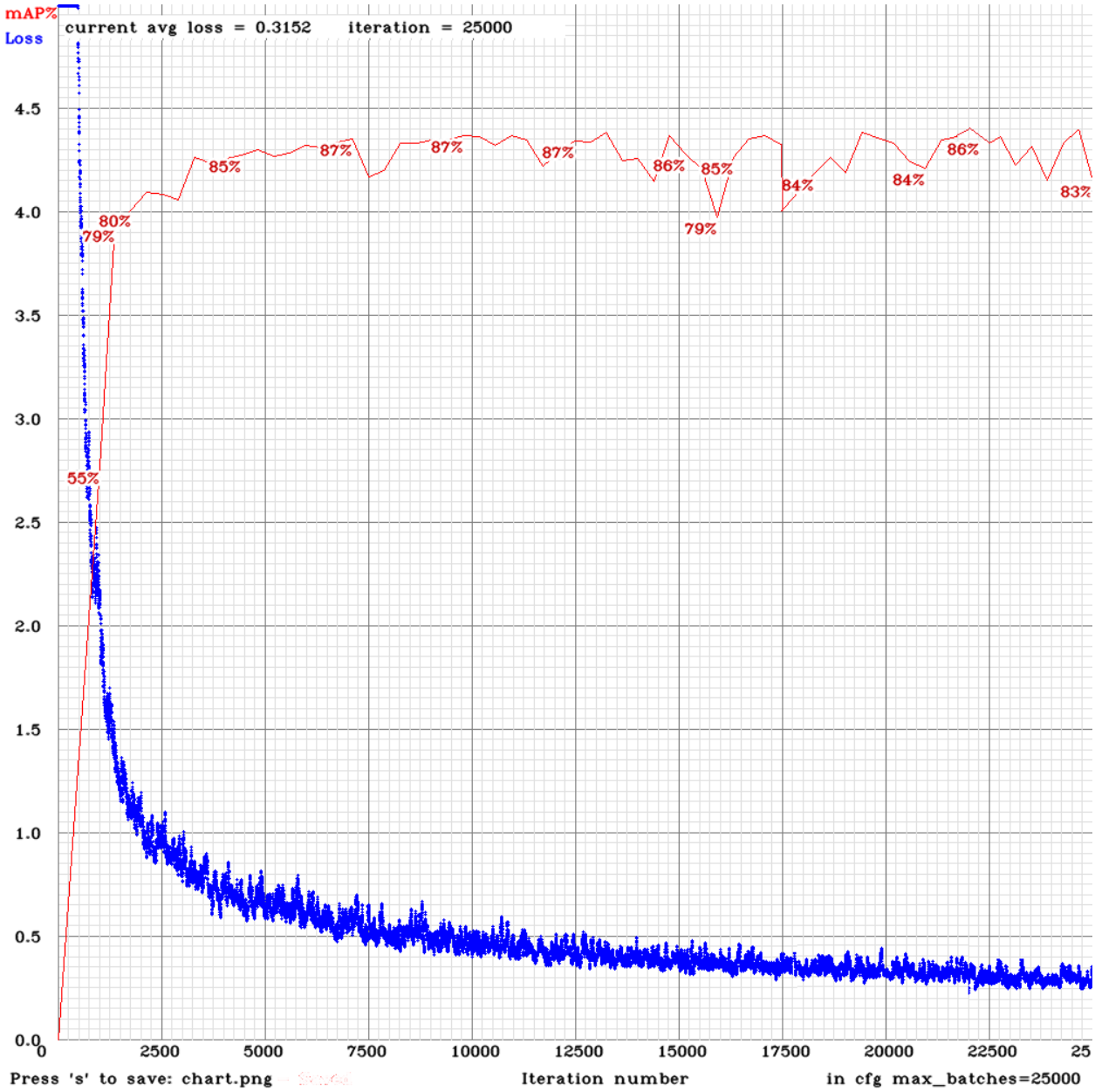
gdje je:

- c_i – vjerojatnost predikcije objekta,
- $\hat{c}_i$ – vrijednost IoU predviđenog i stvarnog graničnog pravokutnika,
- λ_{noobj} – koeficijent smanjenja greške prilikom detekcije pozadine,
- 1_{ij}^{noobj} – komplement od 1_{ij}^{obj}.

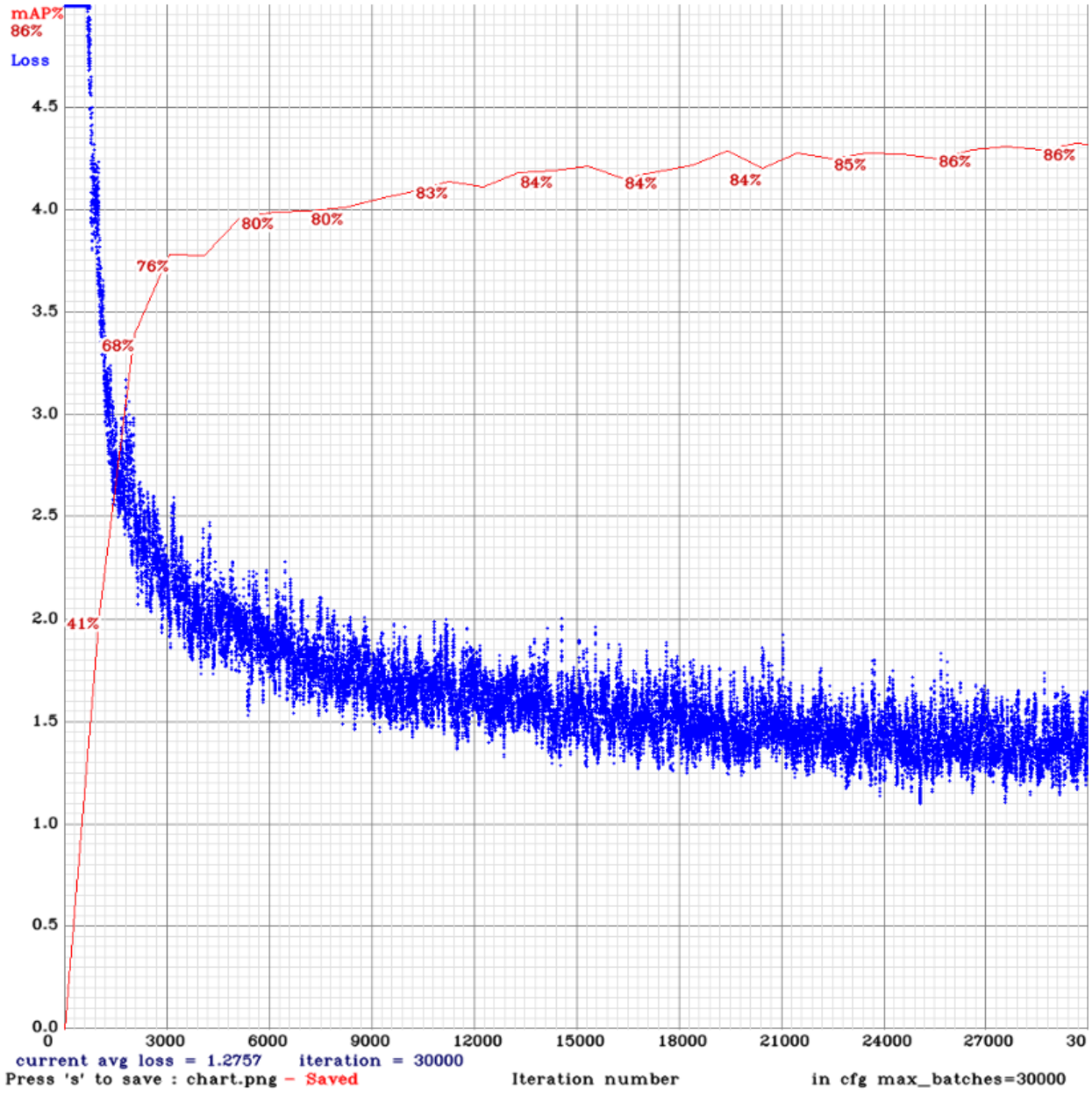
Konačna kriterijska funkcija (KF) se dobije zbrojem prethodno navedenih funkcija grešaka KG, LG i DG (3-4).

$$KF = KG + LG + DG \quad (3\text{-}4)$$

Vrijednost kriterijske funkcije može biti u rasponu $[0{,}0\,, +\infty\rangle$, a cilj ju je minimizirati. Mogućnost pretjeranog usklađivanja na trening podatke, pri čemu dolazi do smanjene sposobnosti generalizacije (engl. *overfitting*) pojavljuje se pri vrijednostima kriterijske funkcije od 0,06 i manje. Vrijednost 0,06 uzeta je kao smjernica (engl. *rule of thumb*) koja ne mora biti točna za svaku CNN [42]. Rezultat treniranja konačne CNN za YOLOv3 algoritam, odnosno graf vrijednosti kriterijske funkcije i prosječne preciznosti detekcije (na slikama 3.10. i 3.11. prikazan kao *mAP%* parametar) u odnosu na broj iteracija pri treniranju prikazan je na slici 3.10. Vrijednost kriterijske funkcije (na slikama 3.10. i 3.11. prikazan kao *current avg loss* parametar) na kraju treniranja YOLOv3 CNN iznosi 0,3152. Rezultat treniranja konačne CNN za *Tiny* YOLOv3 algoritam prikazan je na slici 3.11. Vrijednost kriterijske funkcije na kraju treniranja *Tiny* YOLOv3 CNN iznosi 1,2757. Usporedbom slika 3.10. i 3.11. može se vidjeti da je vrijednost kriterijske funkcije za *Tiny* YOLOv3 CNN višestruko veća od vrijednosti kriterijske funkcije YOLOv3 CNN, što znači da je veća pogreška na trening skupu podataka.



Sl. 3.10. *Graf vrijednosti kriterijske funkcije i prosječne preciznosti detekcija u odnosu na broj iteracija provedenih u postupku treniranja CNN za YOLOv3 algoritam*



Sl. 3.11. *Graf vrijednosti kriterijske funkcije i prosječne preciznosti detekcija u odnosu na broj iteracija provedenih u postupku treniranja CNN za Tiny YOLOv3 algoritam*

3.3. Izrada algoritma zasnovanog na Viola-Jones algoritmu

U ovom potpoglavlju opisana je izrada algoritma za identifikaciju objekata zasnovanog na Viola-Jones algoritmu. Računalni algoritam izrađen je u C++ programskom jeziku u ROS radnom okruženju radi lakše implementacije na ugradbenu platformu. Algoritam koristi višenitnost, odnosno obavlja proces svih 12 detekcija istovremeno. Pri izradi algoritma korištena je OpenCV biblioteka i OpenMP [43] aplikacijsko programsko sučelje za paralelno procesiranje i višenitnost. Prilikom izrade algoritma za identifikaciju, korišten je algoritam za praćenje objekata Median

Flow [44], čime se dobiva ušteda na računalnim resursima jer je praćenje računalno manje zahtjevno od detekcije te je robusnije na promjenu orijentacije objekta [45], [46].

Algoritam za identifikaciju sastavljen je od glavnog modula (*main.cpp*) i modula za detekciju i praćenje (*detect.cpp* i *detect.h*). Unutar glavnog modula deklarirano je polje od 12 struktura tipa *functionArgs* koje sadrže osnovne podatke potrebne funkcijama za detekciju i praćenje:

- *uint8_t ID* – unikatan identifikacijski broj detektora
- *string objectName* – klasa detektora, odnosno ime detektiranih objekata
- *Mat* img* – ulazna slika na kojoj se vrši detekcija
- *CascadeClassifier objectCC* – objekt klase *CascadeClassifier*
- *int minNeighbours* – parametar za funkciju *detectMultiScale,* označava broj detektiranih susjednih graničnih pravokutnika potrebnih da bi se detekcija zadržala
- *double scaleFactor* – parametar za funkciju *detectMultiScale,* označava postotak skaliranja ulazne slike
- *Scalar rectangleColor* – parametar koji određuje boju prikazanog graničnog pravokutnika detektora
- *vector<Rect> rects* – vektor graničnih pravokutnika detektiranih objekata

Inicijalizirane su sve strukture te deklariran broj niti. Zatim je pozvana funkcija za otvaranje video datoteke. Ostatak koda izvršava se unutar *while* petlje čiji je uvjet izvršavanja postojanje ulaznog video okvira. Ulaznom video okviru je promijenjena razlučivost sa 1280 x 720 na 640 x 480 elemenata slike radi bržeg rada algoritma. Uvećan je brojač okvira *frameCounter* za jedan te se ulazi u dio koda koji je paraleliziran pomoću OpenMP sučelja. Definirano je paralelno izvršavanje petlje *for* koja ima 12 iteracija, odnosno poziva se svih 12 detektora. Svakom detektoru prosljeđuje se vlastita struktura s parametrima i redni broj okvira. Sinkronizacija niti je osigurana unutar paralelne regije koda kao i pristup memoriji te rješavanje ulazaka niti u kritični odsječak. Nakon obavljanja detekcije, prikazani su detektirani granični pravokutnici na video okviru te je prikazan broj FPS koji predstavlja broj okvira koji je moguće obraditi u jednoj sekundi.

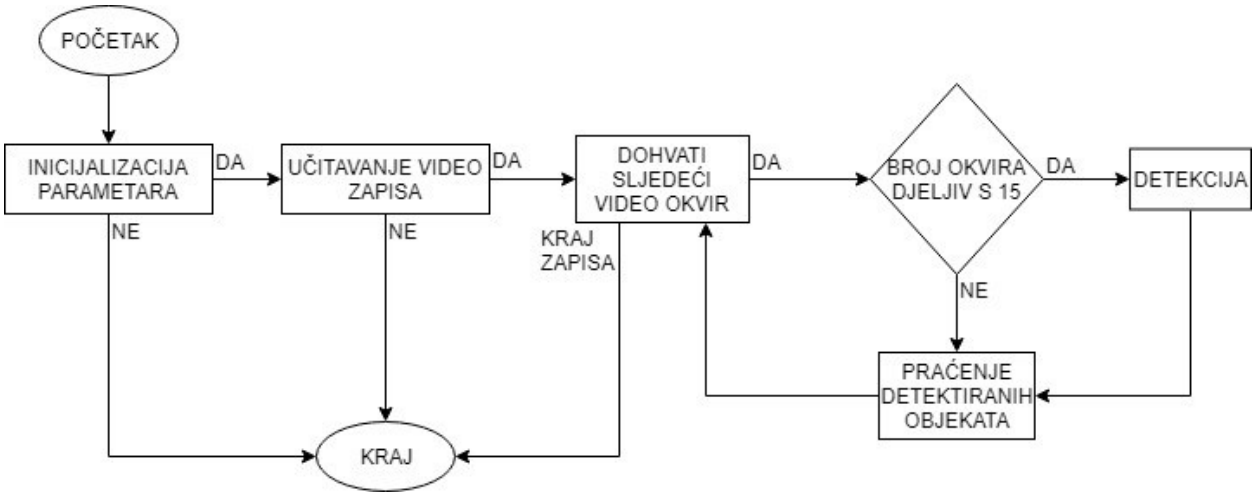
Funkcija za detekciju i praćenje *detectAndTrack* radi na sljedećem principu. Detekcija se obavlja na svakom 15. video okviru, dok se u ostalih 14 obavlja praćenje Median Flow algoritmom. Ulazna slika se prebacuje u nijanse sive boje. Zatim se poziva funkcija za detekciju

detectMultiScale iz OpenCV biblioteke. Funkcija prima ulaznu sliku u nijansama sive boje, polje podataka tipa *Rect* za spremanje detektiranih graničnih pravokutnika te parametre *scaleFactor* i *minNeighbours*. Funkcija se poziva kao metoda objekta *objectCC* proslijeđenog u strukturi *functionArgs*. Pretpostavka je da se unutar 15 okvira, odnosno u pola sekunde neće pojaviti novi objekt (ulazni video je 30 FPS). Na slici 3.12. prikazan je dijagram toka algoritma. Svakom detektiranom objektu kreira se vlastiti *tracker* funkcijom *createTrackerByName* te se on dodaje objektu *multiTracker* klase *MultiTracker* pomoću metode *add.* Metoda *add* za parametre prima objekt *tracker,* okvir na kojem se prati objekt te dimenzije graničnog pravokutnika. Nakon detekcije, funkcijom *update* klase *MultiTracker* ažuriraju se lokacije prethodno praćenih objekata svih *trackera*. Također, svi novo-detektirani objekti su praćeni. Algoritam završava kada ulazni video zapis završi. Kompajliranje koda radi se pomoću naredbe:

mpiCC module.cpp main.cpp –o detection 'pkg-config --cflags --libs opencv' -std=c++11 -fopenmp

a izvršna datoteka (*detection*) se pokreće unutar upravljačkog prozora korištenjem naredbe:

./detection naziv_video_zapisa

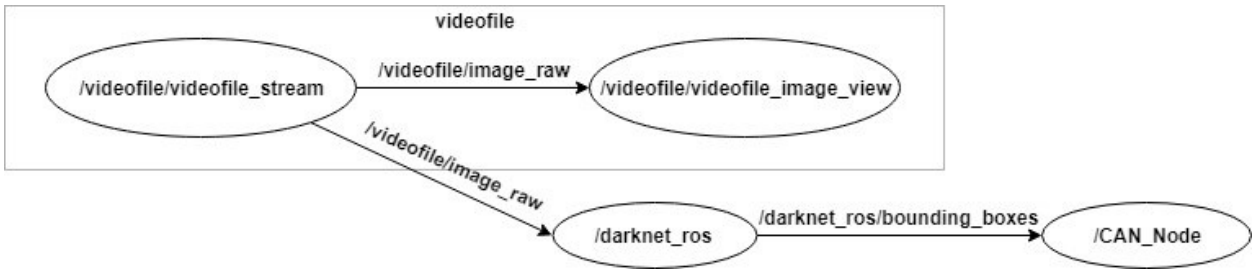


Sl. 3.12. *Dijagram tijeka algoritma zasnovanog na Viola-Jones algoritma*

Sav programski kod algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata je zatim prebačen ROS paket *image_transport,* koji se sastoji od dva čvora: *V-J* čvor, odnosno čvor u kojem je implementiran vlastiti algoritam zasnovan na Viola-Jones algoritmu te *CAN* čvor. Ulazna slika dobiva se pretplaćivanjem na ROS temu */videofile/image_raw video_stream_opencv* paketa.

3.4. Izrada algoritma zasnovanog na YOLOv3 algoritmu

U ovom potpoglavlju detaljno je opisana izrada algoritma za identifikaciju objekata zasnovanog na YOLOv3 algoritmu. Nakon izrade baze slika za treniranje i testiranje te završenog treniranja, kreirana je datoteka *jelic-yolov3.weights* koja sadrži sve težine za CNN YOLOv3. Algoritam detekcije preuzet je sa GitHub stranice [47] autora Marka Bjelonica. To je ROS paket razvijen za detekciju objekata na video okvirima. Unutar ROS paketa može se koristiti YOLO pokretanjem na GPU i procesora (engl. *central processing unit,* CPU). U diplomskom radu koristi se YOLOv3 i *Tiny* YOLOv3. Na slici 3.13. prikazan je dijagram tijeka algoritma preko ROS čvorova i tema dobiven putem ROS alata *rqt_graph* [48].



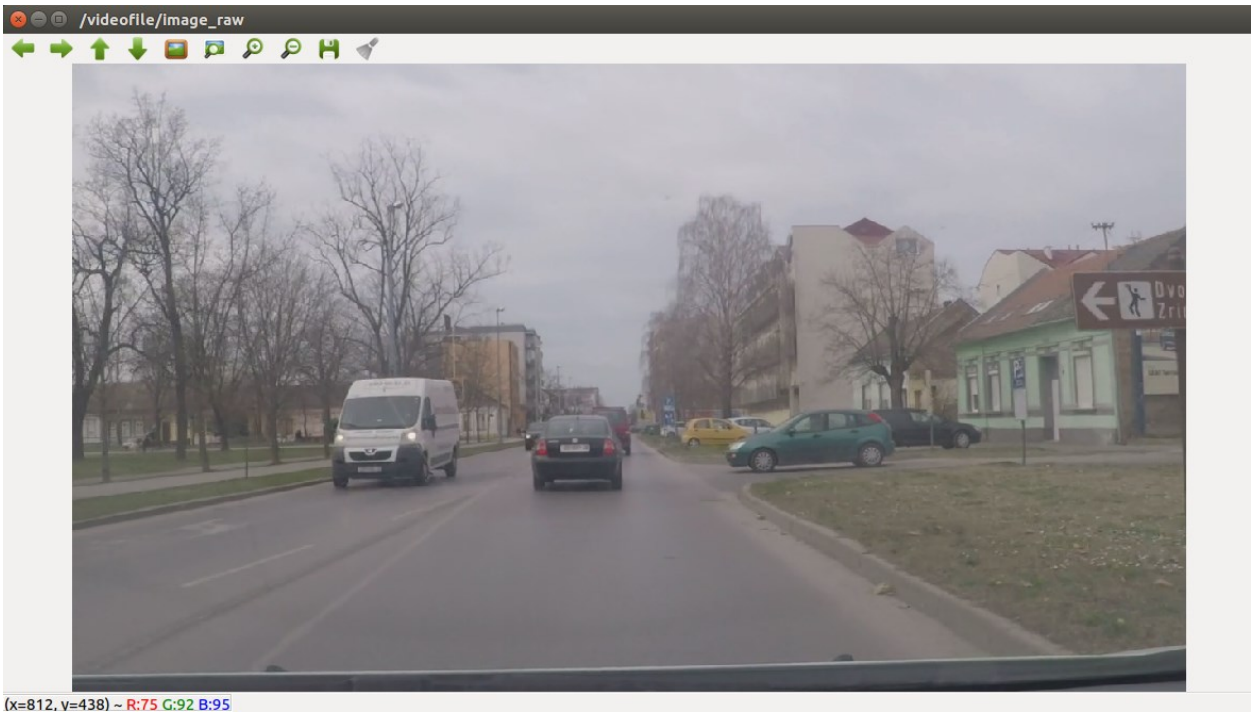
Sl. 3.13. *Dijagram tijeka algoritma preko ROS čvorova i tema*

Za pokretanje i prosljeđivanje video okvira koristi se *video_stream_opencv* paket. Video okviri se šalju u ROS temu */videofile/image_raw* na koju se čvor za detekciju mora pretplatiti. Čvor za detekciju potrebno je konfigurirati da prima video okvire sa odgovarajuće ROS teme te mu proslijediti putanju datoteke s težinama za CNN. To se konfigurira u *ros.yaml* i *yolov3.yaml* datotekama. YAML [49] je standard često korišten za pisanje konfiguracijskih datoteka jer je sintaksa prilagođena korisnicima (engl. *user friendly*) i lagan je za čitanje, a može se koristiti u bilo kakvim aplikacijama gdje se pohranjuju podaci. U *yolov3.yaml* navodi se naziv konfiguracijske datoteke, datoteke sa težinama, prag za prikazivanje detekcije (engl. *threshold*) te popis klasa, dok se u *ros.yaml* datoteci navodi naziv ROS tema za pretplaćivanje, primjerice */videofile/image_raw*, odnosno objavljivanje, primjerice */darknet_ros/bounding_boxes*. Također potrebno je prebaciti datoteku s težinama i konfiguracijsku datoteku u mapu *yolo_network_config.* Potrebno je konfigurirati i datoteku *darknet_ros.launch* ukoliko se kreira vlastita YAML datoteke. Nakon odrađene konfiguracije, testiranje, odnosno detekcija pokreće se naredbom:

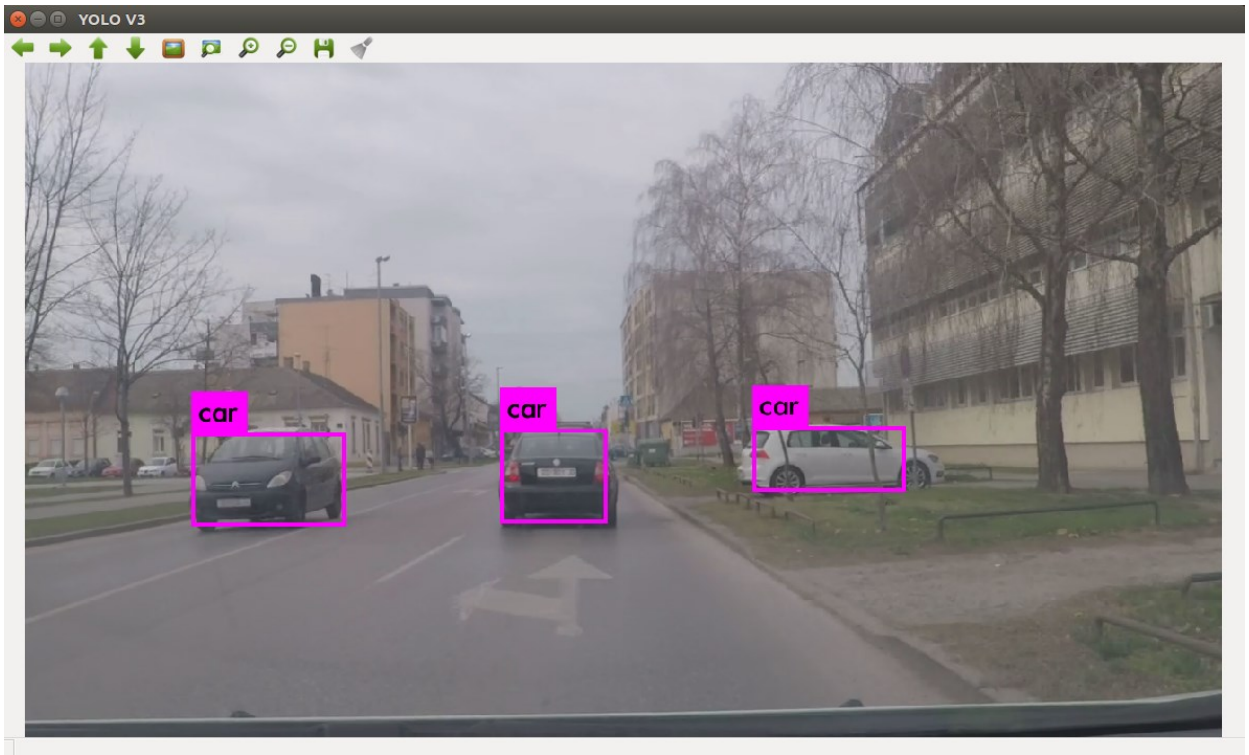
roslaunch darknet_ros darknet_ros.launch

Kreira se CNN prema konfiguracijskoj datoteci te se ulazna slika uzima s ROS teme definirane u *ros.yaml.* Nakon obrade prvog video okvira, na ekranu se prikazuje video okvir s

označenim detekcijama, odnosno graničnim pravokutnicima te nazivom klase. U upravljačkom prozoru se prikazuje broj okvira koji čvor može obraditi u sekundi te vjerojatnost sigurnosti za svaku detektiranu klasu. Na slikama 3.14. i 3.15. prikazan je rad *video_stream_opencv* i YOLOv3 čvorova.



Sl. 3.14. *Prikaz izlaza video_stream_opencv čvora*



Sl. 3.15. *Prikaz izlaza YOLOv3 čvora*

Nakon testiranja inicijalne CNN s vlastitim težinama i konfiguracijom, bilo je potrebno napisati vlastiti ROS čvor koji će dohvaćati podatke o detektiranim objektima te ih slati putem CAN sabirnice. Novi ROS čvor *CAN_Node* pretplaćen je na ROS temu kreiranu od detektora */darknet_ros/bounding_boxes* gdje se šalju podaci o detektiranim objektima u obliku poruka, odnosno *message* (.msg) datoteka. Sadržaj *BoundingBoxes.msg* i *BoundingBox.msg* datoteka prikazan je na slikama 3.16. i 3.17.

```
1. Header header
2. Header image_header
3. BoundingBox[] bounding_boxes
```

Sl. 3.16. *Sadržaj BoundingBoxes.msg datoteke*

```
1. string Class
2. float64 probability
3. int64 xmin
4. int64 ymin
5. int64 xmax
6. int64 ymax
```

Sl. 3.17. *Sadržaj BoundingBox.msg datoteke*

Pristupom polju poruka *bounding_boxes* tipa *BoundingBox* može se za svaku detekciju očitati klasa, vjerojatnost pripadanja pojedinoj klasi te dvije koordinate (x_{min}, y_{min}) i (x_{max}, y_{max}). (x_{min}, y_{min}) označavaju koordinate gornjeg lijevog ugla graničnog pravokutnika, a (x_{max}, y_{max}) koordinate donjeg desnog ugla. Oduzimanjem x_{min} od x_{max} dobiva se širina graničnog pravokutnika. Analogno tome, preko y koordinate dobije se visina graničnog pravokutnika. Preko CAN sabirnice šalje se naziv klase u obliku cijelog broja u rasponu $[0, broj\_klasa\rangle$, koordinate gornjeg lijevog ugla (x, y) te širina i visina. Cjelobrojne oznake klasa odgovaraju redoslijedu navođenja klasa prilikom treniranja CNN u *naziv_mreže.names* datoteci:

- 0 – automobil
- 1 – autobus/kamion
- 2 – pješak
- 3 – biciklist
- 4 – znak obaveznog zaustavljanja
- 5 – znakovi izričitih naredbi s plavom pozadinom
- 6 – znakovi izričitih naredbi s crvenim rubom

- 7 – znakovi opasnosti
- 8 – znak ceste s prednošću prolaska
- 9 – znak raskrižja s cestom s prednošću prolaska
- 10 – semafor.

Struktura CAN poruke prikazana je na slici 3.18. Koristi se ROS paket *ros_canopen* [50].

```
1. Header header
2. uint32 id
3. bool is_rtr
4. bool is_extended
5. bool is_error
6. uint8 dlc
7. uint8[8] data
```

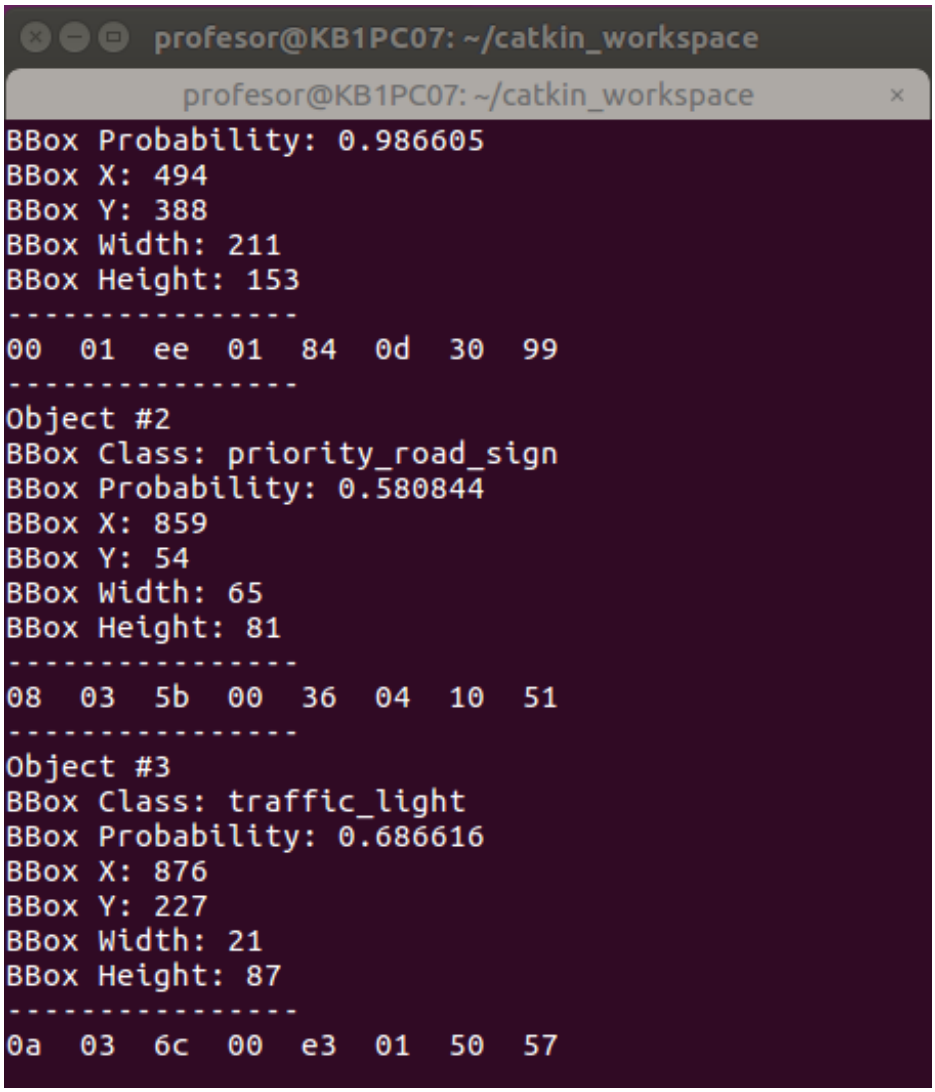
Sl. 3.18. *Struktura CAN poruke*

U polje *data* potrebno je ubaciti prethodno navedene podatke kao cijele brojeve. Zbog CAN standarda, podatkovno polje može biti dugačko 8 bajtova. Struktura podatkovnog polja vlastite CAN poruke prikazana je na slici 3.19.

4 bit	4 bit	5 bit	11 bit	5 bit	11 bit	1 bit	11 bit	1 bit	11 bit
REZ	KLASA	REZ	X KOORD.	REZ	Y KOORD.	REZ	ŠIRINA	REZ	VISINA

Sl. 3.19. *Struktura podatkovnog polja vlastite CAN poruke*

Bitovi u polju REZ su rezervirani bitovi za mogućnost proširenja u budućnosti. Svaki detektirani objekt je jedna CAN poruka s ID-em 0x1213. CAN poruke se šalju drugim modulima u automobilu koji iz njih izvlače potrebne podatke te se ponašaju u ovisnosti o detektiranim objektima. Na slici 3.20. prikazan je izgled upravljačkog prozora s ispisom detekcija i sadržajem podatkovnog polja CAN poruke korištenjem *Tiny* YOLOv3. Za svaki detektirani objekt prikazan je broj objekta na slici, pripadnost klasi (*BBox Class*), vjerojatnost predikcije (*BBox Probability*), koordinate gornjeg lijevog ugla (*BBox X* i *BBox Y*), širina (*BBox Width*) i visina (*BBox Height*) graničnog pravokutnika te zapis CAN poruke u 8 bajtova.



Sl. 3.20. *Prikaz izlaza iz CAN čvora*

4. EVALUACIJA I VERIFIKACIJA IZRAĐENIH ALGORITAMA ZA IDENTIFIKACIJU OBJEKATA U SLIKAMA DOBIVENIM POMOĆU PREDNJE KAMERE NA VOZILU

U ovom poglavlju opisan je proces verifikacije rada izrađenih algoritama i njihova evaluacija na testnom skupu slika i video sekvenci. Inicijalnim pokretanjem oba algoritma nad testnom slikom uzetom iz skupa slika koje nisu korištene prilikom treniranja, Viola-Jones algoritmu treba više od jedne sekunde za inferenciju (proces zaključivanja na temelju iskustava stečenih u procesu treniranja), što izlazi iz okvira rada u stvarnom vremenu. Dodatnim testiranjem isključivo na slikama koje su izdvojeni video okviri prometa u Osijeku, prosječno vrijeme obrade slike i dalje iznosi više od jedne sekunde. S druge strane, testiranjem CNN na istim slikama, izmjereno je vrijeme inferencije od približno 50 milisekundi. U potpoglavlju 4.1. prikazani su rezultati testiranja na skupu slika. U potpoglavlju 4.2. prikazani su rezultati testiranja na video zapisu. U potpoglavlju 4.3. dana je analiza testiranja te su dani određeni zaključci.

4.1. Verifikacija na skupu slika

U ovom potpoglavlju opisana je verifikacija ispravnosti rada algoritma na skupu slika. Prilikom treniranja CNN sprema se datoteka s težinama svakih 1000 iteracija. Prema rezultantnim grafovima prikazanim u odjeljku 3.2.2. u obzir su uzete težine pri kojima je prosječna preciznost bila najveća, ali nakon 16500 iteracija zbog pravila opisanog također u odjeljku 3.2.2. To su iteracije 18900, 19900 i 20900 te konačne težine nakon 25000 iteracija za YOLOv3 CNN, odnosno iteracije 22000, 27000 te konačne težine nakon 30000 iteracija za *Tiny* YOLOv3 CNN. Testni skup slika sadrži 15% od ukupnog skupa slika koji je korišten za treniranje CNN opisanog u odjeljku 3.2.2. Taj skup od 1226 slika nije bio korišten u procesu treniranja, a na njima se nalazi 4781 stvarnih objekata koje je potrebno detektirati. Algoritam zasnovan na Viola-Jones algoritmu za detekciju objekata testiran je na smanjenom skupu od 100 slika koje sadrže 417 stvarnih objekata koje je potrebno detektirati. Test je proveden na smanjenom testnom skupu slika zbog ograničenog vremenskog roka te nemogućnosti automatizacije testiranja kao kod YOLOv3, odnosno *Tiny* YOLOv3 algoritama. Tekstualne datoteke s nazivima slika korištenih u procesu testiranja izrađenih algoritama mogu se pronaći na DVD-u priloženom uz ovaj rad u mapama Prilog P.4.1. i Prilog P.4.2.

Ključni parametri koji su promatrani tijekom usporedbe performansi algoritama su preciznost, odziv, točnost, mjera preciznosti testiranja (F1 mjera), broj točno detektiranih objekata,

prosječni IoU te prosječna preciznost prema klasama kao što je učinjeno u [6] i [23]. Formule za izračun mjera preciznosti, odziva, točnosti i F1 mjere prikazane su u potpoglavlju 2.2.

Parametri *scaleFactor* i *minNeighbours* korišteni za testiranje vlastitog algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata dobiveni su empirijskim putem isprobavanjem na 10 testnih slika, a prikazani su u tablici 4.1.

Tab. 4.1. *Konačni parametri scaleFactor i minNeighbours korišteni pri testiranju vlastitog algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata*

DETEKTOR	*scaleFactor*	*minNeighbours*
PREDNJA/STRAŽNJA STRANA AUTOMOBILA	1,05	5
BOČNA STRANA AUTOMOBILA	1,08	7
PREDNJA/STRAŽNJA STRANA AUTOBUSA	1,3	10
PJEŠAK	1,4	7
BICIKLIST	1,3	7
ZNAK OBAVEZNOG ZAUSTAVLJANJA	1,07	4
ZNAK RASKRIŽJA S CESTOM S PREDNOŠĆU PROLASKA	1,3	9
ZNAKOVI OPASNOSTI	1,025	4
ZNAK CESTE S PREDNOŠĆU PROLASKA	1,07	6
ZNAKOVI IZRIČITIH NAREDBI S CRVENIM RUBOM	1,03	7
ZNAKOVI IZRIČITIH NAREDBI S PLAVOM POZADINOM	1,02	5
SEMAFOR	1,03	5

Rezultati testiranja algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata prikazani su u tablici 4.2. Također je izračunat prosječni *IoU* detektora tako što su zbrojeni *IoU* svih TP detekcija i taj je zbroj podijeljen s brojem TP detekcija. Klase su označene cijelim brojevima opisanim u potpoglavlju 3.4.

Tab. 4.2. *Rezultati testiranja algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata na testnom skupu slika (ukupno testirano na 100 slika)*

KLASA	BROJ TOČNO DETEKTIRANIH OBJEKATA (TP)	BROJ LAŽNO DETEKTIRANIH OBJEKATA (FP)	BROJ NEDETEKTIRANIH OBJEKATA (FN)
0	85	13	83
1	2	8	16
2	15	60	45
3	2	0	6
4	6	5	10
5	15	42	15
6	11	38	17
7	10	4	12
8	4	10	6
9	6	1	13
10	5	2	33
SVOJSTVA DETEKTORA			
PROSJEČNI IoU	54,86%		
PRECIZNOST	0,482		
ODZIV	0,3861		
F1 MJERA	0,4288		

Analizom podataka prikazanih u tablici 4.2., može se vidjeti da su detektori loše detektirali potrebne objekte na testnom skupu slika, jer točno detektiraju ukupno 161 od 417 objekata. Najtočnije detektiraju prometne znakove i automobile, dok pješake i semafore jako loše. U ovom testu dolaze do izražaja nedostatci Viola-Jones algoritma, a to je neotpornost na rotaciju i okluziju (djelomično zaklanjanje objekta), kao i to da prilikom smanjivanja rezolucije slike dolazi do nemogućnosti detekcije ponekih objekata jer su na slikama veličinom manji od najmanje moguće dimenzije za detekciju (pogledati tablicu 3.5. u potpoglavlju 3.2.). Također, prosječni *IoU* je relativno nizak, što znači da se lokacija stvarnog objekta i lokacija detektiranog objekta ne poklapaju dovoljno, čime dolazi do velike lokalizacijske greške i netočnosti algoritma.

Usporedba rezultata YOLOv3 i *Tiny* YOLOv3 CNN je prikazana u tablicama 4.3. i 4.4. Ulazna slika u YOLOv3 CNN je veličine 480 x 480 elemenata slike, dok je ulazna slika u *Tiny* YOLOv3 CNN rezolucije 800 x 800 elemenata slike. Donji dio tablice predstavlja rezultate prilikom uzimanja praga vjerojatnosti predikcije 0,25, dok gornji dio prikazuje postotak točno detektiranih objekata pojedine klase prilikom odabiranja IoU kriterija. Vjerojatnost predikcije

predstavlja sigurnost modela da detektirani objekt pripada određenoj klasi. Izrađena je statistika za tri vrijednosti IoU: 25%, 50% te 75%. Statistika se dobiva pomoću naredbe:

./darknet detector map putanja_do_data_datoteke putanja_do_konfiguracijske_datoteke putanja_do_datoteke_sa_težinama -iou_thresh [iou_thresh]

Analizom dobivenih rezultata, može se vidjeti da su najmanji postotci točnih detekcija u klasama pješak i semafor, nakon čega slijedi klasa biciklist. To je prvenstveno zbog oblika objekata tih klasa, koji su tipično uski te zauzimaju malu površinu na slici. Uspoređujući klasu biciklist i pješak, vrlo ih je lako međusobno zamijeniti (biciklist u paraleli s vozilom sliči na pješaka u hodu). Gledajući postotke točnosti detekcija kod bilo koje vrste prometnih znakova, može se vidjeti da se one kreću uvijek iznad 80% u slučaju YOLOv3 CNN. Prometni znakovi su standardizirani i nisu promjenjivi te je ovaj rezultat očekivan. Rezultat detekcija automobila i autobusa/kamiona je također očekivan. Automobila je najviše u prometu i dolaze u svakakvim oblicima i bojama, no posjeduju jednake karakteristike na sličnim mjestima te ih je jednostavno detektirati. Treba naglasiti da u testnom skupu ima najviše označenih automobila, odnosno da sličan postotak točnosti detekcija automobila i, primjerice semafora, prikazuje samo omjer točno detektiranih objekata i označenih objekata, ne uzimajući u obzir količinu objekata u tom omjeru.

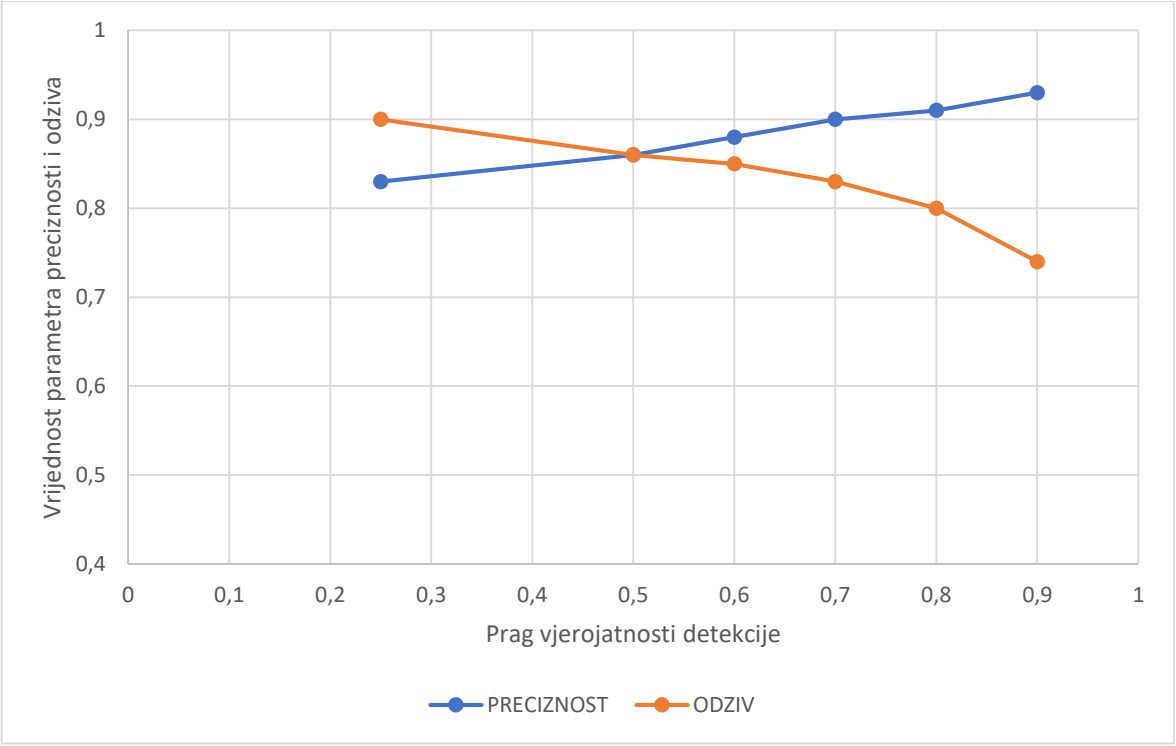
Odabran je set težina YOLOv3 CNN iz iteracije 19900 (označen plavom bojom u tablici 4.3.) iz razloga što ima puno manje lažno negativnih detekcija (FN) i puno više točno pozitivnih detekcija (TP) u odnosu na sljedeće najbolje rezultate prilikom kriterija IoU = 0,5, koji su iz iteracija 18900 i 25000 (označeni žutom bojom u tablici 4.3.). Također, prosječna preciznost detekcije (mAP) je najveća te je najveći i odziv (*recall*). Za *Tiny* YOLOv3 CNN odabran je set težina iz iteracije 30000 (označen plavom bojom u tablici 4.4.) naspram rezultata iteracije 27000 (označen žutom bojom u tablici 4.4.) iz istih razloga. Test je ponovljen više puta koristeći težine iz iteracije 19900, odnosno 30000 tako da se mijenjao prag vjerojatnosti detekcije. Rezultati dobiveni pomoću *map* naredbe za različite pragove vjerojatnosti detekcije prikazani su na grafovima 4.1. i 4.2. Usporedbom rezultata najboljeg seta težina YOLOv3 CNN i *Tiny* YOLOv3 CNN može se vidjeti da su rezultati u slučaju *Tiny* YOLOv3 CNN lošiji. To je i očekivano zbog manje dubine *Tiny* YOLOv3 CNN, odnosno korištenja 13 u odnosu na 75 konvolucijskih slojeva te korištenja samo dva sloja detekcije u odnosu na tri sloja u YOLOv3 CNN. Također, *Tiny* YOLOv3 koristi šest klastera dimenzija, dok YOLOv3 koristi devet.

Tab. 4.3. *Statistika skupova težina YOLOv3 CNN po iteracijama za različite IoU*

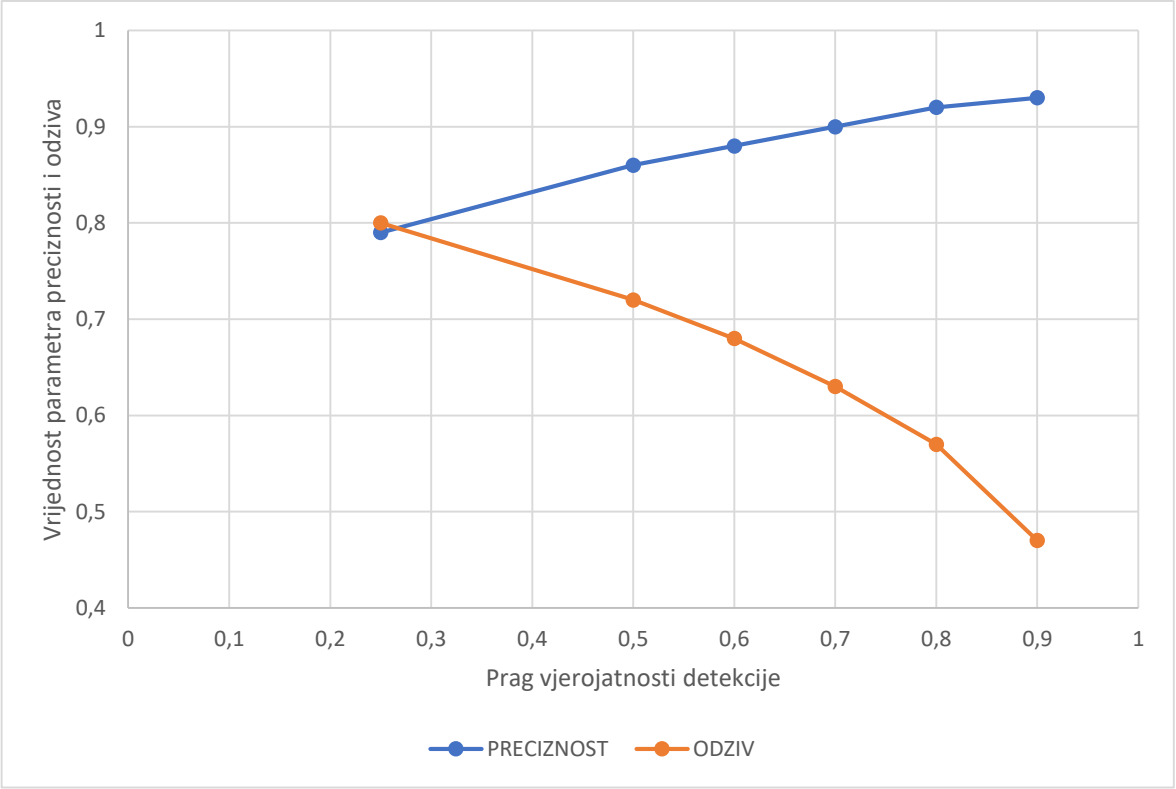
YOLOv3												
ITERACIJA	25000	**25000**	25000	20900	20900	20900	19900	**19900**	19900	18900	**18900**	18900
IoU PRAG	75%	**50%**	25%	75%	50%	25%	75%	**50%**	25%	75%	**50%**	25%
mAP	70,02%	**82,95%**	83,15%	72,28%	84,8%	85,61%	65,9%	**87,35%**	88,05%	70,14%	**84,02%**	84,21%
DETEKCIJE	5540	**5540**	5540	6175	6175	6175	8513	**8513**	8513	5999	**5999**	5999
GROUND TRUTH	4871	**4871**	4871	4871	4871	4871	4871	**4871**	4871	4871	**4871**	4871
0	68,17%	**79,29%**	76,47%	77,13%	78,11%	78,27%	77,8%	**86,29%**	86,47%	77,25%	**78,25%**	78,35%
1	62,84%	**76,72%**	77,07%	66,12%	77,31%	77,47%	57,18%	**79,69%**	81,7%	64,95%	**76,62%**	76,93%
2	37,04%	**69,37%**	70,53%	35,79%	77,53%	78,68%	37,98%	**81,75%**	85,18%	33,74%	**69,6%**	70,81%
3	58,72%	**79,35%**	79,89%	58,14%	80,44%	80,67%	45,43%	**85,79%**	86,85%	42,46%	**79,71%**	80,01%
4	89,78%	**90,91%**	90,91%	90,6%	90,85%	90,85%	89,83%	**90,72%**	90,72%	90,12%	**90,85%**	90,85%
5	79,89%	**90,3%**	90,3%	90,07%	90,55%	90,55%	79,07%	**90,56%**	90,58%	89,27%	**90,08%**	90,08%
6	80,62%	**81,36%**	81,36%	80,37%	88,93%	88,93%	69,89%	**88,84%**	88,89%	79,97%	**88,94%**	88,94%
7	79,21%	**90,21%**	90,21%	80,58%	89,98%	89,98%	87,58%	**90,12%**	90,12%	79,35%	**90,36%**	90,36%
8	81,82%	**90,85%**	90,85%	81,14%	90,85%	90,85%	74,72%	**90,91%**	90,91%	80,98%	**90,67%**	90,67%
9	80,57%	**89,91%**	89,91%	81,1%	89,91%	89,91%	67,25%	**90,49%**	90,49%	81,35%	**90,38%**	90,38%
10	51,28%	**77,17%**	77,18%	54,03%	78,35%	85,52%	38,18%	**85,72%**	86,62%	52,09%	**78,8%**	78,89%
PRAG VJEROJATNOSTI DETEKCIJE 0,25												
PRECIZNOST	0,81	**0,9**	0,91	0,8	0,89	0,9	0,68	**0,83**	0,83	0,81	**0,91**	0,91
ODZIV	0,68	**0,75**	0,76	0,73	0,81	0,82	0,74	**0,9**	0,91	0,69	**0,77**	0,78
F1	0,74	**0,82**	0,83	0,76	0,85	0,85	0,71	**0,86**	0,87	0,75	**0,83**	0,84
TP	3298	**3665**	3683	3544	3951	3970	3609	**4360**	4410	3381	**3765**	3780
FP	757	**390**	372	889	482	463	1674	**923**	873	775	**391**	376
FN	1573	**1206**	1188	1327	920	901	1262	**511**	461	1490	**1106**	1091
PROSJEČNI IoU	70,68%	**76,93%**	77,11%	69,75%	76,05%	76,22%	58,89%	**68,53%**	68,91%	70,81%	**77,18%**	77,32%
PROSJEČNO VR. DET. [ms]	58,46	**59,33**	59,33	57,14	58,46	58,46	59,33	**58,46**	57,59	57,14	**58,87**	58,87

Tab. 4.4. *Statistika skupova težina Tiny YOLOv3 CNN po iteracijama za različite IoU*

TINY YOLOv3									
ITERACIJA	30000	**30000**	30000	27000	**27000**	27000	24000	24000	24000
IoU PRAG	75%	**50%**	25%	75%	**50%**	25%	75%	50%	25%
mAP	45,55%	**81,23%**	85.00%	41,22%	**79,94%**	85,13%	35,03%	78,52%	84,3%
DETEKCIJE	14468	**14468**	14468	16939	**16939**	16939	23534	23534	23534
GROUND TRUTH	4871	**4871**	4871	4871	**4871**	4871	4871	4871	4871
0	63,4%	**83,02%**	83,48%	64,22%	**83,61%**	84,12%	43,31%	83,11%	83,69%
1	40,72%	**74,08%**	74,92%	45,55%	**72,99%**	73,63%	27,92%	69,05%	72,46%
2	10,25%	**54,19%**	70,35%	11,44%	**47,59%**	69,02%	5,91%	40,1%	60,96%
3	34,84%	**83,38%**	87,4%	33,96%	**81,16%**	86,01%	32,15%	79,19%	84,84%
4	76,71%	**90,85%**	90,85%	63,43%	**90,85%**	98,23%	74,32%	90,47%	96,94%
5	56.00%	**90.00%**	90,27%	55,12%	**89,95%**	90,27%	50,06%	89,45%	90,03%
6	55,25%	**87,48%**	88,26%	39,4%	**87,56%**	88,1%	51,01%	86,95%	87,86%
7	49,76%	**88,87%**	89,9%	50,15%	**89,03%**	89,27%	31,15%	88,68%	89,54%
8	40,4%	**89,96%**	90,73%	38,38%	**90,5%**	90,67%	40,32%	88,34%	90,23%
9	57,32%	**90,54%**	90,63%	42,63%	**90,07%**	90,21%	62,07%	89,62%	94,12%
10	16,42%	**61,11%**	78,21%	9,15%	**55,98%**	76,89%	14,39%	58,74%	76,59%
PRAG VJEROJATNOSTI DETEKCIJE 0,25									
PRECIZNOST	0,55	**0,79**	0,82	0,48	**0,75**	0,78	0,42	0,69	0,72
ODZIV	0,56	**0,8**	0,84	0,53	**0,82**	0,85	0,51	0,83	0,88
F1	0,55	**0,8**	0,83	0,5	**0,78**	0,82	0,46	0,76	0,79
TP	2711	**3917**	4070	2561	**3981**	3161	2489	4067	4267
FP	2246	**1040**	887	2764	**1344**	1164	3404	1826	1626
FN	2160	**954**	801	2310	**890**	710	2382	804	604
PROSJEČNI IoU	46,06%	**62,28%**	63,34%	40,36%	**58,04%**	59,16%	39,87%	52,83%	53,84%
PROSJEČNO VR. DET. [ms]	10,6	**10,6**	9,78	9,78	**10,6**	10,6	9,78	9,78	9,78



Graf 4.1. *Graf promjene preciznosti i odziva YOLOv3 algoritma na testnom skupu slika prilikom promjene praga vjerojatnosti detekcije*



Graf 4.2. *Graf promjene preciznosti i odziva Tiny YOLOv3 algoritma na testnom skupu slika prilikom promjene praga vjerojatnosti detekcije*

4.2. Verifikacija na skupu video sekvenci

U ovom potpoglavlju opisana je verifikacija algoritama na skupu video sekvenci. Verifikacija je izvršena na pet video sekvenci prometa iz Osijeka koje nisu korištene u procesu stvaranja baze podataka za treniranje. Video sekvence mogu se pronaći na DVD-u priloženom uz ovaj rad u mapi Prilog P.4.3. Za verifikaciju algoritma zasnovanog na YOLOv3 algoritmu za detekciju objekata korišten je skup težina iz iteracije 19900, odnosno iz iteracije 30000 za *Tiny* YOLOv3 koje su se pokazale najboljim u procesu verifikacije nad skupom slika opisanom u potpoglavlju 4.1. Svaka video sekvenca traje 10 sekundi, osim VIDEO SEKVENCE 5 koja traje 40 sekundi. Svih 11 klasa je pokriveno u tim sekvencama. U tablicama 4.5., 4.6. i 4.7. prikazani su brojevi stvarnih objekata iz svake klase u svakoj video sekvenci u stupcu označenom s TRUTH, a broj detekcija objekata pomoću vlastitih algoritma prikazan je u stupcu označenom s DET. Oznake klasa odgovaraju cjelobrojnim oznakama opisanim u potpoglavlju 3.4. Zbog načina rada Viola-Jones detektora, detekcija stražnje, odnosno prednje strane automobila i bočne strane na istom automobilu se računala kao jedna detekcija, jer se radi o jednom automobilu.

Tab. 4.5. *Detektirani objekti pomoću algoritma zasnovanog na Viola-Jones algoritmu na video sekvencama*

OZNAKA KLASE	VIDEO SEKVENCA 1		VIDEO SEKVENCA 2		VIDEO SEKVENCA 3		VIDEO SEKVENCA 4		VIDEO SEKVENCA 5	
	TRUTH	DET	TRUTH	DET	TRUTH	DET	TRUTH	DET	TRUTH	DET
0	11	12	8	5	9	5	11	5	8	5
1	0	2	0	4	0	2	2	3	2	2
2	4	3	0	8	2	7	4	6	1	13
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	0	1	0	0	1	0	2
5	2	1	0	0	1	1	3	0	1	1
6	0	0	0	0	0	1	3	0	4	2
7	1	0	0	0	0	0	0	0	8	5
8	0	1	1	3	0	2	0	2	0	2
9	1	1	0	1	0	2	0	1	1	4
10	3	0	4	1	2	0	0	0	0	0

Tab. 4.6. *Detektirani objekti pomoću algoritma zasnovanog na YOLOv3 algoritmu na video sekvencama*

OZNAKA KLASE	VIDEO SEKVENCA 1		VIDEO SEKVENCA 2		VIDEO SEKVENCA 3		VIDEO SEKVENCA 4		VIDEO SEKVENCA 5	
	TRUTH	DET	TRUTH	DET	TRUTH	DET	TRUTH	DET	TRUTH	DET
0	11	11	8	7	9	9	11	12	8	10
1	0	0	0	0	0	0	2	3	2	2
2	4	5	0	0	2	2	4	2	1	2
3	1	0	1	1	0	0	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0
5	2	3	0	0	1	1	3	4	1	1
6	0	1	0	0	0	1	3	3	4	4
7	1	1	0	0	0	0	0	0	8	8
8	0	0	1	1	0	0	0	0	0	0
9	1	1	0	0	0	0	0	0	1	1
10	3	3	4	4	2	2	0	0	0	0

Tab. 4.7. *Detektirani objekti pomoću algoritma zasnovanog na Tiny YOLOv3 algoritmu na video sekvencama*

OZNAKA KLASE	VIDEO SEKVENCA 1		VIDEO SEKVENCA 2		VIDEO SEKVENCA 3		VIDEO SEKVENCA 4		VIDEO SEKVENCA 5	
	TRUTH	DET	TRUTH	DET	TRUTH	DET	TRUTH	DET	TRUTH	DET
0	11	12	8	10	9	9	11	10	8	10
1	0	0	0	0	0	0	2	2	2	1
2	4	5	0	1	2	2	4	2	1	2
3	1	1	1	1	0	0	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0
5	2	2	0	0	1	1	3	5	1	1
6	0	1	0	0	0	0	3	3	4	4
7	1	1	0	0	0	0	0	0	8	8
8	0	0	1	1	0	0	0	0	0	0
9	1	1	0	0	0	0	0	0	1	1
10	3	2	4	3	2	2	0	0	0	0

U video sekvencama ukupno ima 100 objekata označenih kao TRUTH u 11 klasa. Detektori objekata korišteni u algoritmu zasnovanom na Viola-Jones algoritmu detektiraju 115 objekata, 40 označenih kao TP i 75 kao FP, a 60 nisu detektirali (FN). YOLOv3 CNN je detektirala 106 objekata, 96 označenih kao TP i 11 kao FP, a 4 nije detektirala (FN). *Tiny* YOLOv3 CNN je detektirala 105 objekata, 93 označenih kao TP i 12 kao FP, a 7 nije detektirala (FN). Uspoređivanjem dobivenih podataka, može se zaključiti da YOLOv3 CNN najbolje detektira objekte u video sekvencama. Koristeći formule (2-2), (2-3) i (2-5), dobiju se rezultati prikazani u tablici 4.8.

Tab. 4.8. *Statistika evaluacije algoritama na skupu video sekvenci*

PARAMETAR	**Algoritam zasnovan na V-J**	**Algoritam zasnovan na YOLOv3**	**Algoritam zasnovan na *Tiny* YOLOv3**
BROJ OBJEKATA U SEKVENCAMA	100		
BROJ DETEKTIRANIH OBJEKATA U SEKVENCAMA	115	106	105
BROJ TOČNO OZNAČENIH OBJEKATA (TP)	40	96	93
BROJ POGREŠNO OZNAČENIH OBJEKATA (FP)	75	10	12
BROJ NEDETEKTIRANIH OBJEKATA (FN)	60	4	7
PRECIZNOST	0,3478	0,9057	0,8857
ODZIV	0,4	0,96	0,93
F1 MJERA	0,3721	0,932	0,9073

Analizom detekcija u video sekvencama, može se zaključiti da mreže vrlo dobro detektiraju objekte. Detektori objekata u algoritmu zasnovanom na Viola-Jones algoritmu ponašaju se slično kao i na testnom skupu slika, detektiraju 40% objekata uz malu preciznost od 34,78%. U tablici 4.8. može se vidjeti da mreže imaju visoku preciznost i odziv. Preciznost je nešto manja zbog krive klasifikacije objekata koji su slični, najčešće prilikom detekcija objekata klase autobus/kamion gdje se objekt prvotno klasificira kao automobil dok se objekt ne približi vozilu. Također, greške prilikom klasifikacije znakova događaju se kada postoje objekti okruglog oblika sa plavom pozadinom u prostoru, npr. logo ili reklama. Primjerice, u VIDEO SEKVENCI 1 mreža detektira više objekata klase 2, 5 i 6 kojih zapravo nema u prostoru. Odziv je poželjno imati što veći, jer to znači da mreža uspješno detektira objekte koji su na sceni, što je u prometu iznimno važno. Dakako, i preciznost je važno svojstvo, jer govori koliki je udio pozitivnih detekcija zapravo objekt.

4.3. Analiza rezultata evaluacije algoritama na testnom skupu slika i video sekvenci

Vlastito rješenje zasnovano na Viola-Jones algoritmu za detekciju objekata je testirano na testnom skupu slika i video sekvenci, no nije zadovoljilo uvjet rada u stvarnom vremenu, jer je vrijeme potrebno za inferenciju više od jedne sekunde. Uz ovo, detektori jako loše detektiraju potrebne objekte, samo 40% točno detektiranih objekata od ukupnih 100 objekata. Razlog tomu je da detektori za detekciju objekata nisu otporni na rotaciju i okluziju objekata koje trebaju detektirati i stoga ne uspijevaju detektirati rotirane i djelomično zaklonjene objekte u slikama. Međutim, rješenje zasnovano na YOLOv3 algoritmu, odnosno *Tiny* YOLOv3 je postiglo prilično visoke performanse. Algoritam zasnovan na YOLOv3 uspješno detektira oko 90% objekata u testnom skupu slika uz preciznost od 83%, dok na video sekvencama uspješno detektira 96% objekata uz preciznost 90%. *Tiny* YOLOv3 detektira oko 80% objekata u testnom skupu slika uz preciznost od 79%, dok na video sekvencama uspješno detektira 93% objekata uz preciznost 88,5%. Može se zaključiti da su CNNs za algoritme zasnovane na YOLOv3 i *Tiny* YOLOv3 jako dobro istrenirane, detektirajući veliku većinu objekata od interesa u prometu. Također, algoritam zasnovan na YOLOv3 algoritmu obrađujući približno 17 FPS prilikom testiranja na video sekvencama, dok *Tiny* YOLOv3 obrađuje približno 94 FPS.

Ne postoji algoritam koji detektira sve objekte uz stopostotnu preciznost. Savršenstvo u detekciji je nemoguće postići jer postoji bezbroj različitih objekata iste klase, primjerice pješaka. Kod algoritama za detekciju objekata potrebno je napraviti kompromis između točnosti detekcije i broja slika koje algoritam može obraditi u jednoj sekundi. Ovisno o primjeni, povećava se jedno svojstvo nauštrb drugog do određene mjere. Primjerice, u slučaju autonomne vožnje vozilo mora detektirati objekt od interesa s vrlo visokom točnošću s radom u stvarnom vremenu. Stoga se mora pronaći kompromis između ta dva svojstva tako da zadovoljavaju rigorozne uvjete sigurnosti u prometu jer je ipak riječ o ljudskim životima. Zbog prethodno navedenih razloga, u ovom trenutku potpuno autonomna vozila još uvijek nisu realnost.

5. ZAKLJUČAK

Autonomna vozila svakog dana obavljaju sve više funkcija bez potrebe za reakcijom vozača, no ključno je da sve te funkcije obavljaju s vrlo visokom točnošću i preciznošću. Identifikacija objekata ispred vozila je jedna od osnovnih funkcija koje vozilo mora obavljati. U ovom diplomskom radu kreirana su tri vlastita rješenja za identifikaciju objekata ispred vozila zasnovana na Viola-Jones algoritmu i YOLOv3 te je načinjena njihova usporedba. Kreirana je vlastita baza podataka za treniranje i testiranje svakog od algoritama te su istrenirani detektori automobila, autobusa i kamiona, pješaka, biciklista, semafora te prometnih znakova u šest kategorija. Također je istrenirana CNN za klasifikaciju u istih 11 klasa. Oba algoritma izrađena su u C++ programskom jeziku u ROS radnom okruženju uz korištenje OpenCV biblioteke. Testiranjem svih triju algoritama, ispitivale su se preciznost, točnost te mogućnost rada u stvarnom vremenu. Algoritam zasnovan na Viola-Jones algoritmu nije uspio postići rad u stvarnom vremenu, jer mu za detekciju svih objekata na slici treba više od jedne sekunde (obrađuje manje od 1 FPS) uz višenitnost pri rezoluciji 640 x 480 elemenata slike. Za praćenje objekata korištena je Median Flow metoda koja je dio OpenCV biblioteke. Viola-Jones algoritam također ponekad detektira dva objekta na lokaciji jednog, dok takvih detekcija nema kod CNN zbog korištenja potiskivanja ne-maksimalne vrijednosti. CNN ima prosječno vrijeme inferencije od 58,46 milisekundi, što je približno 17 FPS-a (YOLOv3 na rezoluciji 480 x 480 elemenata slike) s detekcijom svih klasa, odnosno 10,6 milisekundi, što je približno 94 FPS-a (*Tiny* YOLOv3 na rezoluciji 800 x 800 elemenata slike) korištenjem NVIDIA GeForce 1060 6GB GPU.

Analizom svih triju algoritama, očigledno je da se radi o dva različita pristupa problemu detekcije objekata ispred vozila. Na temelju rezultata testiranja triju algoritama, može se zaključiti da je bolji pristup korištenje CNN. Ubrzana verzija, odnosno *Tiny* YOLOv3 kao takav ima veći potencijal u automobilskoj industriji jer je on nastao kao CNN za korištenje u ugradbenim sustavima gdje su računalni resursi strogo ograničeni. Međutim, prilikom testiranja YOLOv3 se pokazao boljim i preciznijim s radom u relativno stvarnom vremenu, s prosječno 17 FPS-a. Za primjenu *Tiny* YOLOv3 verzije u vozilima, potrebno je povećati preciznost detekcije objekata. To se može postići dodavanjem dodatnih konvolucijskih slojeva čime će se malo povećati vrijeme inferencije, no i dalje bi osiguravalo obradu 30 FPS-a, CNN i dalje radi u stvarnom vremenu. Dodatnim modifikacijama mreže, korištenjem jače grafičke kartice i optimizacijom moguće je dovesti YOLOv3 na veće brzine te proširivanjem baze podataka i njihovom boljom augmentacijom povećati preciznost i točnost detekcije.

LITERATURA

[1] P. Viola, M. Jones, „*Rapid Object Detection using a Boosted Cascade of Simple Features*“, Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Sjedinjene Američke Države, 2001.

[2] P. Viola, M. Jones, *„Robust Real-Time Face Detection”*, International Journal of Computer Vision, vol. 57, no. 2, 2004., str. 137-154

[3] M. Hemmati, M. Biglari-Abhari, S. Berber, S. Niar, "*HOG Feature Extractor Hardware Accelerator for Real-Time Pedestrian Detection*," 2014 17th Euromicro Conference on Digital System Design, Verona, 2014, str. 543-550.

[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, *"You Only Look Once: Unified, Real-Time Object Detection",* 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, Sjedinjene Američke Države, 2016., str. 779-788.

[5] C. Shuhan, W. Ben, L. Jindong, H. Xuelong, "*Semantic image segmentation using region-based object detector*", 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), Yangzhou, 2017, str. 505-510.

[6] H. Sperker, A. Henrich, "*Feature-based object recognition — A case study for car model detection*", 2013 11th International Workshop on Content-Based Multimedia Indexing (CBMI), Veszprem, 2013, str. 127-130.

[7] Computer Science Source, *Computer Vision – The Integral Image*, https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/, 03.09.2010., pristup ostvaren 11.06.2019.

[8] OpenCV Face Detection using Haar Cascades, https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html, pristup ostvaren 11.06.2019.

[9] A. Soto i Serrano, A. M. López Peña, “*YOLO Object Detector for Onboard Driving Images*”, završni rad, Universitat Autònoma de Barcelona, Escola d'Enginyeria, 2017.

[10] Pascal VOC baza slika, http://host.robots.ox.ac.uk/pascal/VOC/, pristup ostvaren 13.06.2019.

[11] R. Grbić, M. Vranješ, predavanje s kolegija Strojno učenje u sustavima autonomnih i umreženih vozila, „Detekcija, segmentacija i *pose estimation*“, FERIT Osijek, 2019.

[12] Medium.com, *Anchor boxes – The Key to quality object detection*, https://medium.com/@andersasac/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9, Anders Christiansen, 15.10.2018., pristup ostvaren 13.06.2019.

[13] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, „*Focal Loss for Dense Object Detection*“, IEEE Transactions on Pattern Analysis and Machine Intelligence, Sjedinjene Američke Države, 23. srpanj 2018.

[14] WiderFace baza podataka, http://shuoyang1213.me/WIDERFACE/, pristup ostvaren 13.06.2019.

[15] J. Redmon, A. Farhadi, „*YOLO9000: Better, Faster, Stronger*“, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, Sjedinjene Američke Države, 2017., str. 6517-6525.

[16] J. Redmon, A. Farhadi, „*YOLOv3: An Incremental Improvement*“, https://pjreddie.com/media/files/papers/YOLOv3.pdf, pristup ostvaren 13.06.2019.

[17] I. Dabbura, „*K-means Clustering: Algorithm, Applications, Evaluation Methods and Drawbacks“,* https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a, pristup ostvaren 12.07.2019.

[18] Cyber AI Lab, A Closer Look at YOLOv3, https://www.cyberailab.com/home/a-closer-look-at-yolov3, pristup ostvaren 13.06.2019.

[19] Darknet-19 neuronska mreža, https://pjreddie.com/darknet/imagenet/#darknet19, pristup ostvaren 13.06.2019.

[20] YOLOv3 Architecture, https://supervise.ly/explore/models/yolo-v-3-coco-1849/overview, pristup ostvaren 13.06.2019.

[21] Z.-R. Wang, Y.-L. Jia, H. Huang, and S.-M. Tang, “*Pedestrian Detection Using Boosted HOG Features*”, IEEE International Conference on Intelligent Transportation Systems, 2008.

[22] I. Laptev, “*Improving object detection with boosted histograms*”, Image and Vision Computing, 2009.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “*Object detection with discriminatively trained part-based models*”, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2010.

[24] P. Gawande, „*Traffic Sign Detection and Recognition Using OpenCV*“, International Research Journal of Engineering and Technology (IRJET), vol. 4., 2017.

[25] P. Kuang, T. Ma, F. Li, Z. Chen, „*Real-Time Pedestrian Detection Using Convolutional Neural Networks*“, International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), 2018.

[26] G. R. Valiati, D. Menotti, „*Detecting Pedestrians with YOLOv3 and Semantic Segmentation Infusion*“, Proceedings of International Conference Systems, Signals and Image Processing (IWSSIP), Osijek, Hrvatska, 2019.

[27] G. Liu, „*Real-Time Object Detection for Autonomous Driving Based on Deep Learning*“, Texas A&M University-Corpus Christi, Corpus Christi, Texas, USA, 2017.

[28] ROS.org, http://www.ros.org/about-ros/, pristup ostvaren 13.06.2019.

[29] OpenCV.org, https://opencv.org/about/, pristup ostvaren 13.06.2019.

[30] OpenCV modul verzija 3.4.x, https://github.com/opencv/opencv/tree/3.4, pristup ostvaren 13.06.2019.

[31] OpenCV_contrib modul verzija 3.4.x., https://github.com/opencv/opencv_contrib/tree/3.4, pristup ostvaren 13.06.2019.

[32] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, „*Man vs. Computer: Benchmarking machine learning algorithms for traffic sign recognition*“, Neural Networks, str. 323-333, 2012.

[33] LISA: Laboratoy for Intelligent & Safe Automobiles, http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html, pristup ostvaren 14.06.2019.

[34] KITTI Vision Benchmark Suite, Karlsruhe Institute of Technology and Toyota Technological Institute, http://www.cvlibs.net/datasets/kitti/, pristup ostvaren 14.06.2019.

[35] ImageNet baza slika, http://www.image-net.org/, pristup ostvaren 14.06.2019.

[36] OpenCV Create Samples aplikacija, https://github.com/opencv/opencv/blob/master/apps/createsamples/createsamples.cpp, pristup ostvaren 14.06.2019.

[37] OpenCV Train Cascade aplikacija, https://github.com/opencv/opencv/blob/master/apps/traincascade/traincascade.cpp, pristup ostvaren 14.06.2019.

[38] J. Ciberlin, „*Detekcija objekata ispred vozila pomoću kamere na prednjoj strani vozila*“, diplomski rad, FERIT Osijek, 2018.

[39] Alat OpenLabeling, https://github.com/Cartucho/OpenLabeling, pristup ostvaren 14.06.2019.

[40] M. Menegaz, Understanding YOLO, https://hackernoon.com/understanding-yolo-f5a74bbc7967, Hackernoon.com, pristup ostvaren 12.07.2019.

[41] GitHub repozitorij DarkNet mreže, https://github.com/AlexeyAB/darknet, pristup ostvaren 12.07.2019.

[42] N. Tijtgat, Understanding YOLOv2 training output, https://timebutt.github.io/static/understanding-yolov2-training-output/, lipanj 2017., pristup ostvaren 18.07.2019.

[43] OpenMP API za C i C++, https://www.openmp.org/wp-content/uploads/cspec20.pdf, verzija 2.0, Ožujak 2002., pristup ostvaren 17.06.2019.

[44] OpenCV dokumentacija Median Flow praćenje, https://docs.opencv.org/3.4/d7/d86/classcv_1_1TrackerMedianFlow.html, pristup ostvaren 12.07.2019.

[45] J. Ciberlin, R. Grbić, N. Teslić, M. Pilipović, „*Object detection and object tracking in front of the vehicle using front view camera*“, IEEE Zooming Innovations in Consumer Electronics Conference 2019, Novi Sad, Srbija, 2019., str. 1-6.

[46] L. Ćosić, M. Vranješ, V. Ilkić, V. Mihić, „*Time to collision estimation for vehicles coming from behind using in-vehicle camera*“, IEEE Zooming Innovations in Consumer Electronics Conference 2019, Novi Sad, Srbija, 2019.

[47] M. Bjelonic, „*YOLO ROS: Real-Time Object Detection for ROS*“, Robotic Systems Lab, https://github.com/leggedrobotics/darknet_ros, ETH Zurich, 2018.

[48] Rqt_graph alat, http://wiki.ros.org/rqt_graph, pristup ostvaren 17.06.2019.

[49] YAML.org, https://yaml.org/, pristup ostvaren 17.06.2019.

[50] Ros_canopen ROS paket, http://wiki.ros.org/ros_canopen, pristup ostvaren 17.06.2019.

SAŽETAK

U radu su opisane određene postojeće metode detekcije objekata u slikama te je dan pregled korištenja istih za primjenu u autonomnim vozilima. Jedna od najvažnijih funkcija koju autonomna vozila trebaju obavljati je pravovremena i precizna detekcija objekata i predstavlja ozbiljan problem za inženjere koji se time bave. U sklopu ovog rada izrađena su tri rješenja za identifikaciju objekata u slici dobivenoj s prednje kamere na vozilu: jedno zasnovano na Viola-Jones algoritmu i dva zasnovana na YOLOv3 i *Tiny* YOLOv3 konvolucijskoj neuronskoj mreži. Objekti koji se mogu detektirati su vozila, pješaci, biciklisti, prometni znakovi podijeljeni u više kategorija i semafori. Vlastiti algoritmi izrađeni su u C++ programskom jeziku unutar ROS radnog okruženja te je provedeno njihovo testiranje na danim skupovima slika i video sekvenci. Rezultati su pokazali da algoritam zasnovan na YOLOv3 postiže znatno bolje rezultate od onog zasnovanog na Viola-Jones algoritmu. Međutim, i on kao takav ima prostora za doradu koja bi mu povećala razinu performansi s obzirom na preciznost, točnost i brzinu inferencije.

Ključne riječi: *identifikacija objekata, klasifikacija objekata, ROS, Viola-Jones, YOLOv3*

IDENTIFYING OBJECTS IN IMAGE CAPTURED FROM THE IN-VEHICLE CAMERA USING ROS

ABSTRACT

This master's thesis describes certain existing object detection methods in images and gives an overview of their usage in autonomous vehicles. One of the most important functions autonomous vehicles must perform is timely and precise object detection, which presents a serious problem for engineers in the field. As a part of this master's thesis, three solutions were developed regarding object detection in images from the vehicle camera, i.e., one based on the Viola-Jones algorithm and the other two based on YOLOv3 and Tiny YOLOv3 convolutional neural networks. Objects that could be detected are vehicles, pedestrians, cyclists, traffic signs divided into multiple categories and traffic lights. The algorithms were made in C++ programming language using ROS framework and tested on a set of test images and video sequences. The results show that the YOLOv3 based algorithm achieves much better results than the Viola-Jones one. However, it too has space for improvements in terms of its performances regarding precision, accuracy and inference speed.

Keywords: *object identification, object classification, ROS, Viola-Jones, YOLOv3*

ŽIVOTOPIS

Borna Jelić rođen je 25. ožujka 1996. godine u Osijeku. Nakon završene III. Gimnazije u Osijeku, 2014. godine ostvaruje direktan upis na preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija, tadašnji Elektrotehnički fakultet u Osijeku. 2017. godine stječe akademski naziv sveučilišni prvostupnik (lat. *baccalaureus*) inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij automobilsko računarstvo i komunikacije. Dvostruki je dobitnik nagrade za uspješnost u studiranju, 2017. i 2019. godine. Izvrsno poznaje engleski jezik u govoru i pismu.

Potpis:

_____

PRILOZI

- P.3.1. Slike korištene za proces treniranja detektora algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata, podijeljene u mape prema nazivu detektora (priloženo na DVD-u uz rad)
- P.3.2. Slike korištene za proces treniranja i testiranja CNN algoritama zasnovanih na YOLOv3 i *Tiny* YOLOv3 algoritmima za detekciju objekata (priloženo na DVD-u uz rad)
- P.3.3. Konfiguracijska, *names* i *data* datoteke korištene u radu CNN (priloženo na DVD-u uz rad)
- P.3.4. Tekstualna datoteka sa zapisom naziva slika korištenih u procesu treniranja CNN (priloženo na DVD-u uz rad)
- P.4.1. Tekstualna datoteka sa zapisom naziva slika korištenih u procesu testiranja algoritma zasnovanog na Viola-Jones algoritmu za detekciju objekata (priloženo na DVD-u uz rad)
- P.4.2. Tekstualna datoteka sa zapisom putanja slika korištenih u procesu testiranja CNN (priloženo na DVD-u uz rad)
- P.4.3. Video sekvence korištene u testiranju izrađenih algoritama (priloženo na DVD-u uz rad)