Algoritam za autonomnu vožnju zasnovan na prepoznavanju prometnih znakova

Mijić, David

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

Sveučilišni studij

ALGORITAM ZA AUTONOMNU VOŽNJU ZASNOVAN NA PREPOZNAVANJU PROMETNIH ZNAKOVA

Diplomski rad

David Mijić

Osijek, 2019.

SADRŽAJ

1. UVOD

Kroz povijest, čovjek je uvijek težio napretku i razvijao svoju okolinu kako bi mu život bio jednostavniji. Jedan od najvažnijih i najkorisnijih izuma u cijelom čovječanstvu je automobil. Prvi automobil pokretan na motor s unutarnjim izgaranjem izumio je Karl Friedrich Benz 1886. godine [1], a to je bio sami početak razvoja automobilske industrije koja je postala jedna od najvećih grana u svjetskoj industriji.

Kako su prometne nesreće jedan od najvećih uzroka smrti u svijetu, broj smrtnih slučajeva uzrokovanih automobilskim nesrećama svakodnevno se pokušava smanjiti razvijanjem tehnologija u automobilima čiji je glavni zadatak sačuvati ljudski život. Neki od primjera su ugradnja zračnih jastuka, senzora za upozoravanje o različitim opasnostima, jača karoserija vozila i slično. Može se reći kako je čovjek u prošlosti pokušavao naći način kako preživjeti automobilsku nesreću. U današnje se pak vrijeme pokušava izbjeći automobilsku nesreću tako da koristi razne pomoćne sustave koji pomažu vozaču u procesu vožnje. Tako se razvijaju razni ADAS (engl. *advanced driver assistance systems*) sustavi – elektronički sustavi u vozilu koji pomažu vozaču pri vožnji i podižu razinu sigurnosti sudionika prometa. Sastoje se od naprednih senzora (dinamički senzori, ultrazvučni senzori, radar, lidar, video kamere, …), aktuatora i snažne računalne podrške i osnova su sustava za autonomnu vožnju. Ideja je da ADAS u potpunosti zamijeni čovjeka i da se dođe do potpunog autonomnog vozila koje ne bi zahtijevalo prisutnost vozača u vozilu.

Prema [2], 94% prometnih nesreća izazvano je ljudskom pogreškom, a većina tih pogrešaka uzrokovane su donošenjem loših odluka u prometnim situacijama i pogrešnim prepoznavanjem situacije prouzročeno različitim ometanjima (npr. zamišljenost vozača, umor, korištenje mobilnog telefona, razgovor s ostalim putnicima u vozilu). Samo 2% uzroka prometnih grešaka pripisuje se mehaničkim kvarovima i na temelju tih podataka razvoju ADAS sustava pridodaje se velika važnost. Prema radu [3] iz 2015. godine, do 2020. godine moglo bi se spriječiti gubitak 5 milijuna ljudskih života i preko 50 milijuna ozbiljnih ozljeda uvođenjem novih i inovativnijih metodologija i ulaganja u sigurnost na cesti, od regionalnih razina do svjetskih razina. Ako se te mjere ne poduzmu, prema procjenama stručnjaka [3], broj smrtnih slučajeva u prometnim nesrećama rast će za 2.4 milijuna godišnje, što će rezultirati time da će prometne nesreće biti na petom mjestu liste najčešćih uzroka smrti u svijetu. Također će se smanjiti zagušenost prometa i riješiti problem traženja parkirnih mjesta jer će autonomna vozila ostaviti putnike na određenoj lokaciji, sama

pronaći slobodno parkirno mjesto i ponovno doći po putnike u određeno vrijeme. Vlade će manje novaca trošiti na prometnu policiju jer će promet biti puno uređeniji, a čovjek će biti produktivniji jer će vrijeme koje je trošio na vožnju automobila moći kvalitetnije iskoristiti, npr. odraditi neki posao ili se opustiti čitajući ili gledajući film.

Kako bi se navedeni zahtjevi mogli ostvariti, vozilu je potrebno ugraditi umjetnu inteligenciju kako bi mogao percipirati okolinu na način sličan kao što to čini čovjek. Ovo je iznimno važno jer se vozilo susreće sa situacijama za čije rješenje ne postoji jasno definiran algoritam s točnim nizom instrukcija koje treba izvršiti, kao što je npr. algoritam za sortiranje podataka. Ti problemi se rješavaju pomoću velikog broja podataka. Npr. kako bi vozilo moglo prepoznavati automobile, potrebno mu je dati veliki broj slika na kojima se nalazi automobil iz kojih bi ono na neki način naučilo što je automobil i kako ga razlikovati od svega što nije automobil. Isti je postupak moguće ponoviti i za ostale objekte od interesa za autonomnu vožnju. Korištenje strojnog učenja (engl. *machine learning*) jedan je od načina kako riješiti ovakve probleme [4].

Strojno učenje je način korištenja algoritama da analiziraju podatke, uče na temelju tih podataka i onda odluče o nečemu ili predvide nešto. Ima veliku ulogu u autonomnoj vožnji gdje je glavni zadatak konstantno praćenje okoline i predviđanja promjena u toj okolini. Strojno učenje koristi se za detekciju, klasifikaciju i lokalizaciju objekata, predviđanje pokreta (engl. *Pose estimation*), segmentaciju slike i sl [4]. U ovom diplomskom radu razmatrat će se problem prepoznavanja i klasifikacije prometnih znakova i upravljanja modelom vozila na temelju prepoznatih znakova. Programsko rješenje potrebno je razviti u programskom jeziku C++, a za realizaciju rješenja koristi se programski okvir ROS (engl. *Robot Operating System*).

U drugom poglavlju opisan je problem prepoznavanja prometnih znakova i dan je osvrt na već gotove radove koji se bave sličnom tematikom, dok je u trećem poglavlju predstavljen vlastiti algoritam za prepoznavanje prometnih znakova i upravljanje modelom vozila. Prvo su opisani alati korišteni za razvoj vlastitog algoritma, a zatim je detaljno opisana baza podataka za trening i verifikaciju algoritma te način rada navedenog algoritma. Nakon toga, u četvrtom poglavlju predstavljeni su rezultati dobiveni testiranjem algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova. Također je opisana korištena baza podataka za testiranje prepoznavanja prometnih znakova i problemi na koje predloženi algoritam nailazi. U petom poglavlju dani su zaključci diplomskog rada.

2. PROBLEM PREPOZNAVANJA PROMETNIH ZNAKOVA U SLIKAMA

Prepoznavanje prometnih znakova jedna je od ključnih stvari koja se mora savladati kako bi autonomna vožnja bila uspješno implementirana. Prometni znakovi daju informacije vozilu kako se treba ponašati na cesti, odnosno kojom maksimalnom brzinom se može kretati, nalazi li se na glavnoj ili sporednoj cesti, mora li stati na određenom križanju ili mora skrenuti u određenom smjeru. Jedan od najvećih problema kod prepoznavanja i klasifikacije prometnih znakova je taj da su mnogi znakovi relativno slični (iste dimenzije, slične boje). Postoje znakovi koji su specifični po boji i obliku (znak ceste s prednošću prolaska, znak raskrižja s cestom s prednošću prolaska) i kao takve, algoritam za prepoznavanje prometnih znakova ih lakše prepoznaje. S druge strane, postoje znakovi koji su jako slični, npr. znak obaveznog smjera kretanja lijevo i desno, znakovi za ograničenje brzine, itd., s kojima algoritam za prepoznavanje prometnih znakova može imati problema prilikom klasifikacije. Postoje mnoga gotova rješenja za ovaj problem zasnovana na različitim pristupima, a većina njih se zasniva na konvolucijskim neuronskim mrežama (engl. *Convolutional Neural Network*, CNN) i metodi potpornih vektora (engl. *Support Vector Machine*, SVM). U sljedećim potpoglavljima opisane su najčešće mjere koje se koriste za analizu uspješnosti algoritama za detekciju objekata u slici, struktura CNN-a i SVM-a i dan je osvrt na radove koji se bave prepoznavanjem prometnih znakova koristeći navedene metode.

2.1. Mjere uspješnosti rada rješenja za detekciju objekata u slici

Kako bi se rad određenog rješenja na smislen način evaluirao, potrebno je uvesti određene kvantitativne mjere koje oslikavaju performanse rada rješenja na određenom skupu podataka. U ovom poglavlju opisane su najčešće mjere korištene pri mjerenju uspješnosti rada rješenja za detekciju objekata, kao što su [5]:

- preciznost (engl. *Precision*)
- odziv (engl. *Recall*)
- F-1 mjera (engl. *F-1 score*)
- IoU (engl. *Intersect over union,*) – omjer površine presjeka detektiranog i stvarnog graničnog okvira (engl. *Bounding box*) i sume površina detektiranog i stvarnog graničnog okvira
- prosječna preciznost (engl. *Average Precision,* AP)
- srednja prosječna preciznost (engl. *mean Average Precision,* mAP).

Prije objašnjavanja ovih mjera, potrebno je objasniti pojmove koji se odnose na detekciju objekta koji može biti:

- točno detektiran objekt (engl. *True positive*, TP) – objekt koji je trebao biti detektiran i detektiran je;
- netočno detektiran objekt tj. njegova prisutnost (engl. *False positive*, FP) – objekt koji nije trebao biti detektiran, a detektiran je;
- nedetektiran prisutni objekt (engl. *False negative*, FN) – objekt koji nije detektiran, a trebao je biti detektiran;
- točno nedetektiran objekt (engl. *True negative, TN*) – objekt koji nije trebao biti detektiran i nije detektiran;

Preciznost se definira kao omjer svih točnih detekcija i ukupnog broja detekcija koje je algoritam dao, a može sadržavati vrijednost u rasponu od 0 do 1 i računa se na sljedeći način:

$$Preciznost = \frac{TP}{TP+FP} \tag{2-1}$$

Odziv označava koliko dobro model pronalazi sve potrebne detekcije, a računa se kao omjer broja točnih detekcija i broja svih detekcija koje je algoritam trebao označiti kao točne. Vrijednost koje odziv može sadržavati također se nalaze u intervalu između 0 i 1 i računa se na sljedeći način:

$$Odziv = \frac{TP}{TP+FN} \tag{2-2}$$

Kada korisnik nije siguran treba li odabrati parametre modela ovisno o preciznosti ili odzivu, koristi se F1 mjera koja je ovisna o preciznosti i odzivu, a računa se kao [5]:

$$F1\ mjera = 2 * \frac{preciznost*odziv}{preciznost+odziv} \tag{2-3}$$

Kao i kod preciznosti i odziva, vrijednost F1 mjere također se nalazi u rasponu od 0 do 1. IoU označava omjer površine preklapanja detektiranog i stvarnog graničnog okvira (engl. *area of overlap*) i sume površina detektiranog i stvarnog graničnog okvira (engl. *area of union*), a računa se prema formuli:

$$IoU = \frac{površina\ preklapanja\ detektiranog\ i\ stvarnog\ graničnog\ okvira}{suma\ površina\ detektiranog\ i\ stvarnog\ graničnog\ okvira} \tag{2-4}$$

Vrijednost IoU-a izražava se kao postotak i može se kretati od 0 do 1. Na slici 2.1. (a) nalazi se grafički prikaz na kojem je stvarni granični okvir objekta prikazan plavom bojom, a algoritmom detektirani granični okvir prikazan je narančastom bojom. Na slici 2.1. (b) prikazano je preklapanje detektiranog graničnog okvira sa stvarnim graničnim okvirom (tamno plava boja), dok suma površina detektiranog i stvarnog okvira (unija) predstavlja sve što je obojano svijetlo plavom i tamno plavom bojom. Prema (2-4), IoU na slici 2.1 iznosi 0.35.

a) b)

Sl. 2.1. *(a) Prikaz detektiranog (narančasti pravokutnik) i stvarnog (plavi pravokutnik) graničnog okvira, (b) Prikaz područja preklapanja (presjeka) i unije detektiranog i stvarnog graničnog pravokutnika [5].*

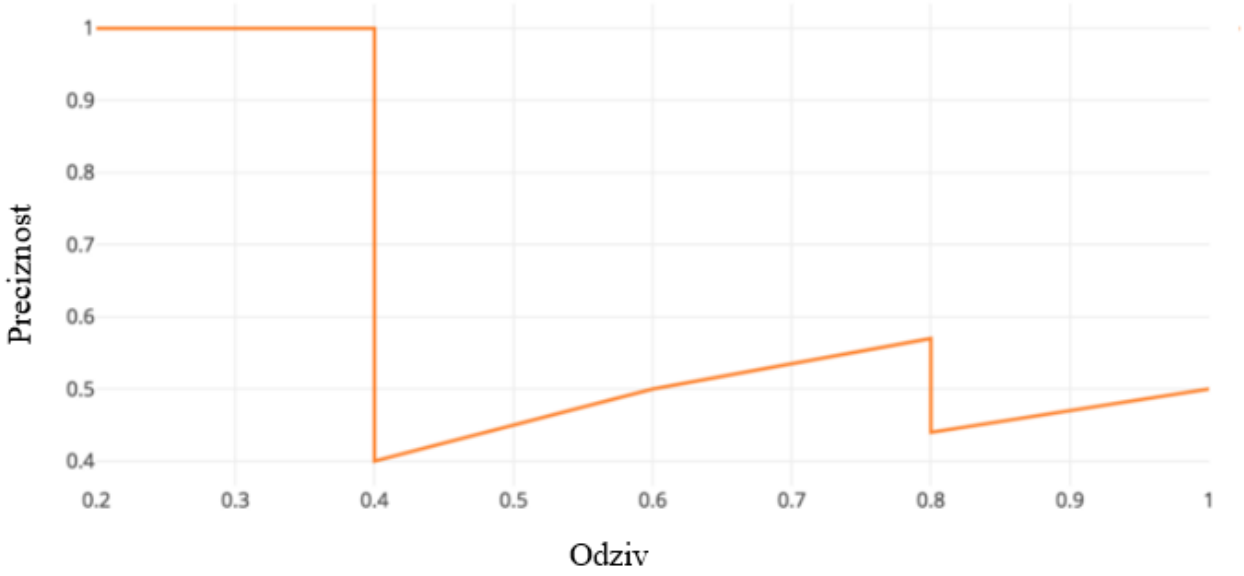
AP je površina ispod krivulje preciznosti ovisne o odzivu, a računa se prema formuli:

$$AP = \int_0^1 p(r)\, dr \tag{2-5}$$

U izrazu (2-5) *p(r)* označava krivulju preciznosti ovisnu o odzivu. Za ilustraciju ovog primjera, u tablici 2.1. nalazi se primjer s različitim vrijednostima preciznosti i odziva, a na slici 2.2. prikazana je krivulja preciznosti ovisna o odzivu prema vrijednostima iz tablice 2.1, a AP za taj primjer iznosi 0.68.

Tab. 2.1. *Različite vrijednosti preciznosti i odziva.*

Preciznost	Odziv
1	0.2
1	0.4
0.67	0.4
0.5	0.4
0.4	0.4
0.5	0.6
0.57	0.8
0.5	0.8
0.44	0.8
0.5	1



Sl. 2.2. *Prikaz krivulje preciznosti u ovisnosti o odzivu [5].*

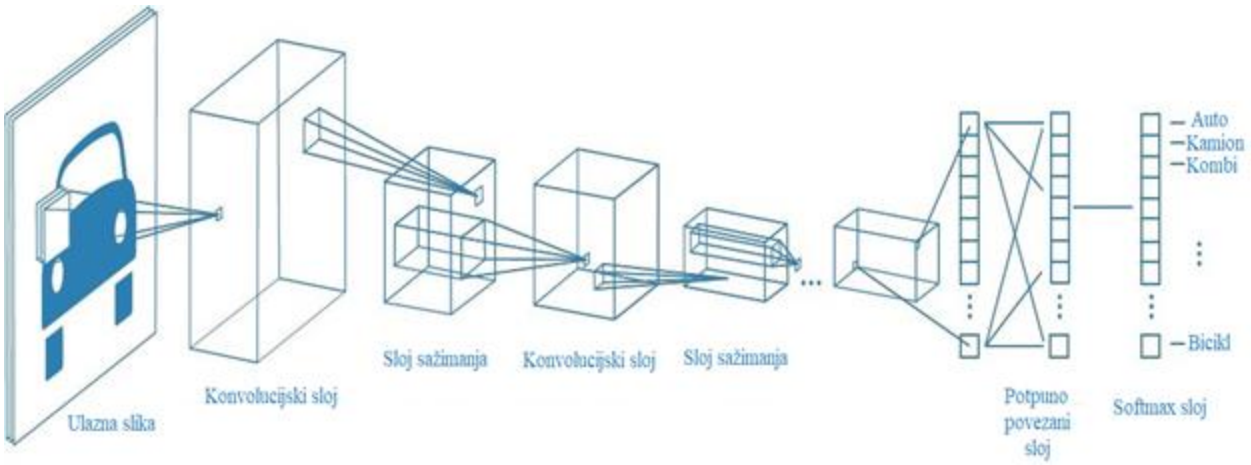
mAP računa se kao zbroj prosječnih preciznosti za svaku klasu objekta koju je potrebno razlikovati:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2-6}$$

gdje je N broj klasa. Vrijednosti AP i mAP kreću se u rasponu od 0 do 1.

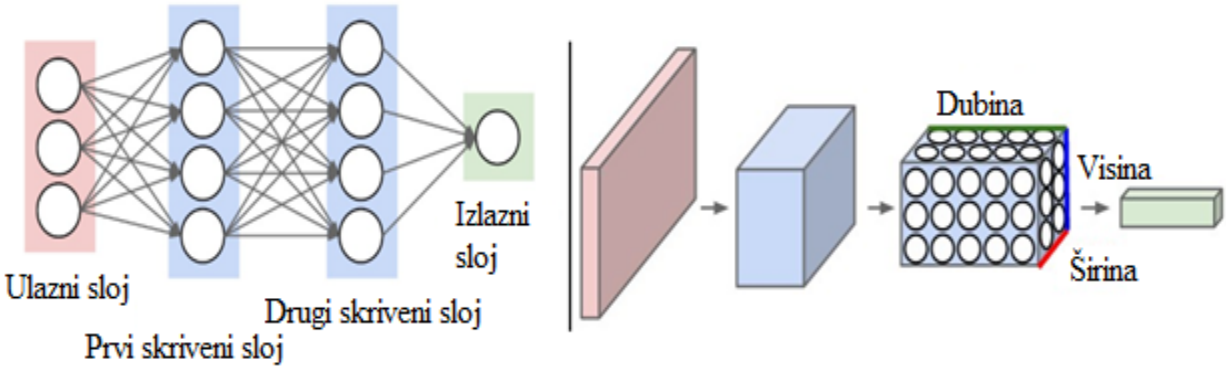
2.2. Opis konvolucijskih neuronskih mreža

CNN sastoji se od više različitih slojeva (konvolucijski slojevi, slojevi sažimanja, popuno povezani slojevi,…), gdje su neuroni posloženi u tri dimenzije. Na slici 2.3. prikazan je izgled CNN s označenim slojevima, koja kao izlaz daje klasu objekta iz prometa detektiranog u slici (npr. automobil, kamion, kombi, bicikl,…).



Sl. 2.3. *Izgled konvolucijske neuronske mreže [6].*

Prednost CNN u odnosu na klasične neuronske mreže je očuvanje prostorne strukture ulaznih podataka. Npr. u klasičnim neuronskim mrežama, sva tri vektora (npr. R, G, B) pojedine komponente boje za određenu ulaznu sliku spremili bi se u jedan zajednički vektor koji bi činio ulaz u samu mrežu. Kod CNN prostorna struktura ulazne slike (dvodimenzionalna) ostaje očuvana i svaka komponenta boje ulazne slike ostaje u zasebnom dvodimenzionalnom zapisu. Ovo se ostvaruje uvođenjem konvolucijskih slojeva. Na slici 2.4. (a) dan je izgled klasične neuronske mreže koja ulazni podatak različitih dimenzija sprema u jedan vektor, dok se na slici 2.4. (b) nalazi izgled konvolucijske neuronske mreže gdje su očuvani struktura i dubina ulaznog podatka.

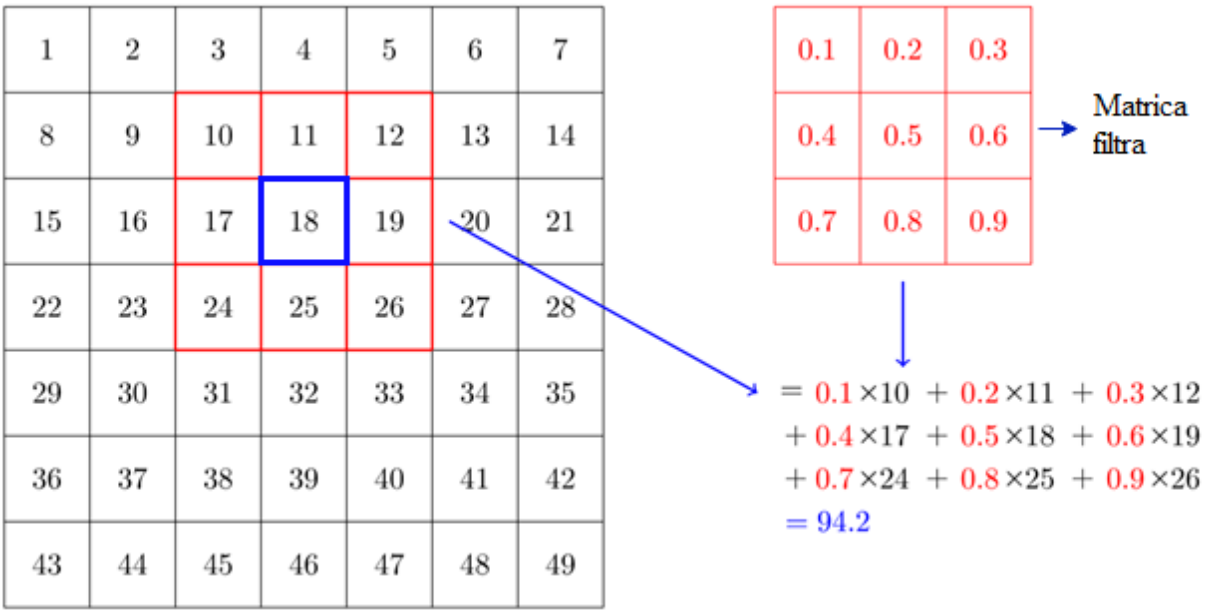


a) *b)*

Sl. 2.4. *Izgled (a) klasične neuronske mreže i (b) konvolucijske neuronske mreže [4].*

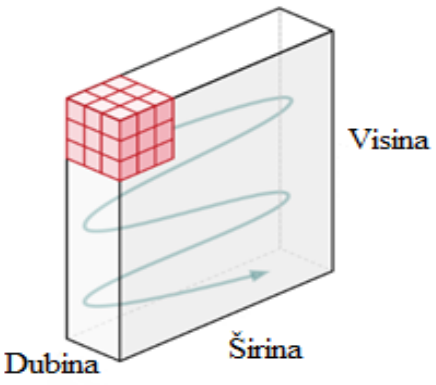
Konvolucijski slojevi sposobni su detektirati određene uzorke, odnosno značajke na slici kao što su krugovi, uglovi, horizontalni i vertikalni rubovi, itd. Te značajke detektiraju se korištenjem filtara. Filtar je matrica određenih dimenzija, npr. 3x3, 5x5, u kojoj se nalaze određene numeričke

vrijednosti. Koristeći tu matricu provodi se operacija konvolucije nad ulaznim podacima predmetnog sloja. Na slici 2.5. nalazi se primjer operacije konvolucije na ulaznom elementu s vrijednošću 18 (na slici uokviren plavom bojom) koristeći filtar 3x3 čija je matrica s iznosima dana također na slici. Rezultat provedene konvolucije na ulaznom elementu uokvirenom plavom bojom je 94.2.



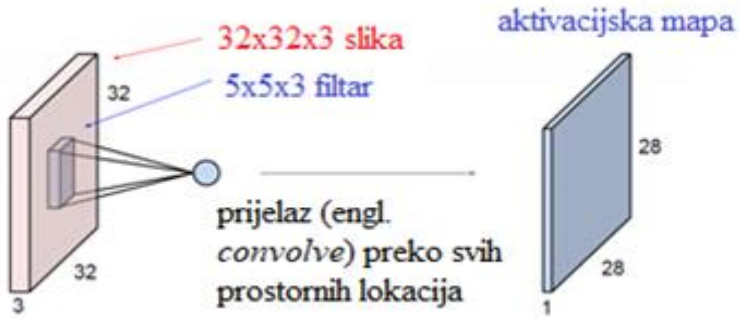
Sl. 2.5. *Primjer operacije konvolucije koristeći filtar 3x3 [7].*

Filtar također ima treću dimenziju, odnosno dubinu, a taj broj jednak je broju dubinskih razina ulaznog podatka [6]. Tako npr. za sliku zapisanu pomoću RGB modela boja filtar ima dubinu 3. Na slici 2.6. dan je grafički prikaz pomicanja konvolucijskog kernela, odnosno filtra dimenzija 3x3x3 kroz ulazni podatak tj. sliku čija je dubina 3.

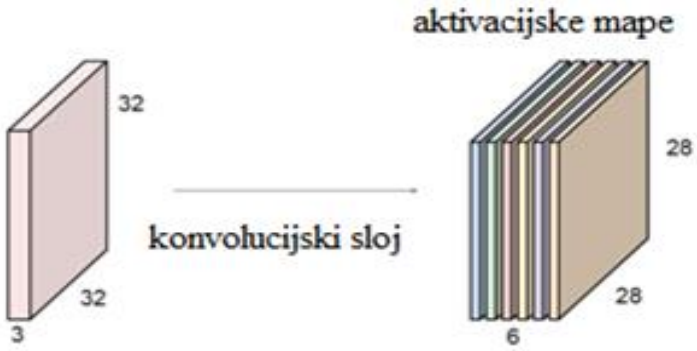


Sl. 2.6. *Prikaz pomicanja konvolucijskog kernela kroz ulazni podatak (sliku) [6].*

U prvim slojevima mreže, filtri detektiraju jednostavne značajke koje su zapravo navedeni osnovi oblici (krugovi, uglovi, horizontalni i vertikalni rubovi, itd), a što se dublje ulazi u mrežu, filtri mogu detektirati sve složenije i specifičnije oblike, kao npr. prometni znak. Jedan konvolucijski sloj može imati više filtara. Kada filtar prođe kroz sve dijelove slike, dobiva se dvodimenzionalna aktivacijska mapa koja sadrži odziv filtra u svim točkama slike. Kada se na sliku dimenzije 32x32x3 primjeni filtar dimenzija 5x5x3, a da pritom nije korištena tehnika dopunjavanja (engl. *padding*) ulazne slike, rezultat je aktivacijska mapa dimenzije 28x28x1 (slika 2.7. (a)). Primjenom više filtara, dobiva se više aktivacijskih mapa koje se slažu jedna iza druge, tj. ako na ulaznu sliku primijenimo šest filtara dimenzija 5x5x3, stvorit će se šest aktivacijskih mapa i nakon konvolucije provede s navedenih 6 filtara ulazna slika dimenzije 32x32x3 bit će reprezentirana podatkom dimenzija 28x28x6 (slika 2.7. (b)).



a)

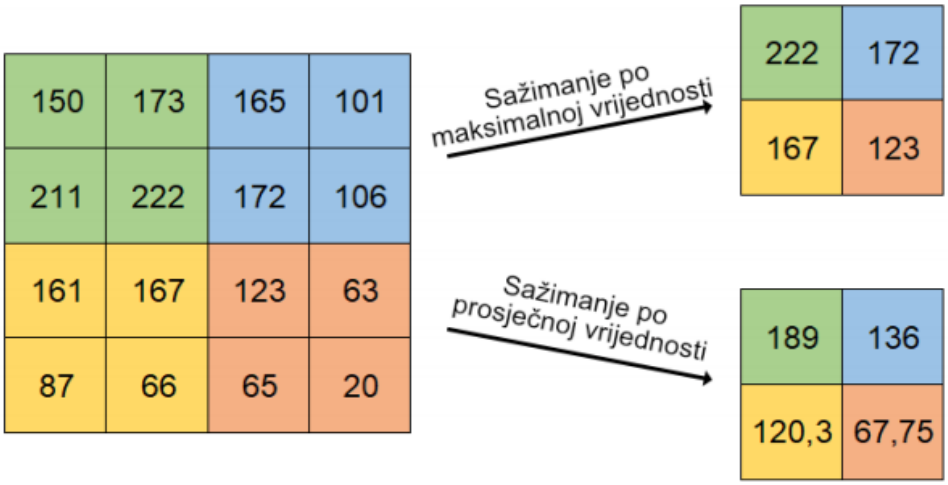


b)

Sl. 2.7. *(a) Stvaranje jedne aktivacijske mape primjenom jednog filtra dimenzija 5x5x3 (b) Stvaranje šest aktivacijskih mapa primjenom šest filtara dimenzija 5x5x3 na ulaznu sliku [4].*

Sloj sažimanja (engl. *Pooling Layer*) služi za smanjivanje dimenzija slike i može se primijeniti na svaku aktivacijsku mapu. Ovaj postupak se radi da bi se smanjili računalni zahtjevi potrebni za obradu podataka, kako bi se stekla invarijantnost na rotaciju i poziciju te kako bi se

smanjio šum [6]. Najčešći oblici sažimanja su sažimanje po najvećoj vrijednosti (engl. *Max Pooling*), gdje se uzima vrijednost najvećeg elementa podmatrice i sažimanje po srednjoj vrijednosti (engl. *Average Pooling*), gdje se uzima srednja vrijednost elemenata podmatrice. Na slici 2.8. nalaze se primjeri sažimanja po maksimalnoj i srednjoj vrijednosti.



Sl. 2.8. *Sažimanje po maksimalnoj i srednjoj vrijednosti [4].*

Zadnji sloj u konvolucijskoj neuronskoj mreži je potpuno povezani *softmax* sloj s funkcijom koja svim neuronima (klasama) u *softmax* sloju pridodjeljuje vrijednost koja označava vjerojatnost da detektirani objekt pripada toj klasi. Broj neurona u *softmax* sloju jednak je ukupnom broju klasa objekata, a neuron s najvećom vjerojatnošću (najvećim brojem u *softmax* sloju) predstavlja klasu koju je CNN detektirala kao onu koja se pojavljuje u ulaznoj slici.

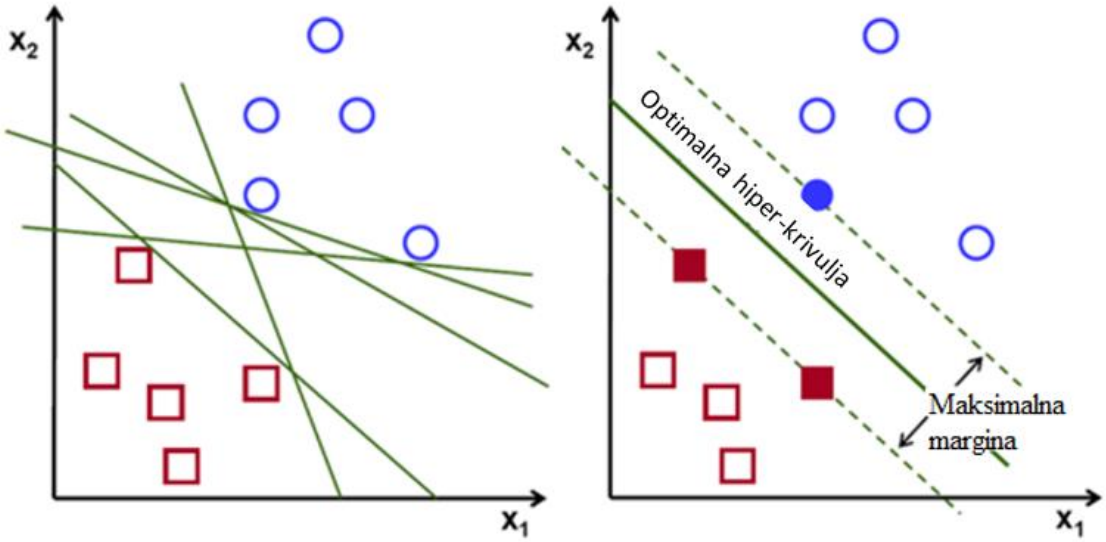
2.3. Opis metode potpornih vektora

SVM jedan je od algoritama nadziranog strojnog učenja koji pruža visoku točnost uz korištenje male količine računalnih resursa. Može se koristiti za probleme regresije, ali najčešće se koristi pri rješavanju problema binarne klasifikacije. Kada se SVM koristi za višeklasnu klasifikaciju, primjenjuje se jedna od sljedećih dviju metoda:

1. Metoda *jedan protiv ostalih* (engl. *one vs the rest*) – za *N* klasa, kreira se *N* binarnih klasifikatora. Prilikom treniranja svakog klasifikatora, slike na kojima se nalazi objekt za koji se klasifikator trenira su pozitivni primjeri, a ostale slike koje sadrže druge objekte su negativni primjeri.
2. Metoda *jedan protiv jednog* (engl. *one vs one*) – trenira se binarni klasifikator za svaki par klasa. Ako postoji *N* klasa, broj klasifikatora iznosi [8]:

$$broj\ klasifikatora = \frac{N*(N-1)}{2} \tag{2-7}$$

Glavni cilj ovog algoritma je naći hiper-krivulju u *N*-dimenzionalnom prostoru koja jasno klasificira ulazne podatke. Među podacima postoji mnogo hiper-krivulja koje uspješno klasificiraju trenutne podatke, no problem je naći optimalnu krivulju koja će najtočnije klasificirati buduće podatke. Ona se određuje tako da se izabere hiper-krivulja s najvećom marginom, odnosno najvećom udaljenošću između najbližih podataka iz obiju klase. Odabirom takve hiper-krivulje, povećava se vjerojatnost da se budući podaci klasificiraju s većom točnošću. Na slici 2.9. (a) nalazi se primjer više hiper-krivulja označenih zelenom bojom koje razdvajaju crvene kvadratiće od plavih kružnica, a na slici 2.9. (b) nalazi se optimalna hiper-krivulja s najvećom mogućom marginom.



a) *b)*

Sl. 2.9. *(a) Prikaz više mogućih hiper-krivulja i (b) odabir optimalne hiper-krivulje [9].*

Prednost korištenja metode potpornih vektora je ta da nije potrebno mnogo trening podataka za određivanje hiper-krivulje. Bitno je da trening podaci budu što različitiji, odnosno da trening slike budu različitih dimenzija, rotirane, različitog intenziteta svjetline, itd., kako bi model mogao što bolje klasificirati nove objekte. Ova metoda daje izvrsne rezultate kada je potrebna binarna klasifikacija, ali kada je potrebna višeklasna klasifikacija, izračun postaje računalno zahtjevan.

Glavna mana ovog algoritma je to što za nove objekte daje informaciju kojoj klasi pripadaju, ali ne daje informaciju kolika je vjerojatnost da taj objekt pripada toj klasi. Također nije pogodna za korištenje kada su trening skupovi jako veliki i kada trening podaci sadrže mjerni šum, odnosno kada se podaci jedne klase nalaze među podacima druge klase.

2.4. Pregled radova iz područja prepoznavanja prometnih znakova

U radu [10] opisana je metoda detekcije kineskih prometnih znakova koristeći CNN i mrežu za predviđanje područja (engl. *region proposal network*, RPN), koja dijeli konvolucijske značajke s mrežom za detekciju, što omogućava predviđena područja uz korištenje male količine računalnih resursa (korištenje memorije i zauzeće procesora). Predviđena područja uvelike poboljšavaju učinkovitost detekcije (brzinu i točnost detekcije). U usporedbi s ostalim metodama detekcije, mreža za predviđanje područja pokazuje određene prednosti u treniranju modela i točnosti detekcije. Ukupni trening set sadrži 18876 slika na kojima se nalazi 48022 prometnih znakova, od kojih se jedna četvrtina koristi za testiranje modela. Koristi se Faster R-CNN metoda kako bi se optimizirali parametri modela, a za početne parametre mreže koristi se predefinirani model na ImageNet bazi slika [11]. Trenirana su tri modela (VGG16 [12], VGG_CNN_M_1024 [13] i ZF [14]) na istoj bazi slika, gdje sva tri modela imaju mAP 0.91. ZF model ima najveću efikasnost detekcije, odnosno njegovo vrijeme detekcije iznosi 60 ms na grafičkoj kartici NVIDIA GTX980Ti sa 6 GB memorije. Za verifikaciju rezultata, model je testiran na 33 video sekvence ulazne rezolucije 640 x 460 elemenata slike. Rezultati pokazuju da metoda radi u stvarnom vremenu gdje je vrijeme detekcije za jedan okvir 51.5 ms s preciznošću iznad 99%.

Slično kao u prethodnom radu, u [15] je također je korištena CNN zajedno s metodom stohastičkog gradijentnog spusta, kako bi se poboljšala učinkovitost treninga. Znakovi su podijeljeni u tri klase: crveni znakovi zabrane (engl. *prohibitory*), žuti znakovi opasnosti (engl. *danger*) i plavi znakovi obaveznih radnji (engl. *mandatory*). Zbog kratkog vremena treniranja i malih zahtjevima za memorijom, AlexNet [16] struktura mreže koristi se za treniranje modela. AlexNet je već istrenirana mreža koja prepoznaje 1000 objekata kao što su miš, tipkovnica, bicikl, itd. U radu se koristi gotova struktura AlexNet mreže, samo su zadnja dva sloja izmijenjena prema potrebama zadatka. Točnost detekcije je 93.96%, a vrijeme detekcije je 0.2-0.3 sekunde po slici dimenzije 1000 x 750 elemenata slike.

Jedan od primjera korištenja SVM-a može se naći u radu [17] u kojem se SVM koristi zajedno s histogramom orijentiranih gradijenata (engl. *Histogram of Oriented Gradients,* HOG) i

pretragom po rešetci (engl. *grid search*). HOG služi za izdvajanje karakteristika prometnih znakova, tehnika pretrage po rešetci se koristi kako bi se optimizirali parametri SVM-a, a znakovi su prepoznati pomoću istreniranih SVM klasifikatora. Računanje HOG značajki odvija se u nekoliko koraka: prvi korak je predstavljanje ulazne slike u tonovima sive boje (engl. *gray scale*), zatim se vrši ispravljanje Gamma faktora zbog normalizacije i smanjivanje efekta sjene i utjecaja svjetlosti, a također se koristi za ispravljanje šuma. Nakon toga, računa se gradijent: komponente gradijenta u horizontalnom i vertikalnom smjeru slike mogu se dobiti korištenjem operatora za detekciju rubova (npr. Sobelov operator). Formule za računanje su sljedeće:

$$G_x(x,y) = H(x+1,y) - H(x-1,y) \tag{2-8}$$

$$G_y(x,y) = H(x,y+1) - H(x,y-1) \tag{2-9}$$

gdje je $H(x, y)$ vrijednost elementa slike (engl. *pixel*) na lokaciji (x,y), x je indeks retka, a y indeks stupca. $G_x(x,y)$ i $G_y(x,y)$ predstavljaju gradijente elementa slike u vertikalnom i horizontalnom smjeru. Prema tome, veličina gradijenta $G(x,y)$ i smjer gradijenta $\alpha(x,y)$ elementa slike (x,y) računaju se prema:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{2-10}$$

$$\alpha(x,y) = tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{2-11}$$

Nakon što se prikupi baza slika, sve slike se prebace u tonove sive boje i podijele se na dva dijela: trening skup i test skup i na svakoj slici odrede se HOG značajke. Na trening skupu također se izvrši optimizacija parametara pomoću pretraživanja po rešetki. Zadnji korak je prolazak kroz SVM klasifikator koji određuje o kojem se prometnom znaku radi. Prema dobivenim rezultatima, postotak prepoznavanja iznosi 97.52%. Također su usporedili tu metodu sa SVM metodom koja ne koristi HOG značajke, a ostali parametri su isti. Za metodu gdje su korištene HOG značajke, postotak svih točno detektiranih znakova (engl. *True positive rate*) doseže 100% kada postotak netočno detektiranih znakova (engl. *False positive rate*) iznosi približno 38%. Kod metode bez HOG značajki, postotak svih točno detektiranih znakova doseže 100% kada postotak netočno detektiranih znakova iznosi 100%. Također, vrijeme treniranja i testiranja duže je kod metode koja koristi HOG značajke, ali je broj potpornih vektora manji. Postoji još mnoštvo radova koji se bave tematikom prepoznavanja znakova u slikama, a neki od njih mogu se naći u [18 – 21].

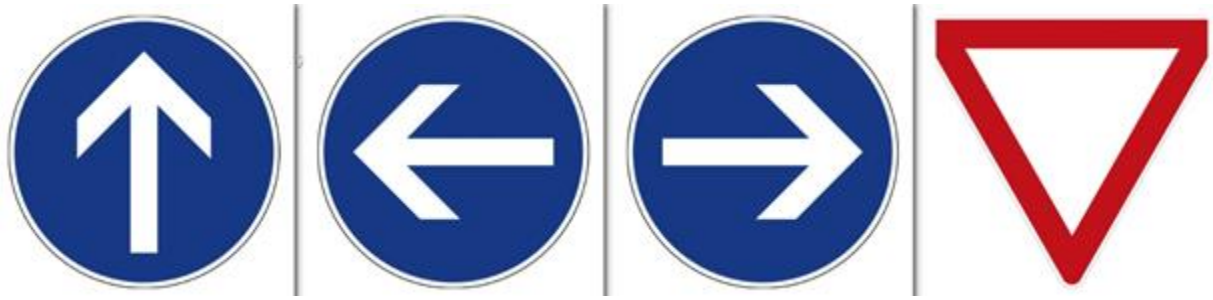
3. ALGORITAM ZA UPRAVLJANJE MODELOM VOZILA ZASNOVAN NA PREPOZNAVANJU PROMETNIH ZNAKOVA

U ovom poglavlju detaljno je opisan algoritam izrađen u sklopu ovog rada, a koji se koristi za rješavanje problema prepoznavanja prometnih znakova i upravljanje modelom vozila na temelju prepoznatih prometnih znakova. Pritom je zadatkom definirano 11 znakova koje je potrebno prepoznati kako bi vozilo izvršilo određenu radnju u simulatoru. Odabrano je 11 znakova koji se često pojavljuju u prometu i na koje bi vozilo u simulatoru trebalo reagirati kada ih detektira:

- Znak ograničenja brzine 30 km/h (slika 3.1. (a))
- Znak ograničenja brzine 50 km/h (slika 3.1. (b))
- Znak obilježenog pješačkog prijelaza (slika 3.1. (c))
- Znak zabrane prometa u jednom smjeru (slika 3.1. (d))
- Znak obaveznog smjera kretanja ravno (slika 3.1. (e))
- Znak obaveznog smjera kretanja lijevo (slika 3.1. (f))
- Znak obaveznog smjera kretanja desno (slika 3.1. (g))
- Znak raskrižja s cestom s prednošću prolaska (obrnuti trokut) (slika 3.1. (h))
- Znak zabrane prometa u oba smjera (slika 3.1. (i))
- Znak ceste s prednošću prolaska (slika 3.1. (j))
- Znak obaveznog zaustavljanja (slika 3.1. (k))

Na slici 3.1. nalaze se svi znakovi koje bi algoritam trebao prepoznati. Radi smanjenja obima posla, odabran je samo podskup svih mogućih prometnih znakova. Shodno tome, zadatak se u budućnosti može poopćiti na sve prometne znakova, međutim to bi iziskivalo puno više vremena za dizajn konačnog rješenja i to nije bilo predmet ovog diplomskog rada. U sljedećim potpoglavljima opisani su alati korišteni pri izradi samog rješenja, a zatim je opisano njihovo korištenje u razvijanju algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakova.

a) *b)* *c)* *d)*



e) *f)* *g)* *h)*

i) *j)* *k)*

Sl. 3.1. *Znakovi koje bi algoritam za upravljanje modelom vozila zasnovanom na prepoznatim prometnim znakovima trebao prepoznati (a) znak ograničenja brzine 30 km/h, (b) znak ograničenja brzine 50 km/h, (c) znak obilježenog pješačkog prijelaza, (d) znak zabrane prometa u jednom smjeru, (e) znak obaveznog smjera kretanja ravno, (f) znak obaveznog smjera kretanja lijevo, (g) znak obaveznog smjera kretanja desno, (h) znak raskrižja s cestom s prednošću prolaska, (i) znak zabrane prometa u oba smjera, (j) znak ceste s prednošću prolaska , (k) znak obaveznog zaustavljanja*

3.1. Opis alata korištenih pri izradi algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima

U ovom potpoglavlju opisani su glavni alati korišteni za funkcionalnost algoritma za upravljanje modelom vozila na temelju prepoznatih prometnih znakova. Detaljno je opisan YOLOv3 (engl. *You Only Look Once*) algoritam [22] i njegov princip rada, programski okvir ROS [23] koji služi za primanje slike s kamere iz simulatora i slanje naredbi za upravljanje modelom vozila i simulator CARLA [24] u kojemu se izvršava simulacija upravlja modelom vozila.

3.1.1. YOLO algoritam

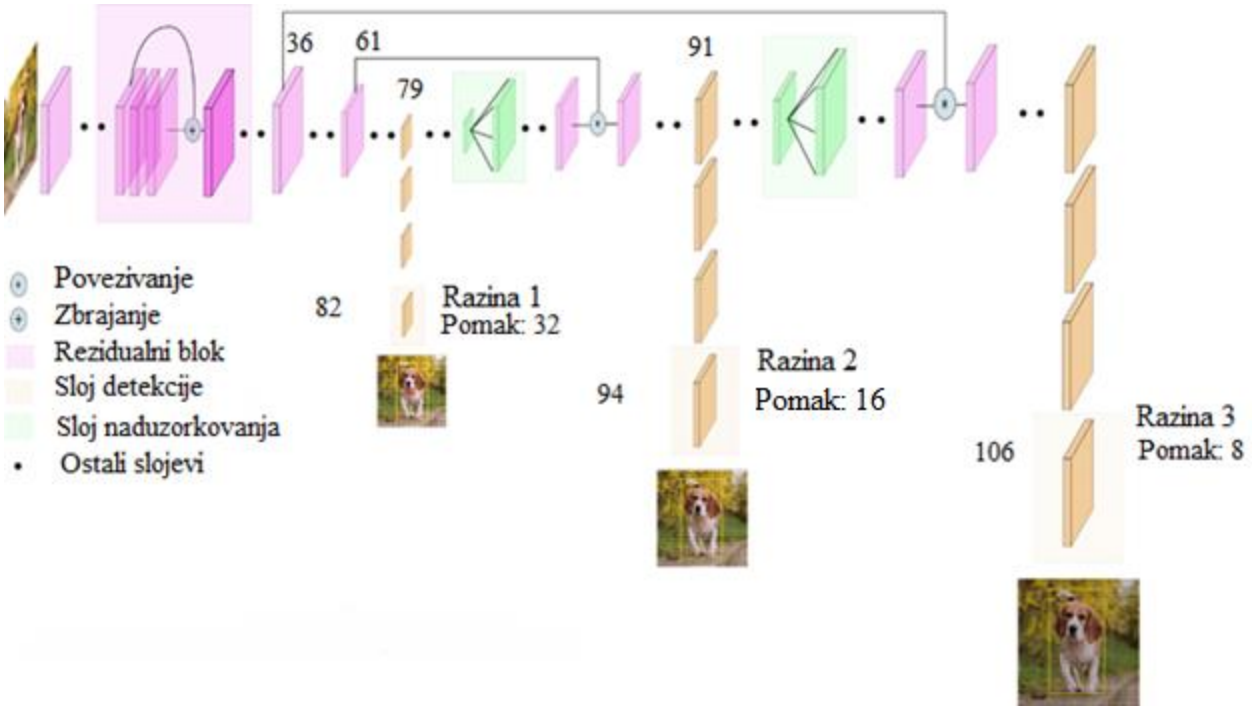
Za detekciju i klasifikaciju prometnih znakova korišten je YOLOv3 algoritam koji je trenutno jedan od najsuvremenijih algoritama u području detekcije objekata [22]. Njegova specifičnost je ta što se zasniva na regresiji – umjesto obrađivanja slike dio po dio koristeći klizajući prozor, klase i granični okviri predviđaju se za cijelu sliku u jednom koraku, odnosno jednim prolaskom kroz sliku (engl. *Single Shot Detector,* SSD) [25]. Zbog toga je YOLO algoritam jedan od najbržih algoritama za detekciju objekata, čak tisuću puta brži od R-CNN (engl. *Region CNN*) mreže i sto puta brži od Fast R-CNN mreže, ali nedostatak mu je što radi više lokalizacijskih pogrešaka.

Ulazna slika dijeli se na mrežu ćelija (engl. *Grid*) dimenzija S x S (ukupno S^2 ćelija) i ako se objekt nalazi u određenoj ćeliji mreže (engl. *Grid cell*), ta ćelija mreže zadužena je za detekciju tog objekta. Svaka ćelija mreže predviđa B graničnih okvira zajedno s razinom pouzdanosti za te okvire. Razina pouzdanosti označava koliko je model siguran da se u ćeliji mreže nalazi točno detektirani objekt. YOLOv3 predviđa granične okvire na tri različite razine, što omogućava algoritmu da detektira objekte različitih veličina (male, srednje i velike). Koristi ukupno devet predviđenih graničnih okvira (engl. *anchor box*), za svaku razinu po tri. Ti okviri se računaju pomoću algoritma K-srednjih vrijednosti koji ima devet klastera. Prva tri najveća predviđena granična okvira dodjeljuju se prvoj razini, sljedeća tri dodjeljuju se drugoj razini i zadnja tri najmanja predviđena granična okvira dodijeljena su trećoj razini. Na slici 3.2. nalazi se arhitektura YOLOv3 mreže. Sastoji se od rezidualnih blokova (engl. *Residual Block*) koji se sastoje od konvolucijskih slojeva i puteva za prečac (engl. s*hortcut path*). Bez toga, mreža bi bila klasična CNN, gdje bi se značajke učile jedna po jedna, a kako se ide sve dublje u mrežu, sve je teže učiti nove značajke. Koristeći put za prečac, slojevi unutar prečaca uče što treba nadodati na staru

značajku kako bi se stvorile nove i bolje značajke. Kao rezultat toga, složena značajka *H(x)*, koja bi se stvorila zasebno, sada je predstavljena pomoću izraza:

$$H(x) = F(x) + x \tag{3-1}$$

gdje je *F(x)* jednostavnija značajka na koju se dodaje neka stara naučena značajka *x*. Ovaj pristup omogućava mreži lakše učenje značajki, pogotovo u dubokim dijelovima mreže. Kako bi se realizirala mogućnost višeznačne klasifikacije (engl. *multilabel classification*), *softmax* sloj zamijenjen je s 1x1 konvolucijskim slojem s logističkom funkcijom. Kod korištenja *softmax* sloja, jedan izlaz pripada samo jednoj klasi, a postoje slučajevi kada jedan objekt može pripadati većem broju klasa (npr. klasa osoba i klasa žena) [26].



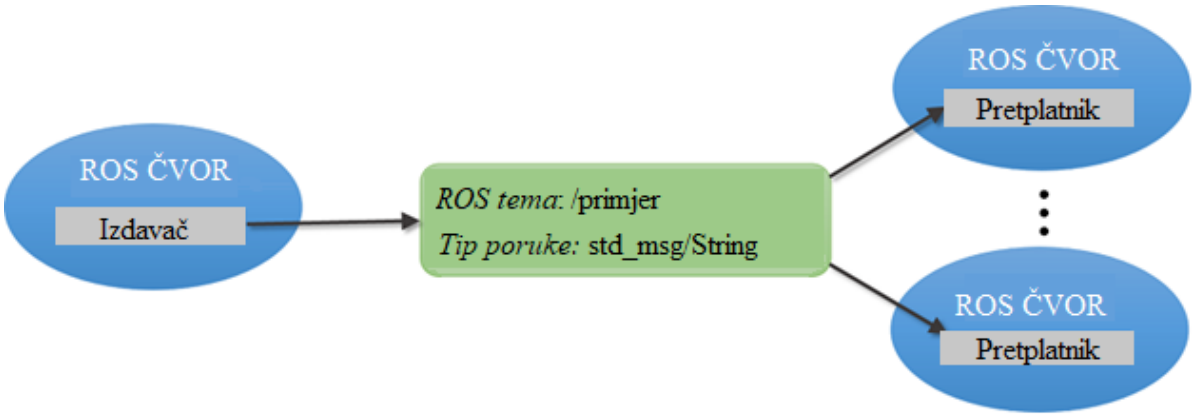
Sl. 3.2. *Arhitektura YOLOv3 algoritma [26].*

U prvoj verziji ovog rada korištena je YOLOv3 inačica algoritma, ali zbog računalnih ograničenja (premala memorija grafičke kartice), za upravljanje u simulatoru korištena je pojednostavljena inačica YOLOv3 Tiny [27], koja zahtjeva manje računalnih resursa, ali je i preciznost detekcije manja. Prilikom korištenja YOLOv3 algoritma zajedno s CARLA simulatorom, model vozila u simulatoru nije se kretao glatko, odnosno obrađivao se premali broj okvira u sekundi (engl. *frames per second,* FPS) i simulator je radio usporeno (engl. *lag*).

Korištenjem YOLOv3 Tiny verzije, također se očituje usporenost simulatora, ali u mnogo manjoj mjeri.

3.1.2. ROS programski okvir

Cijeli algoritam prilagođen je programskom okviru ROS, a to je skup programskih biblioteka otvorenog koda i alata koji služe za razvoj aplikacija u području robotike, u koje spada i autonomna vožnja. Glavni cilj ROS-a je ponovna upotreba koda u svrhu razvoja i istraživanja u ovom području. ROS je raspodijeljeni programski okvir procesa poznatiji kao čvorovi (engl. *nodes*), koji omogućavaju izvršnim datotekama da se samostalno razvijaju, testiraju i zatim da se grupiraju za vrijeme izvođenja. Jedan čvor je zadužen za određeni zadatak, npr. jedan čvor upravlja motorom, drugi računa putanju automobila, treći detektira prepreke itd. Čvorovi koriste ROS klijentsku biblioteku kako bi komunicirali s drugim čvorovima šaljući poruke preko tema (engl. *topic*). Teme su sabirnice s imenima preko kojih čvorovi šalju poruke i mogu se zamisliti kao komunikacijski kanali. Čvor može biti izdavač (engl. *publisher*) i slati poruke temi, a može biti i pretplatnik (engl. *subscriber*) kako bi se pretplatio na temu i primao poruke. Poruke su jednostavne podatkovne strukture koje sadrže osnovne tipove podataka (cjelobrojne vrijednosti, decimalne vrijednosti, logičke vrijednosti). Čvorovi također mogu komunicirati pomoću usluga (engl. *services*). Oni omogućavaju strukturu klijent/poslužitelj tako što čvor može poslati zahtjev i dobiti odgovor na taj zahtjev. Na slici 3.3. nalazi se primjer komunikacije između čvorova.



Sl. 3.3. *Komunikacija čvorova pomoću tema u ROS-u [28].*

Čvorovi se mogu grupirati u pakete (engl. *packages*) koji se mogu lagano dijeliti i distribuirati. Paketi su glavne organizacijske jedinice koda u ROS-u koji mogu sadržavati više čvorova, ROS biblioteke, konfiguracijske datoteke i sve ostalo što je korisno grupirati zajedno.

Paketi se kreiraju prema naputku da imaju dovoljno funkcionalnosti da budu korisni, ali da ne budu previše složeni kako se ne bi mogli koristiti od strane drugih programa.

ROS je dizajniran da bude što „lakši“ – kod napisan za ROS može se iskoristiti za druge programske okvire u području robotike. Shodno tome, ROS je jednostavno integrirati s ostalim programskim okvirima kao što su OpenRAVE, Orocos i Player. ROS podržava programske jezike C++, Python i Lisp, a eksperimentalne biblioteke postoje i u Javi i Lui. Trenutno se može pokretati samo na operacijskim sustavima temeljenim na Unix-u, a najčešći su Ubuntu i Mac OS X operacijski sustavi. Postoji više distribucija ROS-a, gdje svake godine izlazi nova verzija, a neke od verzija su: Kinetic, Lunar, Indigo, Hydro, Groovy i Melodic, koja je ujedno i zadnja verzija. U ovom diplomskom radu koristi se verzija Kinetic Kame, zajedno s operacijskim sustavom Ubuntu 16.04 Xenial [29].

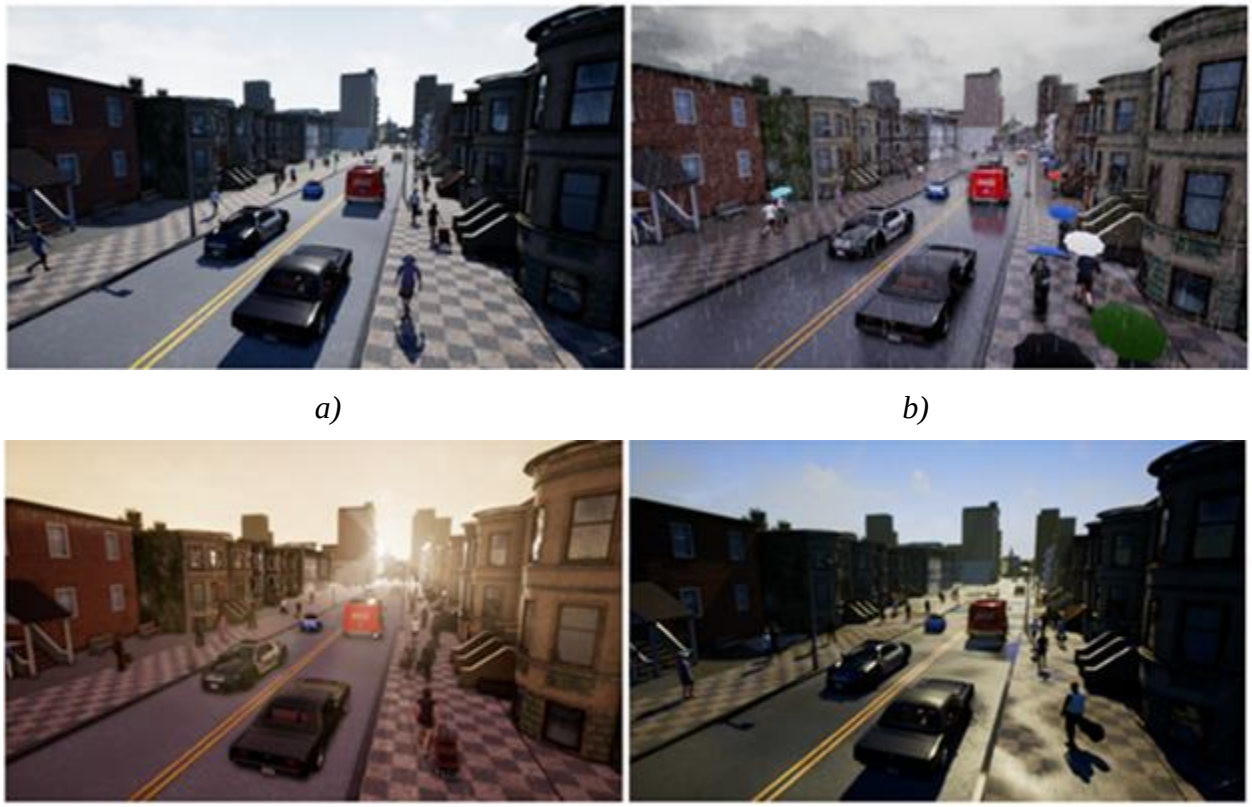
Za implementaciju YOLOv3 algoritma korišten je gotov ROS paket naziva *darknet_ros* [30]. Kako bi se koristio na vlastitom modelu, potrebno mu je samo predati istrenirane parametre modela i konfiguracijsku datoteku. YOLO algoritam može se pokrenuti na procesoru ili grafičkoj kartici, ali ako je potrebna brzina u stvarnom vremenu, mora se koristiti odgovarajuća grafička kartica. Koristeći samo procesor, YOLOv3 Tiny obrađuje približno jedan okvir po sekundi, a ako se koristi grafička kartica (Nvidia GeForce RTX 2060 6GB), obrađuje se približno 70 okvira po sekundi. Glavni čvor u paketu također se naziva *darknet_ros* i sadrži nekoliko tema:

- *camera_reading* - tema koja se pretplaćuje na izvor video sadržaja koji će konstantno slati ulazne podatke u algoritam;
- *object_detector* - šalje broj detektiranih objekata;
- *bounding_boxes* - šalje polje graničnih okvira detektiranih objekata koji daju informaciju o poziciji i veličini detektiranih objekata;
- *detection_image* - šalje sliku detektiranog objekta zajedno s graničnim okvirom.

3.1.3. CARLA simulator

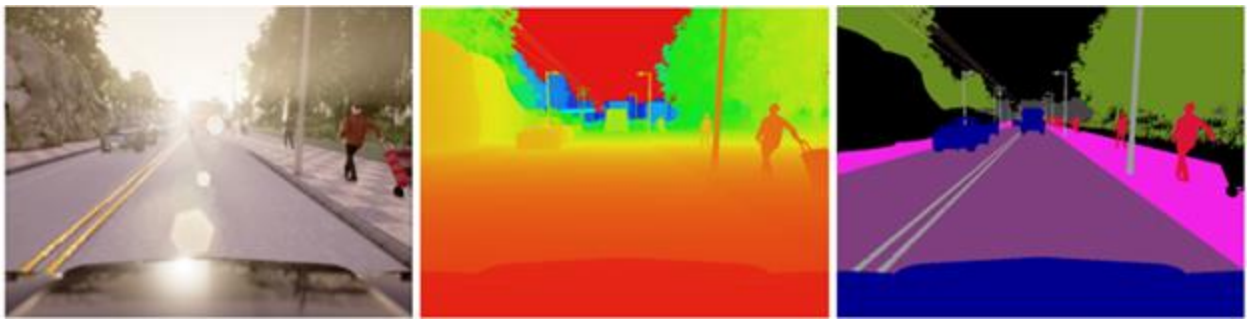
U ovom radu korišten je simulator otvorenog koda CARLA (engl. *Car Learning to Act*) [24] koji je namijenjen za razvijanje i testiranje algoritama u domeni autonomne vožnje. Simulator je dizajniran kao klijent-poslužitelj (engl. *server*) sustav gdje poslužitelj izvršava simulaciju i grafički obrađuje scenu, a korisnik šalje naredbe koje upravljaju vozilom, kao što su brzina, ubrzanje, skretanje i kočenje i za uzvrat prima rezultate očitavanja senzora. Poveznica između klijenta i

poslužitelja je skup aplikacijskih programskih sučelja (engl. *Application programming interface*, API) napisanih u programskom jeziku *Python*, koji omogućavaju korisniku kontroliranje različitih dijelova simulacije, kao što je upravljanje vozilom, reguliranje prometa, ponašanje pješaka, vremenskih uvjeta i senzora. Na slici 3.4. nalazi se prikaz različitih vremenskih uvjeta u simulatoru. CARLA pruža mnogo korisnih mogućnosti kao što su više različitih klijenata u istim ili različitim dijelovima sustava koji mogu kontrolirati različite subjekte u simulaciji. Omogućava velik broj stvarnih senzora koji se nalaze na automobilima, kao npr. lidar, kamere, senzori za procjenu udaljenosti, GPS i mnogi drugi. Koristeći Unreal Engine, može se kreirati grad u kojem će se vozilo kretati, dok u simulatoru dolaze 7 predefiniranih gradova koji se mogu po potrebi modificirati, npr. postavljanjem određenih prometnih znakova.



Sl. 3.4. *Izgled grada u CARLA simulatoru za vrijeme različitih vremenskih uvjeta: (a) sunčano, (b) kiša, (c) zalazak sunca, (d) sunčano nakon kiše [30].*

Također je moguće koristiti tri načina percepcije: prvi je RGB (engl. *Red Green Blue*) kamera gdje je prikazana točna okolina vozila (slika 3.5. (a)), drugi način je procjena dubine slike preuzete s kamere (slika 3.5. (b)) i zadnji način je semantička segmentacija koja grupira objekte istih karakteristika (slika 3.5. (c)).



a) *b)* *c)*

Sl. 3.5. *Prikaz različitih percepcija okoline u CARLA simulatoru: (a) RGB kamera, (b) procjena dubine slike, (c) semantička segmentacija [31].*

Jedna od najvažnijih stvari zbog koje se ovaj simulator koristi u sklopu ovog rada je povezanost s programskim okvirom ROS. CARLA simulator nije direktno povezan s ROS-om, ali postoji ROS paket koji je poveznica između CARLA simulatora i ROS-a, a naziva se CARLA ROS bridge. Omogućava podršku za sva tri tipa kamera, lidar i upravljanje vozilom pomoću posebnih poruka.

3.2. Opis rada vlastitog algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima

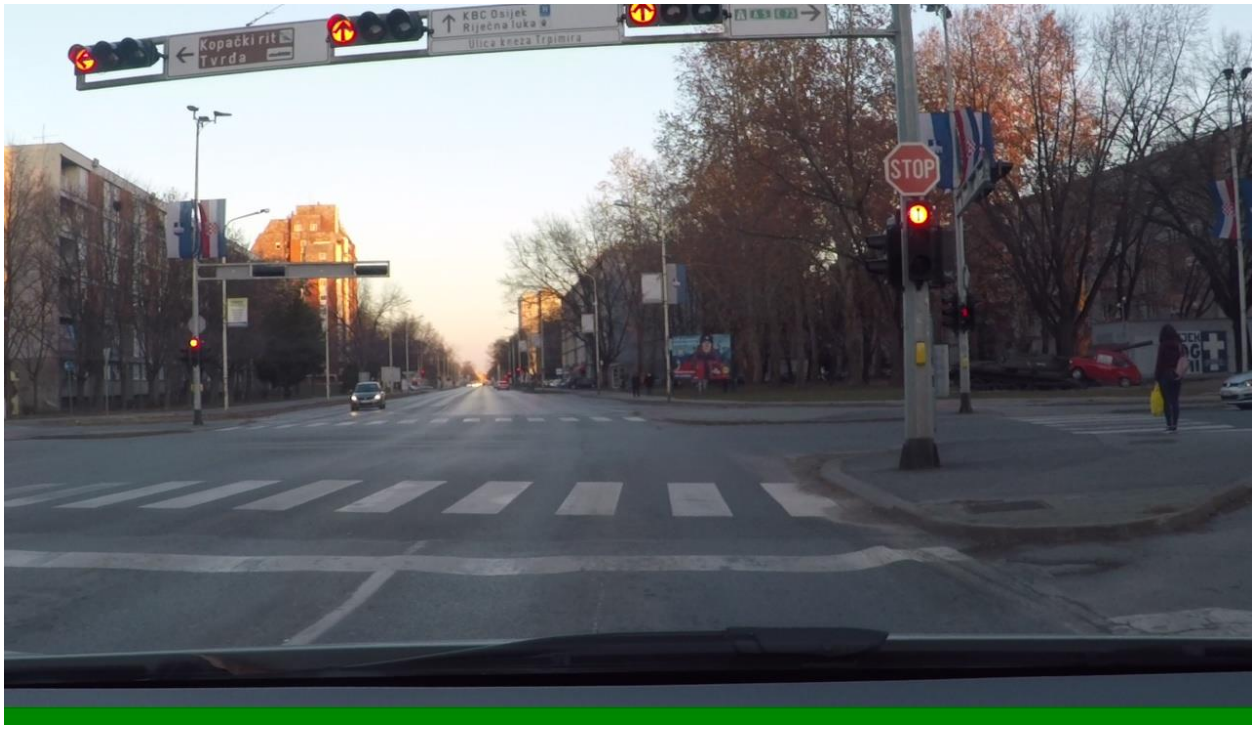
U ovom poglavlju opisano je sljedeće: baza podataka korištena za treniranje algoritma i validaciju ispravnosti njegova rada te postupak treniranja i pokretanja algoritma za upravljanje modelom vozila modelom vozila zasnovanog na prepoznatim prometnim znakovima.

3.2.1. Opis korištene baze podataka za treniranje algoritma i validaciju ispravnosti rada algoritma

Kako je ovo specifičan slučaj prepoznavanja određenih znakova, ne postoji gotov model za detektiranje i prepoznavanje upravo tih znakova. Postoje modeli koji detektiraju sve prometne znakove, ali ih dijele u samo nekoliko grupa, npr. znakove opasnosti (trokutasti oblik i crveni obrub), znakove izričitih naredbi (okrugli oblik) i znakove obavijesti (kvadratni oblik). Takva klasifikacija nije dovoljna za upravljanje modelom vozila u simulatoru i zato je potrebno svaki znak predstaviti kao zasebnu klasu pri treniranju. Kako bi YOLO algoritam uspješno detektirao znakove u prometu, znakovi u trening slikama moraju biti u realnim veličinama, odnosno znak ne smije biti preko cijele slike jer se takva situacija u prometu nikada ne događa.

Za potrebe treniranja i validacije algoritma za prepoznavanje prometnih znakova, kreirana je vlastita baza slika gdje je dio slika korišten iz postojećih baza slika kao što su KITTI baza slika

[32] i baza slika belgijskih znakova [33], ali većina slika prikupljana je vlastitim snimanjem prometnih znakova u gradu Osijeku pomoću *GoPro Hero5* kamere. U ovoj bazi podataka također su dodane slike prometnih znakova iz simulatora. Baza za testiranje i validaciju nalazi se na DVD-u priloženom uz ovaj rad u mapi P.3.1. Na slici 3.6. nalazi se primjer slike iz baze podataka snimljene u gradu Osijeku.



Sl. 3.6. *Znak obaveznog zaustavljanja na jednoj od slika baze podataka prikupljenih GoPro Hero 5 kamerom pri snimanju u vožnji u gradu Osijeku.*

Kako bi algoritam uspješno prepoznao prometne znakove u različitim situacijama, baza podataka sadrži slike za različite vremenske uvjete, kao što su sunčano vrijeme, oblačno vrijeme, kiša i noć. Kako nisu svi prometni znakovi jednako rasprostranjeni u prometu, gdje je npr. znak za obilježeni pješački prijelaz mnogo češći od znaka ograničenja brzine 30 km/h, napravljena je augmentacija nekih prikupljenih slika tako što su im promijenjene svjetlina i oštrina. Na slici 3.7. nalazi se primjer originalno snimljenog znaka ograničenja brzine na 30 km/h, na koju su primijenjene operacije zamućenja (engl. *blur*, slika 3.8.), izoštravanja (engl. *sharpen*, slika 3.9.) i povećanja svjetline (engl. *brightness increase*, slika 3.10).

Kod onih slika koje ne mijenjaju značenje kada se rotiraju (znak zabrane ulaska, prometa, znak za sporednu i glavnu cestu) napravljena je rotacija slika oko vertikalne osi kako bi se dobilo više slika prometnih znakova bez dodatnog snimanja. Npr. ako se na jednoj slici znak za sporednu

cestu nalazi na desnoj strani slike (slika 3.11. (b)), rotiranjem slike oko vertikalne osi znak za sporednu cestu nalazit će se na lijevoj strani slike (slika 3.11. (a)).

Sl. 3.7. *Originalna slika prometnog znaka ograničenja brzine 30 km.*

Sl. 3.8. *Zamućena slika znaka ograničenja brzine 30 km/h.*

Sl. 3.9. *Izoštrena slika znaka ograničenja brzine 30 km/h.*

Sl. 3.10. *Posvijetljena slika znaka ograničenja brzine 30 km/h.*

a) *b)*

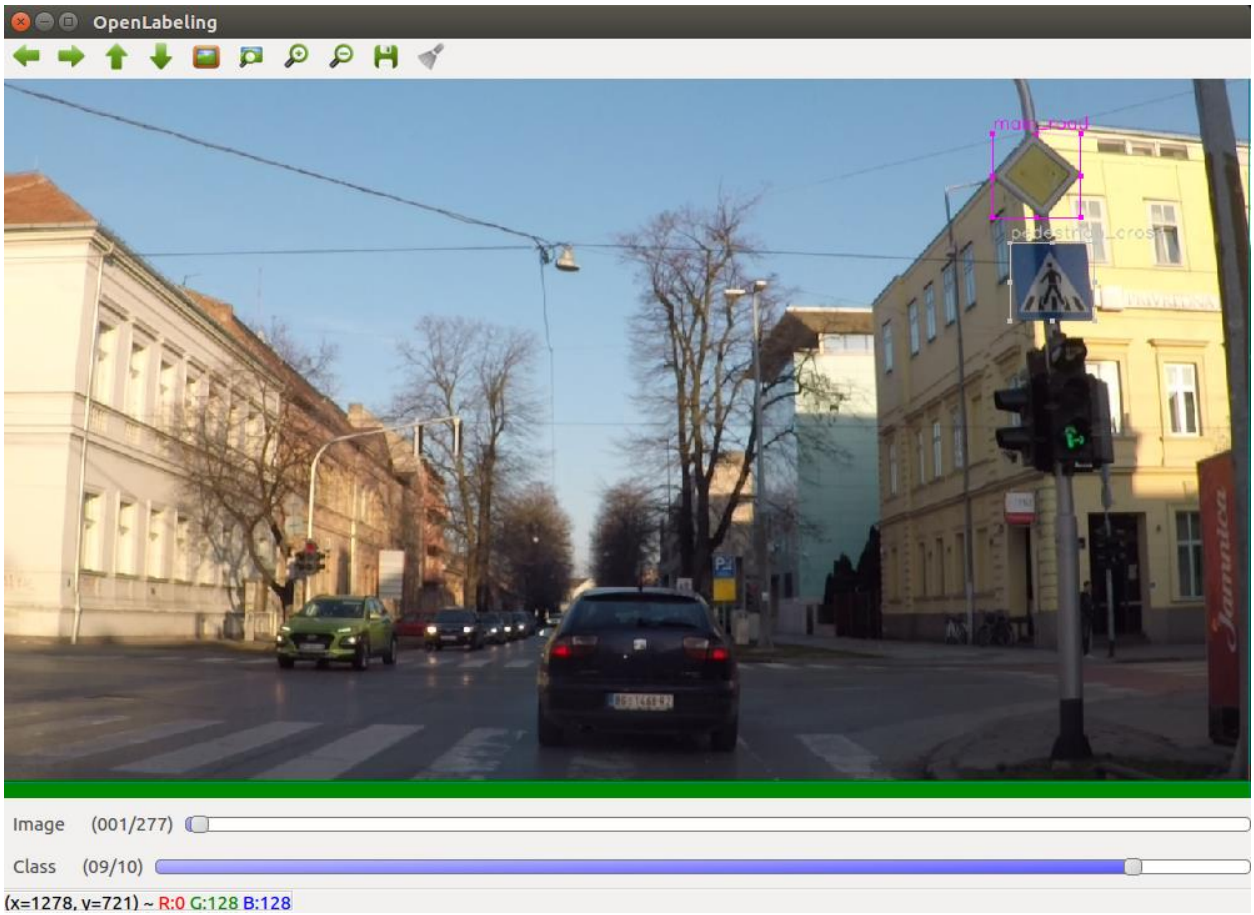
Sl. 3.11. *(b) Originalna slika znaka za sporednu cestu i (a) zarotirana slika znaka za sporednu cestu oko vertikalne osi.*

Rotacija slika također je rađena kod slika koje sadrže znakove za skretanje lijevo i desno jer se rotiranjem slike prometnog znaka za obveznog skretanja lijevo (slika 3.12. (a)) dobije znak za obvezno skretanje desno (slika 3.12. (b)).

a) *b)*

Sl. 3.12. *(a) Originalna slika znaka obaveznog skretanja lijevo i (b) rotirana slika tog znaka koji postaje znak za obavezno skretanje desno.*

Nakon što su prikupljene sve željene slike, čiji je ukupni broj 14224, na svima od njih bilo je potrebno ručno označiti određene prometne znakove kako bi se slike mogle koristiti za treniranje samog algoritma za prepoznavanje navedenih prometnih znakova. Alat za označavanje objekata, odnosno prometnih znakova na slikama naziva se *OpenLabeling* [34]. Potrebno je samo staviti trening slike u mapu *input* i pokrenuti skriptu *main.py* koja automatski generira tekstualne datoteke za svaku označenu sliku. U tim tekstualnim datotekama zapisane su informacije o označenim objektima. Prva vrijednost u tekstualnoj datoteci je broj klase objekta, zatim se upisuju koordinate središta označenog graničnog okvira, a nakon toga se zapisuju širina i visina rukom označenog graničnog okvira. Kroz slike se prolazi pritiskom na tipke *a* i *d*, a klase se biraju tipkama *w* i *s*. Na slici 3.13. nalazi se primjer označenih znakova za glavnu cestu i pješački prijelaz u alatu *OpenLabeling*.



Sl. 3.13. *Slika u alatu OpenLabeling s ručno označenim objektom od interesa (prometnim znakovima).*

3.2.2. Opis postupka treniranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova

U ovom odjeljku detaljno je opisan postupak treniranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova. U tablici 3.1 prikazan je ukupan broj označenih prometnih znakova na svih 14224 prikupljenih slika koje će se raspodijeliti na trening skup i validacijski skup.

Tab. 3.1. *Ukupan broj označenih prometnih znakova za potrebe treniranja i validacije ispravnosti rada algoritma za prepoznavanje prometnih znakova.*

Tip prometnog znaka	Broj oznaka
Znak ograničenja brzine 30 km/h	2260
Znak ograničenja brzine 50 km/h	2140
Znak obaveznog smjera kretanja desno	1908
Znak obaveznog smjera kretanja lijevo	1636
Znak obaveznog smjera kretanja ravno	1231
Znak ceste s prednošću prolaska	1169
Znak obilježenog pješačkog prijelaza	2376
Znak obaveznog zaustavljanja	867
Znak raskrižja s cestom s prednošću prolaska	2100
Znak zabrane prometa u jednom smjeru	952
Znak zabrane prometa u oba smjera	703
Ukupno	**17342**

Nakon što su označene sve slike, potrebno ih je raspodijeliti na skup za trening i skup za validaciju kako bi se dobila mjera točnosti algoritma na nekom skupu slika na kojem on nije treniran. Skup za testiranje algoritma za prepoznavanje prometnih znakova opisan je u sljedećem poglavlju. Potrebno je jednom pokrenuti jednostavnu skriptu koja dijeli bazu podataka na dva dijela u omjeru 9:1. Nasumično odabranih 90 % slika od ukupnog broja označenih slika odvajaju se u skup za treniranje, a apsolutne putanje (engl. *absolute path*) do tih slika spremaju se u tekstualnu datoteku *train.txt*. Ostalih 10 % slika odvaja se u skup za validaciju i njihove apsolutne putanje spremaju se u datoteku *validation.txt*.

Nakon kreiranja trening skupa i validacijskog skupa, potrebno je stvoriti još nekoliko datoteka koje su potrebne kako bi se započelo treniranje: datoteku *traffic-sign-yolov3-tiny.names*

u kojoj se nalaze imena klasa prometnih znakova koji se moraju detektirati, datoteku *traffic-sign-yolov3-tiny.data* koja će sadržavati informaciju o broju klasa, putanju do *train.txt*, *test.txt*, *traffic-sign-yolov3-tiny.names* datoteka i ime mape gdje će se spremati istrenirani parametri mreže pomoću kojih algoritam radi prepoznavanje prometnih znakova. Također je potrebno izmijeniti konfiguracijsku datoteku *yolov3-tiny.cfg* prema vlastitim potrebama: broj klasa potrebno je postaviti na broj klasa algoritma, broj filtera računa se prema formuli [35]:

$$broj\ filtara = (broj\ klasa + broj\ izlaza) * broj\ razina \tag{3-2}$$

gdje je broj klasa 11, broj izlaza 5 (vjerojatnost predikcije, x i y koordinate lijevog gornjeg ugla graničnog okvira, x i y koordinate donjeg desnog ugla graničnog okvira), broj razina je 3 (razine za detekciju malih, srednjih i velikih objekata) pa broj filtara prema (3-2) iznosi 48. Minimalan broj epoha treninga računa se prema sljedećoj formuli [35]:

$$minimalan\ broj\ epoha = 2000 * broj\ klasa \tag{3-3}$$

Prema (3-3), minimalan broj epoha treninga algoritma je 22000, tako da je ukupan broj epoha nakon kojeg se zaustavlja proces treninga postavljen na 24000. Tijekom treninga, nakon svake tisućite iteracije u mapu *backup* sprema se vrijednost istreniranih parametara mreže, tako da korisnik može odabrati parametre mreže koji imaju najveći mAP. Datoteke *traffic-sign-yolov3-tiny.names*, *traffic-sign-yolov3-tiny.data*, *train.txt*, *test.txt* i *yolov3-tiny.cfg* nalaze se na DVD-u priloženom uz ovaj rad u mapi P.3.2.

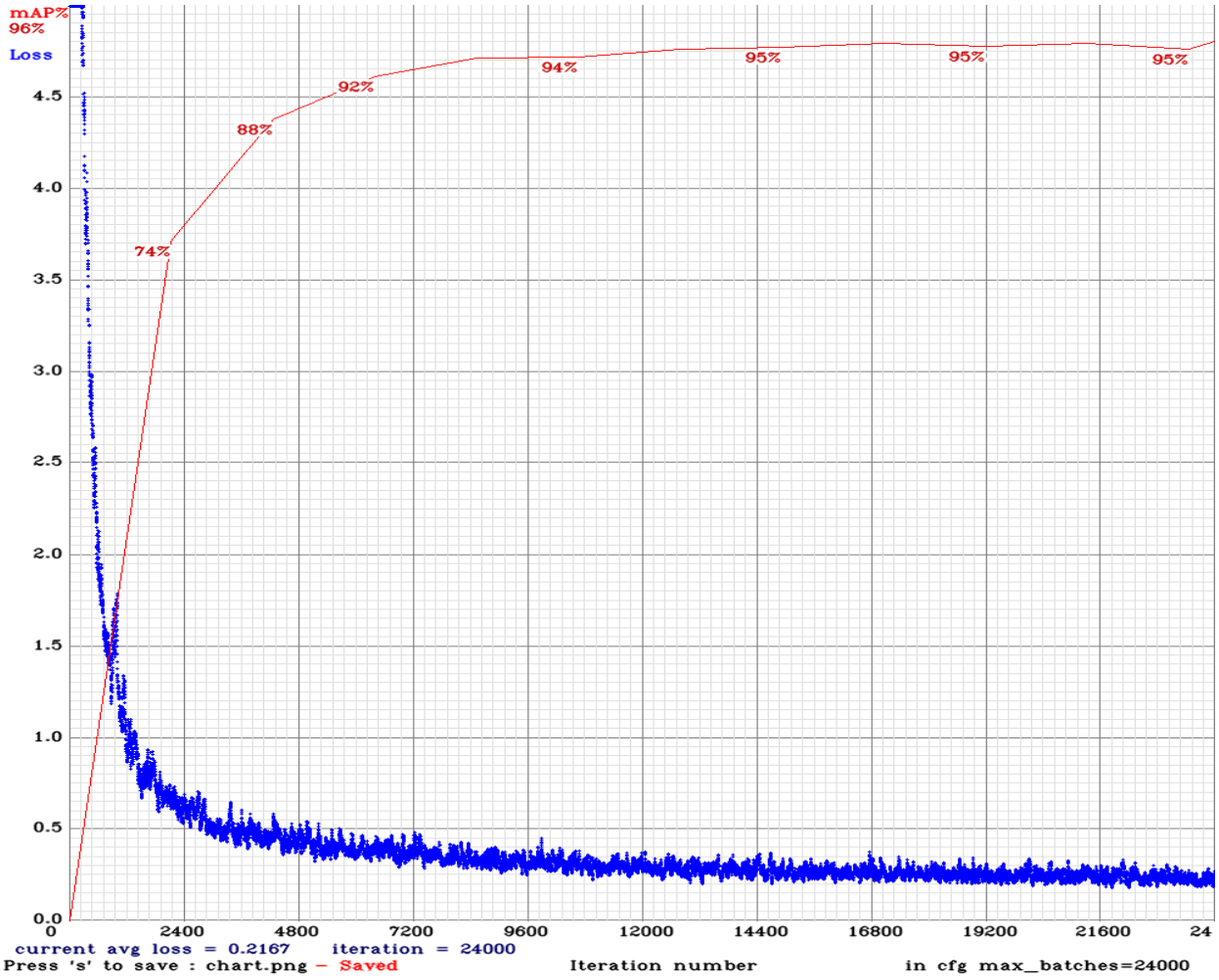
Trening se izvršava na slikama dimenzije 1024 x 768 elemenata slike, a stopa učenja ostavljena je na predefiniranoj vrijednosti 0.001 [27]. Zbog točnije detekcije, preporučuje se izračunavanje vlastitih predviđenih graničnih okvira (engl. *anchor box*). Za izračun istih, potrebno je se pozicionirati u mapu *darknet* i pokrenuti sljedeću naredbu:

./darknet detector calc_anchors cfg/ traffic-sign-yolov3-tiny.data -num_of_clusters 6 -width 608 -height 608 -show

YOLOv3 Tiny koristi šest predviđenih graničnih okvira, tako da je rezultat ove naredbe šest parova koordinata koje označavaju pozicije predviđenih graničnih okvira. Nakon ovoga, potrebno je pokrenuti naredbu za treniranje modela:

./darknet detector train cfg/ traffic-sign-yolov3-tiny.data cfg/ traffic-sign-yolov3-tiny.cfg darknet53.conv.74

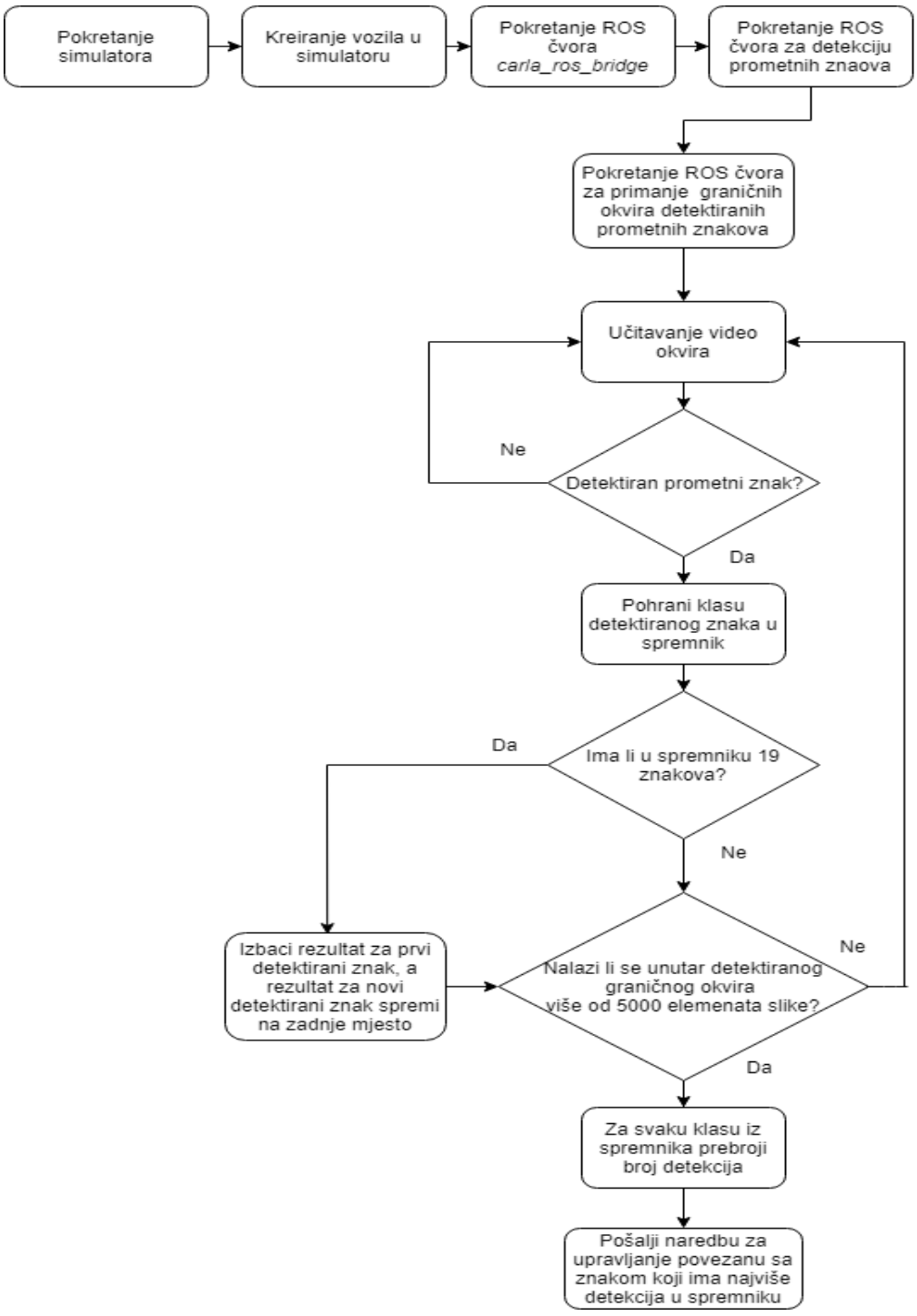
Zadnji parametar prethodne naredbe označava predefinirane parametre za treniranje modela. Kada je proces treninga završen, kreira se slikovna datoteka naziva *chart.png* koja sadrži grafički prikaz promjene iznosa kriterijske funkcije i izračun mAP za različite vrijednosti iteracije. Kriterijska funkcija služi za evaluaciju skupa parametara modela mreže i cilj je da iznos kriterijske funkcije bude što manji jer veća vrijednost kriterijske funkcije označava veću pogrešku modela [36]. Kako se parametri modela spremaju svakih 1000 iteracija, ovaj graf je koristan zato što korisnik može vidjeti kolika je mjera preciznosti algoritma i tako izabrati najbolje parametre modela. U grafu na slici 3.14. vidljivo je da maksimalni mAP iznosi 96% na validacijskom skupu znakova iz prometa. Ako se maksimalna vrijednost mAP-a dostigne nekad prije zadnje iteracije (npr. na 17000. iteraciji), odabiru se parametri modela na toj iteraciji kako bi se spriječilo preveliko usklađivanje modela s trening podacima (engl. *overfitting*). Kako se u ovom slučaju maksimalna vrijednost mAP-a doseže tek na zadnjoj iteraciji, odabrani su parametri modela dobiveni zadnjom iteracijom. Parametri modela nalaze se na DVD-u priloženom uz ovaj rad u mapi P.3.3.



Sl. 3.14. *Graf promjene kriterijske funkcije i mAP-a u odnosu na redni broj iteracije.*

3.2.3. Implementacija algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima zajedno s uputama za pokretanje

Na slici 3.15. nalazi se dijagram toka na kojemu je prikazan način rada algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima.



Sl. 3.15. *Dijagram toka algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima.*

Kada se parametri modela za algoritam istreniraju, potrebno ih je dodati u paket *darknet_ros* zajedno s konfiguracijskim datotekama, a nakon toga u datotekama za pokretanje (engl. *launch file*) *darknet_ros* čvora postaviti putanju do istreniranih parametara modela i konfiguracijskih datoteka kako bi algoritam koristio ispravne datoteke.

Prvi korak prilikom pokretanja algoritma je pokrenuti simulator i skriptu za inicijaliziranje vozila kojim će se upravljati u simulaciji. Potrebno je pozicionirati se u mapu */carla/Dist/0.9.4-21-g3d2e086/LinuxNoEditor* gdje se nalaze izvezene (engl. *exported*) datoteke potrebne za pokretanje simulatora i ostalih API-a. Za pokretanje simulatora koristi se naredba:

./CarlaUE4.sh /Game/Carla/Maps/Town02

Inicijalizacija vozila u simulatoru izvršava se pokretanjem naredbe:

python maunul_control.py

nakon čega se kreira vozilo naziva *hero*, kojim se upravlja pritiskom na tipke *w*, *s*, *a*, *d*. Izgled sučelja za upravljanje vozilom nalazi se na slici 3.16.

Sl. 3.16. *Sučelje za upravljanje vozilom u CARLA simulatoru* [24].

Na slici 3.16 s lijeve strane nalaze se informacije o vozilu i simulaciji, kao što su broj okvira u sekundi, tip vozila, ime grada, brzina kojom se automobil kreće, razina gasa i kočnice, kut zakretaja volana, informacija o sudaru (za slučaj da se sudar dogodi), broj vozila na mapi i slično. Kako bi se omogućilo detektiranje navedenih prometnih znakova s kamere modela vozila iz simulatora, potrebno je povezati temu koja šalje sliku s kamere s temom YOLO algoritma u ROS-u koja prima sliku kamere. Kako CARLA simulator nije direktno povezan s ROS-om, potrebno je pokrenuti *carla_ros_bridge* naredbom:

roslaunch carla_ros_bridge carla_ros_bridge_launch

Pokretanjem *carla_ros_bridge*-a, stvaraju se ROS teme koje dohvaćaju razne vrijednosti iz simulacije, kao što su teme za dobivanje slike sa sve tri različite kamere, podaci prikupljeni preko lidara, teme za slanje naredbi kojim se upravlja vozilom i sl. Na slici 3.17. nalazi se prikaz svih ROS tema.

```
david@dmijicPC:~/carla/Dist/0.9.4-21-g3d2e086/LinuxNoEditor$ rostopic list
/carla/ego_vehicle/camera/depth/front/image_depth
/carla/ego_vehicle/camera/rgb/front/image_color
/carla/ego_vehicle/camera/semantic_segmentation/front/image_segmentation
/carla/ego_vehicle/lidar/front/point_cloud
/carla/hero/ackermann_cmd
/carla/hero/camera/rgb/front/camera_info
/carla/hero/camera/rgb/front/image_color
/carla/hero/enable_autopilot
/carla/hero/gnss/front/fix
/carla/hero/objects
/carla/hero/odometry
/carla/hero/vehicle_control_cmd
/carla/hero/vehicle_info
/carla/hero/vehicle_status
/carla/map
/carla/objects
/carla/vehicle_marker
/carla/vehicle_marker_array
/clicked_point
/clock
/darknet_ros/bounding_boxes
/darknet_ros/check_for_objects/cancel
/darknet_ros/check_for_objects/feedback
/darknet_ros/check_for_objects/goal
/darknet_ros/check_for_objects/result
/darknet_ros/check_for_objects/status
/darknet_ros/detection_image
/darknet_ros/found_object
/initialpose
/move_base_simple/goal
/rosout
/rosout_agg
/tf
/tf_static
```

Sl. 3.17. *Popis svih dostupnih ROS tema u cjelokupnom rješenju kreiranom u sklopu ovog rada.*

Jedna od njih je tema */carla/hero/camera/rgb* koja šalje sliku s kamere iz simulatora. U datoteci *darknet_ros/config/ros.yaml* potrebno je postaviti da se ROS tema *camera_reading*

pretplati na temu */carla/hero/camera/rgb*, kako bi se omogućila detekcija prometnih znakova na slici kamere iz simulatora. Naredbom:

roslaunch darknet_ros yolo_v3.launch

pokreće se čvor za detekciju navedenih prometnih znakova. Pojavljuje se novi prozor u kojemu se nalazi slika s kamere identična kao u prozoru za upravljanje vozilom i na njoj se iscrtavaju granični okviri ako algoritam detektira prometni znak. Algoritam uspijeva obrađivati oko 40 FPS-a što ga čini primjenjivim u stvarnom vremenu. Na slici 3.18. nalaze se primjeri točnih detekcija prometnih znakova.

a) *b)*

c) *d)*

Sl. 3.18. *Točna detekcija (a) znaka obaveznog zaustavljanja, (b) znaka ceste s prednošću prolaska, (c) znaka ograničenja brzine 30 km/h i (d) zakrenutog znaka ograničenja brzine 30 km/h .*

Na slici 3.18. (d), prikazana je detekcija znaka ograničenja brzine 30 km/h iako je znak zarotiran u stranu, a to dokazuje da je algoritam invarijantan na realne rotacije znakova. Realna

rotacija znači rotacija koja je učestala u prometu, npr. blago zakrenut znak, a ne rotacija znaka za 180 stupnjeva. Vozilo mora doći na udaljenost na kojoj je sigurno da je znak ispravno detektiran i da naredba za upravljanje odgovara prometnom znaku i situaciji u prometu. Ne smije se dogoditi da algoritam prerano detektira prometni znak, npr. znak za obavezno skretanje i pošalje naredbu za skretanje, a vozilo još nije došlo do raskrižja na kojem treba skrenuti. Kako su visina prometnog znaka i udaljenost prometnog znaka od ceste u simulatoru uvijek isti, određena je minimalna veličina detektiranog graničnog okvira nakon koje je potrebno poslati naredbu za upravljanje, a ona iznosi 5000 elemenata slike (engl. *pixel*) unutar detektiranog graničnog okvira. Čvor za primanje graničnih okvira detektiranih objekata naziva se *BoundingBox* i pokreće se naredbom:

rosrun darknet_ros BoundingBox

Navedeni čvor je pretplatnik i izdavač jer se pretplaćuje na ROS temu */darknet_ros/bounding_boxes* koja šalje polje detektiranih objekata, a zatim izdaje naredbe za upravljanje modelom vozila koristeći ROS temu */carla/hero/vehicle_control_cmd* tako što šalje razinu signala za dodavanje gasa, kočnice i zakret volana u rasponu od 0 do 1. Na slici 3.19. nalazi se grafički prikaz komunikacije svih čvorova.

Sl. 3.19. *Graf ROS čvorova koji međusobno komuniciraju.*

Programski kod čvora *BoundingBox* napisan je u programskom jeziku C++ i sastoji se od klase naziva *CarControl* koja ima javne funkcije *CarControl*, koja je ujedno i konstruktor, funkciju koja se stalno poziva (engl. *callback function*) *msgCallback* i funkciju *carControlCommand* koja služi za slanje naredbi za upravljanje modelom vozila. Također ima privatne varijable *ros::NodeHandle n_* koja služi za upravljanje ROS čvorom, izdavačku varijablu *ros::Publisher regularPub_* koja služi za izdavanje poruka kojima se upravlja modelom vozila i pretplatničku varijablu *ros::Subscriber sub_* koja se pretplaćuje na ROS temu */darknet_ros/bounding_boxes* kako bi primila granične okvire svih detektiranih objekata. Konstruktor inicijalizira izdavačke i pretplatničke varijable, a funkcija *msgCallback* kao parametar

prima varijablu naziva *msg* i tipa *darknet_ros_msgs::BoundingBoxes* koja sadrži polje graničnih okvira svih detektiranih objekata. Vrijednosti veličine graničnog okvira koje funkcija prima su *xmin*, *ymin*, *xmax*, *ymax* i njih je potrebno pretvoriti u format *x, y, širina, visina*, gdje su *x* i *y* koordinate lijevog gornjeg ugla graničnog okvira, a *širina* i *visina* su vrijednosti širine i visine graničnog okvira, a pretvaranje vrijednosti izvršava se sljedećim izrazima:

$$x = xmin \tag{3-4}$$

$$y = ymin \tag{3-5}$$

$$Širina = xmax - xmin \tag{3-6}$$

$$visina = ymax - ymin \tag{3-7}$$

Programski kod dijela funkcije *msgCallback* u kojem se primaju, parsiraju i ispisuju vrijednosti dobivenih graničnih okvira nalazi se na slici 3.20. U linijama 22 i 23, inicijalizirane su varijable *width* i *height* koje označavaju širinu i visinu graničnog okvira, a njihov izračun dobije se prema (3-3) i (3-4). Površina graničnog okvira sprema se u varijablu *area* i jednaka je umnošku širine i visine graničnog okvira. Nakon toga, u upravljačkom prozoru ispisuju se zaglavlja poruke (linije 25 i 26), mjera sigurnosti s kojom je objekt detektiran (linija 27), klasa detektiranog objekta, odnosno prometni znak koji je detektiran (linija 28), podaci o graničnom okviru u formatu *xmin, ymin, xmax, ymax* (linije 28-32), *x* i *y* koordinate gornjeg lijevog ugla graničnog okvira (linije 34 i 35), širina i visina graničnog okvira (linije 36 i 37) i površina graničnog okvira (linija 38).

```
20      void msgCallback(const darknet_ros_msgs::BoundingBoxes::ConstPtr& msg)
21      {
22          int width = msg->bounding_boxes[0].xmax - msg->bounding_boxes[0].xmin;
23          int height = msg->bounding_boxes[0].ymax - msg->bounding_boxes[0].ymin;
24          int area = width * height;
25          cout << "Bouding Boxes (header):" << msg->header <<endl;
26          cout << "Bouding Boxes (image_header):" << msg->image_header <<endl;
27          cout << "Probability: " << msg->bounding_boxes[0].probability << endl;
28          cout << "Bouding Boxes (Class):" << msg->bounding_boxes[0].Class << endl;
29          cout << "Bouding Boxes (xmin):" << msg->bounding_boxes[0].xmin << endl;
30          cout << "Bouding Boxes (xmax):" << msg->bounding_boxes[0].xmax << endl;
31          cout << "Bouding Boxes (ymin):" << msg->bounding_boxes[0].ymin << endl;
32          cout << "Bouding Boxes (ymax):" << msg->bounding_boxes[0].ymax << endl;
33          cout << endl;
34          cout << "X coordinate: "<< msg->bounding_boxes[0].xmin <<endl;
35          cout << "Y coordinate: "<< msg->bounding_boxes[0].ymin <<endl;
36          cout << "Width: "<< width <<endl;
37          cout << "Height: "<< height <<endl;
38          cout << "Area of Bounding Box: "<< area <<endl;
```

Sl. 3.20. *Programski kod funkcije msgCallback koja služi za primanje detektiranih graničnih okvira.*

Kao dodatni uvjet provjere točnosti detektiranog prometnog znaka, u programskom kodu implementiran je spremnik u koji se pohranjuje 19 zadnjih detekcija objekta. Maksimalna veličina spremnika mora biti neparan broj zato što algoritam donosi odluku o slanju naredbe za upravljanje na temelju znaka koji većinski prevladava u spremniku. Ako je maksimalna veličina spremnika paran broj, npr. 20, moguće je da se u spremniku nalazi 10 detekcija jednog prometnog znaka i 10 detekcija drugog prometnog znaka i algoritam nije siguran koju naredbu za upravljanje treba poslati. Kako bi se taj problem riješio, maksimalna veličina spremnika je neparan broj, čime se može osigurati većinsko prevladavanje jednog prometnog znaka u spremniku u slučaju dvije različite detekcije. Kao početna maksimalna veličina spremnika izabran je broj 25, ali pri većim brzinama kretanja, algoritam detektira prometni znak u premalo okvira, odnosno broj detektiranih prometnih znakova u spremniku manji je od polovine spremnika (u ovom slučaju 13) i algoritam ne može donijeti ispravnu odluku o slanju naredbe za upravljanje modelom vozila. Zbog toga je maksimalna veličina spremnika postepeno smanjivana i empirijskim putem donesen je zaključak da je broj 19 kao maksimalan broj znakova u spremniku najbolji odabir za ovo rješenje.

Kada je spremnik pun, izbacuje se najstariji element spremnika i dodaje se novi. Spremnik funkcionira na principu „prvi uđe – prvi izađe“ (engl. *First In First Out*, FIFO), a za realizaciju spremnika koristi se ugrađeni tip podatka *deque* koji ima različite funkcije, kao što je funkcija *push_back* koja služi za dodavanje elementa na kraj reda. Znakovi se dodaju u spremnik sve dok je broj elemenata slike unutar detektiranog graničnog okvira manji od 5000, a kada broj elemenata slike unutar detektiranog graničnog okvira dostigne tu vrijednost, poziva se funkcija *carControlCommand* u kojoj se šalje odgovarajuća naredba za upravljanje modelom vozila na temelju prepoznatog prometnog znaka. Kako bi se to ostvarilo, unutar funkcije *carControlCommand* poziva se dodatna funkcija *findDetectedSign* kojoj se predaje spremnik detektiranih prometnih znakova. U funkciji se prebroji koliko znakova pojedine klase se nalazi u spremniku, a funkcija vraća ime klase koja je najviše zastupljena u spremniku. Jedan prometni znak mora ispuniti barem pola spremnika, a kako je spremnik veličine 19, znak se mora nalaziti minimalno 10 puta u spremninku kako bi algoritam za upravljanje modelom vozila bio siguran da se radi o tom znaku. Ako u spremniku postoji više različitih detektiranih prometnih znakova, a nijedan znak se ne nalazi 10 puta u spremniku, vozaču se šalje poruka da algoritam nije siguran o kojem se znaku radi i vozilo se zbog sigurnosnih razloga zaustavlja. Moguće je da algoritam točno prepozna prometni znak, ali ne uspije detektirati taj znak minimalno 10 puta (npr. zbog velike brzine kretanja). Zbog toga, dodan je uvjet koji provjerava nalazi li se samo taj znak u spremniku. Ako se nalazi, također se šalje odgovarajuća naredba za upravljanje, a ako se u spremniku nalazi

minimalno jedan drugi prometni znak, vozaču se također šalje poruka da algoritam nije siguran o kojem se znaku radi i vozilo se zbog sigurnosnih razloga zaustavlja.

Naredba za upravljanje vozilom šalje se pomoću poruke tipa *CarlaEgoVehicleControl.msg* čiji se sadržaj nalazi na slici 3.21. U liniji 2, nalazi se varijabla *throttle* koja simulira razinu pritisnutosti papučice gasa. Vrijednosti koje može primiti su u intervalu od 0 i 1, gdje 0 predstavlja nikakav pritisak papučice gasa, a 1 predstavlja maksimalan pritisak papučice gasa. Npr. ako korisnik pošalje vrijednost 0.5, ta vrijednost simulira pritisak papučice gasa do pola maksimalne vrijednosti. Varijabla *steer* nalazi se u liniji 5 i simulira zaokret volana. Interval mogućih vrijednosti je između -1 i 1, gdje -1 označava maksimalan zaokret volana u lijevu stranu, a 1 označava maksimalan zaokret volana u desnu stranu. Zadnja varijabla korištena u ovom algoritmu je varijabla *brake* (linija 8) koja simulira pritisak papučice kočnice. Kao i kod varijable *throttle*, interval vrijednosti koje varijabla *brake* prima mora biti između 0 i 1, gdje 0 predstavlja nikakav pritisak papučice kočnice, a 1 predstavlja maksimalan pritisak papučice kočnice. U poruci postoje dodatne varijable koje se mogu koristiti, kao što su *hand_brake* (korištenje ručne kočnice), *reverse* (kretanje u nazad), *manual_gear_shift* (mogućnost ručnog mijenjanja brzina) i *gear* (broj brzine prilikom ručnog mijenjanja brzina).

```
1   # 0. <= throttle <= 1.
2   float32 throttle
3
4   # -1. <= steer <= 1.
5   float32 steer
6
7   # 0. <= brake <= 1.
8   float32 brake
9
10  # hand_brake 0 or 1
11  bool hand_brake
12
13  # reverse 0 or 1
14  bool reverse
15
16  # gear
17  int32 gear
18
19  # manual gear shift
20  bool manual_gear_shift
```

Sl. 3.21. *Sadržaj poruke CarlaEgoVehicleControl.msg pomoću koje se upravlja modelom vozila u CARLA simulatoru.*

Ako je detektiran znak ograničenja brzine, šalje se eksperimentalno određena razina signala za brzinu, kako bi se brzina vozila prilagodila detektiranom znaku ograničenja brzine. Ako su

detektirani znak označenog pješačkog prijelaza i znak obaveznog zaustavljanja, vozilo se mora zaustaviti na 3 sekunde i nakon toga nastaviti voziti istom brzinom kao i prije zaustavljanja. Ako je detektiran znak raskrižja s cestom s prednošću prolaska, vozilu se 1.5 sekundu šalje maksimalan signal za kočnicu i nakon toga vozilo se kreće 3 sekunde brzinom do 7 km/h, a zatim nastavlja istom brzinom kao i prije zaustavljanja. Detekcijom znakova zabrane prometa u jednom i oba smjera, vozilo se mora zaustaviti nakon čega se više ne smije kretati, dok se kod detekcije znaka obaveznog smjera kretanja ravno i znaka ceste s prednošću prolaska vozilo nastavlja kretati istom brzinom kojom se kreće. Zadnji slučaj je detekcija znakova obaveznog smjera kretanja lijevo ili desno, gdje vozilo mora skrenuti točno u onu stranu prema kojoj detektirani prometni znak pokazuje. Ako se model vozila kreće s maksimalnom vrijednošću signala za pritiskanje papučice gasa i približava se prometnim znakovima nakon kojih mora stati na raskrižju ili skrenuti, potrebno je prvo smanjiti brzinu kretanja kako bi algoritam detektirao prometni znak u većem broju okvira i kako bi mogao uspješno obaviti radnju specifičnu za taj detektirani prometni znak. Kada algoritam 3 puta detektira jedan od prometnih znakova nakon kojih mora stati na raskrižju ili skrenuti, modelu vozila smanjuje se brzina kretanja na približno 30 km/h, a detekcije prometnog znaka nastavljaju se pohranjivati u spremnik. Ako bi se vozilo kretalo s maksimalnom vrijednošću signala za pritiskanje papučice gasa i približavalo se znaku obaveznog zaustavljanja, bez ovog uvjeta usporavanja, vozilo bi se zaustavilo na prevelikoj udaljenosti od znaka obaveznog zaustavljanja. No ako vozilo uspije smanjiti brzinu kretanja prije dolaska do samog prometnog znaka, zaustavit će se na pravilnoj udaljenosti od prometnog znaka. Također, ukoliko vozilo ne uspori prije obveznog skretanja lijevo ili desno, radnja skretanja obavlja se prekasno, tj. vozilo ne stigne skrenuti po cesti, nego tek nakon raskrižja. Iz navedenih razloga implementirano je usporavanje vozila prije provođenja spomenutih radnji.

U paketu *carla_ros_bridge* postoji još jedan tip poruke pomoću kojega je moguće upravljati modelom vozila, a naziva se *AckermannDrive.msg*. Ovaj tip poruke omogućava jednostavnije upravljanje brzinom modela vozila jer se umjesto signala papučice gasa šalje točno određena brzina (u m/s) kojom se model vozila treba kretati, no implementacija ovog tipa poruke nije bila uspješna [37].

Kako nisu svi automobili istih karakteristika (različita veličina, težina, snažniji motori), za svaki automobil šalju se drugačije vrijednosti za pritisak papučica gasa, kočnice i zaokret volana. Težim automobilima potreban je veći zaokret volana prilikom skretanja, dok automobili s jačim motorima ubrzavaju brže i pritisak papučice gasa je drugačiji. Ovaj algoritam je razvijen za točno

određeni automobil, a to je *BMW Grand tourer*. U tablici 3.2. nalaze se točne vrijednosti (do kojih se došlo empirijskim putem u samom simulatoru) koje se šalju određenim aktuatorima prilikom detekcije pojedinog znaka za model automobila *BMW Grand tourer*. Znak *X* u tablici predstavlja vrijeme do detekcije sljedećeg prometnog znaka.

Tab. 3.2. *Vrijednosti koje se šalju aktuatorima automobila za svaki detektirani prometni znak.*

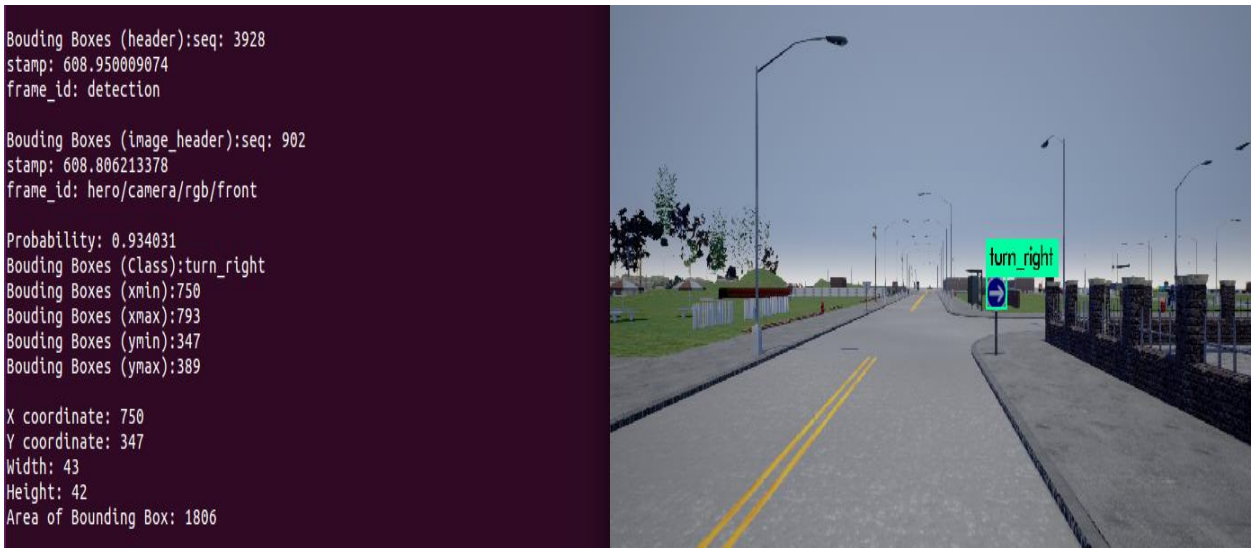
	Razina pritiska papučice gasa (vremensko trajanje)	Razina pritiska papučice kočnice (vremensko trajanje)	Razina zakretaja volana (vremensko trajanje)
Znak ograničenja brzine 30 km/h	0.67 (X)	Ako je brzina manja od 30 km/h: 0 (X)	0 (X)
		Ako je brzina veća od 30 km/h: 0.5 (1 s)	
Znak ograničenja brzine 50 km/h	0.7 (X)	Ako je brzina manja od 50 km/h: 0 (X)	0 (X)
		Ako je brzina veća od 50 km/h: 0.6 (1 s)	
Znak ceste s prednošću prolaska	30 km/h: 0.67 (X)	0 (X)	0 (X)
	50 km/h: 0.7 (X)		
Znak obaveznog smjera kretanja ravno	30 km/h: 0.67 (X)	0 (X)	0 (X)
	50 km/h: 0.7 (X)		
Znak obaveznog smjera kretanja lijevo	0 (1 s)	0.5 (1 s)	0 (1 s)
	0.67 (1.7 s)	0 (1.7 s)	0 (1.7 s)
	0.67 (3 s)	0 (3 s)	- 0.24 (3 s)
Znak obaveznog smjera kretanja desno	0 (1 s)	0.5 (1 s)	0 (1 s)
	0.67 (1.5 s)	0 (1.5 s)	0 (1.5 s)
	0.67 (2.5 s)	0 (2.5 s)	0.35 (2.5 s)

Znak raskrižja s cestom s prednošću prolaska	0 (1.5 s)	1 (1.5 s)	0 (1.5 s)
	0.35 (3.5 s)	0 (3.5 s)	0 (3.5 s)
	30 km/h: 0.67 (X)	0 (X)	0 (X)
	50 km/h: 0.7 (X)		
Znak obaveznog zaustavljanja	0 (3 s)	1 (3 s)	0 (3 s)
	30 km/h: 0.67 (X)	0 (X)	0 (X)
	50 km/h: 0.7 (X)		
Znak obilježenog pješačkog prijelaza	0 (3 s)	1 (3 s)	0 (3 s)
	30 km/h: 0.67 (X)	0 (X)	0 (X)
	50 km/h: 0.7 (X)		
Znak zabrane prometa u jednom smjeru	0 (2 s)	1 (2 s)	0 (2 s)
	0 (X)	1 (X)	0 (X)
Znak zabrane prometa u oba smjera	0 (2 s)	1 (2 s)	0 (2 s)
	0 (X)	1 (X)	0 (X)

Na slici 3.22. nalazi se primjer programskog koda za upravljanje modelom vozila na temelju prepoznatog znaka za obavezan smjer kretanja desno. Kao što je već spomenuto, naredba za upravljanje vozilom šalje se kada je detektirani znak određene veličine, odnosno kada se model vozila nalazi u neposrednoj blizini znaka. Na slici 3.23. prikazana je točna detekcija znaka obaveznog skretanja desno, ali vozilo se ne nalazi na udaljenosti od znaka na kojoj je potrebno poslati naredbu za upravljanje. U upravljačkom prozoru ispisuju se sve informacije o primljenom graničnom okviru, ali se ne šalje naredba za upravljanjem vozila. Za razliku od toga, na slici 3.24. nalazi se situacija kada je vozilo u neposrednoj blizini prometnog znaka gdje je potrebno poslati naredbu za skretanje desno. Ispisan je cijeli spremnik detektiranih znakova i vidljivo je da je u zadnjih 19 okvira algoritam prepoznao znak obaveznog skretanja desno 19 puta i sa sigurnošću se može utvrditi da se radi o znaku obaveznog skretanja desno i modelu vozila šalje se točna naredba za upravljanje.

```
else if (detectedSign == "turn_right")
{
    cout << "Turn right sign detected" << endl;
    // Stop for 3 seconds before moving
    ros::Time start = ros::Time::now();
    while (ros::Time::now() - start < ros::Duration(1))
    {
        regularOutput.throttle = 0.0;
        regularOutput.brake = 0.5;
        regularOutput.steer = 0.0;
        regularPub_.publish(regularOutput);
        loop_rate.sleep();
    }
    // Go straight for 1.5 seconds
    start = ros::Time::now();
    while (ros::Time::now() - start < ros::Duration(1.5))
    {
        regularOutput.throttle = 0.67;
        regularOutput.brake = 0.0;
        regularOutput.steer = 0.0;
        regularPub_.publish(regularOutput);
        loop_rate.sleep();
    }
    // Make a turn to the right
    start = ros::Time::now();
    while (ros::Time::now() - start < ros::Duration(2.5))
    {
        regularOutput.throttle = 0.67;
        regularOutput.brake = 0.0;
        regularOutput.steer = 0.35;
        regularPub_.publish(regularOutput);
        loop_rate.sleep();
    }
    cout << "Turn rignt signal published!" << endl;
    objectQueue.clear();
    area = 0;
    flag = false;
}
```

Sl. 3.22. *Primjer programskog koda za upravljanje modelom vozila na temelju prepoznatog znaka za obavezno skretanje desno.*



Sl. 3.23. *Detekcija udaljenog znaka za obavezno skretanje desno.*

Sl. 3.24. *Detekcija znaka za obavezno skretanje desno i slanje naredbe za upravljanje.*

U sljedećem poglavlju opisana je korištena baza podataka za testiranje rada razvijenog rješenja sastavljena od označenih slika prikupljenih iz simulatora, rezultati testiranja algoritma na toj bazi podataka i rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova u različitim scenarijima kreiranim u simulatoru, gdje vozilo mora ispravno reagirati na prepoznate prometne znakove.

4. VERIFIKACIJA RADA ALGORITMA ZA PREPOZNAVANJE PROMETNIH ZNAKOVA I UPRAVLJANJA MODELOM VOZILA

Tema ovog poglavlja je opisivanje načina testiranja rada predloženog algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima. U 3. poglavlju opisana je baza podataka slika prometnih znakova koja se koristila za treniranje i validaciju, a u ovom poglavlju opisana je baza korištena za testiranje ispravnosti rada algoritma. Opisani su rezultati testiranja na prikupljenim slikama iz simulatora na kojima je također bilo potrebno označiti navedene prometne znakove. Također su opisani rezultati testiranja vozila u različitim scenarijima kreiranim u simulatoru, gdje je vozilo moralo ispravno reagirati na svaki postavljeni prometni znak i poslati odgovarajuću naredbu za upravljanje.

Algoritam za upravljanje modelom vozila zasnovan na prepoznatim prometnim znakovima testiran je na računalu s operacijskim sustavom Ubuntu 16.04, ROS Kinetic, procesorom Intel i7-8700 3.20Ghz, 16 GB RAM-a i grafičkom karticom Nvidia RTX 2060 6GB.

4.1. Opis testnog skupa slika prometnih znakova iz simulatora i rezultata testiranja algoritma za prepoznavanje prometnih znakova na tom skupu

U ovom potpoglavlju opisat će se rezultati testiranja na testnom skupu slika preuzetih iz simulatora. Svi znakovi na slikama iz testnog skupa približno su iste veličine i algoritam bi ih sve trebao detektirati uspješno kako bi mogao poslati naredbu za upravljanjem modelom vozila. Ovaj testni skup sastoji se od 400 slika, gdje se na 335 slika nalazi po jedan ručno označeni prometni znak koji algoritam za upravljanje modelom vozila zasnovan na prepoznatim prometnim znakovima mora prepoznati. Ostalih 65 slika u ovom testnom skupu sadrži prometne znakove koje algoritam za upravljanje modelom vozila zasnovan na prepoznatim prometnim znakovima ne bi smio detektirati jer ne pripadaju nijednoj klasi prometnih znakova koje algoritam mora prepoznati. Na slici 4.1. prikazan je primjer prometnih znakova koji se nalaze u ovom testom skupu.

a) *b)*

c) *d)*

e) *f)*

Sl. 4.1. *Primjer prometnih znakova koje algoritam treba prepoznati (a) znak obaveznog zaustavljanja , (b) znak obaveznog smjera kretanja lijevo, (c) znak obilježenog pješačkog prijelaza, (d) znak ograničenja brzine 50 i primjer prometnih znakova koje algoritam ne bi trebao prepoznati (e) znak zabrane prometa za vozila koja prekoračuju dužinu od 10 metara, (f) znak zabrane skretanja ulijevo korištenih za testiranje rada algoritma za prepoznavanje prometnih znakova.*

Slike iz ovog testnog skupa koje sadrže prometne znakove koje algoritam mora prepoznati također su se morale označiti alatom *OpenLabeling* kako bi se mogla napraviti analiza točnosti detekcije i prebrojati koliko znakova algoritam detektira za različite vrijednosti praga (engl. *threshold value,* TV). U tablici 4.1 nalazi se ukupan broj označenih znakova u ovom testnom skupu. Ovaj testni skup nalazi se na DVD-u priloženom uz ovaj rad u mapi P.4.1.

Tab. 4.1. *Broj prometnih znakova u testnom skupu slika iz simulatora.*

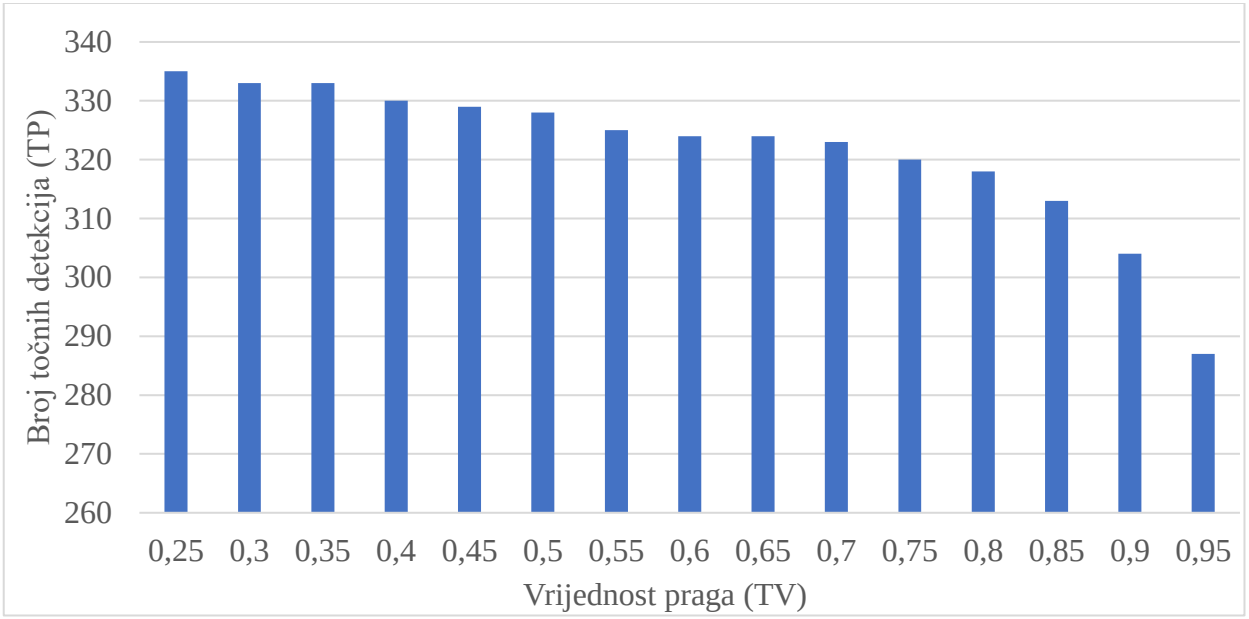
Tip prometnog znaka	Broj oznaka
Znak ograničenja brzine 30 km/h	49
Znak ograničenja brzine 50 km/h	27
Znak obaveznog smjera kretanja desno	25
Znak obaveznog smjera kretanja lijevo	30
Znak obaveznog smjera kretanja ravno	10
Znak ceste s prednošću prolaska	52
Znak obilježenog pješačkog prijelaza	20
Znak obaveznog zaustavljanja	36
Znak raskrižja s cestom s prednošću prolaska	18
Znak zabrane prometa u jednom smjeru	32
Znak zabrane prometa u oba smjera	36
Znakovi koje algoritam ne treba prepoznati	65
Ukupno	**400**

TV predstavlja minimalnu vrijednost sigurnosti s kojom algoritam detektira znakove. Npr. ako je TV postavljen na 0.5, algoritam će prikazati samo one detektirane znakove koji su detektirani sa sigurnošću većom od 50%. Potrebno je naći određeni TV za koji će algoritam detektirati sve označene znakova. Ukupan broj označenih znakova je 335, a model je testiran mijenjajući TV od 0.25 do 0.95, gdje stopa povećanja praga iznosi 0.05. U tablici 4.2. nalazi se broj TP, FP, FN, TN u ovisnosti o promjeni TV-a. Osim tih vrijednosti, u tablici se nalaze i ostale mjere performansi algoritma, kao što su preciznost, odziv, F-1 mjera i prosječni IoU. Također su prikazani rezultati testiranja algoritma za prepoznavanje navedenih prometnih znakova na slikama iz simulatora za različite TV.

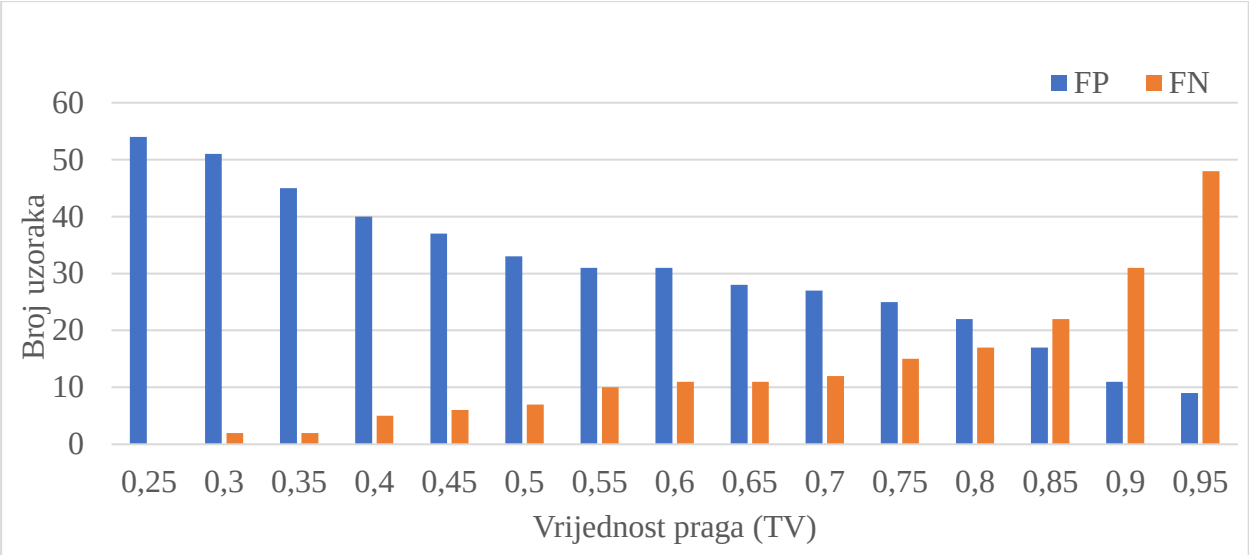
Tab. 4.2. *Rezultati testiranja algoritma za prepoznavanje prometnih znakova na testnom skupu slika prikupljenih iz simulatora.*

TV	Preciznost	Odziv	F-1 mjera	TP	FP	FN	TN	Prosječni IoU %
0,25	0,86	1	0,96	335	54	0	36	76,05
0,3	0,87	0,99	0,97	333	51	2	36	76,64
0,35	0,88	0,99	0,97	333	45	2	39	77,30
0,4	0,89	0,99	0,97	330	40	5	41	77,90
0,45	0,90	0,98	0,97	329	37	6	41	78,59
0,5	0,91	0,98	0,97	328	33	7	42	79,28
0,55	0,91	0,98	0,97	325	31	10	42	79,72
0,6	0,91	0,97	0,97	324	31	11	42	79,71
0,65	0,92	0,97	0,97	324	28	11	43	80,19
0,7	0,92	0,96	0,97	323	27	12	43	80,47
0,75	0,93	0,96	0,97	320	25	15	44	80,73
0,8	0,94	0,95	0,97	318	22	17	46	81,01
0,85	0,95	0,93	0,96	313	17	22	49	81,57
0,9	0,97	0,91	0,95	304	11	31	54	81,84
0,95	0,97	0,86	0,92	287	9	48	56	81,70

Na slici 4.2. nalazi se graf broja točnih detekcija (TP) za različite TV, a na slici 4.3. prikazana je promjena broja netočno detektiranih (FP) prometnih znakova i broja nedetektiranih prometnih znakova (FN) u odnosu na promjenu TV. Vidljivo je da broj TP-a opada kako se TV povećava.



Sl. 4.2. *Grafički prikaz broja točnih detekcija (TP) u odnosu na vrijednost praga (TV).*



Sl. 4.3. *Grafički prikaz broja netočno detektiranih prometnih znakova (FP) i broja nedetektiranih prometnih znakova (FN) u odnosu na vrijednost praga (TV).*

Porastom TV-a, preciznost se povećava zato što algoritam s većom sigurnošću detektira objekte jer odbacuje one detekcije za koje je sigurnost detekcije niska, odnosno ispod TV, a također se smanjuje broj FP detekcija. Suprotno tome, broj FN detekcija raste povećanjem TV-a što rezultira smanjenjem odziva. Broj TN detekcija raste povećanjem TV-a iz istog razloga kao i kod povećanja preciznosti: algoritam s većom sigurnošću detektira objekte jer odbacuje one detekcije za koje je sigurnost detekcije niska, pa tako se i smanjuje broj TN objekata. Za male vrijednosti TV-a, algoritam za npr. znak ograničenja brzine 80 km/h može pretpostaviti da se radi o znaku ograničenja brzine 30 km/h, a za veće vrijednosti TV-a takve netočne detekcije se smanjuju jer algoritam s većom sigurnošću detektira objekte.

Za ovaj algoritam bitno je da su svi prometni znakovi koje algoritam mora prepoznati detektirani i zbog toga TV koji je odabran za pokretanje algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova u simulatoru iznosi 0.25. Prema tablici 4.2., algoritam za prepoznavanje prometnih znakova detektira sve označene prometne znakove na validacijskom skupu za TV iznosa 0.25. Algoritam mora detektirati sve prometne znakove kako bi mogao donijeti odgovarajuću odluku upravljanja na temelju prepoznatog prometnog znaka i ne smije se dogoditi da nijedan izostavi. Međutim, pri ovom TV-u, pojavljuje se 54 FP detekcija prometnih znakova, a to je najčešće miješanje prometnih znakova za obavezno skretanje lijevo i desno, znakova za ograničenje brzine i znakova koje algoritam ne bi trebao prepoznati. Ovaj problem rješava se prebrojavanjem prometnih znakova iz spremnika detektiranih znakova. Npr. ako se model vozila približava znaku za obavezno skretanje desno, može se dogoditi da ga u nekim

okvirima detektira kao znak za obavezno skretanje lijevo. Kada vozilo dođe na mjesto gdje mora donijeti odluku o kojem prometnom znaku se radi, prebrojavaju se detekcije znakova u spremniku: ako se npr. u spremniku nalazi 14 detekcija znaka za obavezan smjera kretanja desno i 5 detekcija znaka za obavezan smjer kretanja lijevo, algoritam za upravljanje modelom vozila zasnovan na prepoznavanju prometnih znakova zaključuje da se radi o znaku za obavezan smjer kretanja desno, čime donosi ispravnu odluku i šalje točnu naredbu za upravljanjem vozilom.

4.2. Opis testnog skupa scenarija u simulatoru i rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima na tom skupu

U simulatoru je kreirano 7 različitih scenarija (gdje su 4 scenarija ponovljena s različitim početnim brzinama kretanja) koji obuhvaćaju sve znakove koje algoritam mora prepoznati. Svi scenariji su predstavljeni su trima različitim vremenskim uvjetima (sunce, kiša i sumrak), što daje ukupno 33 različitih testnih slučajeva (scenarija). Video sekvence svih testnih slučajeva u kojima su uključeni i rezultati rada algoritma za upravljanje modelom vozila zasnovanog na temelju prepoznatih prometnih znakova nalaze se na DVD-u priloženom uz ovaj rad u mapama P.4.2. – P.4.8. U tablicama rezultata, slučajevi u kojima algoritam nije donio točnu odluku o slanju naredbe za upravljanje modelom vozila obojani su sivom bojom.

Scenarij 1.1 – Detekcija znakova za ograničenje brzine (uvjet: početna brzina kretanja vozila veća od 50 km/h)

U početku se modelu vozila šalje maksimalna vrijednost signala za pritiskanje papučice gasa tako da vozilo doseže brzinu kretanja veću od 50 km/h. Model vozila prvo nailazi na znak ograničenja brzine 50 km/h, nakon kojega mora spustiti brzinu kretanja ispod 50 km/h i nakon toga ubrzavati do maksimalnih 50 km/h. Kao podsjetnik, moguće je postaviti točno određenu brzinu kretanja modela vozila pomoću *AckermannDrive.msg* poruka, no kako implementacija tih poruka nije bila uspješna, brzine kretanja modela vozila morale su se regulirati pomoću signala za pritisak papučice gasa i kočnice. Zatim, model vozila nailazi na znak ograničenja brzine 30 km/h nakon kojega spušta brzinu kretanja ispod 30 km/h i ubrzava do maksimalnih 30 km/h. U tablici 4.3. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak ograničenja brzine u ovom scenariju za sva 3 vremenska uvjeta.

Tab. 4.3. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 1.1 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak ograničenja brzine 50 km/h**	**Broj detektiranih znakova u spremniku za znak ograničenja brzine 30 km/h**
Sunčano	Znak ograničenja brzine 50 km/h : 7	Znak ograničenja brzine 30 km/h : 8
Kiša	Znak ograničenja brzine 50 km/h : 6	Znak ograničenja brzine 30 km/h : 6
Sumrak	Znak ograničenja brzine 50 km/h : 4	Znak ograničenja brzine 30 km/h : 6
	Znak ograničenja brzine 30 km/h : 2	

Za vremenske uvjete sunčano i kiša, algoritam ispravno detektira prometne znakove, ali zbog velike brzine kretanja algoritam nije uspio ispuniti barem pola spremnika, a kako se u spremniku nalaze samo ti znakovi, šalje se ispravna naredba za upravljanje modelom vozila. Za vremenski uvjet sumrak, algoritam znak ograničenja brzine 50 km/h detektira 4 puta, ali također ga u 2 slučaja detektira kao znak ograničenja brzine 30 km/h i zbog toga se vozilo zaustavlja jer algoritam nije siguran o kojem se znaku radi. Kao podsjetnik, ako se u spremniku nalazi više detektiranih prometnih znakova, određeni znak mora se nalaziti u spremniku minimalno 10 puta kako bi se poslala naredba za upravljanje koja odgovara tom prometnom znaku. U suprotnome, algoritam će obavijestiti korisnika da se u spremniku nalazi premalo detektiranih prometnih znakova i model vozila će se zaustaviti. Ovu bi pogrešku bilo moguće izbjeći implementacijom nekakvog drugog kriterija o odlučivanju provođenja odgovarajuće naredbe upravljanja vozilom, npr. da se šalje naredba o upravljanju vezana uz onaj znak koji je najviše puta prepoznat. Međutim, da bi se došlo do najboljeg kriterija trebalo bi testove provoditi na puno većem uzorku situacija od onih koje su obuhvaćene ovim diplomskim radom.

Scenarij 1.2 – Detekcija znakova za ograničenje brzine (uvjet: početna brzina kretanja vozila manja od 50 km/h)

Raspored prometnih znakova u ovom scenariju identičan je kao u prethodnom scenariju, ali je početna brzina kretanja vozila manja od 50 km/h. Nailaskom na prometni znak ograničenja brzine 50 km/h, vozilo ubrzava do maksimalne brzine kretanja od 50 km/h i vozi tom brzinom do detekcije prometnog znaka ograničenja brzine 30 km/h, nakon kojega model vozila spušta brzinu kretanja ispod 30 km/h i ubrzava do maksimalnih 30 km/h. U tablici 4.4. nalazi se ispis detektiranih

prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak ograničenja brzine u ovom scenariju za sva 3 vremenska uvjeta.

Tab. 4.4. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 1.2 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak ograničenja brzine 50 km/h**	**Broj detektiranih znakova u spremniku za znak ograničenja brzine 30 km/h**
Sunčano	Znak ograničenja brzine 50 km/h : 10	Znak ograničenja brzine 30 km/h : 7
	Znak ograničenja brzine 30 km/h : 1	
Kiša	Znak ograničenja brzine 50 km/h : 12	Znak ograničenja brzine 30 km/h : 10
		Znak ograničenja brzine 50 km/h : 3
Sumrak	Znak ograničenja brzine 50 km/h : 14	Znak ograničenja brzine 30 km/h : 6
	Znak ograničenja brzine 30 km/h : 1	

U ovom scenariju, algoritam je uspješno detektirao oba znaka ograničenja brzine u svim vremenskim uvjetima i poslao ispravnu naredbu za upravljanjem modelom vozila. U odnosu na scenarij 1.1, vozilo se kreće manjom brzinom i zbog toga uspijeva detektirati prometne znakove u više okvira i samim time nalazi se više detektiranih prometnih znakova u spremniku. Pojavljuje se nekoliko netočnih detekcija, ali kako je ispravan znak detektiran minimalno 10 puta, netočne detekcije nisu utjecale na odluku zbog načina odabira konačne radnje koju vozilo treba provesti.

Scenarij 2 – Detekcija znakova obaveznog zaustavljanja, raskrižja s cestom s prednošću prolaska, obilježenog pješačkog prijelaza i ceste s prednošću prolaska

U početku model vozila ubrzava do 30 km/h i održava tu brzinu kretanja dok ne naiđe na znak obaveznog zaustavljanja kada je potrebno zaustaviti vozilo na 3 sekunde. Zatim, model vozila također ubrzava do 30 km/h i kreće se tom brzinom sve dok ne detektira znak raskrižja s cestom s prednošću prolaska. Modelu vozila šalje se naredba za kočenje u trajanju od 1.5 sekundi nakon čega se vozilo zaustavlja i 3 sekunde se kreće brzinom do 7 km/h nakon čega nastavlja ubrzavati do 30 km/h sve dok ne dođe do znaka za obilježeni pješački prijelaz, gdje postupa identično kao i

kod znaka obaveznog zaustavljanja: zaustavlja se na 3 sekunde, a nakon toga nastavlja ubrzavati do 30 km/h. Model vozila kreće se tom brzinom i nailazi na znak ceste s prednošću prolaska i nastavlja se kretati istom brzinom kao i prije detekcije znaka ceste s prednošću prolaska. U tablici 4.5. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak u ovom scenariju za sva 3 vremenska uvjeta.

Tab. 4.5. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 2 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak obaveznog zaustavljanja**	**Broj detektiranih znakova u spremniku za znak raskrižja s cestom s prednošću prolaska**	**Broj detektiranih znakova u spremniku za znak za obilježeni pješački prijelaz**	**Broj detektiranih znakova u spremniku za znak ceste s prednošću prolaska**
Sunčano	Znak obaveznog zaustavljanja: 11	Znak raskrižja s cestom s prednošću prolaska: 18	Znak za obilježeni pješački prijelaz: 14	Znak ceste s prednošću prolaska: 16
Kiša	Znak obaveznog zaustavljanja: 11 Znak zabrane prometa u jednom smjeru: 6	Znak raskrižja s cestom s prednošću prolaska: 12	Znak za obilježeni pješački prijelaz: 13	Znak ceste s prednošću prolaska: 17
Sumrak	Znak obaveznog zaustavljanja: 11 Znak zabrane prometa u jednom smjeru: 8	Znak raskrižja s cestom s prednošću prolaska: 13	Znak za obilježeni pješački prijelaz: 16	Znak ceste s prednošću prolaska: 19

Algoritam je uspješno prepoznao svaki postavljeni prometni znak u scenariju i tako poslao ispravnu naredbu za upravljanje modelom vozila u simulatoru. Znakovi raskrižja s cestom s prednošću prolaska, obilježenog pješačkog prijelaza i ceste s prednošću prolaska detektirani su bez ijedne netočne detekcije u spremniku. Za vremenske uvjete kiša i sumrak, znak obaveznog zaustavljanja u nekoliko okvira detektiran je kao znak zabrane prometa u jednom smjeru zbog njihove sličnosti (crvena boja pozadine s bijelom bojom u sredini prometnog znaka). Treba napomenuti da se netočne detekcije događaju na većoj udaljenosti, a kada se model vozila približi

znaku obaveznog zaustavljanja, netočne detekcije nestaju i počinju prevladavati točne, što na kraju i dovodi do ispravne odluke o radnji koju treba poduzeti.

Scenarij 3 – Detekcija znakova obilježenog pješačkog prijelaz, obaveznog zaustavljanja, ceste s prednošću prolaska i raskrižja s cestom s prednošću prolaska

U ovom scenariju, prometni znakovi koje model vozila mora detektirati identični su prometnim znakovima iz prethodnog scenarija, ali drugačiji je raspored znakova. Model vozila prvo nailazi na znak za obilježeni pješački prijelaz, zatim na znak obaveznog zaustavljanja, nakon toga na znak ceste s prednošću prolaska i zadnji prometni znak kojeg model vozila mora detektirati je znak raskrižja s cestom s prednošću prolaska. U tablici 4.6. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak u ovom scenariju za sva 3 vremenska uvjeta.

Tab. 4.6. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 3 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak za obilježeni pješački prijelaz**	**Broj detektiranih znakova u spremniku za znak obaveznog zaustavljanja**	**Broj detektiranih znakova u spremniku za znak ceste s prednošću prolaska**	**Broj detektiranih znakova u spremniku za znak raskrižja s cestom s prednošću prolaska**
Sunčano	Znak za obilježeni pješački prijelaz: 19	Znak obaveznog zaustavljanja: 10	Znak ceste s prednošću prolaska: 19	Znak raskrižja s cestom s prednošću prolaska: 12
		Znak zabrane prometa u jednom smjeru: 9		
Kiša	Znak za obilježeni pješački prijelaz: 19	Znak obaveznog zaustavljanja: 8	Znak ceste s prednošću prolaska: 19	Znak raskrižja s cestom s prednošću prolaska: 11
		Znak zabrane prometa u jednom smjeru: 4		

Sumrak	Znak za obilježeni pješački prijelaz: 19	Znak obaveznog zaustavljanja: 11 Znak zabrane prometa u jednom smjeru: 8	Znak ceste s prednošću prolaska: 19	Znak raskrižja s cestom s prednošću prolaska: 11

Algoritam je uspješno prepoznao svaki postavljeni prometni znak u scenariju osim znaka obaveznog zaustavljanja za vremenski uvjet kiša. Za taj slučaj, u spremniku većinski prevladava broj detekcija znaka obaveznog zaustavljanja, no kako je broj detekcija tog prometnog znaka manji od 10, a postoje detekcije drugih znakova u spremniku (znak zabrane prometa u jednom smjeru), model vozila se zaustavio i nije se više kretao. Kao i u prethodnom scenariju, znakovi raskrižja s cestom s prednošću prolaska, obilježenog pješačkog prijelaza i ceste s prednošću prolaska detektirani su bez ijedne netočne detekcije u spremniku. Za vremenske uvjete sunčano i sumrak, znak obaveznog zaustavljanja u nekoliko okvira (na većoj udaljenosti) detektiran je kao znak zabrane prometa u jednom smjeru.

Scenarij 4.1 – Detekcija znakova obaveznog smjera kretanja desno i lijevo (uvjet: brzina kretanja vozila do 30 km/h)

U početku model vozila ubrzava do brzine kretanja 30 km/h i nailazi na prometni znak obaveznog smjera kretanja desno, nakon kojega na križanju skreće desno. Zatim, ponovno ubrzava do brzine kretanja 30 km/h i dolazi do prometnog znaka obaveznog smjera skretanja lijevo, nakon kojega na križanju skreće lijevo i ponovno ubrzava do brzine kretanja 30 km/h. U tablici 4.7. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak obaveznog smjera kretanja lijevo i desno u ovom scenariju za sva 3 vremenska uvjeta.

Tab. 4.7. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 4.1 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak obaveznog smjera kretanja desno**	**Broj detektiranih znakova u spremniku za znak obaveznog smjera kretanja lijevo**
Sunčano	Znak za obavezan smjer kretanja desno: 19	Znak za obavezan smjer kretanja lijevo: 19

Kiša	Znak za obavezan smjer kretanja desno: 18	Znak za obavezan smjer kretanja lijevo: 19
	Znak za obavezan smjer kretanja lijevo: 1	
Sumrak	Znak za obavezan smjer kretanja desno: 19	Znak za obavezan smjer kretanja lijevo: 18
		Znak za obavezan smjer kretanja desno: 1

Algoritam je uspješno prepoznao svaki postavljeni prometni znak obaveznog smjera kretanja u scenariju za sve vremenske uvjete i poslao ispravnu naredbu za upravljanjem modelom vozila.

Scenarij 4.2 – Detekcija znakova obaveznog smjera kretanja desno i lijevo (uvjet: brzina kretanja vozila veća od 50 km/h)

Raspored prometnih znakova u ovom scenarija identičan je kao u prethodnom, ali je početna brzina kretanja vozila veća od 50 km/h. U scenariju 4.2, model vozila kreće se uz maksimalnu vrijednost signala za pritiskanje papučice gasa sve dok 3 puta ne detektira znak obaveznog smjera kretanja desno, nakon čega pokušava smanjiti brzinu kretanja na 30 km/h kako bi mogao uspješno i precizno obaviti radnju skretanja desno na križanju. Nakon toga, vozilo se kreće s maksimalnom vrijednošću signala za pritiskanje papučice gasa sve dok 3 puta ne detektira znak obaveznog smjera kretanja lijevo nakon čega ubrzava do maksimalnih 30 km/h kako bi mogao uspješno i precizno obaviti radnju skretanja lijevo na križanju nakon kojega se nastavlja kretati s maksimalnom vrijednošću signala za pritiskanje papučice gasa. U tablici 4.8. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak u ovom scenariju za sva 3 vremenska uvjeta.

Tab. 4.8. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 4.2 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak obaveznog smjera kretanja desno**	**Broj detektiranih znakova u spremniku za znak obaveznog smjera kretanja lijevo**
Sunčano	Znak za obavezan smjer kretanja desno: 13	Znak za obavezan smjer kretanja lijevo: 19
Kiša	Znak za obavezan smjer kretanja desno: 6	Znak za obavezan smjer kretanja lijevo: 19
	Znak za obavezan smjer kretanja lijevo: 1	

Sumrak	Znak za obavezan smjer kretanja desno: 14	Znak za obavezan smjer kretanja lijevo: 18
		Znak za obavezan smjer kretanja desno: 1

Za vremenski uvjet kiša, algoritam nije prepoznao znak obaveznog smjera kretanja desno i model vozila zaustavio se iz sigurnosnih razloga. Ostale znakove obaveznog smjera kretanja lijevo i desno algoritam je uspješno prepoznao i poslao ispravnu naredbu za upravljanje modelom vozila. U odnosu na prethodni scenarij, vidljivo je da algoritam detektira prometne znakove u manjem broju okvira zbog veće brzine kretanja.

Scenariji 5.1 i 6.1 – Detekcija znakova obaveznog smjera kretanja ravno, zabrane prometa u jednom smjeru i zabrane prometa u oba smjera (početni uvjet: brzina kretanja vozila do 30 km/h)

Scenarij 5.1 sadrži samo jedan prometni znak, a to je znak zabrane prometa u jednom smjeru. Model vozila ubrzava do brzine kretanja 30 km/h i nailazi na znak zabrane prometa u jednom smjeru nakon kojega se mora zaustaviti i ne smije se više kretati. Scenarij 6.1 sadrži prometne znakove obaveznog smjera kretanja ravno i zabrane prometa u oba smjera. Model vozila ubrzava do brzine kretanja 30 km/h i nailazi na znak obaveznog smjera kretanja ravno, nakon kojega zadržava istu brzinu kojom se kretao prije detekcije tog prometnog znaka, a zatim nailazi na znak zabrane prometa u oba smjera nakon kojega se šalje identična naredba kao pri detekciji znaka zabrane prometa u jednom smjeru. U tablici 4.9. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak u oba scenarija za sva 3 vremenska uvjeta.

Tab. 4.9. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 5.1 i Scenarij 6.1 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak zabrane prometa u jednom smjeru (Scenarij 5.1)**	**Broj detektiranih znakova u spremniku za znak obaveznog smjera kretanja ravno (Scenarij 6.1)**	**Broj detektiranih znakova u spremniku za znak zabrane prometa u oba smjera (Scenarij 6.1)**
Sunčano	Znak zabrane prometa u jednom smjeru: 19	Znak obaveznog smjera kretanja ravno: 16	Znak zabrane prometa u oba smjera: 19
		Znak obaveznog smjera kretanja lijevo: 1	

Kiša	Znak zabrane prometa u jednom smjeru: 19	Znak obaveznog smjera kretanja ravno: 13	Znak zabrane prometa u oba smjera: 16
Sumrak	Znak zabrane prometa u jednom smjeru: 19	Znak obaveznog smjera kretanja ravno: 17	Znak zabrane prometa u oba smjera: 15

Algoritam je uspješno prepoznao svaki postavljeni prometni znak u oba scenarija za sve vremenske uvjete i poslao ispravnu naredbu za upravljanjem modelom vozila.

Scenariji 5.2 i 6.2 – Detekcija znakova obaveznog smjera kretanja ravno, zabrane prometa u jednom smjeru i zabrane prometa u oba smjera (početni uvjet: brzina kretanja vozila veća od 50 km/h)

Raspored prometnih znakova u ova 2 scenarija identičan je kao u prethodna 2 scenarija, ali je početna brzina kretanja vozila veća od 50 km/h. U scenariju 5.2, model vozila kreće se s maksimalnom vrijednošću signala za pritiskanje papučice gasa sve dok 3 puta ne detektira znak zabrane prometa u jednom smjeru, nakon čega pokušava smanjiti brzinu kretanja na 30 km/h kako bi se mogao zaustaviti neposredno nakon znaka zabrane prometa u jednom smjeru nakon čega se ne smije više kretati. U scenariju 6.2, model vozila kreće se uz maksimalnu vrijednost signala za pritiskanje papučice gasa i dolazi do znaka obaveznog smjera kretanja ravno, nakon kojega se također nastavlja kretati s maksimalnom vrijednošću signala za pritiskanje papučice gasa sve dok 3 puta ne detektira znak zabrane prometa u oba smjera nakon čega pokušava smanjiti brzinu kretanja na 30 km/h kako bi se mogao zaustaviti neposredno nakon znaka zabrane prometa u oba smjera nakon čega se ne smije više kretati. U tablici 4.10. nalazi se ispis detektiranih prometnih znakova koji se nalaze u spremniku za svaki postavljeni prometni znak u oba scenarija za sva 3 vremenska uvjeta.

Tab. 4.10. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 5.2 i Scenarij 6.2 u različitim vremenskim uvjetima.*

Vremenski uvjet	**Broj detektiranih znakova u spremniku za znak zabrane prometa u jednom smjeru (Scenarij 5.2)**	**Broj detektiranih znakova u spremniku za znak obaveznog smjera kretanja ravno (Scenarij 6.2)**	**Broj detektiranih znakova u spremniku za znak zabrane prometa u oba smjera (Scenarij 6.2)**
Sunčano	Znak zabrane prometa u jednom smjeru: 15	Znak obaveznog smjera kretanja ravno: 11	Znak zabrane prometa u oba smjera: 7
Kiša	Znak zabrane prometa u jednom smjeru: 9	Znak obaveznog smjera kretanja ravno: 6	Znak zabrane prometa u oba smjera: 8
Sumrak	Znak zabrane prometa u jednom smjeru: 12	Znak obaveznog smjera kretanja ravno: 9	Znak zabrane prometa u oba smjera: 9

U scenariju 5.2 za vremenske uvjete sunčano i sumrak, model vozila uspijeva spustiti brzinu kretanja na približno 30 km/h nakon čega se uspijeva na vrijeme zaustaviti u neposrednoj blizini uspješno prepoznatog prometnog znaka zabrane prometa u jednom smjeru. Za vremenski uvjet kiša, model vozila uspješno je prepoznao znak zabrane prometa u jednom smjeru, ali ne uspijeva na vrijeme spustiti brzinu kretanja na približno 30 km/h i tako se zaustavlja na prevelikoj udaljenosti od prometnog znaka.

U scenariju 6.2, vozilo u sva tri vremenska uvjeta uspješno prepoznaje sve prometne znakove i šalje odgovarajuću naredbu upravljanja za svaki prepoznati prometni znak, no zbog velike brzine kretanja ne uspijeva na vrijeme spustiti brzinu kretanja na 30 km/h i zaustavlja se na prevelikoj udaljenosti od prometnog znaka zabrane prometa u oba smjera. Također, vidljivo je da model vozila pri nižim brzinama sadrži veći broj detektiranih prometnih znakova u spremniku.

Scenariji 7 – Detekcija prometnih znakova koje algoritam ne bi smio detektirati

U ovom scenariju postavljeno je 8 prometnih znakova za koje algoritam upravljanja modelom vozila zasnovan na prepoznatim prometnim znakovima nije treniran, odnosno ne bi ih

trebao prepoznati. Model vozila kreće se brzinom 30 km/h i nailazi na prometne znakove koje ne treba detektirati. U tablici 4.11. nalazi se broj detekcija prometnih znakova iz ovog scenarija za sva 3 vremenska uvjeta.

Tab. 4.11. *Rezultati testiranja algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova za Scenarij 7 u različitim vremenskim uvjetima.*

Vremenski uvjet	Broj detektiranih prometnih znakova/ Ukupan broj prometnih znakova u scenariju
Sunčano	2/8
Kiša	3/8
Sumrak	3/8

Za vremenski uvjet sunčano, algoritam 4 prometna znaka ne detektira u nijednom okviru i ne donosi nikakvu naredbu za upravljanje modelom vozila, osim one za kretanje brzinom 30 km/h koja se šalje kako bi se vozilo moglo kretati. Algoritam 2 prometna znaka (znak zabrane skretanja ulijevo i znak prestanka obvezne upotrebe zimske opreme) detektira samo u 2, odnosno 3 okvira, a kako se u spremniku mora nalaziti minimalno 6 znakova, model vozila ne donosi nikakvu naredbu za upravljanje modelom vozila, osim one za kretanje brzinom 30 km/h koja se šalje kako bi se vozilo moglo kretati. Problem nastaje u tome što se spremnik samo čisti u slučaju slanja naredbe za upravljanjem vozila, pa tako ovih 5 detekcija ostaju u spremniku sve dok se on ne očisti. Kao rezultat toga, vozilo nailazi na još jedan prometni znak (znak zabrane prometa za vozila koja prekoračuju dužinu od 10 metara) kojega netočno detektira u 5 okvira, a kako se od prije u spremniku već nalazi 5 detektiranih prometnih znakova, vozilo ima dovoljan broj detektiranih znakova u spremniku da donese naredbu za upravljanje. Kako u spremniku nijednog detektiranog prometnog znaka nema minimalno 10 puta, vozilo se zaustavlja iz sigurnosnih razloga jer algoritam nije siguran o kojem prometnom znaku se radi. Nakon toga, algoritam je ponovno pokrenut nakon zadnjeg netočno detektiranog prometnog znaka, gdje nailazi na novi prometni znak (znak ograničenja brzine 130 km/h) kojeg netočno prepoznaje kao znak ograničenja brzine 50 km/h i šalje netočnu naredbu za povećanje brzine kretanja na 50 km/h.

Za vremenski uvjet kiša, algoritam 5 prometnih znakova ne detektira u nijednom okviru i ne donosi nikakvu naredbu za upravljanje modelom vozila, osim one za kretanje brzinom 30 km/h koja se šalje kako bi se vozilo moglo kretati. Znak zabrane skretanja ulijevo netočno detektira kao znak zabrane prometa u oba smjera i šalje netočnu naredbu za zaustavljanje vozila. Zatim, znak

ograničenja brzine 130 km/h netočno detektira kao znak ograničenja brzine 50 km/h i šalje netočnu naredbu za povećanje brzine kretanja na 50 km/h, dok znak zabrane prometa za vozila koja prekoračuju dužinu od 10 metara detektira netočno, ali nije siguran o kojem znaku se radi i zaustavlja vozilo iz sigurnosnih razloga.

Za vremenski uvjet sumrak, kao i u prethodna 2 vremenska uvjeta, algoritam 5 prometnih znakova ne detektira u nijednom okviru i ne donosi nikakvu naredbu za upravljanje modelom vozila, osim one za kretanje brzinom 30 km/h koja se šalje kako bi se vozilo moglo kretati. Jedan prometni znak (znak zabrane skretanja ulijevo) netočno detektira kao znak zabrane prometa u oba smjera i šalje netočnu naredbu za zaustavljanje vozila, dok ostala 2 prometna znaka (znak ograničenja brzine 130 km/h i znak zabrane prometa za vozila koja prekoračuju dužinu od 10 metara) detektira netočno, ali nije siguran o kojim znakovima se radi i zaustavlja vozilo iz sigurnosnih razloga.

Kako bi se izrazio postotak broja točno obavljenih radnji nakon prepoznavanja znaka, uvedena je nova mjera točnosti rada algoritma, odnosno omjer broja točno obavljenih radnji nakon prepoznavanja znaka i ukupnog broja ispitnih radnji:

$$Točnost = \frac{broj\ točno\ obavljenih\ radnji\ nakon\ prepoznavanja\ znaka}{ukupan\ broj\ ispitnih\ radnji} \quad (4\text{-}1)$$

Od ukupno 90 postavljenih znakova u svim scenarijima, algoritam je donio ispravnu naredbu upravljanja za 79 znakova, što prema (4-1) daje točnost od 0.87, odnosno 87 %.

Uzevši u obzir iste scenarije prometnih znakova, ali s različitim brzinama, dolazi se do zaključka da algoritam pri većim brzinama kretanja ne uspijeva uvijek točno obaviti zadanu radnju upravljanja modelom vozila. Ovaj problem moguće je pripisati usporenosti simulatora, odnosno premalom broju obrađenih FPS-a. Ako bi simulator obrađivao veći broj FPS-a, ne bi radio usporeno i brže bi mogao detektirati prometni znak i pravovremeno usporiti brzinu kretanja kako bi model vozila ispravno reagirao na nadolazeći prometni znak. Također, povećanjem računalnih mogućnosti testnog računala (grafička kartica boljih performansi), moguće je trenirati slike na većim dimenzijama, pa samim time i povećati dimenzije ulazne slike koju YOLO algoritam obrađuje i tako detektirati manje objekte, odnosno prometne znakove na većim udaljenostima. U scenariju s prometnim znakovima koje algoritam ne bi smio prepoznati, vidljivo je da algoritam netočno detektira nekoliko prometnih znakova koji svojim izgledom i sadržajem podsjećaju na

neke znakove koje algoritam može prepoznati (npr. znakovi ograničenja brzine). Ovaj problem može se riješiti povećanjem TV-a čime bi se smanjio broj netočnih detekcija, no teže bi bilo ispuniti spremnik detektiranim znakovima zato što je broj detekcija umanjen. Također, pri prikupljanju baze podataka za treniranje algoritma, kao negativne primjere moguće je dodati znakove koje algoritam ne bi trebao prepoznati, a često se pojavljuju u prometu i imaju slične karakteristike kao neki znakovi koje algoritam treba prepoznati. Tako bi algoritam postao otporniji na detekciju tih prometnih znakova i detektirao bi ih u manjem broju slučajeva.

Kao što je već spomenuto, algoritam ponekad zamjenjuje znakove za obavezno skretanje lijevo i desno i znakove ograničenja brzine, a znak obaveznog zaustavljanja sa znakom zabrane prometa u jednom smjeru samo na većim udaljenostima. Razlog je taj što su ti znakovi vrlo slični, a postoji nekoliko rješenja tog problema. Jedan od njih je već predstavljen u ovom radu, a to je korištenje spremnika koji sprema određen broj detekcija i na temelju toga odlučuje o kojem se znaku radi. Drugo rješenje je proširenje trening skupa znakovima s kojima algoritam za prepoznavanje prometnih znakova ima problema pri prepoznavanju. Još jedno od rješenja je i dodatna obrada detektiranog objekta, odnosno slike prometnog znaka koji se nalazi unutar graničnog okvira, npr. znakove za ograničenje brzine moguće je točno odrediti koristeći prepoznavanje optičkih znakova (engl. *Optical Character Recognition*, OCR). Pomoću OCR bilo bi moguće ispravno prepoznati sve znakove ograničenja brzine i ne bi bilo potrebno označavati slike za određene znakove ograničenja brzine. Ako algoritam zamjenjuje znakove samo na većim udaljenostima, moguće je postaviti minimalnu vrijednost broja elemenata slike unutar detektiranog graničnog okvira za koju bi se detektirani prometni znakovi spremali u spremnik.

5. ZAKLJUČAK

U ovom radu obrađen je problem prepoznavanja prometnih znakova i upravljanja modelom vozila na temelju prepoznatih znakova. Stvoren je vlastiti algoritam za upravljanje modelom vozila zasnovan na prepoznatim prometnim znakovima koji za ulaz prima sliku s kamere iz simulatora i na temelju prepoznatog prometnog znaka šalje naredbu za upravljanje modelom vozila. Za prepoznavanje prometnih znakova korišten je YOLOv3 algoritam za koji je bilo potrebno istrenirati parametre modela kako bi mogao prepoznavati određene prometne znakove, dok se za upravljanje vozilom koristi programski okvir ROS. Za provođenje simulacija koristi se simulator otvorenog koda CARLA. Tijekom testiranja algoritma pokazalo se da su moguće pogreške prilikom detekcije sličnih znakova, kao što su znakovi za ograničenje brzine (međusobno 30 km/h i 50 km/h), znakovi za obavezno skretanje (lijevo i desno), te znak obaveznog zaustavljanja i znak zabrane prometa u jednom smjeru. Zbog toga je implementiran spremnik u kojem se prebrojavaju detektirani prometni znakovi i na temelju znaka s najvećim brojem detekcija šalje se naredba za upravljanjem modela vozila.

Točnija detekcija znakova za obavezno skretanje mogla bi se dobiti dodavanjem većeg broja tih znakova u trening skup, a znakovi ograničenja brzine uspješno bi se mogli detektirati koristeći optičko prepoznavanje znakova. Također je moguće povećati broj znakova koje algoritam za prepoznavanje znakova može prepoznati tako što bi se dodale označene slike tih znakova u skupove za treniranje i testiranje. Za dobivanje točnijih rezultata, potrebno je proširiti bazu podataka za testiranje.

Algoritam je testiran na testnom skupu od 400 slika od kojih svaka sadrži po jedan ručno označeni prometni znak i na kojima je uspješno prepoznao sve prometne znakove uz TV od 0.25. Za ovu vrijednost TV pojavilo se 54 FP-a. Algoritam je također testiran na 90 znakova u simulatoru u sklopu 11 različitih scenarija, gdje je od potrebnih 90 radnji koje je vozilo trebalo poduzeti nakon nailaska na određeni znak, ispravno poduzelo 79 radnji, a postotak točno obavljenih radnji iznosi 87%. Algoritam uspijeva obrađivati približno 40 FPS, što ga čini primjenjivim u stvarnom vremenu.

LITERATURA

[1] Daimler – prvi automobil, https://www.daimler.com/company/tradition/company-history/1885-1886.html [29. lipanj 2019.]

[2] KPMG, „I see. I think. I drive. (I learn). How Deep Learning is revolutionizing the way we interact with out cars“, https://assets.kpmg/content/dam/kpmg/se/pdf/komm/2016/se-isee-ithink-idrive-ilearn.pdf [29. lipanj 2019.]]

[3] K. Bimbraw, Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology. ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings. 1., str. 191-198.

[4] M. Vranješ, R. Grbić, predavanja iz kolegija Strojno učenje u sustavima autonomnih i umreženih vozila, studij Automobilsko računarstvo i komunikacije, ak.god. 2018/2019., https://loomen.carnet.hr/pluginfile.php/1946392/mod_resource/content/1/DeepLearning.pdf [1. srpanj 2019.]

[5] J. Hui, mAP (mean Average Precision) for Object Detection, https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173 [30. srpanj 2019.]

[6] S. Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [1. srpanj 2019.]

[7] S. Gugger, Convolution in depth, https://sgugger.github.io/convolution-in-depth.html [28. srpanj 2019.]

[8] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks , vol. 13, str. 415 – 425, 7. kolovoz 2002.

[9] R. Gandhi, Support Vector Machine — Introduction to Machine Learning Algorithms, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [2. srpanj 2019.]

[10] X. Changzhen, W. Cong, M Weixin, S. Yanmei, A traffic sign detection algorithm based on deep convolutional neural network, 2016 IEEE International Conference on Signal and Image Processing (ICSIP), str. 676-679, Beijing, China, 13.-15. kolovoz 2016.

[11] ImageNet baza slika, http://www.image-net.org/ [25. srpanj 2019.]

[12] VGG16 - Convolutional Network for Classification and Detection , https://neurohive.io/en/popular-networks/vgg16/ [25. srpanj 2019.]

[13] MatConvNet, Pretrained CNNs, http://www.vlfeat.org/matconvnet/pretrained/ [25. srpanj 2019.]

[14] S. Das, CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more …, https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[15] S. Xu, D. Niu, B. Tao, G. Li, Convolutional Neural Network Based Traffic Sign Recognition System, 2018 5th International Conference on Systems and Informatics (ICSAI), str. 957-961, Nanjing, China, 10.-12. studeni 2018

[16] S. Nayak, Understanding AlexNet, https://www.learnopencv.com/understanding-alexnet/ [25. srpnja 2019.]

[17] Y. Chang, W. Feng, C. Hou-jin, H. Xiao-li, S. Yan, Traffic sign recognition using HOG-SVM and grid search, 2014 12th International Conference on Signal Processing (ICSP), str. 962-965, Hangzhou, China, 19.-23. listopad 2014

[18] K.T. Phu, L.L. Oo, Traffic sign recognition system using feature points, 2018 12th International Conference on Research Challenges in Information Science (RCIS), Nantes, Fance, 29.-31. svibanj 2018

[19] Y. Li, L. Ma, Y. Huang, J. Li, Segment-Based Traffic Sign Detection from Mobile Laser Scanning Dana, IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, str. 4607-4610, Valencia, Spain, 22.-27. kolovoz 2018

[20] C. Wang, Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning, str. 150-152, Changsha, China, 26.-27. svibanj 2018

[21] E. A. Roxas, J. N. Acilo, R. R. P. Vicerra, R. P. Dadios, A. A. Bandala, Vision-based traffic sign compliance evaluation using convolutional neural network, str. 120-123, Chiba, Japan, 13.-17. travanj 2018

[22] YOLO: Real-Time Object Detection, https://pjreddie.com/darknet/yolo/ [5. srpanj 2019.]

[23] ROS official website, https://www.ros.org/ [7. srpanj 2019.]

[24] CARLA, Open-source simulator for autonomous driving research, http://carla.org/ [10. srpanj 2019.]

[25] M. Maj, What is object detection? Introduction to YOLO algorithm, https://appsilon.com/object-detection-yolo-algorithm/ [5. srpanj 2019.]

[26] A. Kathuria, What's new in YOLO v3?, https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b [5. srpanj 2019]

[27] YoloV3 Tiny, https://github.com/pjreddie/darknet/blob/master/cfg/yolov3-tiny.cfg [26. srpanj 2019.]

[28] Mathworks, Exchange Data with ROS Publishers and Subscribers https://www.mathworks.com/help/robotics/examples/exchange-data-with-ros-publishers.html;jsessionid=b22ff99110c254f3e9e3e85f0656 [7. srpanj 2019.]

[29] ROS Documentation, http://wiki.ros.org/Documentation [7. srpanj 2019.]

[30] M. Bjelonic "YOLO ROS: Real-Time Object Detection for ROS", : https://github.com/leggedrobotics/darknet_ros , [10. srpanj 2019.]

[31] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An Open Urban Driving Simulator, Proceedings of the 1st Annual Conference on Robot Learning, str. 1-16, 2017

[32] The KITTI Vision Benchmark Suite, http://www.cvlibs.net/datasets/kitti/ [14. srpanj 2019.]

[33] BelgiumTS Dataset, https://btsd.ethz.ch/shareddata/ [14. srpanj 2019.]

[34] J. Cartucho, OpenLabeling, https://github.com/Cartucho/OpenLabeling [14. srpanj 2019.]

[35] Yolo-v3 and Yolo-v2 for Windows and Linux, https://github.com/AlexeyAB/darknet [17. srpanj 2019]

[36] N. Laskaris, New State-of-the-art in Logo Detection Using YOLOv3 and Darknet, https://platform.ai/blog/page/7/new-state-of-the-art-in-logo-detection-using-yolov3-and-darknet [30. srpanj 2019]

[37] CARLA Ackermann problem s porukama, https://github.com/carla-simulator/ros-bridge/issues/27 [3. rujan 2019.]

SAŽETAK

U ovom diplomskom radu razvijen je algoritam za upravljanje modelom vozila zasnovan na prepoznatim prometnim znakovima. YOLO algoritam korišten je za prepoznavanje određenih prometnih znakova, a parametri algoritma za prepoznavanja prometnih znakova trenirani su na označenim slikama iz kreirane baze podataka određenih prometnih znakova. Programski okvir ROS korišten je za upravljanje modelom vozila u CARLA simulatoru otvorenog koda. Čvor za primanje graničnih okvira detektiranih objekata u spremnik pohranjuje zadnjih 25 detektiranih prometnih znakova i na temelju znaka koji se najviše puta pojavljuje u spremniku šalje se naredba za upravljanje modelom vozila. Rješenje je testirano na slikama prikupljenim iz simulatora na kojima prepoznaje prometne znakove u različitim vremenskim uvjetima. Manji problemi se pojavljuju prilikom detekcije sličnih znakova, ali ti problemi su uspješno riješeni korištenjem spremnika za detekciju.

Ključne riječi: prepoznavanje prometnih znakova, YOLO, ROS, CARLA, detekcija objekata, autonomna vožnja

AUTONOMOUS DRIVING ALGORITHM BASED ON TRAFFIC SIGNS RECOGNITION

ABSTRACT

In this graduate thesis, an autonomous driving algorithm based on traffic signs recognition is developed. For a traffic sign recognition, the YOLO algorithm is used and weights for a traffic sign recognition algorithm are trained with a created database of specific traffic signs. ROS framework is used for controlling a vehicle model in CARLA open source simulator. A node used for receiving bounding boxes of detected objects uses a buffer in which he stores last 25 detected traffic signs and based on a traffic sign which appears the most time in a buffer, a command for controlling vehicle model is sent. The solution is tested on images from real life traffic and images collected from a simulator. The results are satisfying for different weather conditions. Small problems are noticed while detecting similar traffic signs, but these problems are successfully solved using a detection buffer.

Keywords: traffic sign recognition, YOLO, ROS, CARLA, object detection, autonomous driving,

ŽIVOTOPIS

David Mijić rođen je 18. studenog 1995. u Hamm-u, SR Njemačka. U Tenji završava osnovnu školu „Tenja“ te 2010. upisuje I. gimnaziju Osijek. Nakon gimnazije, 2014. godine upisuje preddiplomski studij na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek, smjer računarstvo. Nakon završenog preddiplomskog studija 2017. godine upisuje diplomski studij – Automobilsko računarstvo i komunikacije.

Potpis:

_____

PRILOZI

- P.3.1. – Baza podataka slika trening skupa i validacijskog skupa
- P.3.2. – Konfiguracijske datoteke s postavkama parametara za treniranje algoritma za upravljanje modelom vozila zasnovanog na prepoznatim prometnim znakovima
- P.3.3. – Parametri modela algoritma za upravljanje modelom vozila zasnovanog na prepoznavanju prometnih znakova
- P.4.1. – Baza podataka slika testnog skupa
- P.4.2. – Video sekvence scenarija 1
- P.4.3. – Video sekvence scenarija 2
- P.4.4. – Video sekvence scenarija 3
- P.4.5. – Video sekvence scenarija 4
- P.4.6. – Video sekvence scenarija 5
- P.4.7. – Video sekvence scenarija 6
- P.4.8. – Video sekvence scenarija 7