Algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine

Ljepić, Srđan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila

Diplomski rad

Srđan Ljepić

Osijek, 2019.

Sadržaj

1. Uvod

Automobilska industrija trenutno bilježi ubrzan razvoj gdje je fokus stavljen na automatizaciju što je moguće većeg broja aktivnosti koje bi inače trebao obavljati sam vozač, tj. na razvoj naprednih sustava za pomoć vozaču (engl. ***A**dvanced **D**riving **A**sistance **S**ystems* - ADAS). ADAS sustavi pomoću različitih senzora percipiraju okolinu, te uz određenu sklopovsku i programsku izvedbu obrađuju dobivene podatke i šalju ih nizu aktuatora koji poduzimaju odgovarajuće aktivnosti čime pripomažu vozaču u vožnji ili ga u potpunosti zamjenjuju. Konačni cilj navedene automatizacije procesa vožnje je smanjenje broja prometnih nesreća koje često rezultiraju smrtnim ishodima ili teškim tjelesnim ozljedama osoba uključenih u prometnu nesreću.

Sustavi za pomoć vozaču dijele se na šest osnovnih razina autonomije, gdje nulta razina odgovara sustavu koji ne posjeduje niti jednu automatiziranu funkciju. Prva razina odnosi se na pomoć vozaču prilikom izvršavanja osnovnih operacija, gdje je i dalje potrebna potpuna angažiranost vozača. Druga razina je razina djelomične automatizacije gdje se više sustava za pomoć vozaču mogu izvršavati u isto vrijeme, dok ostatak aktivnosti i dalje obavlja vozač. Razina tri je razina kojoj odgovara trenutni stadij razvoja većine proizvođača unutar automobilske industrije i gdje se omogućava potpuna automatizacija vožnje u određenim uvjetima i situacijama gdje sustav preuzima kontrolu nad vozilom, ali vozač i dalje mora biti spreman za preuzimanje kontrole nad vozilom, ako je to potrebno. Razina četiri se odnosi na automatizaciju vožnje u određenim scenarijima, odnosno situacijama unutar kojih sustav preuzima zadatak u svim aspektima vožnje, ali i dalje postoje određena ograničenja kao što je na primjer zemljopisna lokacija. Konačni cilj automatizacije je razina pet gdje računalni sustav u automobilu u potpunosti preuzima sve moguće operacije u svim situacijama i gdje prisutnost vozača nije niti potrebna.

Razvoj takvih sustava posjeduje brojna ograničenja, kao što su na primjer ograničenost fizičkog prostora za smještanje upravljačkog sustava u automobil, računalnih zahtjeva i slično. Zbog toga vrlo je teško odgovoriti osnovnim zahtjevima, kao što je na primjer vremensko ograničenje izvođenja algoritma, jer pri velikim brzinama i minimalno kašnjenje izvođenja

algoritma koji obavlja neku od kritično sigurnosnih funkcija može rezultirati katastrofalnim posljedicama. Također, sama primjena uvelike ovisi i o okolini u kojoj se izvodi gdje je često potrebno obaviti određene infrastrukturne promjene kako bi se prometnice prilagodile i omogućile što preciznije i brže izvođenje određenih operacija. Osim odgovarajućeg računalnog sustava ADAS sustav treba imati i senzorski sustav kojim percipira okolinu. Senzorski sustav može činiti jedna ili više kamera koje mogu biti montirane na prednjoj strani vozila, unutar samog vozila ili unutar pomoćnih ogledala. Navedene kamere koriste se najčešće za: detekciju vozila, pješaka, uzdužnih oznaka na kolniku i prometnih znakova u toku vožnje, dobivanje pogleda prostora oko automobila, pomoć prilikom parkiranja vozila i brojne slične namjene. Osim kamera često su u primjeni i ultrazvučni senzori, kojima se može procijeniti udaljenost od prepreke zbog čega se najčešće primjenjuju za pomoć prilikom parkiranja vozila ili detekcije prepreke u mrtvom kutu vozila. Također, koriste se i lidar-i (engl. ***Li****ght* ***D****etection* ***A****nd* ***R****anging*) koji mjere udaljenost do površine objekata u okolini vozila tako da ih osvijetle pulsirajućim laserom i mjere vrijeme proteklo između odašiljanja i prijema zrake koja je odbijena od objekta. Zbog većeg dometa i preciznosti mjerenja udaljenosti od ultrazvučnih senzora, lidar se koristi za detekciju i procjenu udaljenosti od pješaka ili vozila, a često se i koristi u sustavu adaptivne kontrole kretanja (engl. ***A****daptive* ***C****ruise* ***C****ontrol – **ACC***). Osim lidara koristi se i radar (engl. ***RA****dio* ***D****etection* ***A****nd* ***R****anging*) koji za mjerenje udaljenosti od objekta koristi radiovalove. Radar se koristi za slične primjene kao i lidar, ali se razlikuje od lidara zbog toga što ima veći domet i manju osjetljivost na vremenske uvjete, ali ima manju razlučivost.

Jedna od zadaća koju je potrebno ispuniti prilikom autonomnog kretanja vozila je detekcija vozne trake na kolniku, gdje je, zahvaljujući preciznoj detekciji, moguće zadržavati vozilo unutar vozne trake i planirati buduće kretanje. Da bi se taj zadatak uspješno izvršio potrebno je dohvatiti sliku s kamera montiranih na prednjoj strani vozila. Izdvajanje oznaka na kolniku na temelju takve slike nije trivijalan zadatak i potrebno je koristiti različite tehnike obrade slike kako bi se ispravno izdvojili dijelovi koji sadrže oznake na kolniku. Također, vozilo treba pravovremeno detektirati objekt (npr. pješak ili vozilo) unutar vozne trake unutar koje se kreće te procijeniti udaljenost od potencijalne prepreke, kako bi dalje u planiranju kretanja bilo moguće prilagoditi brzinu

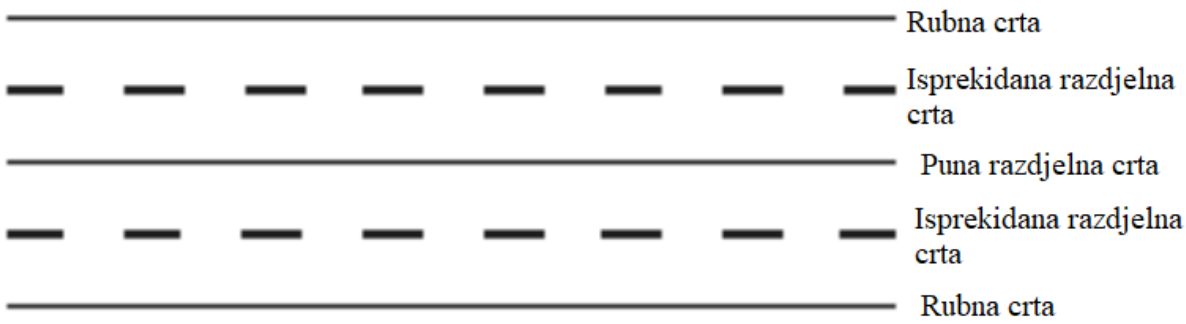
automobila ovisno o udaljenosti prepreke i kako bi se u konačnici automobil uspješno zaustavio ako je to potrebno.

U okviru ovog diplomskog rada osmišljen je algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila. Algoritam kao ulaz koristi sliku dobivenu pomoću optičkog senzora, odnosno kamere, gdje se obavlja niz različitih operacija kako bi se pronašla i prikazala vozna traka kojom se vozilo kreće. Navedeno područje se zatim pretražuje kako bi se ustvrdilo postoji li nekakva prepreka i ako postoji algoritam procjenjuje udaljenost do iste. Za razvoj navedenog algoritma korišten je programski jezik C++ te biblioteka *OpenCV*. Rješenje je implementirano unutar robotskog operacijskog sustava (engl. ***R****obot* ***O****perating* ***S****ystem* – ROS) i verificirano unutar *Carla* simulatora. Prilikom verifikacije algoritma kao prepreke su korišteni automobili i pješaci.

U drugom poglavlju predstavljen je opis problema detekcije prepreka unutar vozne trake i prilagodbe brzine vozila. Također je dan pregled postojećih rješenja, načina na koji su ostvareni i njihove učinkovitosti. Opisani su algoritmi koji se oslanjaju na primjenu neuronskih mreža kako bi ispunili zadatak, ali i algoritmi koji su dobiveni klasičnim metodama u području računalnog vida. U trećem poglavlju predstavljeno je vlastito rješenje algoritma za detekciju prepreke unutar vozne trake i prilagodbe brzine vozila. Korak po korak prikazan je cijeli postupak od dohvaćanja video okvira sve do konačnog izlaza koji predstavlja prikaz detektirane vozne trake, procijenjene udaljenosti do prepreke ako prepreka postoji i procesa kočenja kako bi se vozilo zaustavilo pred detektiranom preprekom. Uz to opisana je *OpenCV* biblioteka, kao i ROS okruženje koji su korišteni za implementaciju algoritma i *Carla* simulator koji je korišten prilikom verifikacije algoritma. Unutar četvrtog poglavlja predstavljena je verifikacija rada rješenja, uz prezentirane rezultate rješenja pomoću kojih je moguće doći do zaključka danog u petom poglavlju.

2. Detekcija vozne trake i prepreka unutar vozne trake

Algoritmi za detekciju vozne trake i prepreke unutar vozne trake zasnivaju se na uzastopnoj obradi slika koje se dobivaju s kamere montirane na prednjoj strani vozila. Detekcija vozne trake predstavlja prepoznavanje područja na takvim slikama koje odgovara voznoj traci dok nam detekcija prepreke daje odgovor na pitanje postoji li unutar vozne trake prepreka. Detekcija vozne trake se najčešće zasniva na prepoznavanju uzdužnih oznaka na kolniku [1]. Uzdužne oznake na kolniku mogu biti razdjelne ili rubne. Rubne oznake predstavljaju krajnje desno ili krajnje lijevo pozicionirane oznake na kolniku koje predstavljaju sami rub kolnika i prelaskom kojih vozilo silazi s kolnika. Razdjelne oznake na kolniku koriste se za razdvajanje kolničkih površina prema smjerovima kretanja. Kolničke crte čine sve kolničke oznake koje pripadaju istoj rubnoj ili isprekidanoj crti. Podjelu oznaka na kolniku moguće je vidjeti na slici 2.1.



Slika 2.1. Rubne i razdjelne uzdužne kolničke oznake [1].

Vozilu najbliža lijeva i najbliža desna kolnička oznaka zajedno čine voznu traku kojom se vozilo kreće. Osnovni problem prilikom pronalaska kolničkih oznaka je velika količina ostalih informacija, takozvanih šumova koji su prisutni u gotovo svakom okviru ulaznog videa, a koji ne pripadaju željenom izlazu, tj. elementima slike koji pripadaju kolničkim oznakama koje je potrebno detektirati. Također, potrebno je utvrditi semantičku vezu između povezanih segmenata isprekidane ili rubne crte kako se ne bi povezivale kolničke oznake koje pripadaju različitim crtama. Kada su dobivene lijeva i desna kolnička crta, tada se područje između njih definira kao vozna traka kojom se kreće vozilo. Tada se područje vozne trake koristi za pronalazak prepreka

ako postoje na putanji automobila. Ako ta prepreka postoji, vrlo je važno odrediti na kojoj je udaljenosti od vozila kako bi se vozilo moglo zaustaviti na vrijeme ispred prepreke bez ugrožavanja bilo kojeg sudionika prometa. Taj problem postaje vrlo složen kada se radi o većim udaljenostima, gdje je prepreku vrlo teško uočiti.

2.1. Postojeći algoritmi detekcije vozne trake i prepreka na putu

Detekcija uzdužnih kolničkih oznaka, odnosno vozne trake, kao i detekcija prepreka na putu neizostavne su funkcije računalnih sustava autonomnih vozila. Povijest razvoja i korištenja navedenih sustava seže do 1990-tih godina kada su korišteni relativno primitivni računalni sustavi, gdje bi se vizualnim ili zvučnim signalima upozorilo vozača ako bi došlo do prelaska neke od kolničkih oznaka vozne trake kojom se vozilo to tada kretalo. Često su u upotrebi i sustavi koji prate i susjedne vozne trake i obavještavaju vozača kada se može sigurno prestrojiti. U današnjoj proizvodnji postoje i vozila koja navedenu funkciju ispunjavaju tako da osim kamere kombiniraju informacije dobivene s drugih senzora kao što su npr. radari, lidari ili ultrazvučni senzori [2]. Velika prednost prilikom izvedbe sustava za prepoznavanje kolničkih oznaka i prepreka pomoću kamere je znatno manja cijena u odnosu na sustave koji koriste i ostale senzore, naročito ako se koristi lidar. Veliki pomak u razvoju sustava je nastao 2015. godine kada je američka kompanija Tesla pustila u pogon potpuno autonomno vozilo, međutim, zbog pojedinih prometnih nesreća koje su završile smrtnim ishodom, vidljivo je da je potrebno dodatni fokus staviti na sigurnost navedenih sustava.

U okviru ovog diplomskog rada za dohvaćanje informacija iz okoline vozila koriste se isključivo kamere koje su montirane na prednjoj strani vozila. Kamere omogućavaju dohvaćanje ogromne količine podataka, gdje današnje prosječne kamere mogu snimati rezolucijom od 1920×1080 elemenata slike što daje više od dva milijuna elemenata slike po video okviru. Zbog toga, sama obrada slika dobivenih pomoću takvih kamera je vrlo složen zadatak u odnosu na obradu podataka koje bilježe radari ili lidari. Međutim, prilikom detekcije kolničkih oznaka kamere se koriste zbog toga što je na podacima zabilježenim video kamerom moguće razlikovati

boje što olakšava detekciju kolničkih oznaka koje su uvijek bijele ili žute boje. U literaturi se mogu prepoznati dva ključna smjera istraživanja. Prvi smjer zasnovan je na klasičnim algoritmima iz domene računalnog vida i obrade slike, dok je drugi smjer zasnovan na primjeni strojnog učenja, odnosno dubokih neuronskih mreža za obradu slike.

2.2.1 Detekcija uzdužnih kolničkih oznaka i prepreka na cesti pomoću računalnog vida

Detekcija uzdužnih kolničkih oznaka kompleksan je zadatak transformacije ulaznog okvira snimljenog pomoću kamere fiksirane na vozilo u izlaz unutar kojega su sadržane koordinate prisutnih kolničkih oznaka. Prvi smjer istraživanja zasnovan je na metodama i algoritmima računalnog vida, odnosno obrade slike i videa. Jedan od značajnijih radova tog smjera istraživanja je rad [3] gdje su autori predstavili jednostavan algoritam za detekciju kolničkih oznaka unutar stvarnog vremena. Ulaz u algoritam je video okvir dobiven kamerom fiksiranom na vozilo. Prvi korak algoritma je transformacija slike u takozvani ptičji pogled što se postiže metodom inverznog mapiranja. Osim toga u procesu predobrade korišten je i postupak dvodimenzionalnog filtriranja pomoću *Gaussovog* filtra kako bi se dodatno filtrirao šum. Nadalje, korištena je metoda *Houghove* transformacije koja može izdvojiti specifičan geometrijski oblik koji se nalazi na slici, a koja se najčešće primjenjuje za detekciju ravnih linija, elipsi ili krugova. Osim *Houghove* transformacije korištena je i RANSAC (engl. ***RAN****dom* ***SA****mple* ***C****onsensus*) metoda. RANSAC koristi za ulaz prisutne elemente slike, te nakon određenog broja iteracija daje pravac koji najbolje opisuje ulazne elemente slike kako bi se dobili pravci koji predstavljaju kolničke oznake. U procesu naknadne obrade izlaznih vrijednosti dobivenih kombinacijom *Houghove* transformacije i RANSAC algoritma odbacuju se pravci koji ne zadovoljavaju određene geometrijske kriterije te se za izlaz dobiju pravci koji su procjena pozicije kolničkih oznaka prisutnih unutar scene. Algoritam je testiran od strane autora nad različitim video materijalima koji uključuju različite scenarije urbane vožnje, s ravnim i zakrivljenim kolničkim oznakama. Navedena metoda uspješno detektira kolničke oznake u 96.34% slučajeva kada na kolniku postoje dvije oznake, odnosno u 90.89% slučajeva kada postoje četiri oznake. Međutim, prilikom primjene algoritma pojavljuje se veliki

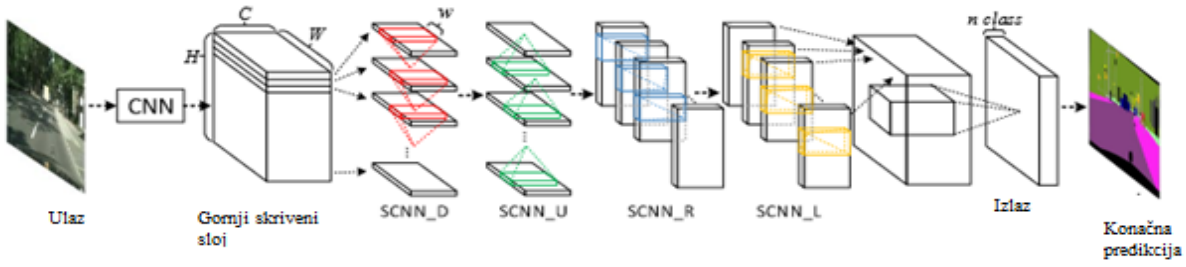
broj lažno pozitivnih segmenata (engl. *false positive*). Unatoč tomu, algoritam je postigao značajno bolje rezultate u usporedbi s ostalim sličnim algoritmima koji su razvijeni prošlog desetljeća, a također je vrlo važan jer je omogućio primjenu u stvarnom vremenu zbog svoje jednostavne izvedbe.

U radu [4] predložena je metoda koja koristi RANSAC algoritam za detekciju uzdužnih kolničkih oznaka. Zbog distorzije objektiva kamere, prvo se obavlja kalibracija kamere. Kada je video okvir uspješno dohvaćen, tada se pretvara u tonove sive boje. Takva slika koristi se kao ulaz za detekciju rubova pomoću *Canny* operatora, gdje se kao izlaz dobiva binarna slika s detektiranim rubovima. Sljedećim korakom vrši se transformacija perspektive. Kada je dobivena slika u ptičjoj perspektivi, provodi se morfološka obrada binarne slike. Koristi se tzv. zatvaranje (engl. *closing*) koje se sastoji od dva osnovna koraka, dilatacije i erozije binarne slike. Prilikom izvođenja tih operacija povezuju se nepovezane regije koje se nalaze unutar slike, što za cilj ima spajanje svih dijelova pojedinog objekta. Ovo se pokazalo kao vrlo korisno prilikom detekcije kolničkih oznaka ako se na video okvirima pojavljuju isprekidane kolničke oznake. Nakon toga svaka slika se dijeli na dvije regije kako bi se razdvojile lijeva i desna kolnička oznaka. Navedena operacija obavlja se pomoću metode grupiranja „K najbližih susjeda“ (engl. *k-means*), gdje se računaju elementi slike koje pripadaju jednom od dva definirana centara točaka na osnovu dobivenog 1-D histograma boja. Nadalje, RANSAC se koristi za dobivanje pravaca ili parabola koje predstavljaju kolničke oznake budući da je robustan na prisustvo stršećih vrijednosti. Zadnji korak algoritma je primjena *Gauss-Newtonove* metode kako bi se izračunali parametri jednadžbe pravca ili kružnice kojima se prikazuju detektirane uzdužne kolničke oznake. Navedeno rješenje implementirano je i testirano na različitim video materijalima snimljenim kako za vrijeme dnevne, tako i noćne vožnje. Rezultati su pokazali da navedena metoda uspješno detektira sve prisutne kolničke oznake u 90.9% slučajeva, što je znatno veći postotak od do tada postignutih rezultata sličnih metoda koje koriste RANSAC algoritam.

U radu [5] su predstavljene tehnike koje se koriste kako za praćenje vozne trake unutar kolnika, tako i za detekciju prepreka koje se nalaze unutar detektirane vozne trake. Navedeni rad oslanja se na sustav koji se zasniva na korištenju "*fish-eye*" kamera i lidar senzora. Algoritam se sastoji od dva osnovna koraka, gdje je prvi korak detekcija kolničkih oznaka. Unutar tog koraka se detektiraju rubne točke kolničkih oznaka te se vrši geometrijski proračun pozicije kolničkih oznaka unutar dvodimenzionalnog koordinatnog sustava u odnosu na poziciju vozila. To se obavlja tako da se najprije ukloni prisutni šum koji se nalazi na video okvirima kako bi se uklonili segmenti koji mogu utjecati na preciznost detekcije rubova kolničkih oznaka. Nakon obavljenog filtriranja, sljedeći korak je detekcija rubova kako bi se dobila binarna slika. Unutar rada korišteni su različiti operatori detekcije rubova, kao što su *Canny*-jev, *Prewitt*-ov i *Sobel*-ov operator. Na navedenu binarnu sliku se zatim primjenjuje *Houghova* transformacija koja se primjenjuje za detekciju pravaca koji predstavljaju kolničke oznake. *Houghova* transformacija kao izlaz daje veći broj različitih pravaca, gdje je potrebno odvojiti lijevu i desnu kolničku oznaku i onda predstaviti ih s po jednim pravcem od svih detektiranih. Drugi korak algoritma je detekcija i izbjegavanje prepreka koje se nalaze unutar detektirane vozne trake. Detekcija prepreke, odnosno procjena udaljenosti prepreke u voznoj traci se obavlja pomoću lidar-a. Navedene su i prednosti korištenja lidar-a, kao što su velika preciznost i brzina izvođenja detekcije udaljenosti od prepreke. Autori su testirali navedeni pristup snimajući video materijale pomoću "*fish-eye"* kamera montiranih na testnu maketu automobila. Prilikom obrade, svaki video okvir podijeljen je po visini na dva dijela kako bi se dio koji je bliži samom vozilu koristio za obradu slike i pronalazak kolničkih oznaka. Stoga, sustav je testiran unutar strogo kontroliranog okruženja gdje su simulirani stvarni prometni scenariji i gdje je navedeni algoritam uspješno pratio voznu traku, ali i uspješno reagirao na prepreku uočenu unutar iste.

2.2.2. Detekcija vozne trake i/ili prepreka unutar vozne trake pomoću konvolucijskih neuronskih mreža

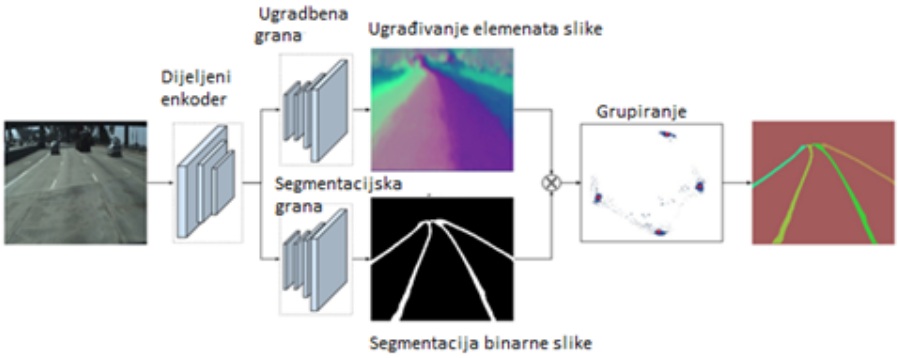
State-of-the-art metode koje se koriste za detekciju vozne trake ili kolničkih oznaka i prepreka unutar voznih traka većinom su zasnovane na korištenju konvolucijskih neuronskih mreža. U radu [6] autori su predložili prostornu konvolucijsku mrežu koja se koristi za segmentaciju različitih slika snimljenih u prometu. Konvolucijske mreže su se pokazale vrlo korisne prilikom uočavanja semantike različitih objekata u različitim situacijama. Navedeno rješenje može prepoznati četiri kolničke oznake, dvije koje predstavljaju voznu traku gdje je vozilo pozicionirano, te dvije koje se nalaze u susjednim trakama, gdje je svaka traka predstavljena posebnom klasom. Za ostvarenje navedene mreže korišten je veliki broj video materijala koje su autori prikupili koristeći dostupne video sekvence na internetu, kao i video materijali koje su autori sami snimili. Korištene su scene koje su snimljene u urbanim područjima, ruralnim područjima, kao i scene snimljene na autocesti. Zahvaljujući navedenoj metodi riješen je vrlo čest problem kada slika dobivena s kamerom montiranom na prednjoj strani vozila ne sadrži kompletne kolničke oznake jer su zaklonjene od strane drugih vozila. U tom je slučaju potrebno procijeniti pozicije kolničkih oznaka iz samog konteksta slike. Na slici 2.2. moguće je vidjeti izgled navedene prostorne konvolucijske mreže.



Slika 2.2. Prikaz prostorne konvolucijske mreže korištene u radu [6].

Na slici 2.2. moguće je vidjeti da se ulazna slika najprije transformira u trodimenzionalni tenzor dimenzija C × H × W gdje navedene vrijednosti predstavljaju broj kanala (C), redaka (H) i stupaca (W). Tenzor se tada dijeli na H dijelova, gdje se svaki koristi kao ulaz u konvolucijsku mrežu koja posjeduje C kernela veličine C × w gdje je w širina kernela. Za razliku od većine ostalih rješenja koje koriste konvolucijske mreže i gdje se izlaz iz konvolucijskog sloja prosljeđuje sljedećem sloju, unutar navedene metode izlaz iz jednog sloja konvolucijske mreže se na nadodaje na sljedeći dio iz skupa H. Tako nastaje ulaz u sljedeći sloj neuronske konvolucijske mreže i taj proces se nastavlja sve dok se i posljednji dio skupa H ne iskoristi kao ulaz u sloj konvolucijske neuronske mreže. Također, koristi se nelinearna aktivacijska funkcija *ReLU*. Navedena metoda postigla je bolje rezultate od često korištenih neuronskih mreža kao što su ReNet [7], ResNet [8], DenseCRF [9] i sličnih, gdje je zamijećen napredak unutar gotovo svih kategorija pripadajućih scenarija i trenutno predstavlja najpouzdaniju *state-of-the-art* metodu koja se koristi za detekciju kolničkih oznaka. Algoritam je testiran pomoću 1525 slika prikupljenih iz *Cityscapes* skupa podataka, gdje je u 96.53% slučajeva metoda ispravno detektirala sve oznake na kolniku.

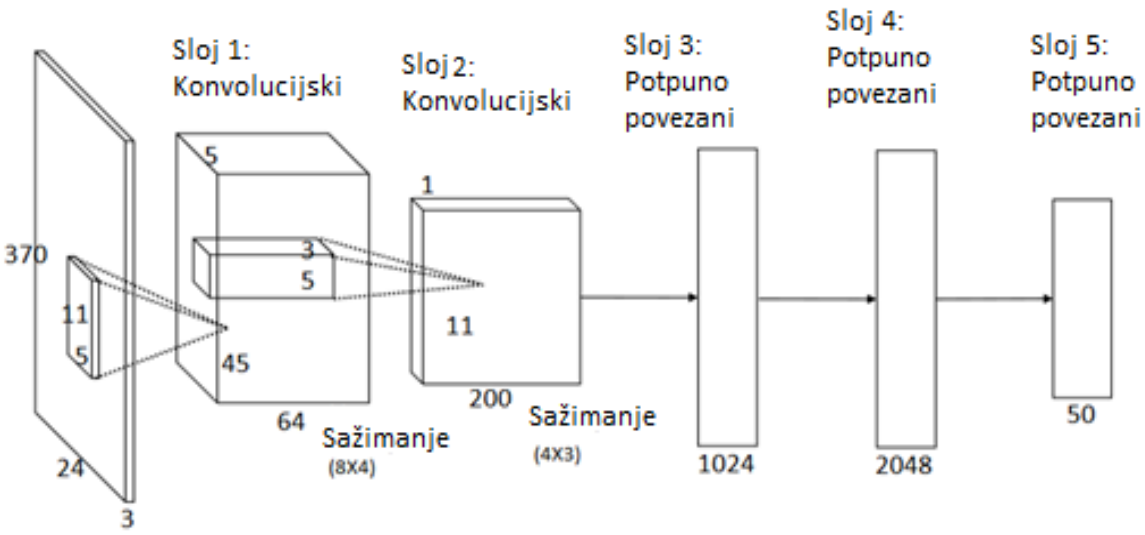
U radu [10] predstavljena je konvolucijska mreža *LaneNet* koja je korištena za segmentaciju video okvira kako bi se detektirala vozna traka kojom se vozilo kreće. Za ulaz se koristi video okvir dohvaćen pomoću kamere montirane na prednjoj strani vozila. Arhitektura mreže se sastoji od dvije „grane“ gdje se jedna grana koristi za segmentaciju slike i koja je „istrenirana“ kako bi se proizvela binarna slika unutar koje su označene procjene kolničkih oznaka pomoću elemenata slike koji su predstavljeni bijelom bojom i gdje je okolina prezentirana elementima slike crne boje. Druga grana koristi se za takozvano ugrađivanje elemenata slike (engl. *pixel embbeding*), gdje se iznosi elemenata slike koji pripadaju istoj voznoj traci prikazuju istim vrijednostima, dok su druge trake prikazane različitim vrijednostima. Tako se maskiraju pozadinski elementi slike koristeći binarnu segmentacijsku mapu koja se koristi unutar segmentacijske grane. Na slici 2.3. moguće je vidjeti arhitekturu *LaneNet* konvolucijske mreže.



Slika 2.3. Arhitektura *LaneNet* mreže [10].

Autori su navedenu metodu testirali pomoću slika iz *tuSiple* skupa podataka koji predstavlja gotov skup podataka za treniranje i testiranje neuronskih mreža koje služe za detekciju oznaka na kolniku. Za testiranje autori su koristili 2782 slike gdje je metoda u 96.4% uspješno detektirala sve prisutne oznake na kolniku.

U okviru rada [11] autori su predložili metodu koja se koristi za detekciju prepreka unutar kolničke trake pomoću metode dubokog učenja i konvolucijske mreže *StixelNet.* Navedeni algoritam bavi se zadatkom segmentacije scene gdje je potrebno svakom elementu slike dodijeliti klasu objekta kojemu pripada kako bi se odvojili kolnik, pješaci ili vozila od oznaka na cesti ili okoline prisutne na video okviru. Algoritam razvijen unutar navedenog rada predstavlja doradu postojećih rješenja [12] detekcije prepreka pomoću elemenata slike koji predstavljaju prepreke i koji se nazivaju *stixel*-i, gdje se za razliku od prethodnih radova koristi samo jedna kamera. Video okviri dobiveni pomoću kamere se podijele po širini na stupce, i svaki stupac predstavlja ulaz u algoritam. Pri tome za svaki stupac se računaju takozvani *stixel*-i koji se dobiju tako da se pronađe najniži piksel koji odstupa od teksture kolnika. Tada se koristi metoda regresije kako bi se iz dobivenih *stixel*-a dobio model kojim se opisuju detektirane prepreke. Taj postupak se obavlja tako da se koristi neuronska mreža *StixelNet* prikazana na slici 2.4. Prilikom treniranja navedene mreže predani su različiti stupci slika s označenim pozicijama prepreke. Kako bi se „istrenirala“ konvolucijska mreža autori su predstavili novu funkciju koja se bazira na polu-diskretnoj reprezentaciji vjerojatnosti o točnoj poziciji prepreke koristeći sigmoidnu funkciju.



Slika 2.4. Arhitektura *StixelNet* konvolucijske mreže [11].

Kada je obavljena regresija, dobivena je mapa vjerojatnosti pozicije prepreke. Tada se obavljaju dva koraka. Prvi korak predstavlja računanje pozicije svakog elementa slike u odnosu na susjedne stupce. Drugi korak se obavlja pomoću uvjetnih nasumičnih polja (engl. ***C****onditional* ***R****andom* ***F****ields*) [13] pomoću kojih je dobivena segmentacija slike gdje su odvojeni kolnik i prepreke unutar vozne trake. *StixelNet* konvolucijska mreža testirana je pomoću *KITTI* baze gdje su kolničke oznake uspješno detektirane u 81.23% slučajeva, što je predstavljalo pomak u odnosu na pojedine konvolucijske mreže do tada predložene za segmentaciju prometne scene.

3. Algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila

Predloženi algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila sastoji se od više koraka. Kao ulaz u algoritam koriste se video okviri zabilježeni unutar simulatora *Carla* (engl. ***Car Learning to Act***) pomoću kamere montirane na prednjoj strani vozila. Dohvaćanje video okvira obavlja se pomoću robotskog operacijskog sustava (eng. ***Robotic Operating System*** - *ROS*). Na navedenu ulaznu sliku primijenjene su metode predobrade, kao što su inverzna transformacija perspektive, filtriranje prema boji, *Gauss*-ovo filtriranje i *Canny* detekcija rubova. Navedena predobrada obavljena je pomoću funkcija biblioteke *OpenCV (**Open Source Computer Vision library***). Nakon što su navedene metode primijenjene, dobivena je binarna slika čiji se elementi slike koriste za ulaz u RANSAC algoritam. RANSAC algoritam korišten je kako bi se iz predobrađene slike dobile jednadžbe pravaca kojima su estimirane kolničke oznake koje pripadaju voznoj traci unutar koje se nalazi vozilo. Tada je izdvojeno područje koje je omeđeno navedenim pravcima, odnosno vozna traka. Zatim se provodi pretraživanje vozne trake gdje su obavljeni postupci kojima se klasificira prepreka kao vozilo ili pješak te se računa udaljenost od iste, ako se nalazi na udaljenosti do 80 metara. Posljednji korak obavlja se pomoću robotskog operacijskog sustava tako što se vozilu šalje signal da počne kočiti sve dok se ne zaustavi.

3.1. Korišteni programski alati

3.1.1. OpenCV biblioteka

OpenCV je biblioteka otvorenog koda koja se primjenjuje u području računalnog vida, kao i u području strojnog učenja. Unutar biblioteke postoji preko 2500 optimiziranih algoritama koje je moguće na jednostavan način koristiti prilikom obrade slike. Takvi algoritmi se mogu koristiti za različite namijene, kao što su na primjer detekcija i prepoznavanje lica, identifikacija objekata,

klasificiranje i predviđanje kretanja objekata ili ljudi, izvlačenje 3D modela objekata, kreiranje 3D oblaka točaka pomoću kamere, spajanje više slika unutar jedne i slično [14].

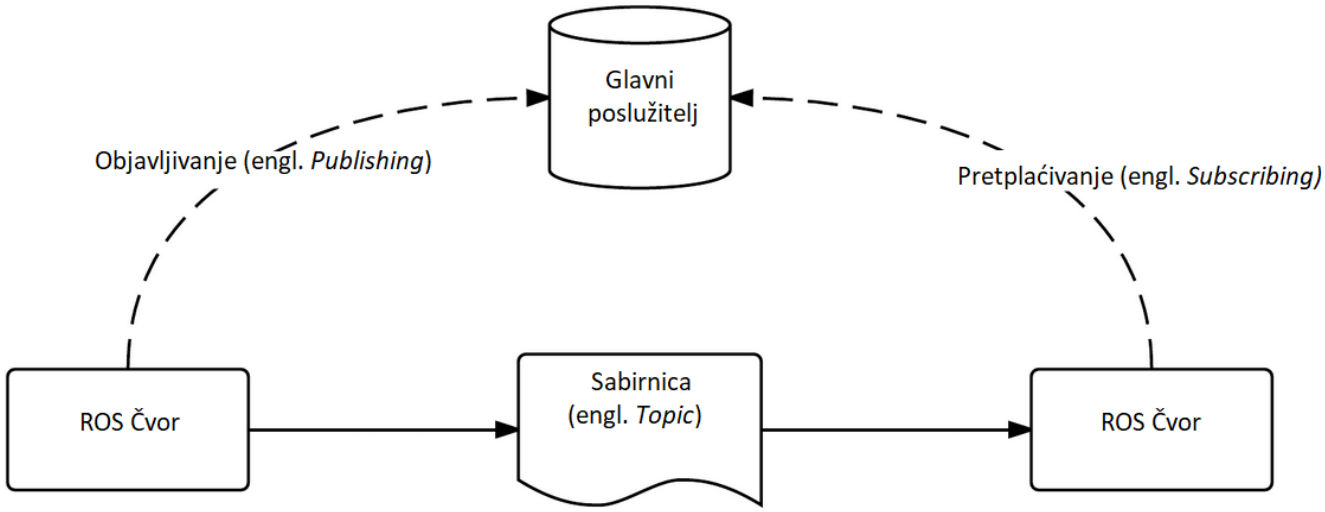
U radu je korištena verzija *OpenCV* 3.4.3 koja je instalirana na operacijskom sustavu Linux Ubuntu 16.04. Algoritam predložen u ovom radu je izveden korištenjem tipova podataka, kao i različitih funkcija koje su definirane unutar *OpenCV* biblioteke. Prilikom kreiranja bilo koje nove slike unutar algoritma koristi se instanca klase `cv::Mat` unutar koje je moguće definirati veličinu i tip slike gdje se spremaju brojevne vrijednosti unutar matrice dimenzije određenog broja elemenata slike po širini i visini. Također, postoji i veći broj formata kojima je moguće prikazati sliku, gdje se prilikom navođenja definira broj bitova, tip podataka svakog elementa i broj kanala, tako da npr. `CV_32UC3` predstavlja 32-bitni neoznačeni cjelobrojni tip, a definirana su i tri kanala. Također, korišteni su i osnovni tipovi podataka koji se koriste za reprezentaciju elementa slike na slici. Element slike se zapisuje kao vektor koji sadrži dvije vrijednosti, poziciju na *x*-osi slike, i poziciju na *y*-osi slike pri čemu je ishodište koordinatnog sustava u gornjem lijevom kutu. Različit broj ugrađenih funkcija također je korišten prilikom izrade algoritma. Pomoću bitovnih operacija moguće je primijeniti neku od logičkih operacija kao što su I, ILI, XILI, NE… Navedenoj funkciji se kao parametri prilažu dvije ulazne slike gdje se navedena operacija vrši za sve elemente slike koji se nalaze na identičnim koordinatama. Bitovne operacije korištene su npr. prilikom izdvajanja detektirane kolničke oznake pomoću RANSAC algoritma. Također, korištena je funkcija `cvtColor`, koja omogućava pretvorbu slike iz jednog prostora boja u drugi. Prilikom vizualizacije slike, korištene su gotove funkcije za prikaz detektiranih kolničkih oznaka. Osim navedenih osnovnih operacija, korištene su i dodatne funkcije ugrađene unutar *OpenCV* biblioteke, kao što su one pomoću kojih je moguće izvesti filtriranje *Gaussovim* filtrom, detekciju rubova *Canny*-evim operatorom ili transformaciju perspektive.

3.1.2. Robotski Operacijski Sustav (ROS)

Robotski operacijski sustav predstavlja razvojni okvir koji se vrlo često koristi prilikom razvoja programske podrške u području robotike. Preciznije, ROS predstavlja skup alata, API-ja i biblioteka koji se koriste kako bi se olakšao zadatak kreiranja i implementacije složenih upravljačkih algoritama robota ovisno o njihovima različitim zadaćama, kao i ovisno o platformama koje se koriste. ROS se sastoji od više dijelova, pri čemu su osnovni dijelovi skup upravljačkih programa, zbirka osnovnih algoritama, skup alata za vizualizaciju i ROS ekosustav. Pomoću upravljačkih programa obavlja se čitanje podataka sa senzora, kao i slanje naredbi aktuatorima. Postoji veliki broj sklopovskih uređaja koji su kompatibilni s robotskim operacijskim sustavom, a samim time široka je i rasprostranjenost čak i komercijalnih robotskih sustava koji koriste robotski operacijski sustav [15]. Dostupni veliki skup alata omogućava povezivanje različitih programskih komponenti robota i ugradnju vlastitog algoritma. Omogućena je vizualizacija stanja robota, ispravljanje pogrešnog ponašanja robota, kao i pogrešno snimljenih senzorskih podataka. Ekosustav ROS-a predstavlja opsežan skup resursa, koji omogućava dokumentiranje ROS okvira. Uz navedeno robotski operacijski sustav je široko korišten jer podržavaj veliki broj programskih jezika kao što su Python, Ruby, LISP, MATLAB, ali najčešće se koristi programski jezik C++, koji od navedenih pruža najbolje performanse. Osim toga, ROS je besplatan i javno otvoren, a potiče se i izrada samostalnih biblioteka što omogućava upotrebu razvijenih algoritama izvan ROS-a, kao i ubrzavanje i pojednostavljivanje izrade automatiziranih testova.

Osnovni koncepti ROS razvojnog okvira su poruke (engl. *messages*) koje su ostvarene pomoću podatkovnih struktura, gdje se navode tipovi elemenata strukture, kao i nazivi varijabli koje se šalju unutar poruke. Poruke se razmjenjuju između takozvanih čvorova (engl. *nodes*), gdje svaki ROS čvor predstavlja zasebni proces. Svaki čvor obavlja svoju samostalnu operaciju koje se međusobno povezuju pomoću svojevrsnih kanala, odnosno sabirnica ili tema (engl. *topics*) koje omogućavaju razmjenu različitih poruka između čvorova. Protok poruka se obavlja isključivo u jednom smjeru od pošiljatelja prema primatelju, a više pošiljatelja može istodobno slati poruke,

dok također može biti i više primatelja iste poruke. Komunikacija se temelji na modelu poslužitelja (engl. *publisher*) i klijenta (engl. *subscriber*). Kako bi se omogućila ispravna komunikacija definiraju se dvije poruke, gdje se jedna koristi kao zahtjev i koju šalje klijent, a druga se koristi kao odgovor poslužitelja klijentu. Stoga, svaki poslužiteljski čvor nudi uslugu (engl. *service*) pod određenim imenom. Klijent tada šalje zahtjev kako bi iskoristio pruženu uslugu. Tako se svakom čvoru omogućava komunikaciju s drugim čvorovima. Format datoteke unutar kojega se spremaju podaci koji se šalju predstavlja se unutar takozvanog spremnika (engl. *bag*). Navedenim mehanizmom moguće je pohraniti podatke koji su dobiveni s različitih senzora iz okoline i na osnovu kojih se određuje daljnje ponašanje robota. Kako bi se omogućila komunikacija između čvorova, usluga i poruka, koristi se glavni poslužitelj (engl. *ROS master*). Navedeni glavni poslužitelj koristi se često unutar distribuiranih sustava, a sva komunikacija između članova pojedine mreže može se izvršiti na glavnom poslužitelju [16]. Jednostavni prikaz komunikacije između dva čvora pomoću sabirnice za prijenos poruka prikazan je na slici 3.1.



Slika 3.1. Način prijenosa poruke između dva ROS čvora [17].

Za izvedbu navedenog algoritma korištena je *Kinetic Kame* distribucija, koja je deseta ROS distribucija puštena u pogon sredinom 2016. godine i koja je instalirana na Linux Ubuntu 16.04 distribuciji.

3.1.3. Carla simulator

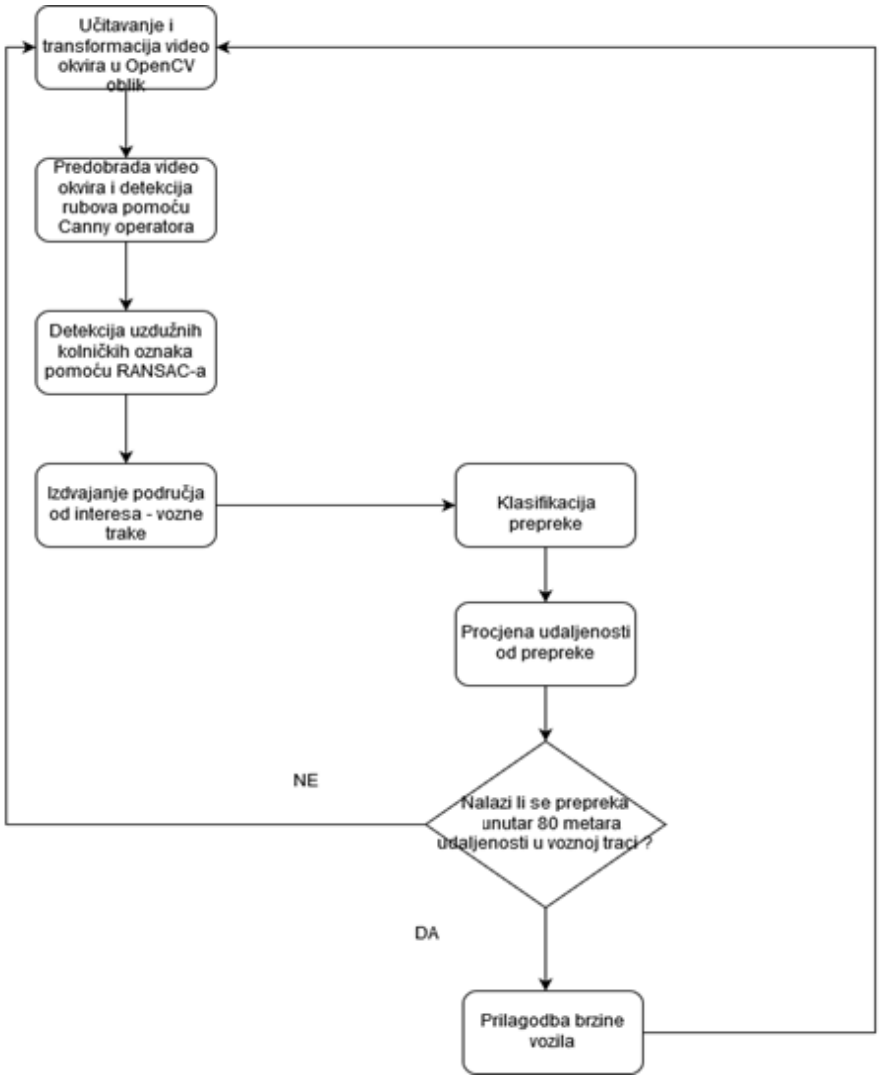
Prije nego pojedini sustav autonomne vožnje uspije zaživjeti u širokoj upotrebi, takav sustav je potrebno vrlo detaljno testirati. Testiranje sustava u stvarnim uvjetima vrlo je skupo i rizično, stoga su razvijeni različiti simulatori koji nastoje što realističnije prikazati stvarne uvjete koje je moguće sresti prilikom svakodnevne vožnje. Prilikom evaluacije predloženog rješenja korišten je simulator *Carla*. *Carla* je simulator otvorenog koda koji se koristi za razvoj i validaciju sustava autonomne vožnje. Simulator pruža mogućnost testiranja vožnje u različitim uvjetima, gdje je dostupno nekoliko predefiniranih mapa koje pokrivaju niz različitih scenarija. Stoga, moguće je testirati sustav autonomne vožnje kako u urbanom, tako i u ruralnim okruženju. Moguće je simulirati promet koji sadrži različita vozila koja se kreću unutar prometnica, a moguće je uključiti i pješake u promet. Također, unutar Carla simulatora postoji veliki broj senzora, kao što su RGB kamera, stereo kamera, kamera koja vrši semantičku segmentaciju scene, lidar, ugrađeni sustav koji upozorava na koliziju s ostalim objektima, sustav koji signalizira prelazak neke od kolničkih oznaka vozne trake kojom se vozilo kreće, kao i sustav kojim se omogućava detekcija i procjena udaljenosti od prepreke. Osim toga, Carla simulator pruža niz različitih vremenskih uvjeta unutar kojih je moguće testirati sustave autonomne vožnje, kao što su različita razdoblja dana, uključujući svitanje, podne i sumrak, ali i ista razdoblja dana prilikom kišnog vremena s odabirom suhog ili vlažnog kolnika. Prilikom testiranja odabire se jedno vozilo kojim je moguće upravljati na različite načine. Moguće je definirati kut pod kojim vozilo skreće, silu koju koristi prilikom kočenja kao i ubrzavanje vozila. Unutar simulatora vrlo su detaljno razrađeni fizikalni zakoni kako bi se maksimizirala realnost simulacije i kretanja vozila i pješaka. Implementiran je kao dodatni sloj otvorenog koda zasnovan na *Unreal Engine 4* programskim alatima koji su besplatno dostupni za nekomercijalno korištenje. Također, osim trodimenzionalnih modela različitih vozila i pješaka, dostupni su i različiti trodimenzionalni modeli statičnih objekata kao što

su zgrade, vegetacija, prometni znakovi, infrastruktura i slično. Također, *Carla* simulator je kompatibilan s robotskim operacijskim sustavom jer posjeduje takozvani ROS most pomoću kojeg je moguće koristiti robotski operacijski sustav kako bi se upravljalo vozilom. Pomoću dostupnih kamera moguće je unutar stvarnog vremena dohvatiti video okvire koji se potom obrađuju i na osnovu njih moguće je dalje planirati kretanje vozila. Kako bi se navedeni simulator uspješno instalirao potrebno je pratiti uputstva priložena na službenoj stranici simulatora gdje se nalazi i dodatna dokumentacija [18]. Na slici 3.2. moguće je vidjeti jedan video okvir zabilježen unutar *Carla* simulatora.

Slika 3.2. Prikaz video okvira snimljenog unutar simulatora *Carla* [18].

3.2. Predloženi algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila

Predloženi algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila može se podijeliti na sedam osnovnih koraka, gdje se većina koraka može podijeliti na manje korake. Na slici 3.3. moguće je vidjeti grafički prikaz ključnih koraka algoritma za detekciju vozne trake i prepreka unutar vozne trake.



Slika 3.3. Ključni koraci algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila

Prvi korak se odnosi na dohvaćanje video okvira pomoću RGB kamere implementirane unutar vozila u *Carla* simulatoru. Prilikom tog koraka koristi se ROS kako bi se pomoću dostupne sabirnice dohvatili video okviri zabilježeni RGB kamerom. Unutar prvog koraka potrebno je također pretvoriti video okvire u format kojeg je moguće obrađivati algoritmima koji su sadržani unutar *OpenCV* biblioteke. Kada je video okvir uspješno dohvaćen i transformiran u odgovarajući oblik, tada se obavljaju različiti postupci predobrade slike. Pri tome korištena je metoda inverznog mapiranja, kako bi se dobio ptičji pogled. Nakon transformacije perspektive provodi se filtriranje prema boji kako bi se izdvojili segmenti koji su bijele ili žute boje. Također, korišten je i *Gaussov* filtar koji se koristi za zamućivanje (engl. *bluring*) slike te *Canny* operator koji se koristi kako bi se dobila binarna slika koja sadrži rubove objekata. Navedenim postupcima smanjuje se broj elemenata slike koje se koriste kao ulaz u trećem koraku. Treći korak predstavlja primjenu RANSAC algoritma na binarnu ulaznu sliku. Kako bi se detektirale uzdužne kolničke oznake, definiraju se pravci kao matematički modeli kojima je moguće opisati iste. Iterativnim postupkom se određuju parametri pravaca koji estimiraju uzdužne kolničke oznake vozne trake. Tada se prostor koji se nalazi između dobivenih pravaca izdvaja na binarnoj slici dobivenoj postupcima predobrade slike. Kada su izdvojeni elementi binarne slike koji pripadaju voznoj traci potrebno je provjeriti postoje li prepreke unutar istih. Predloženi algoritam razlikuje dvije vrste prepreka unutar vozne trake, a to su vozilo i pješak. Kada se obavi klasifikacija, tada se primjenjuju različite metode koje za cilj imaju pronalazak elementa slike koji pripada prepreci i koji ima najveću vrijednost y koordinate (pri čemu je ishodište u gornjem lijevom kutu slike). Ta se vrijednost na y-osi predaje funkciji koja je dobivena empirijskim putem, a procjenjuje udaljenost prepreke od vozila. Na osnovu izračunate udaljenosti, dalje je potrebno prilagoditi kretanje vozila kako bi se vozilo zaustavilo na određenoj udaljenosti pred preprekom i čime bi se spriječila eventualna kolizija s drugim vozilom ili pješakom.

3.2.1. Učitavanje video okvira

Kako bi mogli procijeniti položaj kolničkih oznaka, kao i prepreka koje se nalaze unutar istih potrebno je pomoću optičkog senzora, odnosno kamere zabilježiti podatke koji se nalaze ispred vozila. Za dohvaćanje video okvira koristi se RGB kamera implementirana unutar *Carla* simulatora. Kameri je moguće pristupiti pomoću sabirnice `image_color` koja šalje video okvire formata `sensor_msg/Image`. Da bi takvu poruku mogli pretvoriti u oblik koji je moguće obraditi pomoću algoritama sadržanih u *OpenCV* biblioteci koristi se takozvani *OpenCV* most (engl. *OpenCV bridge*). *OpenCV* most definira posebnu vrstu slike koja sadrži sliku u *OpenCV* formatu koja je enkodirana pomoću ROS zaglavlja. Da bi se slika iz ROS formata pretvorila u *OpenCV* format deklarira se klasa koja sadrži potrebne rukovatelje čvorom, kao i potrebne poslužitelje i klijente. Tada se pomoću funkcije `toCvCopy` iz formata `sensor_msg/Image` dobiva slika u formatu kojeg podržava biblioteka *OpenCV* i nad kojom je moguće obaviti preostale korake algoritma.

3.2.2. Transformacija perspektive ulaznog video okvira

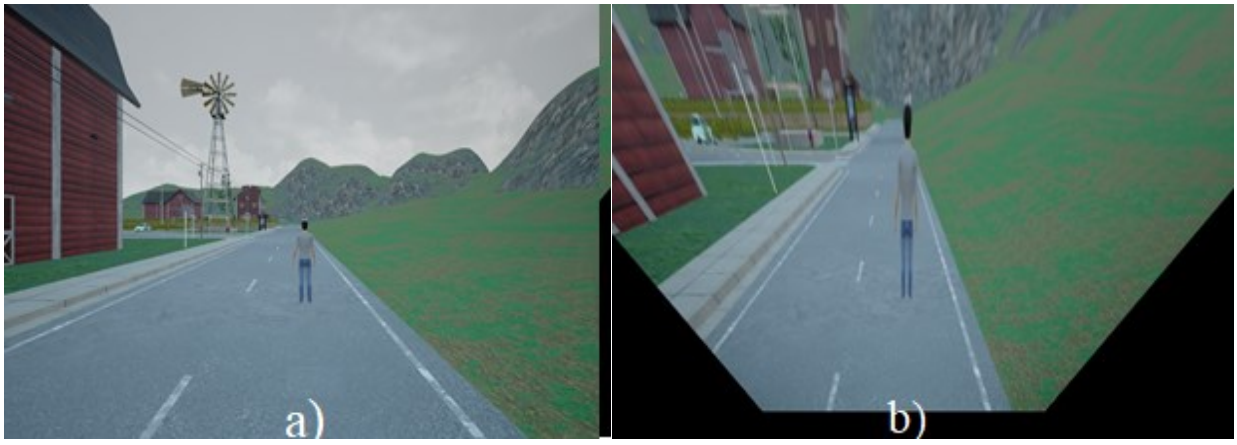
Prvi korak unutar procesa predobrade ulaznih video okvira unutar kojih je potrebno detektirati uzdužne kolničke oznake je transformacija perspektive. Ta metoda se sastoji od toga da se na osnovu ulazne slike koja je snimljena s prednje strane kamere odaberu četiri točke koje se koriste za transformaciju i zatim se primjenjuju postupci kojima se dobije transformirana slika unutar koje su prisutni svi elementi slike omeđeni odabranim točkama, ali iz druge perspektive. Korištenjem metode dobiva se perspektiva koja odgovara izgledu video okvira kada bi kamera bila postavljena iznad kolničkih oznaka što se naziva i ptičji pogled. Kada se promatraju dvije kolničke oznake koje pripadaju istoj voznoj traci iz ptičjeg pogleda, tada su one gotovo paralelne, čime je pojednostavljena detekcija istih. Tako se također eliminiraju i određeni dijelovi scene koji ne pripadaju regiji od interesa za promatranu scenu čime se smanjuje vrijeme obrade i povećava preciznost detekcije kolničkih oznaka. Navedeni postupak obavlja se pomoću biblioteke *OpenCV,*

gdje se koriste funkcije `getPerspectiveTransform` i `warpPerspective`. Funkcija `getPerspectiveTransform` prima na ulazu četiri ulazne točke koje predstavljaju četverokut slike koji se izdvaja kako bi se nad njim izvela transformacija perspektive. Prilikom odabira točki vrlo je važno odabrati područje dovoljno široko i dovoljno visoko da bi se mogle vidjeti kolničke oznake u svim mogućim scenarijima vožnje. Također, među argumentima funkcije nalazi se i skup točaka unutar kojega se dobiveni rezultat prikazuje na izlaznoj slici. Rezultat funkcije je transformacijska matrica dimenzije 3x3, koja se prosljeđuje funkciji `warpPerspective`, zajedno s ulaznom i izlaznom slikom. Tada se vrši transformacija perspektive pomoću izraza [19]:

$$dst\ (x,y) = src\ (\frac{\mathbf{M}_{11}x + \mathbf{M}_{12}y + \mathbf{M}_{13}}{\mathbf{M}_{31}x + \mathbf{M}_{32}y + \mathbf{M}_{33}}, \frac{\mathbf{M}_{21}x + \mathbf{M}_{22}y + \mathbf{M}_{23}}{\mathbf{M}_{31}x + \mathbf{M}_{32}y + \mathbf{M}_{33}}), \tag{3-1}$$

gdje *src* predstavlja ulaznu sliku, *dst* predstavlja izlaznu sliku, *x* i *y* određeni stupac, odnosno redak ulazne, odnosno izlazne matrice, dok $\mathbf{M}_{ij}$ predstavljaju pojedine elemente transformacijske matrice **M**.

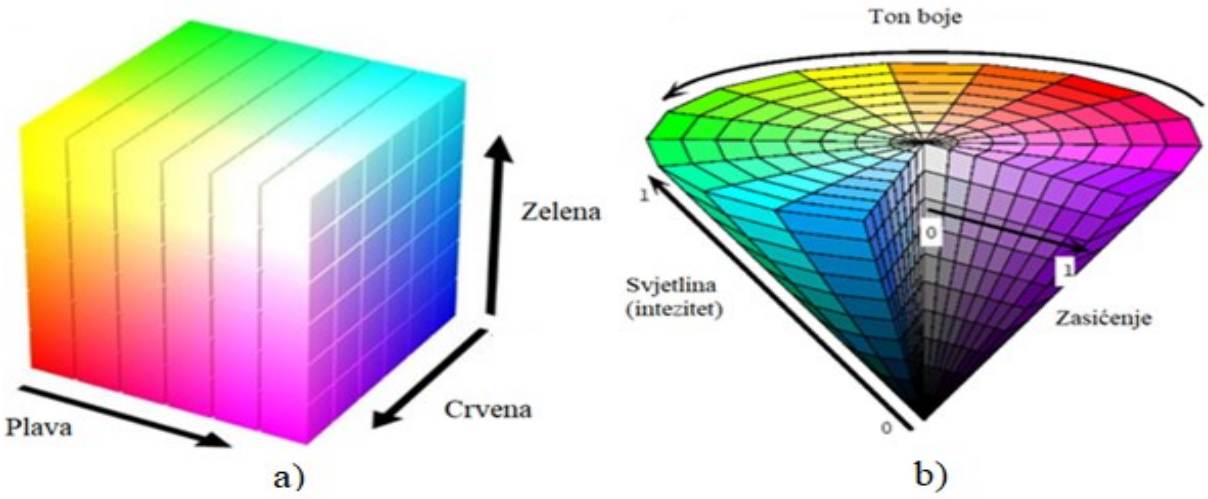
Na slici 3.4. b) moguće je vidjeti sliku koja se dobije nakon inverzne transformacije perspektive ulazne slike 3.4. a).



Slika 3.4. Slika koja je korištena za inverznu transformaciju perspektive a) i slika koja je dobivena kao izlaz inverzne transformacije b).

3.2.3. Filtriranje prema boji

Kada se transformacijom perspektive izdvoji određeno područje, to područje najčešće sadrži veliku količinu šuma. Filtriranjem prema boji može se reducirati količina šuma, čime se može povećati efikasnost detekcije uzdužnih kolničkih linija koja se odvija u narednim koracima priloženog algoritma. Svaki video okvir bilježi se unutar RGB modela boja, gdje se svaki element slike opisuje pomoću tri vrijednosti koje predstavljaju udio crvene, zelene i plave boje (vidi sliku 3.5. a)). Prije filtriranja dobiveni video okviri se transformiraju iz RGB prostora boja u HSV (engl. ***H**ue **S**aturation **V**alue*) prostor boja (vidi sliku 3.5. b)) koristeći *OpenCV* naredbu `cvtColor`. HSV prostor boja se prikazuje pomoću tri vrijednosti. Prva vrijednost je ton boje koji se kreće unutar raspona između 0 i 180. Osim tona boje koriste se komponente kao što su zasićenje boje i svjetlina ili intenzitet koje se kreću u rasponu od 0 do 255. Filtriranje se vrši unutar HSV prostora boja zato što navedeni prostor boja razlikuje intenzitet svjetlosti od komponenti boje. Zbog toga je navedeni prostor boja znatno robusniji na različito osvjetljenje objekata, što je također učinkovito ako na slici postoje sjenke koje prekrivaju objekte od značaja. Korištenjem prikladnih raspona vrijednosti koje odgovaraju bojama koje pripadaju kolničkim oznakama, moguće je filtrirati objekte određene boje unutar HSV modela boja pomoću *OpenCV* naredbe `inRange`, koja prima ulaznu sliku, kao donje i gornje pragove svih kanala određenog modela boja. Kako bi se filtrirale bijela, odnosno žuta boja unutar HSV modela, korišteni su donji pragovi iznosa [150, 150, 150] i gornji pragovi iznosa [255, 255, 255].



Slika 3.5. Prikaz RGB (a) i HSV (b) prostora boja.

3.2.4. Gaussov filtar

Proces filtriranja podrazumijeva linearnu transformaciju, odnosno primjenu maske na određenu grupu elemenata slike. Pri tome postoje dvije vrste filtriranja, visokopropusno i niskopropusno filtriranje, kojemu pripada *Gaussov* filtar. Niskopropusno filtriranje se radi tako da se u frekvencijskoj domeni prigušuje dio spektra signala iznad određenog praga, dok se dio spektra koji ima frekvenciju manju od praga propušta. Tako niskopropusni filtri vrše zamućivanje ulaznih slika, čime se smanjuje broj detalja, što ujedno smanjuje šum prisutan na slici. *Gaussov* filtar koristi konvolucijsku masku koja ima oblik *Gaussove* krivulje. *Gaussov* filtar temelji se na sljedećem izrazu [20]:

$$h_g(x,y) = e^{\frac{-(x^2+y^2)}{2*\sigma^2}}, \tag{3-2}$$

gdje h_g predstavlja odziv niskopropusnog filtra, σ predstavlja standardnu devijaciju koju je moguće mijenjati kako bi se dobile različite vrijednosti niskopropusnog filtra, a x i y predstavljaju određeni

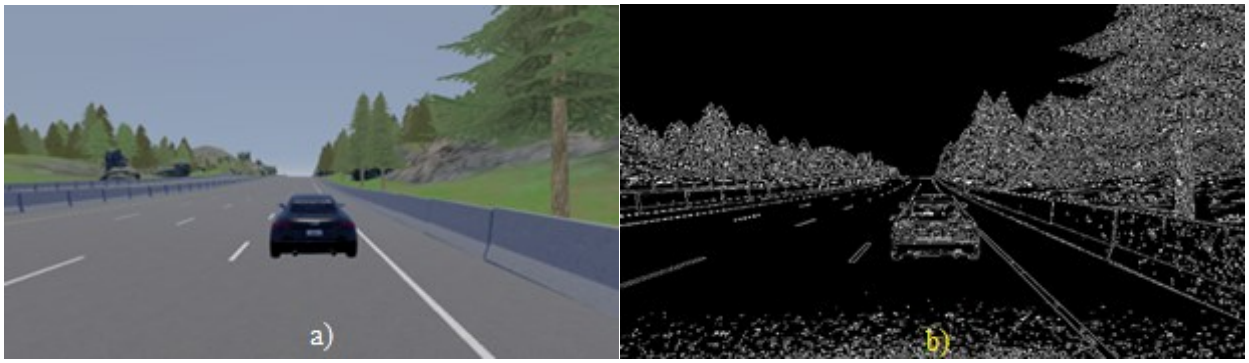
redak, odnosno stupac slike. Kako bi se aproksimirao *Gaussov* filtar, odabire se vrijednost standardne devijacije (najčešće $\sigma = 1$), kao i dimenzije konvolucijske matrice (npr. 3x3, 4x4, 5x5, …) gdje se za svaki element matrice računa koeficijent dobiven pomoću *Gaussove* funkcije (3-2). Temeljni zadatak *Gaussovog* filtra je zamućenje slike, čime se gube detalji na slici jer se prigušuju visokofrekvencijske komponente. *Gaussov* filtar najčešće se koristi zajedno metodama detekcije rubova gdje je cilj očuvati stvarne rubove, a nekakvi nevažni detalji kao što je tekstura na kolniku se izbacuju iz daljnjeg procesiranja. Na slici 3.6. b) moguće je vidjeti primjer video okvira nakon primjene *Gaussovog* filtra nad video okvirom na slici 3.6. a).

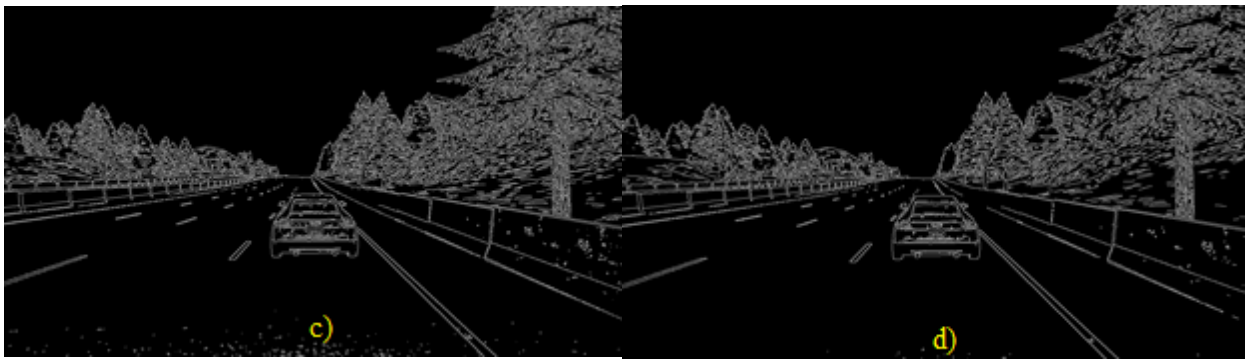
Slika 3.6. Prikaz ulaznog video okvira (a) i video okvira nad kojim je primijenjen *Gaussov* filtar dimenzija konvolucijske matrice 3x3 (b).

3.2.5. *Canny* detekcija rubova

Nakon što je slika filtrirana, potrebno je detektirati rubove objekata. Ako se ispravno detektiraju rubovi objekata moguće je odvojiti određene objekte od značaja od pozadine. U radu je korišten takozvani *Canny* operator. Prvi korak koji se izvodi kod primjene *Canny* operatora je primjena metode gradijenta gdje se dobiju četiri matrice horizontalnih, vertikalnih i dijagonalnih rubova na osnovu kojih se radi mapa položaja rubova i orijentacije svakog ruba. *Canny* operator zahtijeva dva parametra koji predstavljaju donji i gornji prag s histerezom. Kada se usporede dobiveni gradijenti s gornjom granicom, moguće je zaključiti koji će rubovi sigurno biti označeni kao ispravno detektirani rubovi. Konačno, promatraju se susjedni elementi oko označenog ruba kao i smjer ruba i ako im je vrijednost između donjeg i gornjeg praga, tada se mogu proglasiti rubovima [20]. Korištenje *Canny*-evog operatora unutar *OpenCV*-a je omogućeno pomoću ugrađene funkcije `Canny` koja prima za parametre ulaznu sliku, izlaznu sliku, vrijednost donjeg praga i vrijednost gornjeg praga.

Na slikama 3.7. b) do d), moguće je vidjeti primjenu *Canny* detekcije rubova na sliku 3.7. a) uz različite vrijednosti donjeg i gornjeg praga. S dobivenih binarnih slika jasno je vidljivo kako povećanje donjeg tako i gornjeg praga smanjuje broj segmenata koji će biti detektirani kao rubovi. Iako su u svim slučajevima sami rubovi kolničkih oznaka uspješno detektirani, na slici 3.7.b), gdje je iznos pragova nizak u blizini vozila pojavljuje se veliki broj lažnih segmenata.





Slika 3.7. Prikaz ulaznog video okvira (a), kao i primjene *Canny-jevog* operatora za detekciju rubova s iznosom pragova 20 i 40 (b), 40 i 60 (c), 50 i 80 (d).

3.2.6. RANSAC (RANdom SAmple Consensus)

RANdom SAmple Consensus (RANSAC) je iterativna metoda koja služi za procjenu parametara unaprijed definiranog modela na temelju mjernih podataka. Pritom je cilj RANSAC-a obuhvatiti što je veći broj moguće pripadajućih vrijednosti (engl. *inliers*). Ako se za matematički model odabere pravac tada je dovoljno odabrati dvije točke iz skupa podataka koje jednoznačno određuju pravac na slici. Također se definiraju gornja i donja granica odstupanja pomoću kojih se dobiva područje oko pravaca za koje se smatra da vrijednosti koje mu pripadaju predstavljaju pripadajuće vrijednosti za konkretni pravac. Kada se odrede navedeni parametri prebroji se koliko se ukupno točaka odnosno vrijednosti nalazi unutar dozvoljenog odstupanja od definiranog pravca. Postupak se ponavlja predodređeni broj puta te se za svaki dobiveni pravac bilježi broj pripadajućih vrijednosti. Na kraju postupka odabire se pravac koji ima najviše pripadajućih vrijednosti kao pravac koji najbolje opisuje dane podatke.

RANSAC je nedeterministički algoritam što znači da postoji prihvatljiv rezultat s određenom vjerojatnošću koji može odstupati od idealnog rješenja. Kako bi se povećala vjerojatnost dobivanja točnog rješenja, potrebno je povećavati broj iteracija. Međutim nakon određene vrijednosti veći broj iteracija neće značajno povećati preciznost rezultata, ali će povećati

vrijeme izvođenja algoritma što u sustavima stvarnog vremena može imati značajan utjecaj. Potreban broj iteracija N kod RANSAC-a može se izračunati na sljedeći način [21]:

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}, \tag{3-3}$$

da bi se detektirao pravac s vjerojatnošću p. Varijabla e predstavlja vjerojatnost da je vrijednost stršeća (engl. *outlier*), a pomoću S predstavljen je broj točaka kojima se estimira matematički model ulaznih podataka, što u slučaju pravca odgovara broju dva.

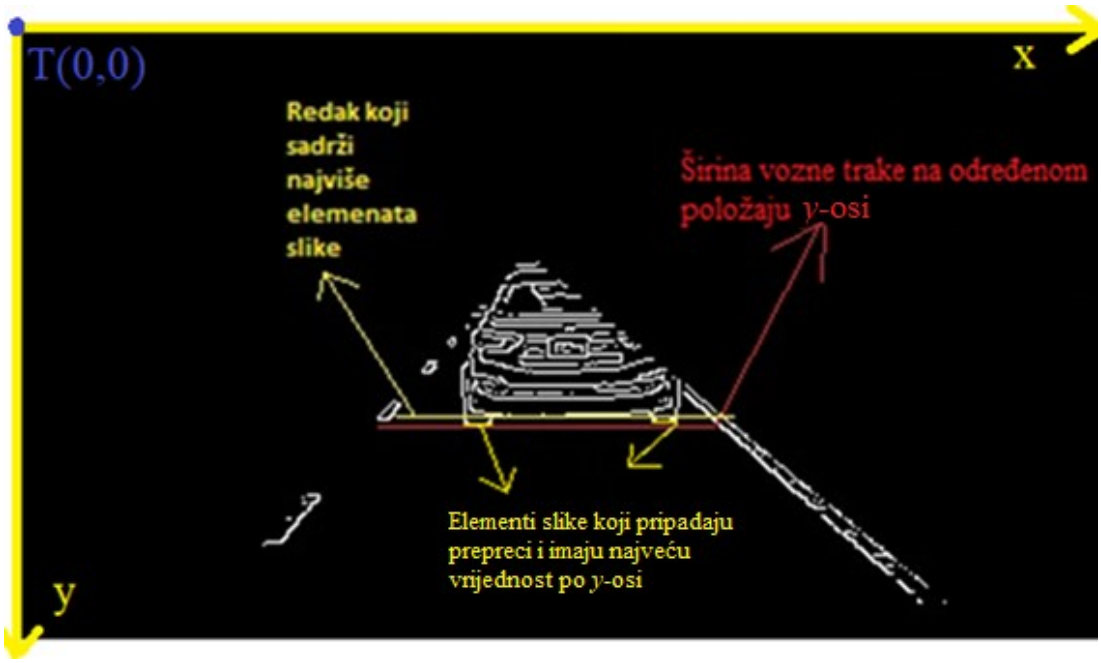
Kao ulaz u ovaj dio algoritma koristi se binarna slika dobivena detekcijom rubova pomoću *Canny* operatora. Detekcijom rubova smanjen je broj ulaznih podataka za RANSAC algoritam što omogućuje smanjivanje broja iteracija kako bi se detektirale kolničke oznake. Zbog toga što je potrebno detektirati voznu traku, odnosno dvije kolničke oznake, potrebno je primijeniti RANSAC algoritam dva puta. Za izvođenje ovog algoritma koristi se 250 iteracija prilikom oba izvođenja. Kada je detektirana prva kolnička oznaka tada se pozitivni elementi slike brišu s binarne slike koja je ulaz prilikom drugog izvođenja RANSAC-a. Tako se sprječava ponovna detekcija iste kolničke oznake, a ujedno se i smanjuje broj ulaznih podataka za drugo izvođenje RANSAC algoritma. Dodatno je moguće detektirati i kolničke oznake susjednih voznih traka, ali zbog brzine izvođenja cijelog algoritma postavljeni su uvjeti kojima se traže elementi slike koji se nalaze u određenom rasponu širine video okvira, gdje bi se prema pretpostavci trebale nalaziti kolničke oznake vozne trake (ako vozilo ne napušta svoju voznu traku). Na temelju usporedbe vrijednosti po x-osi elemenata slike određuje se koji pravac odgovara lijevoj, a koji odgovara desnoj kolničkoj oznaci. Izlaz iz ovog dijela algoritma su dva pravca koji predstavljaju kolničke oznake vozne trake na ulaznoj slici. Zbog toga što se često na kolniku nalaze isprekidane kolničke oznake koristi se dobivena jednadžba pravca kako bi se detektirana kolnička oznaka produžila do dna slike. Također, pomoću jednadžbi pravaca pronalazi se točka u kojoj se detektirani pravci sijeku. Tako je dobiveno područje koje omeđuje voznu traku.

Prilikom izvođenja algoritma korišten je tip podataka `cv::Point2f` koji je definiran unutar *OpenCV* biblioteke. Navedeni tip predstavlja vektor koji sadrži dvije vrijednosti kojima se prikazuju koordinate pojedinog elementa slike. Osim toga, korištene su logičke funkcije I za izdvajanje područja za traženje pripadajućih vrijednosti i ILI za brisanje detektirane kolničke oznake s binarne slike. Funkcija `line` korištena je kako bi se pravci prikazali na izlaznoj slici.

3.2.7. Klasifikacija i procjena udaljenosti od prepreke unutar detektirane vozne trake

Kada je pomoću RANSAC algoritma detektirana vozna traka kojom se vozilo kreće, potrebno je procijeniti nalazi li se eventualna prepreka unutar tog područja. Za izdvajanje rubova unutar vozne trake koristi se logička operacija I (engl. ***AND***), gdje se kao prvi ulaz koristi binarna slika s označenom regijom od interesa, dok se kao drugi ulaz koristi binarna slika koja je dobivena primjenom *Canny*-evog operatora za detekciju rubova nad ulaznim video okvirom.

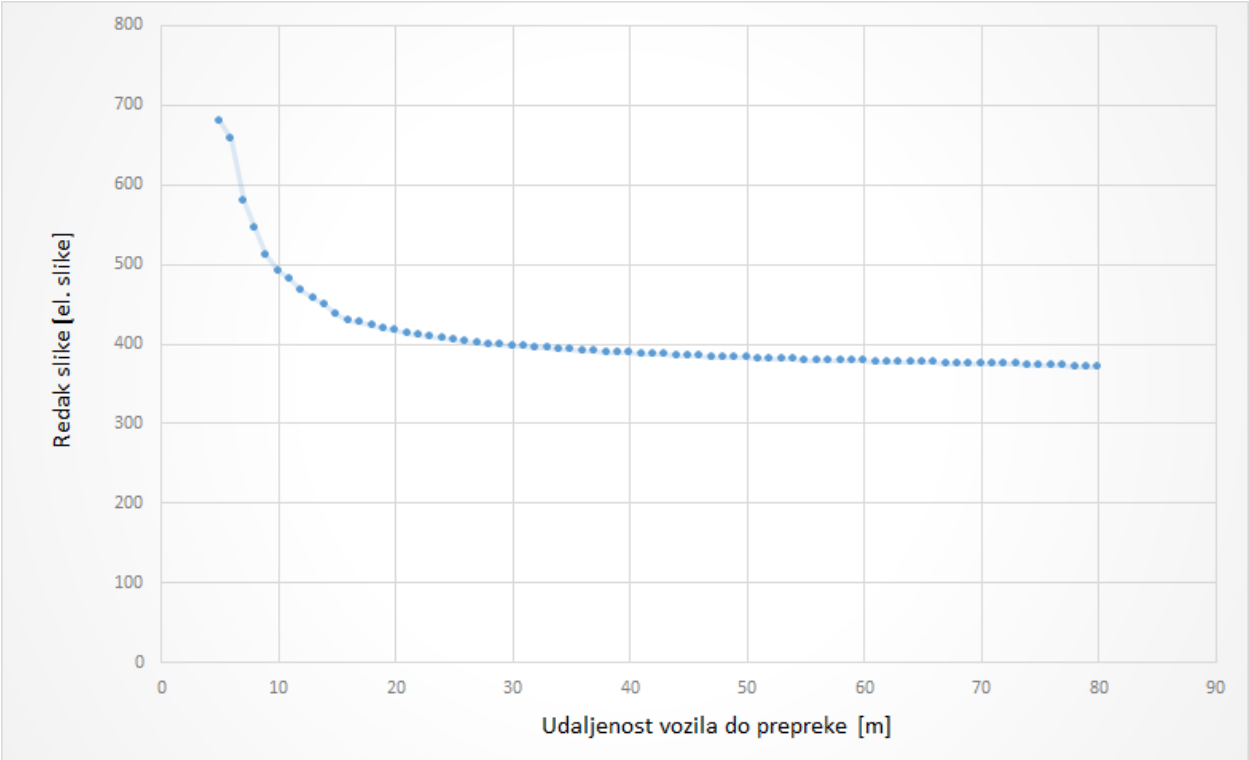
Da bi se procijenila udaljenost do prepreke, važno je definirati nekoliko vrijednosti koje su naznačene na slici 3.8. Za procjenu udaljenosti traži se redak slike koji ima najveću vrijednost po *y*-osi na kojem postoje elementi slike koji pripadaju prepreci. Također, prilikom klasifikacije prepreke i računanja udaljenosti koriste se širina objekta na određenom retku i širina vozne trake na određenom retku. Širina vozne trake za određenu udaljenost dobivena je tako da se za neku vrijednost *y*-osi oduzme vrijednost na *x*-osi elementa slike koji pripada lijevoj kolničkoj oznaci od vrijednosti na *x*-osi elementa slike koji pripada desnoj kolničkoj oznaci. Za procjenu širine objekta na svakom retku slike izračunat je broj pozitivnih elemenata slike unutar vozne trake.



Slika 3.8. Binarna slika koja se koristi za procjenu udaljenosti automobila od prepreke.

Prilikom razvoja algoritma definirana je funkcija udaljenosti prepreke od automobila u odnosu na poziciju elementa slike za koji se smatra da predstavlja rub prepreke, a ima najveću vrijednost po *y*-osi slike. Da bi se dobila funkcijska ovisnost navedene udaljenosti o položaju elementa slike po *y*-osi koji pripada rubu objekta unutar vozne trake, zabilježeni su video okviri s različitim udaljenostima prepreke, gdje su položaji elementa slike određeni prikladnim alatima za obradu slike koji prikazuju koordinate elemenata slike. Stvarna udaljenost je dobivene pomoću sustava ugrađenog unutar simulatora. Prilikom određivanja navedene funkcijske povezanosti uočeno je kako mjerenja dobivena pomoću sustava u simulatoru mjere udaljenost od centra mase vozila pa je potrebno oduzeti četiri metra kako bi se dobila udaljenost od stražnjeg kraja vozila. Slično, kada se radi o pješaku, tada je potrebno oduzeti dva metra od detektirane udaljenosti kako bi se ponovno dobila točna udaljenost. Na slici 3.9. prikazan je dobiveni graf ovisnosti položaja elementa slike prepreke u odnosu na udaljenost prepreke od automobila, gdje je na *x*-osi stvarna udaljenost, dok je na *y*-osi iznos retka elementa slike koji pripada prepreci i koji ima najveću vrijednost na *y*-osi slike.

a



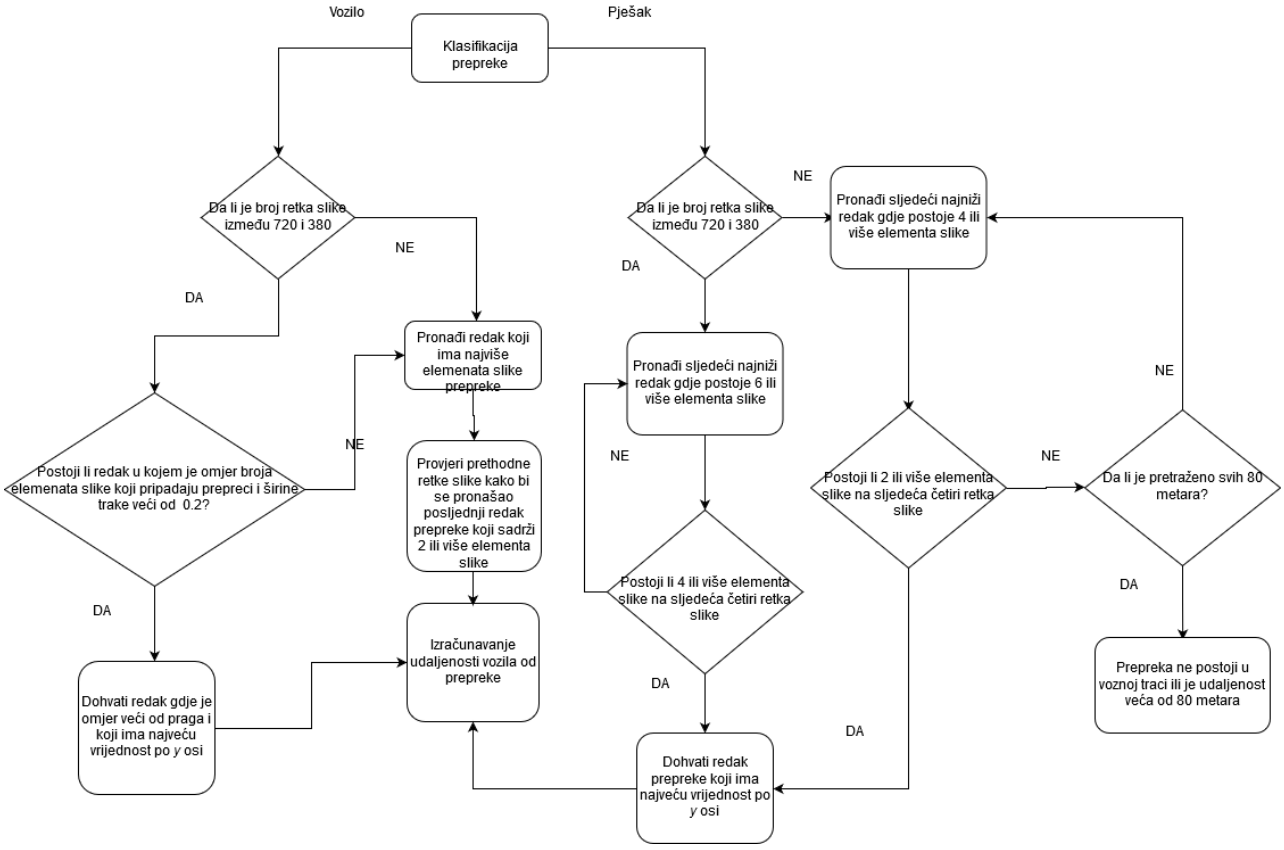
Slika 3.9. Prikaz ovisnosti položaja prepreke na slici o stvarnoj udaljenosti od iste.

Slike su dimenzije 1280x720, što znači da je po visini slika podijeljena na 720 redaka. Unutar *OpenCV* biblioteke elementi slike koji pripadaju najnižem retku na slici imaju vrijednost na *y*-osi jednaku 720, dok elementi slike koji pripadaju najvišem retku na slici imaju vrijednost 0. Kao što je vidljivo iz slike 3.9., dobivena ovisnost udaljenosti prepreke od automobila nije linearna. Također pri većim udaljenostima više udaljenosti mogu biti predstavljene istim položajem elementa slike koji pripada prepreci i ima najveću vrijednost po *y*-osi zbog konačne rezolucije slike.

Da bi se procijenila udaljenost do prepreke, potrebno je klasificirati da li je prepreka pješak ili vozilo. Vozilo je znatno šire od pješaka, a ta razlika je znatno veća na bližim udaljenostima. Kako bi se obavila klasifikacija provjerava se omjer broja elemenata slike prepreke na nekoj

vrijednosti *y*-osi sa širinom vozne trake na toj vrijednosti *y*-osi te se uspoređuje s iznosom praga. Prag nije fiksne veličine nego se za područje s vrijednosti na *y*-osi od 720 do 380 koristi prag 0.4, dok se za vrijednosti od 379 do 0 koristi prag od 0.6. Ako na pojedinom video okviru postoji pet ili više vrijednosti na *y*-osi gdje je omjer veći od praga, tada se zaključuje da je prepreka vozilo, a u suprotnom se prepreka klasificira kao pješak. Ako se prepreka klasificira kao pješak, ali metodama procjene udaljenosti se ne nađe udaljenost do pješaka, tada se zaključuje da prepreke nema u voznoj traci ili je udaljenost od iste veća od 80 metara.

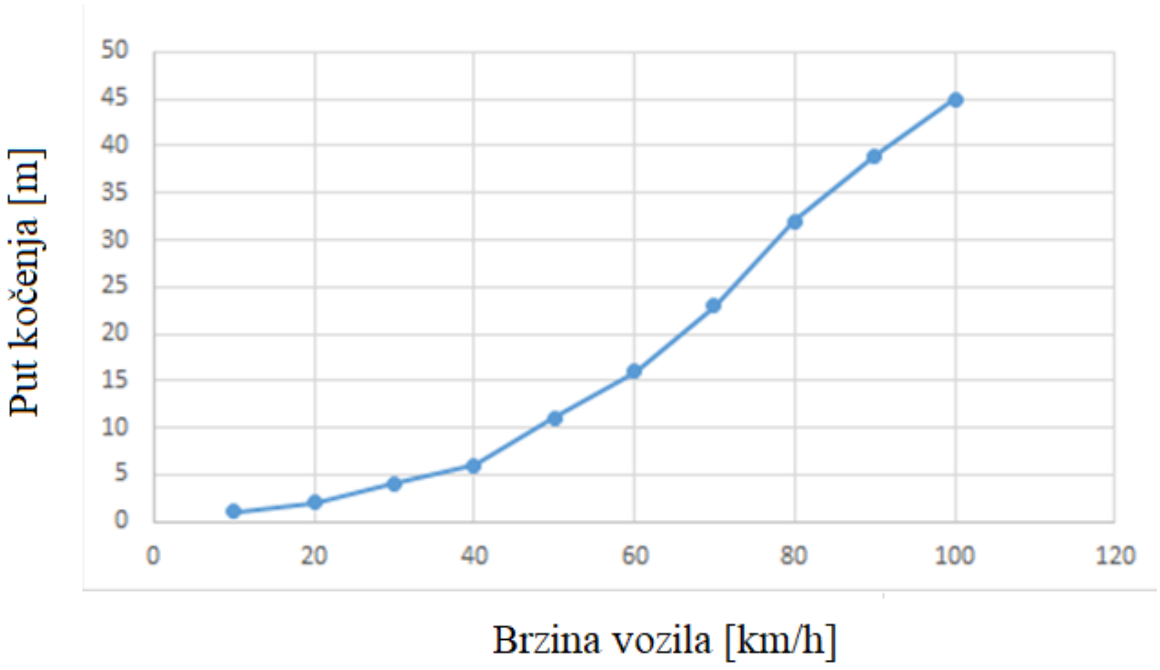
Kada je prepreka klasificirana kao vozilo ili pješak, koriste se različiti načini kako bi se dobio najviši iznos na *y*-osi koji sadrži elemente slike koje pripadaju prepreci. Na slici 3.10. prikazan je dijagram toka izvođenja ovog dijela algoritma nakon što je prepreka klasificirana.



Slika 3.10. Blok dijagram dijela algoritma koji se koristi za izračunavanje udaljenosti od prepreke.

3.2.8. Zaustavljanje vozila kada je detektirana prepreka

Kada je dobivena vrijednost udaljenosti od prepreke, tada je potrebno prilagoditi brzinu vozila u odnosu na tu udaljenost, kako bi se sigurno zaustavilo vozilo. Signal za kočenje šalje se samo ako je uspješno detektirana prepreka i procijenjena udaljenost prepreke je manja od 80 metara (vidi sliku 3.9.). Prilikom procjene treba li vozilo kočiti, uzima se u obzir brzina kojom se vozilo kreće, kao i udaljenost od prepreke. Ako se definira udaljenost na kojoj vozilo treba zakočiti pred preprekom, potrebno je na tu udaljenost nadodati zaustavni put potreban da bi se vozilo pri određenoj brzini zaustavilo na definiranoj udaljenosti. Zaustavni put eksperimentalno je utvrđen unutar simulatora. Mjerenja su izvedena na ravnoj suhoj cesti za automobil Audi Etron tako što se mjerio zaustavni put za različite brzine pri čemu je korištena samo opcija maksimalnog kočenja zbog ograničenja simulatora i nedostupne dokumentacije. Na slici 3.11. je prikazan graf ovisnosti puta kočenja o trenutnoj brzini.



Slika 3.11. Graf ovisnosti puta kočenja o brzini vozila.

Da bi se vozilu unutar simulatora poslala naredba gasa, odnosno kočenja potrebno je kreirati poslužitelj tipa `CarlaEgoVehicleControl` koji sadrži dva parametra `throttle` i `brake` koji predstavljaju papučicu gasa, odnosno kočnicu. Kako bi bilo omogućeno upravljanje vozilom potrebno se povezati na sabirnicu kojom se pristupa pomoću sljedeće putanje `/carla/ego_vehicle/vehicle_control_cmd`. Komunikacija s navedenom sabirnicom obavlja se tako što se šalju poruke formata `carla_msgs`.

3.3. Pokretanje algoritma

Kako bi pokrenuli algoritam, potrebno je prvo pokrenuti simulator. Najprije je potrebno pozicionirati se u direktorij iz kojeg će biti pokrenut server simulatora. Pozicioniranje se obavlja naredbom:

```
cd UnrealEngine_4.22/carla/Dist/CARLA_Shipping_0.9.6-4-ga91foda-dirty/LinuxNoEditor
```

Nakon toga pokreće se server naredbom:

```
./CarlaUE4.sh /Game/Carla/Maps/Town04 -windowed -Res=320 -Res=240 -benchmark -fps=40
```

Tada je potrebno se pozicionirati u direktorij kako bi se pokrenuo klijent naredbom:

```
cd UnrealEngine_4.22/carla/Dist/CARLA_Shipping_0.9.6-4-ga91foda-dirty/LinuxNoEditor/PythonAPI/examples
```

Klijent se pokreće pomoću Python skripte naredbom:

```
python manual.py –rolename=ego_vehicle
```

Kako bi ubacili prepreku na željeno mjesto pokreće se drugi klijent:

```
python manual.py –rolename=obstacle
```

Ako želimo kao prepreku koristiti pješaka potrebno je pokrenuti naredbu:

```
python manual.py –rolename=obstacle –filter=walker
```

Carla ROS most se pokreće naredbom:

```
roslaunch carla_ros_bridge carla_ros_bridge.launch
```

Kada je simulator pokrenut potrebno je izgraditi potrebne pakete tako što se pozicioniramo u direktor catkin_ws i pokrenemo naredbu:

```
catkin build
```

Posljednji korak je pokretanje programskog koda sadržanog u main.cpp koji je dan u prilogu P.3.1. naredbom:

```
rosrun ros_directory main
```

Ako su ispravno instalirani simulator *Carla*, ROS okruženje i *OpenCV* biblioteka moguće je pokrenuti algoritam. Pri tomu preporučuje se korištenje 0.9.4 verzije *Carla* simulatora, kao i 0.9.5 verzije *Carla* ROS mosta.

4. Verifikacija rada algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila

Verifikacija algoritma predstavljenog u prethodnom poglavlju učinjena je najprije na slikama dobivenim u *Carla* simulatoru, gdje je ispitana efikasnost rješenja u smislu detekcije vozne trake, estimacije udaljenosti vozila od prepreke te klasifikacija prepreke. Za testiranje navedenog algoritma korištene su slike snimljene unutar *Carla* simulatora pri čemu je za snimanje korištena ugrađena RGB kamera. Ukupno je snimljeno 152 različite slike pri čemu se vozilo s kamerom nalazi u sredini vozne trake, a prepreka (drugo vozilo ili pješak) se nalazila na određenoj udaljenosti ispred vozila u voznoj traci. U slučaju kada je prepreka drugo vozilo, korišteni su osobni automobili dostupni unutar simulatora. Rezolucija snimljenih slika je 1280x720 elemenata slike. Također, evaluacija rješenja provedena je i u simulatoru, tako da se vozilo kretalo po kolniku s obilježenim uzdužnim oznakama s preprekama u voznoj traci. Tako je provjerena i ispravnost zadnjeg koraka algoritma (prilagodba brzine vozila). Testiranje je provedeno na osobnom računalu sa sljedećim hardverskim i softverskim karakteristikama: Linux operacijski sustav Ubuntu 16.04, Intel i7-5820 procesor, 32 GB RAM i grafičku karticu Nvidia GeForce GTX 1060. Također su korišteni Carla 0.9.4., Carla ROS Bridge 0.9.5. OpenCV 3.4.3, kao i ROS Kinetic Kame.

4.1. Verifikacija rada predloženog algoritma na slikama dobivenim u simulatoru Carla

Kako bi se verificirao rad predloženog algoritma snimljen je set slika koji se sastoji od 152 slike. Dakle, snimljene su 76 slike kada drugo vozilo predstavlja prepreku unutar vozne trake, odnosno dodatne 76 slike kada pješak predstavlja prepreku unutar vozne trake. Pri tomu, svaka prepreka snimljena je na udaljenosti od 5 do 80 metara s korakom od jednog metra. Unutar tablica 4.1., 4.2. i 4.3. moguće je vidjeti rezultate koji su dobiveni testiranjem algoritma na osnovu navedenih slika, gdje je moguće vidjeti rezultat klasifikacije prepreke, stvarnu udaljenost u metrima koja je dobivena u simulatoru, te izmjerenu udaljenost primjenom algoritma.

Tablica 4.1. Rezultati dobiveni verifikacijom algoritma za detekciju prepreka unutar vozne trake na udaljenostima prepreke od 5 do 39 metara.

	Automobil		Pješak	
Stvarna udaljenost *d* [m]	Predviđena klasa	Procijenjena udaljenost [m] - $\hat{d}$	Predviđena klasa	Procijenjena udaljenost [m] - $\hat{d}$
5	Automobil	5	Pješak	5
6	Automobil	6	Pješak	6
7	Automobil	7	Pješak	7
8	Automobil	8	Pješak	8
9	Automobil	9	Pješak	9
10	Automobil	11	Pješak	10
11	Automobil	11	Pješak	11
12	Automobil	12	Pješak	12
13	Automobil	13	Pješak	13
14	Automobil	14	Pješak	13
15	Automobil	15	Pješak	15
16	Automobil	17	Pješak	17
17	Automobil	18	Pješak	16
18	Automobil	17	Pješak	19
19	Automobil	18	Pješak	18
20	Automobil	21	Pješak	20
21	Automobil	22	Pješak	22
22	Automobil	21	Pješak	24
23	Automobil	24	Pješak	23
24	Automobil	25	Pješak	24
25	Automobil	27	Pješak	29
26	Automobil	29	Pješak	28
27	Automobil	30	Pješak	28
28	Automobil	31	Pješak	29
29	Automobil	28	Pješak	31
30	Automobil	29	Pješak	31
31	Automobil	30	Pješak	33
32	Automobil	32	Pješak	33
33	Automobil	36	Pješak	36
34	Automobil	37	Pješak	36
35	Automobil	36	Pješak	37
36	Automobil	36	Pješak	41
37	Automobil	36	Pješak	38
38	Automobil	37	Pješak	41
39	Automobil	39	Pješak	41

Tablica 4.2. Rezultati dobiveni verifikacijom algoritma za detekciju prepreka unutar vozne trake na udaljenostima prepreke od 40 do 74 metara.

	Automobil		Pješak	
Stvarna udaljenost d [m]	Predviđena klasa	Procijenjena udaljenost - $\hat{d}$ [m]	Predviđena klasa	Procijenjena udaljenost - $\hat{d}$ [m]
40	Automobil	40	Pješak	45
41	Automobil	44	Pješak	46
42	Automobil	42	Pješak	46
43	Automobil	42	Pješak	46
44	Automobil	48	Pješak	46
45	Automobil	48	Pješak	51
46	Automobil	44	Pješak	49
47	Automobil	50	Pješak	51
48	Automobil	45	Pješak	51
49	Automobil	48	Pješak	54
50	Automobil	50	Pješak	54
51	Automobil	50	Pješak	55
52	Automobil	50	Pješak	55
53	Automobil	54	Pješak	58
54	Automobil	54	Pješak	58
55	Automobil	54	Pješak	58
56	Automobil	54	Pješak	55
57	Automobil	62	Pješak	63
58	Automobil	57	Pješak	63
59	Automobil	54	Pješak	63
60	Automobil	57	Pješak	63
61	Automobil	62	Pješak	66
62	Automobil	69	Pješak	63
63	Automobil	65	Pješak	66
64	Automobil	65	Pješak	66
65	Pješak	63	Pješak	66
66	Automobil	65	Pješak	66
67	Pješak	63	Pješak	67
68	Automobil	69	Pješak	67
69	Automobil	69	Pješak	75
70	Automobil	65	Pješak	63
71	Automobil	69	Pješak	75
72	Automobil	69	Pješak	75
73	Pješak	74	Automobil	80
74	Automobil	69	Automobil	80

Tablica 4.3. Rezultati dobiveni verifikacijom algoritma za detekciju prepreka unutar vozne trake na udaljenostima prepreke od 75 do 80 metara.

75	Automobil	69	Pješak	75
76	Automobil	73	Pješak	75
77	Automobil	73	Pješak	80
78	Automobil	73	Nema prepreke	-
79	Pješak	78	Pješak	79
80	Pješak	80	Nema prepreke	-

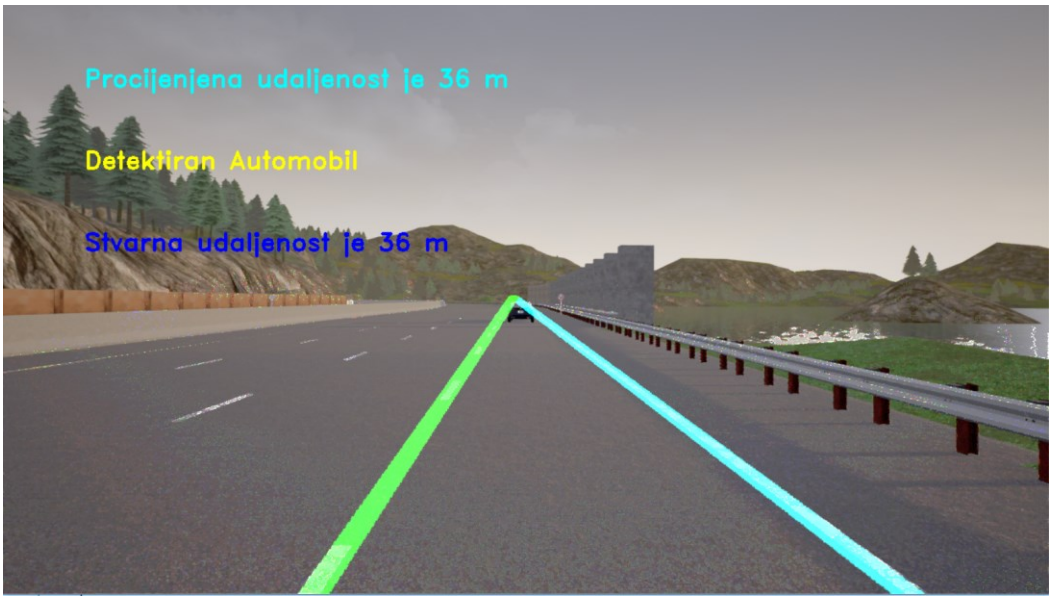
Srednju apsolutnu pogrešku procjene udaljenosti prepreke od vozila PO moguće je dobiti pomoću izraza:

$$PO = \frac{\sum_{i=5}^{n}|(d_i - \hat{d}_i)|}{N}, \tag{4-1}$$

gdje $\hat{d}$ predstavlja procijenjenu vrijednost pomoću predloženog algoritma, d predstavlja stvarnu vrijednost koja se na ulaznom skupu prikazuje vrijednostima od 5 do 80, dok je N ukupan broj slika, a n maksimalna stvarna udaljenost. Primjenom formule (4-1) dobivena je srednja apsolutna pogreška kada je automobil prepreka $PO_{automobil}$ koja iznosi 1.68 metra, dok je u slučajevima kada je prepreka pješak prosječno odstupanje $PO_{pješak}$ iznosi 2.3 metra. Očekivano, manje je odstupanje kada je prepreka automobil zbog toga što je većih dimenzija i lakše je procijeniti udaljenost od njega. Analizom rezultata također je moguće vidjeti da kada je stvarna prepreka automobil, algoritam na testnom skupu slika pet puta, odnosno u 6.58% slučajeva pogrešno klasificira prepreku kao pješaka. Kada je stvarna prepreka pješak, tada algoritam u četiri slučaja pogrešno detektira da je prepreka automobil ili da nema prepreke, što odgovara pogrešci od 5.26%. Prepreka je osim u dva slučaja uspješno detektirana, međutim postoji pogreška klasifikacije koja se pojavljuje prilikom većih udaljenosti jer su manji omjeri između elemenata slike prepreke i širine vozne trake pa i mala pogreška kod detekcije uzdužnih prometnih oznaka može ukloniti dio elemenata slike koji pripadaju prepreci. Veću točnost procjene udaljenosti prepreke od vozila algoritam ima kada je prepreka na stvarnoj udaljenosti manjoj od 40 metara. Ipak, moguća su manja odstupanja od stvarnih vrijednosti. Kao što je navedeno, vrijednost koja se koristi za stvarnu

udaljenost unutar simulatora je zaokružena na cjelobrojnu vrijednost u metrima čime stvarna vrijednost može varirati +/-0.5 metra. Ipak, rasponi elemenata slike unutar kojih se mogu naći prepreke pri određenoj udaljenosti u blizini vozila dovoljno su veliki kako bi se u velikom broju slučajeva dobio točan rezultat. Također, u situacijama gdje je prepreka na udaljenosti većoj od 40 metara moguće je vidjeti da se pogreška u procjeni udaljenosti povećava. To se događa iz više razloga , a jedan od njih je to što se vozilo ili pješak kreće u trodimenzonalnom prostoru, dok je slika prikazana u dvodimenzinalnom, pa kao što je već navedeno, ovisnost stvarne udaljenosti o položaju prepreke nije linearna. Stoga, na većim udaljenostima, kao što su na primjer udaljenosti preko 60 metara postoji vrlo mala razlika između položaja objekta na različitim udaljenostima. Specifično, postoje slučajevi kada je za više različitih udaljenosti moguće vidjeti da se položaj zadnjeg retka prepreke razlikuje za iznos manji od jednog elementa slike. Maksimalno odstupanje kada je prepreka automobil na testnom skupu iznosi 7 metara, dok maksimalno odstupanje kada je prepreka pješak iznosi 9 metara. Iako taj iznos nije zanemariv, navedene pogreške dogodile su se pri velikim udaljenostima i moguće ih je ispraviti u sljedećim video okvirima kada vozilo prilazi prepreci i kada točnost estimacije udaljenosti raste.

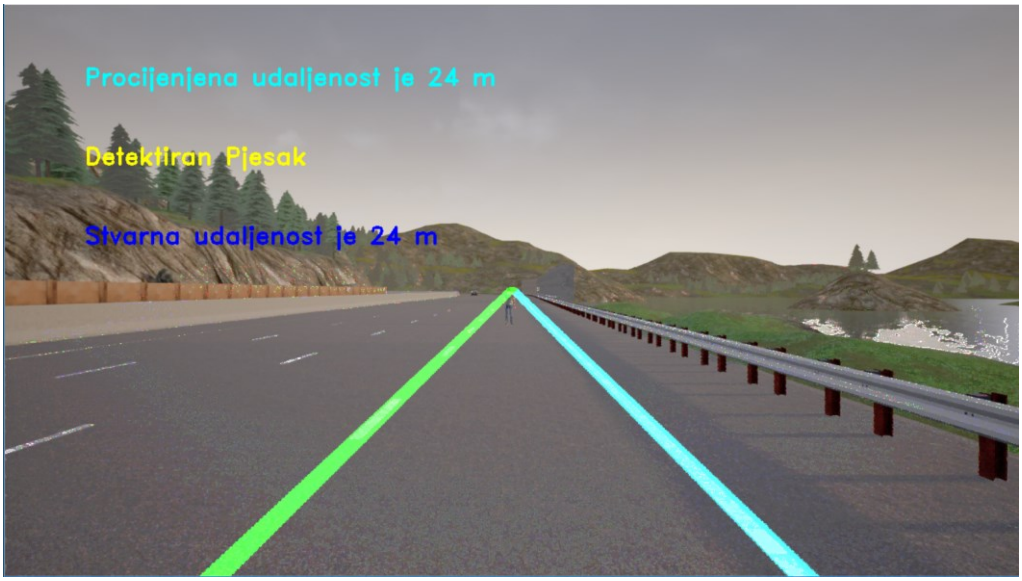
Na slikama 4.1. do 4.4., prikazani su primjeri primjene predloženog algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila. Na svakoj slici zelenom i bijelom bojom su prikazane detektirane kolničke oznake (lijeva i desna) i ispisan je rezultat klasifikacije prepreke te procijenjena i stvarna udaljenost prepreke od vozila. U prilogu P.4.1. priložene su sve slike dobivene primjenom predloženog algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila nad testnim skupom.



Slika 4.1. Rezultat algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila kada je prepreka prepoznata kao automobil i kada je udaljenost od iste uspješno detektirana.

Slika 4.2. Rezultat algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila kada je automobil detektiran kao pješak.

Slika 4.3. Rezultat algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila kada je pješak detektiran kao automobil.



Slika 4.4. Rezultat algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila kada je prepreka prepoznata kao pješak i kada je udaljenost od iste uspješno detektirana.

4.2. Verifikacija rada predloženog algoritma simulacijom vožnje unutar simulatora Carla

Dodatna verifikacija predloženog algoritma provedena je simulacijama vožnje unutar simulatora *Carla*, pri čemu je cilj zaustaviti vozilo na određenoj udaljenosti od prepreke u voznoj traci. Kako bi se izvelo testiranje postavljeno je vozilo u sredinu vozne trake i na koje je moguće programski utjecati na brzinu kretanja (gas i kočnica), te je postavljena prepreka u voznu traku (vozilo ili pješak). Zbog ograničenja simulatora, testiranje je provedeno na brzinama kretanja vozila od 10 do 70 kilometara na sat. Testiranje je obavljeno tako da je vozilo postavljeno na određenu udaljenost potrebnu da se postigne željena brzina, ali da se također pokrije i put kočenja, kao i sigurna udaljenost zaustavljanja do prepreke. U prilogu P.4.2. priložene su video sekvence koje prikazuju simulacije vožnje unutar simulatora pri verifikaciji rada predloženog algoritma. Unutar tablice 4.4. prezentirani su rezultati simulacije unutar simulatora gdje su u obzir uzeta dva slučaja, kada je potrebno zaustaviti vozilo na 5 i 10 metara od prepreke.

Tablica 4.4. Izmjerena udaljenost vozila od prepreke nakon zaustavljanja kada je brzina vozila u rasponu od 10 do 70 km/h i kada je prepreka automobil.

Brzina [km/h]	Izmjerena udaljenost nakon zaustavljanja (očekivano 5 m)	Izmjerena udaljenost nakon zaustavljanja (očekivano 10 m)
10	5	10
20	4	9
30	3	8
40	2	7
50	2	6
60	KOLIZIJA	3
70	KOLIZIJA	KOLIZIJA

Na temelju tablice 4.4. moguće je vidjeti da se vozilo uspješno zaustavi na definiranoj udaljenosti od prepreke samo u slučaju vrlo niskih brzina (10 km/h). Za veće brzine kretanja vozila,

vozilo se zaustavlja na manjoj udaljenosti od prepreke nego što je definirano. Ovo može nastati zbog pogreške u određivanju funkcijske ovisnosti udaljenosti prepreke i elemenata slike koji pripadaju prepreci (što je vidljivo iz rezultata priloženih u tablici 4.1.) i zbog pogreške u eksperimentalnom određivanju zaustavnog puta (slika 3.11.). Pri većim brzinama vozila problem nastaje zbog nedovoljne brzine obrade video okvira (engl. ***F**rames **p**er **s**econd*), zbog čega pri brzinama od 60 i 70 km/h dolazi i do kolizije vozila s preprekom kada očekivana udaljenost od prepreke nakon zaustavljanja vozila iznosi 5 metara.

Osim navedenih testiranja kada je prepreka vozilo, provedena su testiranja kada je prepreka pješak. Unutar tablice 4.5. prikazani su rezultati simulacija brzinom od 10 do 70 km/h kada je prepreka pješak.

Tablica 4.5. Izmjerena udaljenost vozila od prepreke nakon zaustavljanja kada je brzina vozila u rasponu od 10 do 70 km/h i kada je prepreka pješak.

Brzina [km/h]	Izmjerena udaljenost nakon zaustavljanja (očekivano 5 m)	Izmjerena udaljenost nakon zaustavljanja (očekivano 10 m)
10	5	10
20	4	9
30	3	8
40	1	5
50	KOLIZIJA	3
60	KOLIZIJA	1
70	KOLIZIJA	KOLIZIJA

Kod scenarija s pješakom kao preprekom zabilježene su pri nižim brzinama slične vrijednosti kao i kada je prepreka vozilo. Međutim, pri većim brzinama potrebno je ranije započeti s kočenjem, što znači da je potrebno estimirati udaljenost od prepreke pri znatno većim vrijednostima udaljenosti. Zbog toga što predloženi algoritam lošije procjenjuje udaljenost pješaka

nego automobila (vidi rezultate u tablici 4.1.), moguće je uočiti veća odstupanja od očekivanih udaljenosti nakon zaustavljanja vozila. Kao i kada je prepreka vozilo, postoji određena pogreška u eksperimentalnom određivanju zaustavnog puta. Također, pri većim brzinama ponovno je došlo do kolizije s preprekom što je posljedica nedovoljne brzine obrade video okvira u sekundi.

Iz navedenog moguće je zaključiti da je navedena metoda donekle uspješno ispunila temeljnu zadaću. Međutim, prilikom izvođenja RANSAC-a svaki put se uzimaju nasumični elementi slike, zbog čega njegov izlaz neće uvijek biti jednak, pa zbog toga rezultati testiranja mogu varirati. Samo izvođenje iteracija RANSAC-a zahtijeva određeno vrijeme, koje ovisi u o broju elemenata slike koji se trenutno procesiraju, što može utjecati na broj okvira u sekundi koji se obrađuju. Također, zbog toga što je sam simulator vrlo zahtijevan, broj okvira u sekundi koji se obrađuju je dodatno smanjena. Međutim, algoritam je ipak uz određeno odstupanje u većini slučajeva približno detektirao udaljenost od prepreke i prilagodio kretanje vozila kako bi zakočio bez kolizije s preprekom. Ipak, zbog povećanja sigurnosti, ako se radi o većim brzinama potrebno je definirati veću udaljenost od prepreke na kojoj je potrebno zakočiti.

5. Zaključak

Detekcija vozne trake, odnosno udaljenosti od prepreka koje se nalaze unutar vozne trake važni su zadaci sustava autonomnog kretanja vozila jer omogućavaju planiranje budućeg kretanja vozila i pravovremenu reakciju na situaciju unutar vozne trake. U okviru rada razvijen je i testiran algoritam za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila. Za ispunjenje funkcionalnosti korišteni su C++ programski jezik, ROS i *OpenCV* biblioteka, dok je za verifikaciju rada korišten *Carla* simulator. Predloženi algoritam se sastoji od više koraka, gdje je važno detektirati voznu traku te klasificirati i estimirati udaljenost vozila od prepreke unutar vozne trake. Detekcija vozne trake nije trivijalan zadatak jer se unutar svakog video okvira pojavljuje šum, koji otežava pronalazak pravaca koji predstavljaju kolničke oznake. Detekcija pravaca se obavlja tako da se nakon dohvaćanja video okvira prvo koriste metode koje služe za predobradu slike i detekciju rubova. Tada se na osnovu elemenata slike binarne slike primjenjuje RANSAC algoritam kako bi se dobili pravci koji najbolje opisuju kolničke oznake. Pomoću detektiranih kolničkih oznaka definirano je područje vozne trake. Područje vozne trake se zatim pretražuje kako bi se klasificirala prepreka, ali i procijenila udaljenost od iste. Algoritam je za testiran na skupu slika pri čemu je prepreka na udaljenostima od 5 do 80 metara gdje je u većini slučajeva ispravno klasificirao prepreku. Također na osnovu izmjerenih vrijednosti kreiran je model procjene udaljenosti gdje je algoritam kada je drugo vozilo bilo prepreka procijenilo udaljenost uz srednje odstupanje od 1.68 metra, dok je srednje odstupanje iznosilo 2.3 metra kada je prepreka pješak. Odstupanje procijenjene od stvarne udaljenosti raste porastom udaljenosti, međutim ponovnom detekcijom u sljedećim video okvirima gdje je prepreka bliže moguće je ispraviti dobivene pogrešne udaljenosti. Također, algoritam je verificiran pomoću simulacije vožnje unutar simulatora brzinama od 10 do 70 km/h. Rezultati simulacija su pokazali da se upravljano vozilo pri nižim brzinama zaustavlja na unaprijed definiranoj udaljenosti od prepreke, dok je za veće brzine stvarna udaljenost na kojoj se zaustavi vozilo manja od definirane pa čak može nastupiti i kolizija zbog pogrešaka prilikom procjene udaljenosti od prepreke, ali i nedovoljne brzine obrade video okvira u sekundi.

Literatura

[1] Prometna signalizacija: Uzduzne oznake, https://www.prometna-signalizacija.com/horizontalna- signalizacija/uzduzne-oznake/, pristup ostvaren: Kolovoz 2019.

[2] Autonomous Vehicles and Their Sensors, https://www.azocleantech.com/article.aspx?ArticleID=834, pristup ostvaren: Srpanj 2019.

[3] M. Aly: Real time Detection of Lane Markers in Urban Streets, IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, Proceedings. 7 - 12. str., 2008.

[4] Y. Choi, Ju H. Park, Ho-Y. Jung: Lane Detecton Using Labeling Based RANSAC Algorithm, World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering Vol:12, No:4, 245-248. str., 2018.

[5] P. Amaradi, N. Sriramoju, Li Dang, G. S. Tewolde and J. Kwon: Lane following and obstacle detection techniques in autonomous driving vehicles, 2016 IEEE International Conference on Electro Information Technology (EIT), Grand Forks, The Netherlands, 674.-679. str., 2016.

[6] X. Pan, J. Shi, P. Luo, X. Wang, X. Tang, Spatial As Deep: Spatial CNN for Traffic Scene Understanding, AAAI Conference on Artificial Intelligence (AAAI), New Orleans, USA, 7276.-7283.str., Veljača 2018.

[7] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, Y. Bengio: ReNet: A Reccurent Neural Network Based Alternative to Convolutional Network, arXiv preprint arXiv:1505.00393, 2015.

[8] K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition, The Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 770.-778. str. 10. Prosinac 2015.

[9] P. Krahenbuhl, V. Koltun, Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, Advances in Neural Information Processing Systems 24(2011), 109.-117. str., 20. Listopad 2012.

[10] Z. Wang, W. Ren, Q. Qiu, LaneNet: Real-Time Lane Detection Networks for Autonomous Driving, arXiv preprint arXiv:1807.01726, 2018.

[11] D. Levi, N. Garnett, E. Fetay: StixelNet: A Deep Convolutional Network for Obstacle Detection and Road Segmentation, British Machine Vision Conference 2015., Swansea, Wales, Paper 109 – 1.-12. str., Siječanj 2015.

[12] H. Badino, U. Franke, and D. Pfeiffer. The stixel world - a compact mediumlevel representation of the 3d-world, Pattern Recognition, 31st DAGM symposium, Jena, Germany, 51.-60. str., 2009.

[13] A. J. Viterbi: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.IEEE Transactions on Information Theory, 260.-269. str., 1967.

[14] OpenCV 3.4.3 documentation, https://docs.opencv.org/3.4.3/, pristup ostvaren: Srpanj 2019.

[15] M. Šarić, Robotski operacijski sustav ROS, Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku, Osijek, 2018.

[16] ROS.org Documentation, http://wiki.ros.org/Documentation, pristup ostvaren: Srpanj 2019.

[17] Don Wilcher, ROS 101: An Intro to the Robot Operating System, https://www.designnews.com/gadget-freak/ros-101-intro-robot-operating-system/107053141061075, pristup ostvaren: Srpanj 2019.

[18] Carla simulator documentation, https://carla.readthedocs.io/en/latest/, pristup ostvaren: Kolovoz 2019.

[19] Geometric image transformations, OpenCV documentation, https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html, pristup ostvaren: Srpanj 2019.

[20] S. Rimac-Drlje, M. Vranješ, predavanja iz kolegija Digitalna obrada slike i videa za autonomna vozila, studij Automobilsko računarstvo i komunikacije, ak.god. 2018/2019., https://loomen.carnet.hr/pluginfile.php/1536272/mod_resource/content/2/3.%20Postupci%20za%20predobradu%20slike.pdf , pristup ostvaren: Srpanj 2019.

[21] Robert Collins, Robust Estimation : RANSAC, Lecture 15., Penn State, USA http://www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf, pristup ostvaren: Kolovoz 2019.

Sažetak

Tema ovog diplomskog rada je izrada algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila. Algoritam se temelji na obradi slika koje se dobivaju s kamere montirane na prednjoj strani vozila pomoću ROS-a. Obrada slika obavljena je metodama računalnog vida, koje su ostvarene pomoću programskog jezika C++ i biblioteke *OpenCV*. Kada je slika dohvaćena, izvršavaju se metode predobrade slike, kao i detekcija rubova kako bi se smanjio šum. Osim toga, koristi se RANSAC algoritam pomoću kojeg se definira područje vozne trake koje se koristi za klasifikaciju i estimaciju udaljenosti od prepreke ako ona postoji, kako bi se pomoću ROS-a omogućilo zaustavljanje vozila. Prilikom verifikacije korišten je *Carla* simulator, gdje je zaključeno da algoritam na slikama vrlo precizno detektira udaljenost od prepreke. Analizom simulacija vožnje utvrđeno je da algoritam zbog nedovoljne brzine obrade video okvira bez dodatnog proširenja nije prikladan za izvođenje pri većim brzinama vozila.

Ključne riječi: C++, ROS, Carla, OpenCV, detekcija vozne trake, detekcija prepreke, autonomna vožnja, ADAS, računalni vid

AN ALGORITHM FOR DETECTING OBSTACLES WITHIN THE LANE AND VEHICLE SPEED ADJUSTMENT

Abstract

The topic of this thesis is the development of an algorithm for obstacle detection within the lane and adjusting vehicle speed. The algorithm is based on the processing of images obtained from a camera mounted on the front of the vehicle using ROS. Image processing was done using computer vision methods, which were accomplished using the C++ programming language and the OpenCV library. Once the image is retrieved, image pre-processing methods are performed as well as edge detection to reduce noise. Moreover, a RANSAC algorithm is used to define the lane area used to classify and estimate the distance from an obstacle, to stop the vehicle by using ROS if necessary. Carla simulator was used for verification, where it was concluded that the algorithm on pictures detects the distance from the obstacle very accurately. Analyzing the results of driving simulations, it is concluded that due to insufficient processing speed of the video frames without future work, the algorithm is not suitable for execution at higher vehicle speeds.

Keywords: C++, ROS, Carla, OpenCV, driving lane detection, obstacle detection, autonomous driving, ADAS, computer vision

Životopis

Srđan Ljepić rođen je 8. ožujka 1996. u Vukovaru. Završio je Osnovnu školu „Tenja“ u Tenji. Nakon toga upisuje II. gimnaziju u Osijeku koju uspješno završava 2014. godine. Iste godine upisuje preddiplomski smjer Računarstva na Fakultetu Elektrotehnike, Računarstva i Informacijskih Tehnologija koji završava 2017. godine. Trenutno pohađa drugu godinu diplomskog studija Automobilskog računarstva i komunikacija.

Prilozi

Prilog P.3.1. Programski kod

Programski kod predloženog rješenja za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila koji se sastoji od ImageConverter klase, main getMeterDistance, getPointPositions i findLanesFunction funkcija nalazi se na priloženom CD-u.

Prilog P.4.1. Skup slika koje su dobivene primjenom predloženog algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila nalazi se na priloženom CD-u

Prilog P.4.2. Skup video sekvenci koje predstavljaju simulacije vožnje dobivene primjenom predloženog algoritma za detekciju prepreka unutar vozne trake i prilagodbu brzine vozila nalazi se na priloženom CD-u