Detekcija voznih traka ispred vozila pomoću kamere i upozoravanje na nepoželjno napuštanje trake

Špoljar, Domagoj

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

Sveučilišni studij računarstva

DETEKCIJA VOZNIH TRAKA ISPRED VOZILA POMOĆU KAMERE I UPOZORAVANJE NA NAPUŠTANJE TRAKE

Diplomski rad

Domagoj Špoljar

Osijek, 2019. godina.

SADRŽAJ

1. UVOD

U posljednjih desetak godina došlo je do znatne promjene trendova u automobilskoj industriji. Autonomna vozila su sve više prisutna na cestama dok razina samostalnosti takvih vozila s vremenom raste. Porast samostalnosti vozila u donošenju odluka u konačnici bi trebala dovesti do potpuno autonomne vožnje, čime vozilo zamjenjuje vozača u vožnji, a samim time bi trebala biti povećana i sigurnost vožnje. Autonomna vozila opremljena su raznim senzorima od kojih je jedan od najvažnijih kamera koja se nalazi na prednjem kraju automobila. Pomoću kamere vozilo može vidjeti što se nalazi ispred njega te detektirati razne objekte koji se nalaze na cesti. Tako npr., kako bi automobil ostao unutar svoje vozne trake tijekom vožnje, potrebno je pravilno detektirati vlastitu voznu traku, kao i susjedne vozne trake.

Metoda detekcije vozne trake zasnovana na obradi slike ključna je u suvremenim sustavima za pomoć u vožnji (*engl. Advanced Driver Assistance Systems - ADAS*). ADAS sustavi se u posljednje vrijeme sve više koriste te se sve više inženjera bavi izradom programskih rješenja za napredne ADAS algoritme. Cilj tih algoritma je povećanje sigurnosti za sve sudionike prometa i samim time smanjenje broja nesreća i broja stradalih u prometu. Jedan od takvih naprednih ADAS algoritama je onaj za detekciju voznih traka, čime se pomaže vozaču da zadrži vozilo u svojoj voznoj traci. Međutim, otkrivanje voznih traka je i dalje izazovno iz nekoliko razloga. Voznu traku definiraju dvije paralelne linije na cesti kojima se označava prostor same vozne trake. Za početak, izgled trake obično je vrlo jednostavan, zbog čega ne postoje nekakve složene ili karakteristične značajke kojima se linije vozne trake mogu otkriti. Time se dramatično povećava rizik od lažne pozitivne detekcije vozne trake. Nadalje, različiti uzorci linija, kao npr. pune linije, isprekidane linije, dvostruke linije ili spajajuće trake, čine računalno modeliranje trake iznimno kompliciranim. Ako vozilo može uspješno detektirati vozne trake, moguće je razviti sustav upozoravanja vozača o napuštanju vlastite vozne trake i samim time spriječiti potencijalnu koliziju s drugim vozilima i objektima u prometu.

U ovom radu predložen je novi računalni algoritam koji detektira vozne trake na cesti ispred vozila na osnovu slike dobivene s kamere na prednjoj strani vozila te upozorava vozača ukoliko dolazi do napuštanja trenutne vozne trake. Algoritam detektira položaje vlastite i dviju susjednih voznih traka na cesti. Mogućnost detekcije voznih traka mora biti neovisna o uzorku same linije koja definira voznu traku. Uz to, potrebno je da algoritam bude prilagodljiv

promjenama svjetlosnih i vremenskih uvjeta, položaju ostalih cestovnih vozila te vrstama cesta. Cijeli algoritam izrađen je u višem programskom jeziku, točnije u Python programskom jeziku na Ubuntu operacijskom sustavu.

U nastavku rada, u drugom poglavlju detaljnije su opisane postojeće metode detekcije voznih traka. Objašnjeni su općeniti načini rada dosadašnjih algoritma kao i njihove prednosti i nedostatci. Nakon toga u trećem poglavlju objašnjen je kompletan proces izrade novog vlastitog algoritma za detekciju voznih traka i upozorenje o napuštanju vozne trake, sa svim koracima i korištenim metodama. Četvrto poglavlje opisuje način verifikacije ispravnosti rada algoritma i rezultate testiranja na video sekvencama iz prometa. U petom poglavlju dan je zaključak rada.

2. PROBLEM DETEKCIJE VOZNIH TRAKA U SLICI DOBIVENOJ S KAMERE NA VOZILU

Računalna lokalizacija granica vozne trake može se podijeliti na dva pod-zadatka: detekcija vozne trake te praćenje vozne trake. Detekcija vozne trake je problem lociranja granica vozne trake bez prethodnog poznavanja geometrije ceste. Algoritmi za detekciju voznih traka proučavaju se već duže od 20 godina [1]. Postoji mnogo radova koji različitim pristupom i metodama detektiraju vozne trake na cesti. Općenito, metode detekcije voznih traka mogu se podijeliti u tri kategorije: metode zasnovane na postavljanu modela voznih traka iz detektiranih rubova (*engl. model based methods),* metode zasnovane na značajkama u slici (*engl. feature based methods*) te metode zasnovane na modelima neuronskih mreža (*engl. neural network based methods*). Napisano je nekoliko preglednih radova [2], [3] koji opisuju različite postojeće algoritme detekcije voznih traka te će se u ovom radu opisati samo nekoliko predstavnika pojedinog pristupa koji se koristi pri detekciji voznih traka. Generalno, svi pristupi imaju zajedničku karakteristiku, koja je zapravo ulazni podatak u algoritam u obliku slike snimljene kamerom na prednjoj strani automobila. S druge strane, praćenje vozne trake je problem praćenja rubova vozne trake od okvira do okvira u snimci, s obzirom na postojeći model geometrije ceste. Mnoge tehnike se koriste za praćenje traka. Među njima se može spomenuti Kalmanovo filtriranje [4] koje je zapravo algoritam koji koristi niz mjerenja promatranih tijekom vremena te izrađuje procjene nepoznatih varijabli, u ovom slučaju položaja linija voznih traka, koje su obično preciznije od onih koje se zasnivaju na samo jednom mjerenju. Praćenje vozne trake je lakši problem od detekcije trake, budući da prethodno poznavanje geometrije ceste omogućuje algoritmima za praćenje prometnih traka prilično jaka ograničenja na vjerojatno mjesto i orijentaciju rubova vozne trake u novoj slici. Algoritmi za detekciju traka, s druge strane, moraju locirati rubove voznih traka bez snažnog modela geometrije ceste, i to u situacijama u kojima može biti mnogo smetnji na slici. Uz to, algoritmi za detekciju voznih traka mogu se susresti s raznim izazovima prilikom rada [3]. Često će kod razvijanja algoritma za detekciju voznih traka doći do problema loših oznaka linija voznih traka na kolniku (npr. izlizane oznake). Isto tako moguće je da su oznake linija voznih traka zaklonjene drugim vozilima na cesti. U navedenim situacijama je vrlo teško detektirati linije voznih traka jer značajke po kojima se linije voznih traka detektiraju nisu prisutne u okvirima snimke snimljene kamerom na prednjem kraju vozila. Još jedan izazov s kojim se algoritmi za detekciju voznih traka mogu susresti je otežana detekcija linija voznih traka uzrokovana lošim vremenskim

uvjetima na slici. Pojava kiše ili magle smanjuje vidljivost ceste te stvara dodatni šum na slici zbog kojeg linije voznih traka nisu jasno vidljive ili uopće nisu vidljive. Isto tako pojava snijega na cesti može prouzrokovat netočnu detekciju linija voznih traka jer izgled snijega sadrži vrlo slične značajke kao i oznake linija voznih traka (npr. boja snijega i oznaka linija voznih traka bijele boje je ista). Zbog toga je važno u početnoj fazi razvoja vlastitog algoritma obratiti pozornost na navedene probleme kako bi se oni mogli eliminirati u radu vlastitog algoritma. U nastavku će se detaljnije opisati nekoliko postojećih metoda detekcije voznih traka.

U prvim radovima koji se bave detekcijom voznih traka korištene su tehnike koje su se zasnivale na postavljanu modela voznih traka iz detektiranih rubova. Modeli linija voznih traka koji se najčešće koriste uključuju linearni model [5], parabolični model [6] ili model hiperbole [7]. Kod linearnog modela problem u radu algoritma stvaraju vozne trake koje su zakrivljene, zbog čega se model ne može ispravno prilagoditi na zakrivljenu liniju budući da je ograničen činjenicom da može modelirati samo ravnu liniju, tj. pravac. Bolju detekciju ostvaruju parabolični model koji može ispravno modelirati zakrivljene vozne trake te hiperbolični modeli koji može ispravno modelirati bilo koji oblik zakrivljenosti vozne trake poput „S krivulje“. Zbog toga se najčešće koriste parabolični ili hiperbolični modeli u modeliranju voznih traka. Većina metoda otkrivanja traka u ovom pristupu zasniva se na korištenjem metoda detekcija rubova na slici (*engl. edge-based methods*). Pomoću detektiranih rubova na slici, moguće je jednostavno odrediti parametre geometrijskog modela. Aly u svom radu [8] opisuje korištenje prikladnog geometrijskog modela za opisivanje linija voznih traka, a zatim dobivanja parametra geometrijskog modela pomoću konsenzusa slučajnog uzorka (*engl. random sample consensus* - RANSAC) [9]. Za evaluaciju performansi rada koristio je skup podataka koji sadrži razne prikaze scena gradske vožnje snimljene kamerom na prednjoj strani automobila. U navedenom radu prvi puta je postignuta detekcija svih voznih traka na ulaznoj slici. Nedostatak tog rada je činjenica da je ostvarena samo detekcija bez praćenja voznih traka te pojava lažnih pozitivnih detekcija linija vozne trake na mjestima gdje se one ne nalaze. Osim RANSAC-a, u metode dobivanja parametra geometrijskog modela spadaju i razne druge metode poput metode najmanjih kvadrata [10], Houghove transformacije [11] i slično. Glavna prednost ovakvog pristupa je da se vozna traka može pratiti raznim statističkim metodama, tako da su lažne detekcije ovim pristupom gotovo u potpunosti izbjegnute. Obrada slike i izračun svih parametara kod ovakvog pristupa računalno su vrlo zahtjevni, zbog čega algoritmi nisu prikladni za primjenu u sustavima koji trebaju raditi u stvarnom vremenu (*engl. real-time*

systems). Uz to, algoritmi zasnovani na postavljanu modela voznih traka sadrže ručno izrađene značajke za detekciju voznih traka, čime se unaprijed određuju značajke različitih tipova linija voznih traka u programskom kodu. Zbog toga takav sustav nije potpuno prilagodljiv za detekciju voznih traka u svim mogućim situacijama u prometu.

Uz algoritme koji koriste geometrijske modele voznih traka postoje i algoritmi koji se zasnivaju na metodama zasnovanim na značajkama na slici. Metode zasnovane na značajkama obično izoliraju i lokaliziraju vozne trake na slikama filtrirajući iz njih značajke niske razine. Neke značajke niske razine kao što je boja [12], širina [13], rub [14], tekstura [15] i gradijentna promjena [16] koriste se za izdvajanje linije trake s područja ceste. U radu [17] slike prikaza ceste u boji se pretvaraju u HSV prostor boja (*engl. colorspace*) s ciljem boljeg detektiranja značajki prometnih traka bijele boje. HSV prostor boja je način zapisa boja u obliku tri kanala H, S i V gdje H predstavlja ton boje, S zasićenje boje a V intenzitet boje. Prednosti korištenja HSV prostora boja u navedenom radu je taj što se njime linije vozne trake na slici jednostavnije segmentiraju nego kad se koriste ostali prostora boja. Metode zasnovane na značajkama na slici zahtijevaju definiranje određenog praga (*engl. threshold*) koji služi za izdvajanje značajka niske razine iz slike. U navedenom radu korišten je određeni prag kojim se izdvajaju značajke linija voznih traka iz HSV kanala prostora boja. U mnogim scenama na cestama nije moguće odabrati prag koji eliminira sve smetnje, a da pritom ne eliminira mnoge točke interesa na rubovima vozne trake. Zbog toga modeli zasnovani na značajkama nisu prilagodljivi za razne promjene osvjetljenja okoline na cesti.

Proteklih godina došlo je do značajnog broja inovacija u detekciji i praćenju voznih traka zasnovanih na modelima umjetnih neuronskih mreža i dubokom učenju (*engl. Deep learning*). Umjetna neuronska mreža predstavlja niz algoritama koji nastoje prepoznati temeljne odnose u skupu podataka kroz proces koji oponaša način na koji ljudski mozak djeluje [18]. Neuronske mreže mogu se prilagoditi promjenama ulaza tako da mreža stvara najbolji mogući rezultat bez potrebe za redizajnom izlaznih kriterija. Duboka neuronska mreža je neuronska mreža koja osim ulaznog i izlaznog sloja sadrži dodatne, skrivene, slojeve koji povećavaju razinu složenosti mreže [19]. Duboke neuronske mreže uzrokovale su značajan napredak u području računalnog vida u proteklom desetljeću budući da je povećana računalna sposobnost dovela do napretka u učinkovitim strukturama dubokih neuronskih mreža. Duboka računalna struktura pokazala se korisnom u širokom rasponu područja primjene, uključujući prepoznavanje govora [20], obradu

prirodnog jezika [21], i relevantno za ovaj rad, detekciju objekata na slici [22]. D. Neven i njegovi suradnici [23] predstavljaju algoritam za detekciju voznih traka koji se zasniva na konvolucijskoj neuronskoj mreži, obliku umjetne neuronske mreže koji je specijaliziran za obradu višedimenzionalnih podataka koji imaju prostornu hijerarhiju, poput slike. Ulazna slika ceste prolazi kroz neuronsku mrežu gdje se kao izlaz iz mreže dobiva binarna slika koja označava vozne trake na slici i samim time pruža detekciju voznih traka. Novost u navedenom radu je posebna struktura neuronske mreže koja uz stvaranje binarne slike linija voznih traka kreira i segmentacijsku sliku ceste koja zajedno s binarnom slikom omogućuje detekciju voznih traka na slikama gdje linije nisu vidljive ili jasno označene na cesti. Kako bi neuronska mreža ispravno prepoznavala i izdvajala vozne trake sa slike, potrebno je prvotno istrenirati neuronsku mrežu sa slikama ceste na kojima su već označene vozne trake. Ovakav pristup zahtjeva veliki broj slika kojim se neuronska mreža trenira, često reda nekoliko tisuća ili desetaka tisuća slika. Prednost ovakvog pristupa je njegova velika brzina obrade slike i točnost detekcije voznih traka. U navedenom radu [23] koriste *tuSimple* bazu podataka [24] za treniranje neuronske mreže te su nakon treniranja uspjeli postići brzinu rada mreže do 50 okvira u sekundi (*engl. Frames per second - FPS*), što je pogodno za implementaciju u razne sustave stvarnog vremena.

3. VLASTITI ALGORITAM ZA DETEKCIJU VOZNIH TRAKA I UPOZORENJE O NAPUŠTANJU VOZNE TRAKE

Zadatak ovog rada je osmisliti i implementirati računalni algoritam za detekciju voznih traka i upozoravanje vozača o napuštanju trenutne vozne trake. Potrebno je detektirati vlastitu voznu traku kao i susjedne vozne trake. Vozne trake se detektiraju pomoću obrade okvira (*engl. frame*) iz snimke snimljene kamerom na prednjoj strani vozila. Uz to, potrebno je osmisliti način detekcije napuštanja trenutne vozne trake u kojoj se nalazi vozilo te upozoriti vozača o napuštanju vozne trake ispisivanjem teksta upozorenja na izvornoj snimci.

Kompletni algoritam sastoji se od nekoliko jednostavnijih koraka koji sekvencijalno obrađuju okvir snimke i prosljeđuju potrebne informacije kroz algoritam kako bi se naposljetku prikazale detektirane linije u okviru snimke. Kreiran je i dodatni proces u algoritmu koji omogućuje dinamički pronalazak najboljih parametara obrade okvira slike kako bi se omogućila prilagodljivost detekcije voznih traka u novonastalim uvjetima na cesti. U nastavku su opisani korišteno razvojno okruženje te podešavanje i instalacija potrebnih biblioteka za rad algoritma. Nakon toga opisan je svaki korak algoritma zasebno, s popratnim slikama rezultata pojedinih koraka.

3.1. Podešavanje i instalacija potrebnih biblioteka

Vlastito rješenje za detekciju voznih traka i upozorenje o napuštanju vozne trake izrađeno je u Python programskom jeziku uz OpenCV i NumPy biblioteke na Ubuntu 16.04 LTS operacijskom sustavu.

OpenCV (*engl. Open Source Computer Vision*) je biblioteka otvorenog koda koja sadrži programske funkcije za računalni vid i strojno učenje. Izgrađen je kako bi osigurao zajedničku infrastrukturu za aplikacije računalnog vida i ubrzao korištenje strojne percepcije u komercijalnim proizvodima. Biblioteka sadrži više od 2500 optimiziranih algoritama, među kojima je i mnogo najsuvremenijih algoritama u području računalnog vida i strojnog učenja. OpenCV se primjenjuje u razne svrhe: od uklanjanja „crvenih očiju" s fotografija, praćenja pokreta očiju, prepoznavanja krajolika i lica na fotografijama, pa sve do praćenja objekata u pokretu te stvaranja 3D modela objekta i slično. Postoje implementacije za C++, Python i Java programske jezike uz podršku za Windows, Linux, Android i MacOS operacijske sustave [25]. Za instalaciju OpenCV biblioteke na Ubuntu operacijskom sustavu dovoljno je u terminalu napisati naredbu:

sudo pip install opencv-python

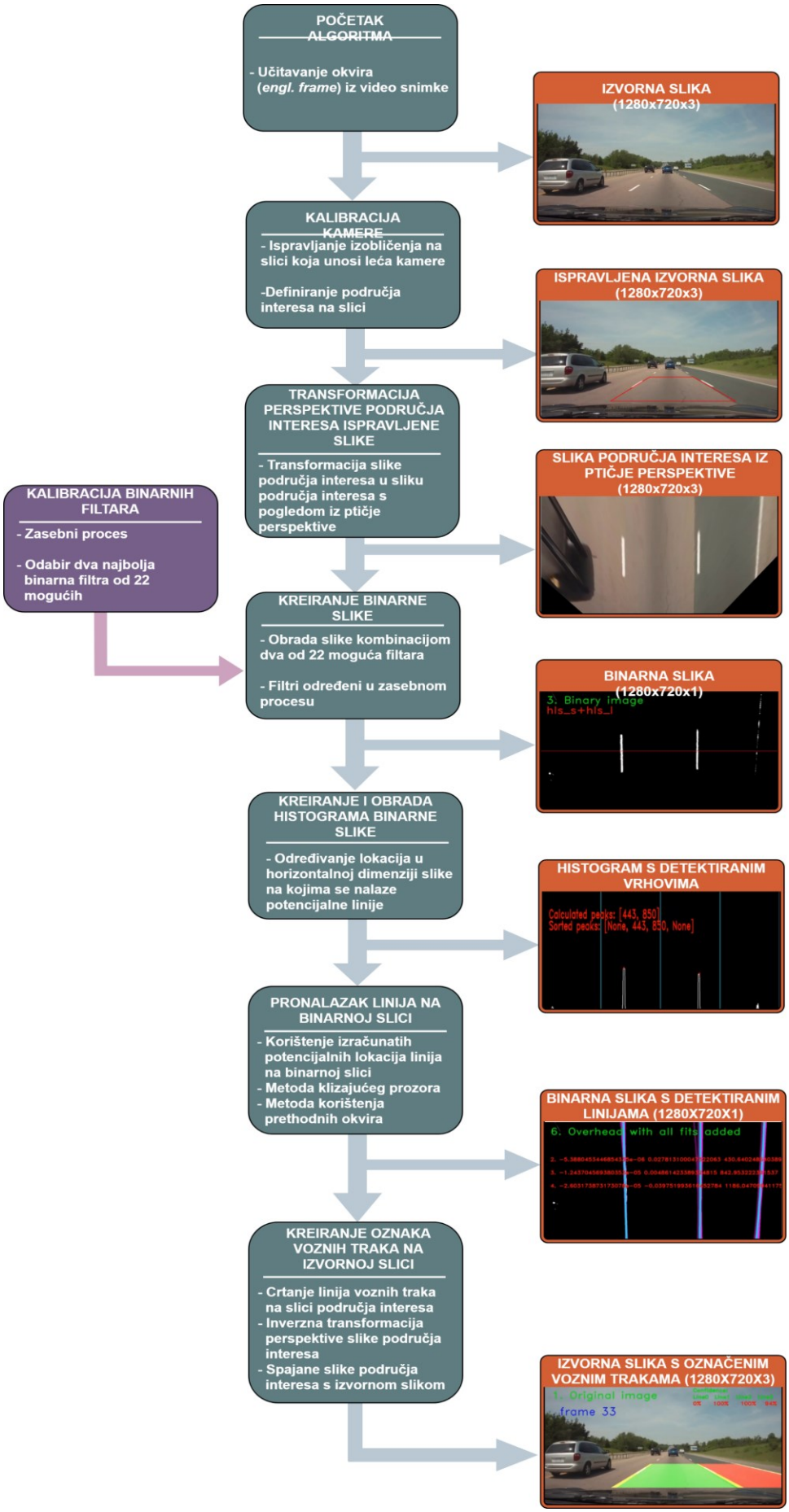
NumPy jest biblioteka otvorenog koda za znanstveno računanje u Python programskom jeziku. Sadrži podršku za velike, višedimenzionalne nizove i matrice s velikom zbirkom matematičkih funkcija visoke razine za rad na tim nizovima. Korištenje NumPy u Python-u omogućuje funkcionalnosti koje su usporedive s korištenjem MATLAB programa. Osim očiglednih znanstvenih primjena, NumPy se također može koristiti kao učinkovit višedimenzionalni spremnik generičkih podataka. Pomoću njega mogu se definirati proizvoljni tipovi podataka. To omogućuje NumPy-ju da se bez problema integrira s velikim brojem baza podataka. Za instalaciju Numpy biblioteke na Ubuntu operacijskom sustavu potrebno je u terminal upisati naredbu [26].

sudo python -m pip install --user numpy

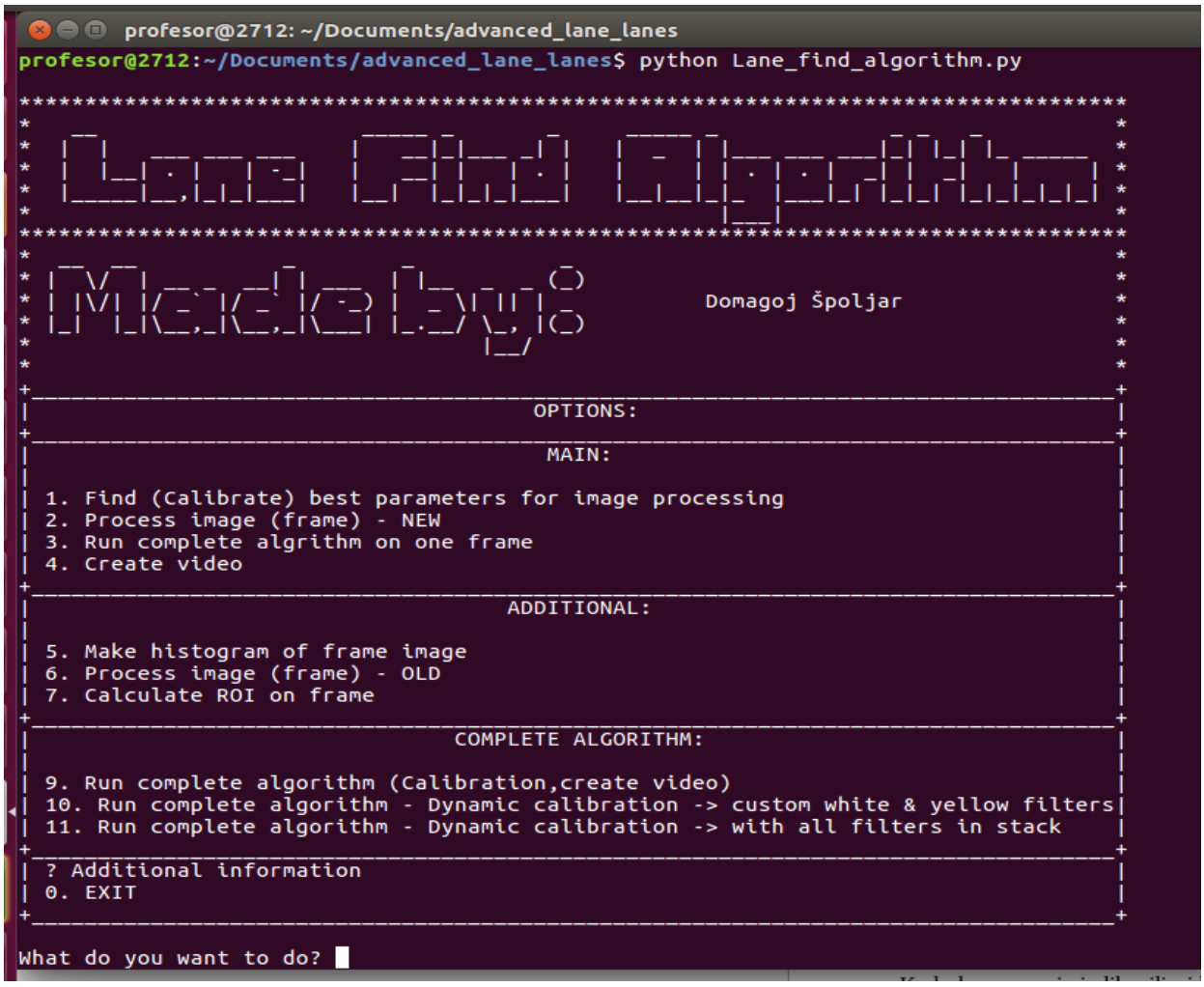
3.2. Opis rada algoritma

Kompletan algoritam sastoji se od više koraka koji se izvršavaju sekvencijalno obradom ulazne slike. U sami algoritam ulazi izvorna fotografija ili video snimljen prednjom kamerom na automobilu, dok se na izlaz dobiva ista fotografija ili video, samo sa označenim voznim trakama. Na slici 3.1. prikazan je dijagram toka rada samog algoritma s informacijama koje se u svakom koraku prosljeđuju na idući korak. U sljedećim dijelovima detaljno je opisan svaki korak u radu algoritma.

Kompletan se algoritam sastoji od nekoliko manjih potprograma koji izvršavaju određenu funkciju. *Lane_find_algorithm.py* je glavni program koji objedinjuje sve kreirane potprograme u jedan te omogućuje razne funkcionalnosti ovisno o željenom odabiru unutar programa. Slika 3.2. prikazuje ispis u upravljačkom prozoru nakon pokretanja glavnog programa sa svim mogućim potprogramima koje sadrži i koji se mogu pokrenuti. Korisnik nakon pokretanja glavnog programa može unosom broja pokrenuti određeni potprogram i testirati različite funkcionalnosti samog algoritma. Ovaj dio je napravljen kako bi se olakšalo ispravljanje grešaka tijekom razvoja vlastitog algoritma.



Slika. 3.1. Dijagram toka rada algoritma za detekciju voznih traka



Slika. 3.2. Ispis glavnog programa Lane_find_algorithm.py u terminalu.

Algoritam se zasniva na obradi svakog okvira unutar snimke. Prvo je potrebno izdvojiti svaki okvir iz snimke pomoću potprograma *frames_to_video.py*. Nakon što je svaki okvir izdvojen i spremljen kao slika, moguće je pokrenuti glavni program te testirati razne funkcionalnosti. Algoritam prolazi redom kroz svaku kreiranu sliku okvira te vrši detekciju linija.

3.2.1. Kalibracija kamere

Kada kamera snimi fotografiju ili video, objektiv kamere ne uhvati pravu sliku, već uhvaćena slika bude na određeni način izobličena. Nastajanje izobličenja može biti posljedica optičkog dizajna leće objektiva, različitog lomljenja zraka svijetlosti pri različitim položajima objektiva i sl. [27]. Zbog deformacije snimljene slike, točke u središtu slike imaju manje izobličenje od točaka udaljenih od središta koje imaju veće izobličenje. Stoga linije koje su ravne u stvarnom svijetu možda više neće biti ravne na snimljenim fotografijama. Dva najčešća oblika distorzije

su pozitivna i negativna radijalna distorzija. Pozitivnu radijalnu distorziju čini „savijanje“ zrake svjetlosti od centra leće prema rubovima, gdje se kutovi slike ugibaju prema unutra (jer je put zrake do njih najduži), pa stranice snimljenog objekta izgledaju kao da su izbočene. Ta vrsta distorzije se još zove i bačvasta distorzija (*engl. barrel image distortion*) [28]. Negativna radijalna distorzija je suprotna pojava gdje se rubne zrake svjetlosti otklanjaju od centra leće, čime se rubovi i kutovi slike izdužuju. Ta pojava se još naziva i jastučasta distorzija (*engl. pincustion image distortion*) [28]. Slika 3.3. prikazuje primjere izobličenja slike šahovnice s pozitivnom (b) i negativnom (c) radijalnom distorzijom.

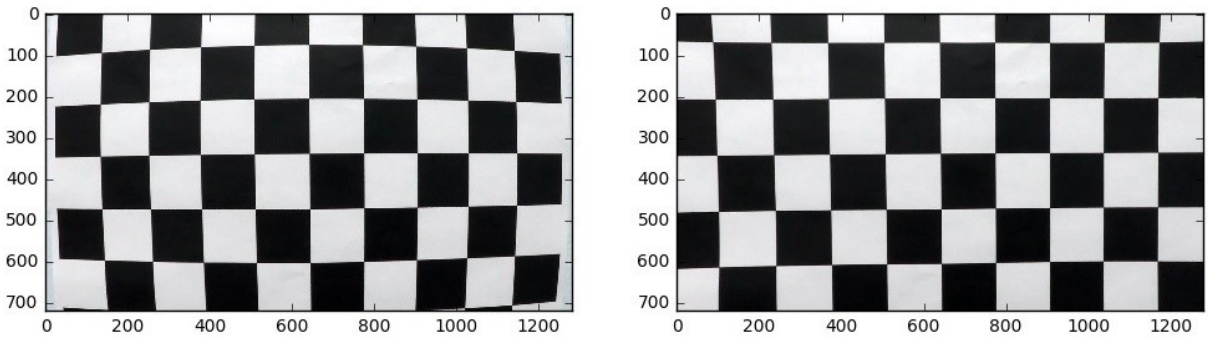
a) b) c)

Slika. 3.3. Primjer izobličenja slike, prikaz šahovnice (a) bez distorzije, (b) s pozitivnom radijalnom distorzijom, (c) s negativnom radijalnom distorzijom.

Zbog potencijalne pojave distorzije zbog leće na ulaznoj slici, prvi je korak u radu algoritma kalibracija kamere kojom se ispravlja navedena distorzija. OpenCV nudi jednostavan način da se to učini pomoću njegovih ugrađenih funkcija. Prvo je potrebno pronaći niz fotografija potrebnih za kalibraciju parametara kamere snimljenih istom kamerom. Vrlo uobičajena tehnika jest snimanje tiskane šahovske ploče iz različitih kutova i izračunavanje distorzija koje je proizvela kamera na temelju očekivane orijentacije šahovnice na fotografiji. Koristeći najmanje deset fotografija, moguće je uspješno izvršiti kalibraciju kamere. Važan ulazni podatak potreban za kalibraciju kamere jest skup 3D točaka iz stvarnog svijeta i odgovarajućih 2D koordinata tih točaka na slici. 3D točke nazivaju se još i objektnim točkama, a 2D točke se nazivaju točkama slike. Točke 2D slike mogu se lako pronaći na slikama šahovske ploče jer su to mjesta gdje se dva crna kvadrata međusobno dodiruju. Pomoću *cv2.findChessboardCorners()* funkcije moguće je pronaći unutrašnje kutove kvadrata šahovnice na snimljenim fotografijama i samim time 2D točke. Potrebno je imati i 3D točke iz stvarnog svijeta. Stoga je potrebno znati X, Y, Z vrijednosti, gdje X vrijednost predstavlja koordinatu točke po X osi, Y vrijednost koordinatu točke po Y osi i Z vrijednost koordinatu

točke po Z osi Kartezijevog trodimenzionalnog koordinatnog sustava. Međutim, zbog jednostavnosti, može se reći da je šahovska ploča ostala nepomična na XY ravnini, tako da je uvijek Z = 0. Ovo razmatranje omogućuje pronalazak samo X, Y vrijednosti koje su zapravo unutarnji kutovi šahovske ploče. Te 3D točke bit će iste za svaku kalibracijsku sliku budući da se očekuje ista šahovska ploča u svakoj slici [29].

Kada su objektne točke i točke slike poznate, idući je korak kalibracija kamere. Najprije je potrebno izračunali matricu izobličenja kamere i parametre izobličenja pomoću navedenih slika šahovnice. Može se koristiti funkcija *cv2.calibrateCamera()* koja pomoću objektnih točaka i točaka slika izračunava koeficijente matrice izobličenja kamere i parametre izobličenja. Nakon što su navedene matrice izračunate, moguće je pomoću *cv2.undistort()* funkcije dobiti sliku ispravljene šahovnice (slika 3.4.). Nakon postupka kalibracije dobiveni su parametri za ispravljanje slike koji se koriste na slikama dobivenim s kamere na prednjoj strani automobila. Na slici 3.5. prikazana je fotografija snimljena pomoću kamere na prednjoj strani automobila i ispravljena slika nakon uklanjanja distorzije koju je unosila kamera. Promjena je vidljiva na donjem dijelu slike gdje se kod ispravljene slike npr. manje vidi prednja strana automobila.



a) b)

Slika. 3.4. Primjer ispravljanja izobličenja slike šahovnice, a) izvorna slika s izobličenjem koje unosi leća kamere, b) ispravljena slika bez izobličenja nakon postupka kalibracije kamere.

a) b)

Slika. 3.5. Primjer ispravljanja izobličenja slike snimljene na prednjoj strani vozila, a) izvorna slika s izobličenjem koje unosi leća kamere, b) ispravljena slika bez izobličenja nakon postupka kalibracije kamere.

3.2.2. Transformacija perspektive slike

Kada se golim okom promatra prikaz određene scene, predmeti koji su bliže izgledaju veći nego predmeti koji se nalaze u daljini. Razlog tomu je taj što oči pretvaraju 3D prostor u 2D sliku čime se gube određene informacije o objektima u prostoru, poput nejednake veličine istih predmeta u daljini u odnosu na veličine tih predmeta u blizini gledišta. Općenito, pogled na prostor iz neke pozicije naziva se perspektivom [30]. Princip na kojemu radi kamera zapravo je sličan principu na kojem djeluje i ljudski vid. Kod prikaza prostora snimljenog kamerom na prednjoj strani automobila, linije voznih traka, koje su u stvarnosti paralelne, na slici ne izgledaju paralelno. Nepravilna reprezentacija voznih traka na 2D slici uzrokuje neispravnu detekciju i modeliranje vozne trake kod daljnje obrade slike u algoritmu. Kako bi se ispravno detektirali položaji i zakrivljenost voznih traka, potrebno je transformirati perspektivu snimljene slike u prikaz ceste iz ptičje perspektive (*engl. bird's eye perspective*). Time zapravo linije voznih traka izgledaju paralelnije naspram linija voznih traka na izvornoj slici.

OpenCV biblioteka sadrži funkcije *getPerspectiveTransform()* i *warpPerspective()* s kojima je moguće uspješno transformirati sliku u neku drugu perspektivu. Kako bi to bilo moguće, potrebno je odrediti koordinate, SRC i DST, kojima se određuju točke transformacije. SRC koordinate predstavljaju četiri točke iz izvorne slike kojima se omeđuje prostor koji se želi transformirati. Te točke u osnovi predstavljaju trapezoid kojim se označavaju pozicije vozne trake s lijeve i desne strane automobila u neposrednoj blizini vozila i u točki nestajanja (*engl. vanishing point*) vozne trake kod horizonta (slika 3.6.). Dio slike omeđen trapezoidom naziva se i područjem interesa (*engl. ROI – Region of Interes,* u daljnjem tekstu: ROI). DST koordinate su pozicije SRC koordinata u transformiranoj slici. Za izradu ovoga rada DST koordinate poprimaju vrijednosti zbog kojih transformirana slika ima istu razlučivost kao i ulazna slika. Funkcije *getPerspectiveTransform()* i *warpPerspective()* kod transformacije ulazne slike će npr. za sliku razlučivosti 1280x720 elemenata slike kreirati transformiranu sliku s razlučivosti od 1280x720 elemenata slike.

Za lakše određivanje ROI-a kao i SRC i DST koordinata razvijen je potprogram s nazivom *click_and_crop.py* s kojim je pojednostavljeno određivanje točaka SRC matrice pomoću odabira točaka pritiskom miša na izvornoj slici. SRC matrica je zapravo skup svih SRC koordinata spremljenih u jedan tip podatka. Važno je napomenuti kako je dovoljno jednom

odrediti ROI za transformaciju perspektive slika na snimkama snimljenim istom kamerom na istoj poziciji u automobilu. Ako se koriste različite kamere na različitim pozicijama na prednjoj strani automobila, potrebno je zasebno odrediti ROI za svaku kameru. Slika 3.6. prikazuje primjer transformacije perspektive ROI-ja ispravljene slike snimljene na prednjoj strani vozila.

a) b)

Slika. 3.6. Primjer transformacije perspektive slike: (a) ispravljena slike bez izobličenja kojeg unosi kamera, s označenim ROI-jem, (b) ROI sa slike (a) iz ptičje perspektive, nakon provedene transformacije perspektive

3.2.3. Kreiranje binarne slike i kalibracija binarnih filtara

Nakon što je transformacija perspektive obavljena, potrebno je dobivenu sliku obraditi kako bi se dobila binarna slika koja prikazuje samo bijele i žute prometne linije na slici. Binarna slika je vrsta slike kod koje elementi slike (*engl. pixel*) sadrže samo jednu od dvije vrijednosti (0 ili 1) [31]. Obično se za prikaz tih vrijednosti koristi crna (za 0) i bijela (za 1) boja. Objekti koji su potrebni i važni na slici nakon obrade izvorne slike bit će bijele boje dok će ostatak slike koji nije važan biti crne boje. Za obradu izvorne slike i dobivanje binarne slike linija voznih traka koristit će se razne funkcije za detekciju rubova i filtriranje slike pomoću pragova boja. Za detekciju rubova na slici u razvijenom algoritmu korišten je Sobelov operator [32], dok su za filtriranje slike pomoću pragova boja korišteni razni kanali RGB (*engl. Red, Green, Blue*), HLS (*engl. Hue, Lightness, Saturation*), LAB (*engl. Lightness, A-first color dimension, B-second color dimension*) prostora boja slike. RGB, HLS i LAB prostori boja predstavljaju različit zapis prikaza boja koji omogućuje izvlačenje određenih informacija iz slike na jednostavniji način [33]. U sklopu ovog rada kreirano je sveukupno 22 filtra u obliku funkcija koje obrađuju dobivenu sliku na različite načine. Tablica 3.1. prikazuje detaljnije informacije o svim kreiranim filtrima. OpenCV sadrži sve potrebne funkcije za kreiranje filtara i obradu izvorne slike. Za bolje rukovanje i snalaženje u obradi binarnih slika kreiran je potprogram

Process_image_final.py kojim je moguće obraditi izvornu sliku sa svim definiranim filtrima i prikazati sve binarne slike na ekranu, kao što je vidljivo na slici 3.7. Na slici je također vidljivo da ne daju svi filtri istu binarnu sliku, što je i razumljivo. Ovisno o slici i onome što slika prikazuje (vremenski uvjeti, doba dana, vidljivost na cesti, boja linije, itd.), vidljivo je da neki filtri bolje detektiraju vozne trake za razliku od ostalih. Međutim, već za nekakve druge uvjete na cesti i u samoj okolini, rezultati mogu biti potpuno drugačiji. Konačna binarna slika je rezultat kombinacije dviju binarnih slika dobivenih primjenom različitih filtara, a konačna binarna slika dobiva se pomoću logičke operacije "ili", na način da će nova binarna slika imati vrijednosti 1 na lokacijama elemenata slike koje sadrže vrijednost 1 na jednoj ili objema originalnim binarnim slikama. Odabir koja dva binarna filtra će se koristiti za konačnu sliku opisan je u sljedećem odlomku.

Tablica 3.1. Popis svih kreiranih binarnih filtara za detekciju linija

Naziv filtra	**Metode filtriranja**	**Vrijednosti konstanti raspona vrijednosti pragova**	**Koristi se za detekciju**
rgb_r	Raspon vrijednosti R kanala RGB prostora boja	Min_threshold= 200 Max_threshold= 255	Bijele linije
hls_s	Raspon vrijednosti S kanala HLS prostora boja	Min_threshold= 125 Max_threshold= 255	Bijele linije
hls_l	Raspon vrijednosti L kanala HLS prostora boja	Min_threshold= 220 Max_threshold= 255	Bijele linije
lab_l	Raspon vrijednosti L kanala LAB prostora boja	Min_threshold= 190 Max_threshold= 255	Bijele linije
lab_b	Raspon vrijednosti B kanala LAB prostora boja	Min_threshold= 190 Max_threshold= 255	Žute linije
sobel_mag	Kombinacija sobel operatora	Min_threshold=25 Max_threshold=255 Kernel_size=25	Žute linije Bijele linije
sobel_abs	Kombinacija sobel operatora	Min_threshold=25 Max_threshold=255	Žute linije Bijele linije
sobel_dir	Kombinacija sobel operatora	Min_threshold=0 Max_threshold=0.11 Kernel_size=7	Žute linije Bijele linije
hsv_white	Raspon vrijednosti H, S i V kanala HSV prostora boja	White_hsv_low= [0, 0,160] White_hsv_high=[255,80,255]	Bijele linije
hsv_yellow	Raspon vrijednosti H, S i V kanala HSV prostora boja	Yellow_hsv_low= [0, 0,160] Yellow_hsv_high=[255,80,255]	Žute linije
white_tight	Raspon vrijednosti V kanala HSV prostora boja	Min_threshold=175 Max_threshold=255	Bijele linije
white_loose	Raspon vrijednosti V kanala HSV prostora boja	Min_threshold=175 Max_threshold=255	Bijele linije
yellow_edge_pos	Kombinacija Sobel operatora i raspon vrijednosti V i S kanala HSV prostora boja	Vmin_threshold = 20 Vmax_threshold=120 Smin_threshold = 10 Smax_threshold=100	Žute linije

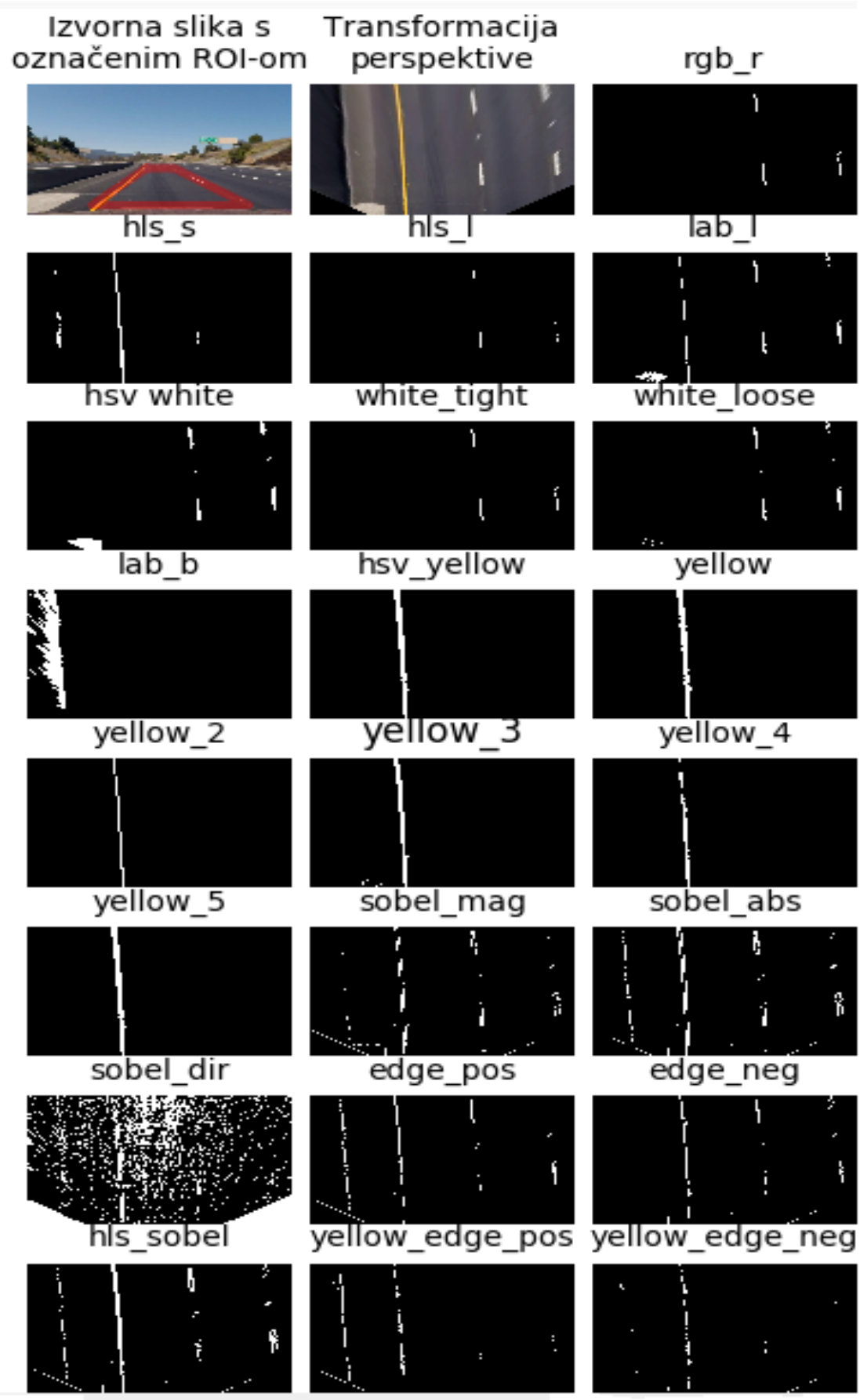
yellow_edge_neg	Kombinacija Sobel operatora i raspona vrijednosti V i S kanala HSV prostora boja	Vmin_threshold = 20 Vmax_threshold=120 Smin_threshold = 10 Smax_threshold=100	Žute linije
edge_neg	Kombinacija Sobel operatora i raspona vrijednost V kanala HSV prostora boja	Vmin_threshold = 20 Vmax_threshold=120	Žute linije Bijele linije
edge_pos	Kombinacija Sobel operatora i raspona vrijednosti V kanala HSV prostora boja	Vmin_threshold = 20 Vmax_threshold=120	Žute linije Bijele linije
hls_sobel	Kombinacija Sobel operatora i raspona vrijednosti L i S kanala HLS prostora boja	Lmin_threshold = 120 Lmax_threshold=255 Smin_threshold = 150 Smax_threshold=255 S_Sobel_min=20 S_Sobel_max=100	Žute linije Bijele linije
yellow	Kombinacija raspona vrijednosti H i S kanala HSV prostora boja	Hmin_threshold = 15 Hmax_threshold=35 Smin_threshold = 10 Smax_threshold=100	Žute linije
yellow_2	Kombinacija raspona vrijednosti H, S i V kanala HSV prostora boja	Yellow_hsv_low= [0, 80, 200] Yellow_hsv_high=[40, 255, 255]	Žute linije
yellow_3	Kombinacija raspona vrijednosti H, S i V kanala HSV prostora boja	Yellow_hsv_low= [15, 38, 115] Yellow_hsv_high= [35, 204, 255]	Žute linije
yellow_4	Kombinacija raspona vrijednosti H, S i V kanala HSV prostora boja	Yellow_hsv_low= [20, 120, 80] Yellow_hsv_high= [45, 200, 255]	Žute linije
yellow_5	Kombinacija raspona vrijednosti H, S i V kanala HSV prostora boja	Yellow_hsv_low= [0, 100, 100] Yellow_hsv_high= [50, 255, 255]	Žute linije

Tijekom izrade kompletnog algoritma primijećeno je da dva filtra odabrana na početku snimke najčešće nisu prikladni za cijelu snimku koja se obrađuje. Promjene svjetline na cesti, vremenskih uvjeta ili pojava novih žutih voznih linija uzrokuju poteškoće ili nemogućnost pronalaska voznih linija na slici. Kako bi se omogućio ispravan rad algoritma u što više mogućih scena i situacija snimljenih kamerom postavljenom na prednjoj strani automobila, osmišljen je dodatni proces koji radi uz glavni proces rada algoritma. Dodatni proces konstantno vrši kalibraciju tj. odabir najboljih binarnih filtara za detekciju voznih traka. Način na koji se to izvodi je taj da se obradi ROI slika iz ptičje perspektive, opisana u dijelu 3.2.2, pomoću svih mogućih filtara, njih ukupno 22, nakon čega se stvara jedna slika koja je kombinacija svih slika obrađenih filtrima. Elementi slike na istim koordinatama obrađenih slika koji imaju vrijednost 1 se zbrajaju, čime maksimalna vrijednost elementa slike u novoj slici može biti 22. Nakon što je kombinirana slika kreirana, vrši se provjera maksimalne vrijednosti elemenata slike u slici. Maksimalna vrijednost elemenata slike na slici potrebna je zbog izračuna vrijednosti gornje

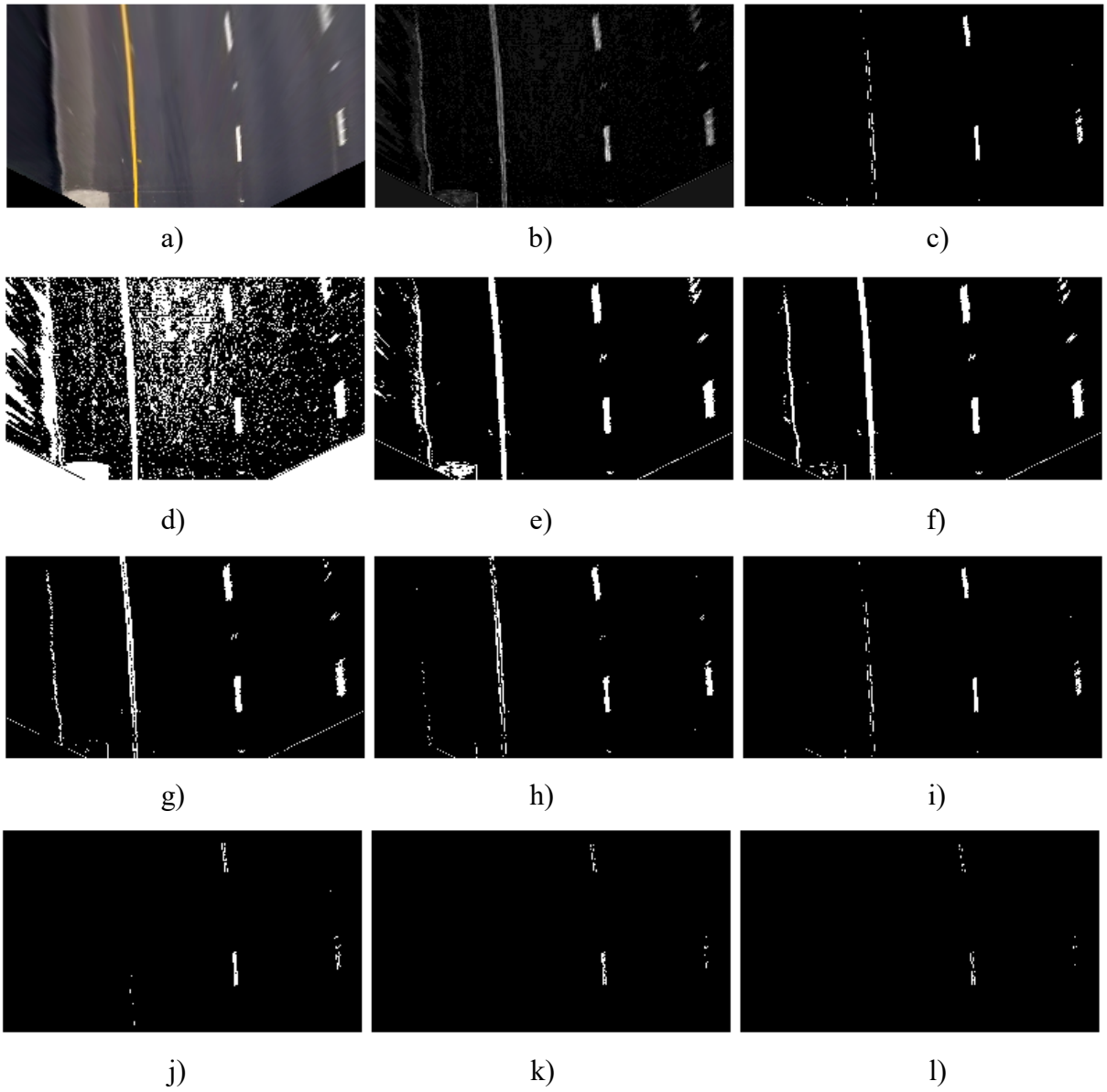
granice filtriranja kombinirane slike. Gornja granica filtriranja kombinirane slike omogućava da se filtrira kombinirana slika na način da elementi slike s vrijednošću iznad gornje granice ostaju na slici dok vrijednosti ispod granice poprimaju vrijednost 0. Pretpostavka je da će linije vozne trake biti detektirane od strane nekoliko filtara, dok će pojava šumova na slikama biti različita i vezana za karakteristike obrade slike određenog filtra. Ako na više filtara linije budu detektirane, vrijednost na lokacijama tih elemenata slike u kombiniranoj slici bit će veća nego vrijednosti šumova na kombiniranoj slici. Vrijednost gornje granice određuje funkcija *threshold_equation()* koja za danu maksimalnu vrijednost elemenata slike vraća vrijednost gornje granice. Na taj se način dobiva konačna binarna slika koja najbolje prikazuje detektirane prometne linije. Slika 3.8. prikazuje rezultate kreiranja kombinirane slike iz svih kreiranih binarnih slika nakon primjene svih 22 kreiranih filtara u jednu binarnu sliku. Uz navedenu kombiniranu sliku prikazane su i slike nastale filtriranjem navedene kombinirane slike uz različite iznose gornje granice filtriranja, gdje se ona mijenjala od 1 do 9. Vidljivo je da od svih mogućih vrijednosti gornje granice slika sa vrijednošću gornje granice 6 (slika 3.8. (i)) najbolje prikazuje linije vozne trake s najmanjom pojavom šumova na slici. Slika 3.8. (c) prikazuje konačnu binarnu sliku dodatno filtriranu kroz automatsku izračunatu vrijednost (pomoću funkcije *threshold_equation()*) gornje granice filtriranja 6 što je zapravo ista slika kao i 3.8. (i). Vidljivo je da je funkcija *threshold_equation()* zapravo izračunala najbolju vrijednost gornje granice.

Nakon izračuna konačne binarne slike filtrirane po automatskoj izračunatoj vrijednosti gornje granice od 6 (slika 3.8. (c)), ta se slika označava kao referentna binarna slika te se uspoređuje s 22 slike koje su svi filtri kreirali. Za usporedbu binarnih slika kreirana je funkcija *compare_binary_images()* koja prolazi kroz sve elemente slike redom te uspoređuje vrijednost elementa slike na referentnoj slici s vrijednošću elementa slike na slikama obrađenim pojedinim od 22 filtra. Radi lakšeg izračuna, kreirano je polje varijabli koje predstavljaju zbroj jednakih vrijednosti elemenata slike na referentnoj slici i na slikama obrađenim pojedinim od 22 filtra. Polje se sastoji od 22 elementa koji predstavljaju zbroj jednakih vrijednosti elemenata slike kod svakog filtra posebno. Ako je kod slike obrađene pojedinim filtrom vrijednost elementa slike 1, a vrijednost elementa slike referentne slike na istoj koordinati je isto 1, dodaje se vrijednost 1 na ukupan zbroj točnih elemenata slike u element polja koji predstavlja taj filtar. Isto tako ako je kod slike obrađene pojedinim filtrom vrijednost elementa slike 0, a vrijednost elementa slike referentne slike na istoj koordinati je isto 0, dodaje se vrijednost 1 na ukupan zbroj točnih elemenata slike u element polja koji predstavlja taj filtar. Kada se vrijednost elementa slike kod

slike obrađene pojedinim filtrom i referentne slike razlikuju, oduzima se vrijednost 1 na ukupan zbroj točnih elemenata slike u element polja koji predstavlja taj filtar. Nakon toga se sortira polje varijabli koje predstavljaju zbroj jednakih vrijednosti elemenata slike na referentnoj slici i na slikama obrađenim pojedinim od 22 filtra od najveće vrijednosti prema najmanjoj. Pripadni nazivi filtra koji odgovaraju prvom i drugom elementu sortiranog polja se spremaju i prosljeđuju kao rezultat funkcije. Rezultat funkcije zapravo daje naziv dva filtra koja najbolje odgovaraju za detekciju voznih linija u obrađivanom okviru te se ti nazivi zapisuju u varijablu koju glavni proces koristi kako bi odabrao odgovarajuće filtre za rad algoritma u glavnom procesu. Usporedba binarnih slika i odabir dvaju najboljih filtara odvija se u zasebnom procesu u petlji dok traje rad glavnog procesa na način da zasebni proces nakon usporedbe prvog okvira slike provjerava okvir slike koji glavni proces trenutno obrađuje te ga kopira i ponovo vrši zasebni proces od početka, samo s novim okvirom slike i tako sve do kraja rada glavnog procesa. Budući da se dva najbolja filtra ne odabiru samo jednom, moguće je dinamički detektirati pojavljivanje žutih voznih linija na određenim dijelovima ceste. Uz to, kompletan se algoritam vrlo lako može prilagoditi promjeni vanjskog osvjetljenja ili promjeni vremenskih uvjeta. Razlog zašto je ovaj korak odvojen u zasebni proces jest njegova računalna kompleksnost. Budući da je obrada izvorne slike sa svih 22 definiranih filtara poprilično računalno zahtjevna operacija, vrijeme izračuna cijelog procesa traje duže nego vrijeme obrade jednog okvira snimke kroz kompletni algoritam u glavnom procesu. Tako se glavni i zasebni proces izvršavaju bez međusobnog čekanja jednog procesa na drugi te kad zasebni proces izvrši sve izračune za određeni okvir koji je obrađivao, promijene se samo nazivi dvaju najboljih filtra u glavnom procesu. To znači da se kompletan algoritam može prilagoditi novonastaloj sceni na cesti. Konačna binarna slika koja se prosljeđuje u sljedeće korake rada algoritma je rezultat kombinacije dviju binarnih slika dobivenih primjenom najboljih izračunatih filtra u zasebnom procesu.



Slika. 3.7. Prikaz svih kreiranih binarnih slika s njihovim nazivima nakon primjene odgovarajućih filtara.

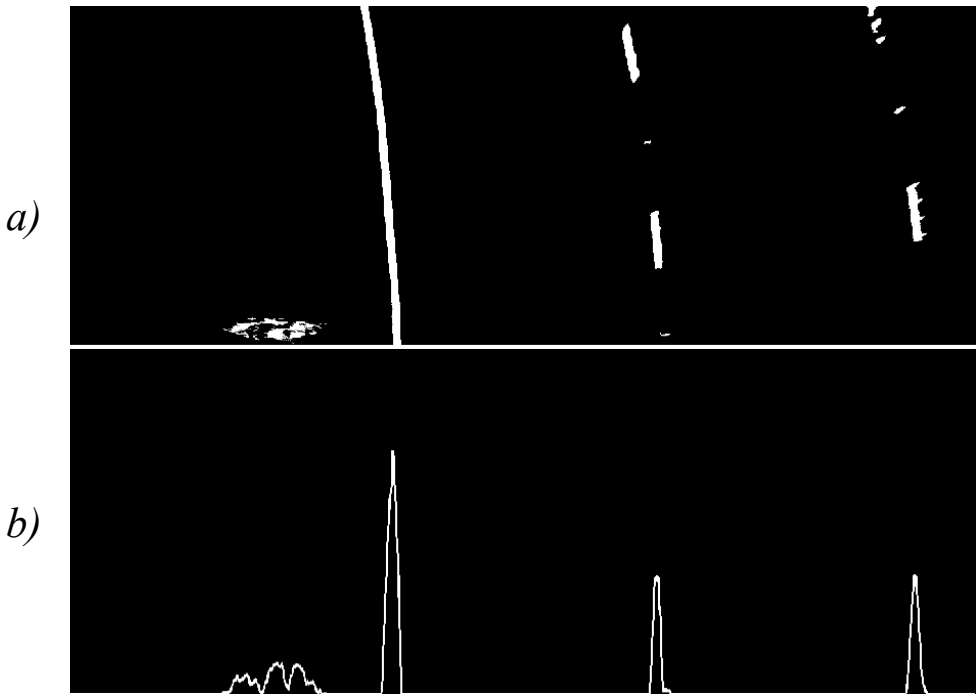


Slika. 3.8. Dobivanje binarne slike primjenom 22 različita filtra (a) slika područja interesa (ROI) iz ptičje perspektive koja se obrađuje filtrima; (b) konačna binarna slika dobivena kombinacijom svih 22 filtra; (c) konačna binarna slika dodatno filtrirana kroz automatsku izračunatu vrijednost gornje granice filtriranja 6; (d)-(l) konačna binarna slika dodatno filtrirana uz vrijednosti gornje granice filtriranja: 1,2,3,4,5,6,7,8 i 9.

3.2.4. Kreiranje i obrada histograma binarne slike

Sljedeći je korak izrada histograma konačne binarne slike koja je rezultat kombinacije dviju binarnih slika dobivenih primjenom najboljih izračunatih filtra u zasebnom procesu. Histogram jest grafički prikaz učestalosti pojave rezultata između dviju granica koje definiraju skup kontinuiranih podataka, u ovom slučaju binarne slike [34]. Ako se binarna slika promatra kao

dvodimenzionalna matrica s X stupaca i Y redaka, gdje X predstavlja širinu slike u broju elemenata slike, a Y predstavlja visinu slike u broju elemenata slike, i uz to svaki element slike može poprimiti vrijednost 0 ili 1, sliku je moguće sažeti na način da se zbroje svi elementi Y redaka za određeni X. stupac i na taj je način moguće reducirati podatke koje slika sadrži u jednostavniji oblik, čime se saznaje učestalost pojavljivanja svih elemenata u Y redaka X-tog stupca koji imaju vrijednost 1. Slika 3.9. (a) prikazuje izgled binarne slike, a slika 3.9 (b) prikazuje izgled histograma te binarne slike. Vidljivo je da su na mjestima gdje su elementi slike bijele boje prisutni skokovi u histogramu koji odgovaraju broju elemenata slike koji imaju vrijednost 1 za svih Y elemenata u odgovarajućem X-tom stupcu.



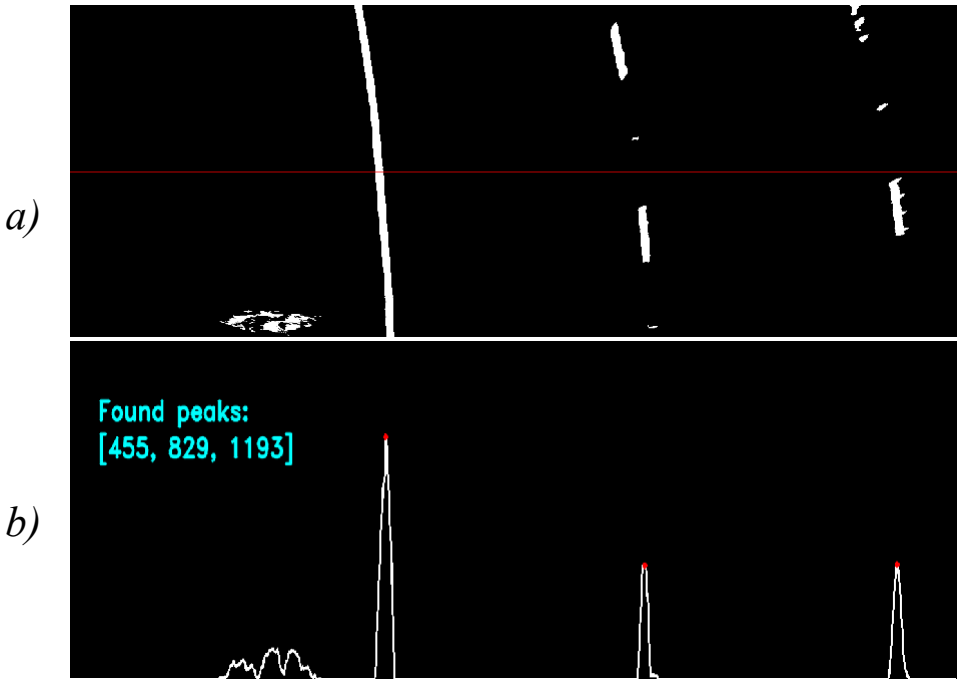
Slika. 3.9. (a) izgled binarne slike, (b) histogram te binarne slike

Binarna slika prikaza linija voznih traka neće uvijek "propustiti" samo vozne trake. Često će se na slici pronaći elementi slike bijele boje koji ne predstavljaju vozne trake nego su zapravo smetnje nastale istim značajkama u drugim objektima u slici ili nepravilnim odabirom filtara koji su obradili sliku. Budući da histogram sažima sve Y vrijednosti za svaki X element duž X osi, pojava smetnji postaje smanjena ili čak i neprimjetna na samom histogramu. Na taj su način pomoću histograma reducirane smetnje nastali u kreiranju binarne slike. Na slici 3.9. (a) je vidljivo da su na binarnoj slici u donjem lijevom kutu prisutne smetnje, a slika 3.9 (b) prikazuje i izgled te smetnje na histogramu te binarne slike. Vidljivo je da su prisutne smetnje na binarnoj slici zapravo beznačajne na histogramu. Uz to je važno naglasiti da je nepotrebno pretraživati cijelu binarnu sliku kako bi se utvrdilo koji elementi slike pripadaju određenoj voznoj traci.

Dovoljno je informacija sadržano u donjoj polovici binarne slike koja predstavlja cestu bliže autu. Zato je umjesto određivanja histograma cijele slike izračunat histogram samo donje polovice binarne slike. Uz to se reducira vrijeme izračuna samog histograma te se smanjuje vrijeme računalne obrade kompletnog algoritma.

Nakon što je histogram kreiran, potrebno ga je obraditi radi izračunavanja X koordinata potencijalnih voznih traka. To je riješeno određivanjem vrhova koji predstavljaju maksimume u obliku "šiljaka" na slici histograma. Ideja je da će najvjerojatnije linije vozne trake biti na mjestima gdje su prisutni vrhovi na slici histograma. Ti vrhovi predstavljaju najveću koncentraciju bijelih elemenata slike po X osi u binarnoj slici, a samim time i lokaciju potencijalne vozne trake.

Za izračunavanje histograma koristi se numPy funkcija *np.sum()* koja zbraja elemente matrice po odabranoj osi, u ovom slučaju X osi. Za određivanje vrhova slike histograma kreirana je funkcija *find_histogram_peaks()* koja za dani histogram određuje maksimalno 4 vrha te vraća X koordinate na kojima se nalaze pronađeni vrhovi. Uz to je osmišljena funkcija *draw_histogram()* koja od izračunatog histograma stvara sliku histograma te određuje vrhove histograma radi lakše dijagnostike u daljnjem radu algoritma. Slika 3.10. prikazuje rezultat funkcije *draw_histogram()* tj. izgled histograma donje horizontalne polovice binarne slike s detektiranim vrhovima.



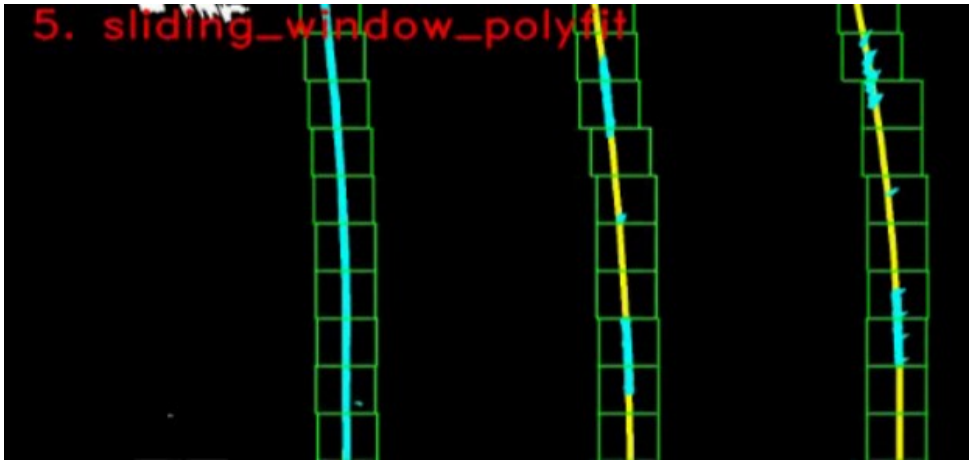
Slika. 3.10. a) izgled binarne slike, b) izgled histograma te binarne slike s detektiranim i označenim (crvene oznake) vrhovima na slici.

Nakon što su X koordinate vrhova slike histograma izračunate, pretvaraju se u nove X koordinate na način da se sortiraju u polje od četiri elementa koji predstavljaju položaj linija voznih traka na cesti. Prvi element polja predstavlja koordinatu lijeve linije lijeve susjedne vozne trake, drugi element predstavlja desnu liniju lijeve susjedne vozne trake, tj. lijevu liniju vozne trake u kojoj se automobil nalazi, treći element predstavlja desnu liniju vozne trake u kojoj je automobil tj. lijevu liniju desne susjedne vozne trake, a zadnji element predstavlja desnu liniju desne susjedne vozne trake. Način na koji se koordinate sortiraju jest taj da se prvo odredi rezolucija ulazne slike te se vrijednost širine slike podjeli na četiri jednaka dijela i označe X koordinate granica tih dijelova. Nakon toga se provjerava X koordinata vrhova s granicama podijeljenih dijelova i ako je vrijednost vrha unutar granica prvog podijeljenog dijela, vrh pripada prvom elementu polja, ako je vrijednost vrha unutar granica drugog dijela vrh pripada drugom elementu polja, ako je vrijednost vrha unutar granica trećeg dijela vrh pripada trećem elementu polja i ako je vrijednost vrha unutar granica četvrtog dijela vrh pripada četvrtom elementu polja. Ako neka linija nije pronađena, zapisuje se vrijednost "None" u polje sortiranih koordinata. Time se određuje koja je vozna traka pronađena s obzirom na 4 vozne trake koje je moguće detektirati. Pritom je iz same postavke (lokacije) kamere utvrđeno da se svaka od 4 potencijalne linije nalaze upravo u određenim četvrtinama same slike, gledano po širini. Tako npr., za širinu slike od 1280 elemenata, razmatraju se četiri područja po X koordinatama, 1-320, 321-640, 6 41-960, 961-1280. U svrhu sortiranja X koordinata, kreirana je funkcija pod nazivom *allocate_peaks_to_4lanes()* koja za predane X koordinate vraća navedeno polje od četiri elementa. Nakon toga, ostatku se programa prosljeđuju sortirane X koordinate koje predstavljaju pozicije potencijalnih voznih traka te su potrebne za sljedeće korake u radu algoritma.

3.2.5. Pronalazak voznih traka

Idući je korak pronalazak voznih traka na temelju sortiranih X koordinata. To je riješeno implementacijom metode "klizajućih prozora" (*engl. sliding window*). Metoda se zasniva na kreiranju skupa od *n* prozora (pravokutnika) povezanih tj. spojenih po dužini na binarnoj slici (slika 3.11.), gdje je *n* broj pronađenih X koordinata. Ti prozori su određene visine i širine kojima se omeđuju linije na binarnoj slici, počevši od dna slike.

Nakon što se kreira prvi klizajući prozor na dnu slike za određenu liniju iz sortiranih X koordinata, identificiraju se svi elementi slike koji sadrže vrijednost 1 unutar definiranog prozora te se spremaju njihove koordinate u polje. Nakon toga potrebno je pronaći srednju vrijednost po X osi svih pronađenih elemenata slike unutar prozora, čime se dobiva središnji položaj linije u tom rasponu. U sljedećem koraku ponovno se kreira novi klizajući prozor iznad prvog prozora, ovaj put sa središtem po x osi u prethodnoj izračunatoj srednjoj vrijednosti. Nakon toga se ponavlja pronalazak svih elemenata slike koji sadrže vrijednost 1 kao i računanje srednje vrijednosti. Navedeni koraci se ponavljaju sve dok zadnji klizajući prozor ne obrubi vrh binarne slike. U ovom radu za jednu liniju sveukupno se kreira 10 klizajućih prozora visine 72 i širine 60 elemenata slike (dobivene empirijski eksperimentalnim putem). Metoda se vrši za svaku izračunatu X koordinatu koja predstavlja potencijalnu liniju posebno. Ovim pristupom omogućeno je učinkovito "praćenje" linije od izračunate X koordinate u prethodnom koraku do vrha binarne slike, čime se ubrzava obrada same binarne slike na način da se traže samo aktivni elementi slike preko malog dijela slike. Slika 3.11. prikazuje kreiranje pravokutnika (prozora) na binarnoj slici pomoću metode klizajućeg prozora.



Slika. 3.11. Prikaz kreiranja pravokutnika (prozora) na binarnoj slici pomoću metode klizajućeg prozora

Sljedeći je korak izračun parametara polinoma drugog reda tj. rješavanje kvadratne jednadžbe iz svih identificiranih koordinata spremljenih u polju koje predstavljaju jednu liniju. Polinom drugog reda tj. kvadratna jednadžba je definirana izrazom:

$$ax^2 + bx + c = y \tag{3-1}$$

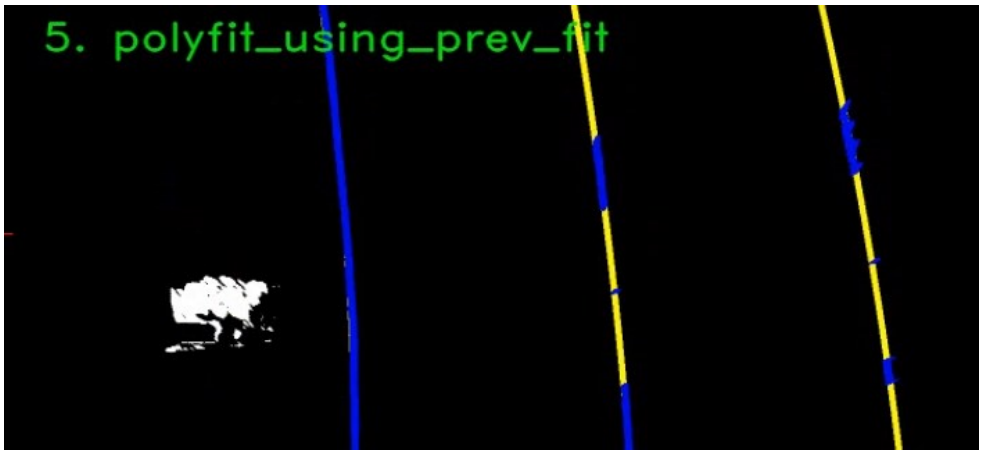
gdje je :

- a – kvadratni koeficijent
- b – linearni koeficijent
- c – slobodni koeficijent.

Pomoću NumPy funkcije *polyfit()* moguće je na temelju predanog polja dobiti tri koeficijenata *a*, *b* i *c* [35] koji predstavljaju koeficijente pripadajuće kvadratne jednadžbe. Nakon toga vrši se provjera izračunatih koeficijenta za svaku detektiranu liniju na način da se zanemaruju polinomi drugog reda gdje je koeficijent a>0.001 ili b>1. Na taj se način eliminiraju horizontalne linije koje se ovim korakom mogu odrediti kao potencijalne linije. Potrebno je naglasiti da je unutar glavnog programa kreirana klasa *Line* koja sadrži razne algoritmu potrebne informacije o samoj detektiranoj liniji, poput koeficijenata polinoma drugog stupnja, detekcije same linije, pouzdanosti pronalaska linije i sl. Uz to je kreirano globalno polje od 4 objekta klase *Line* koje predstavljaju četiri linije koje je moguće detektirati na slici. Algoritam u bilo kojem trenutku može pristupiti polju i raznim informacijama o linijama kako bi što bolje detektirao linije u novim slikama. Algoritam također sprema informacije o detektiranoj liniji u određeni *Line* objekt za protekle okvire, njih maksimalno deset, te na izlaz prosljeđuje konačnu liniju kao srednju vrijednost svih detektiranih linija u proteklim okvirima. Nakon svega, linije se dodaju u polje s četiri objekta klase *Line()* s kojim je pojednostavljen pristup svim objektima klase kao i njihovim informacijama u bilo kojem dijelu programskog koda. Cijeli navedeni korak vrši funkcija *sliding_window_polyfit ()* koja za predanu X koordinatu određenog vrha histograma i binarnu sliku vraća tri koeficijenta polinoma drugog stupnja ako je pronađen na predanoj X koordinati, u suprotnom vraća vrijednost "None". Navedena funkcija se izvodi četiri puta za svaku sortiranu X koordinatu vrha histograma posebno.

Ako se u obzir uzme činjenica da se linije ne mijenjaju znatno u uzastopnim slikama na snimkama s 30 FPS, jedan je od načina za poboljšanje algoritma spremanje prethodno izračunatih koeficijenata voznih linija i pokušaj pronalaženja bijelih elemenata slike na novoj slici iz tih koeficijenata. Način na koji je to implementirano jest kreiranje funkcije *polyfit_using_prev_fit()* koja se izvršava umjesto funkcije *sliding_window_polyfit()* za linije koje su pronađene u prethodnim slikama. Pomoću prethodno određenih koeficijenta polinoma, crta se krivulja polinoma na binarnoj slici s debljinom 60 elemenata slike, koja zapravo predstavlja marginu oko krivulje izvornog polinoma. Vrijednost od 60 elemenata slike za marginu je odabrana kao najbolja vrijednost nakon provedenog eksperimentiranja vrijednosti

margine tokom razvijanja vlastitog algoritma. Nakon toga unutar nacrtanog polinoma potrebno je pronaći bijele elemente slike te ih spremiti u novo polje, nakon čega se ponovno vrši određivanje novih koeficijenta polinoma NumPy funkcijom *polyfit()*. Time se smanjuje površina obrade slike za pronalazak linija u okvirima u čijim su prethodnim okvirima već pronađene linije. Slika 3.12 prikazuje rezultat primjene *polyfit_using_prev_fit()* metode gdje su plavom bojom označeni detektirani elementi slike s vrijednošću 1 unutar definirane margine, a žutom bojom označeni polinomi drugog stupnja kojima je linija vozne trake modelirana. Kod lijeve plave linije žutom bojom je označena linija ispod plave boje pa linija poprima samo plavu boju. Bijelom bojom su označeni elementi slike koje program ignorira jer predstavljaju smetnju.



Slika. 3.12. Prikaz rezultata primjene metode detekcije linija s prethodnim izračunatim koeficijentima

3.2.6. Kreiranje oznaka voznih traka na izvornoj slici

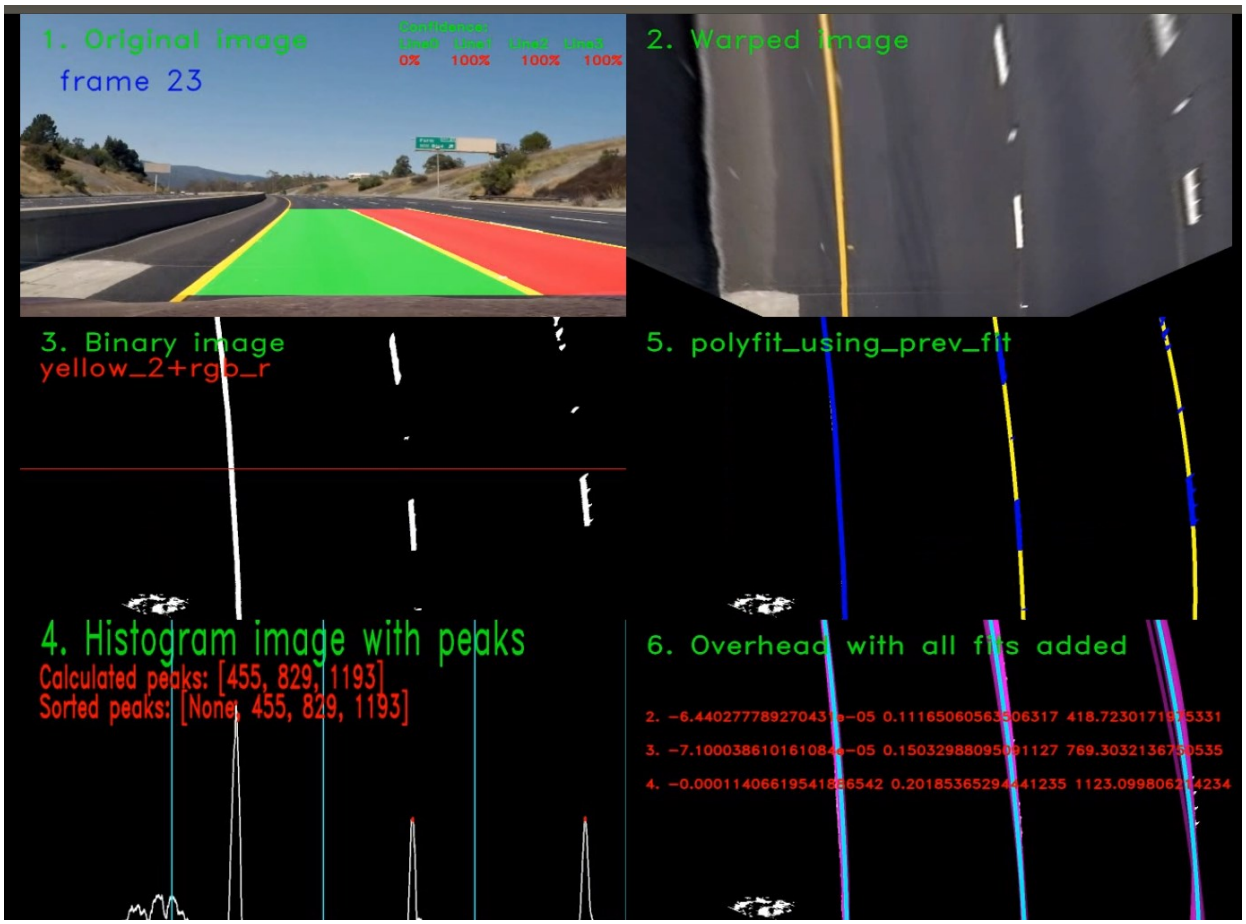
Nakon što je algoritam pronašao linije voznih traka i samim time vozne trake, potrebno ih je označiti na izvornoj slici. Prvo se linije kreiraju na binarnoj slici (slika 3.13) te se označavaju plavom bojom. Nakon iscrtavanja svih potrebnih linija, vrši se inverzna transformacija perspektive kako bi se paralelne linije s binarne slike transformirale za prikaz linija na izvornoj slici. OpenCV funkcija *warpPerspective()* omogućava jednostavnu inverznu transformaciju perspektive. Nakon toga na izvornoj slici se osim nacrtanih linija ispunjava i prostor između pronađenih linija vozne trake određenom bojom kako bi se označile detektirane vozne trake. Ispunjavanje prostora između dviju linija izvršava se pomoću OpenCV funkcije *fillpoly()* (slika 3.14.). U algoritmu je još dodana varijabla vrijednosti postotka sigurnosti (pouzdanosti) detekcije linije za svaku liniju na izvornoj slici. Ta varijabla može poprimiti vrijednost od 0 do 100, čime se zapravo predstavlja postotak sigurnosti detekcije linije od 0% do 100%. Varijabla

sigurnosti detekcije pojedine linije funkcionira na način da se prvo kreira navedena varijabla kada se linija ispravno detektira u određenom okviru slike. Pod ispravnom detekcijom se smatra uspješno odrađeni svi koraci obrade okvira slike te uspješno detektirani koeficijenti polinoma 2. stupnja koji predstavljaju određenu liniju. Ako se linija ispravno detektira u idućim okvirima snimke, nadoda se vrijednost od 5, koja predstavlja 5%, na vrijednost pouzdanosti detekcije pojedine linije za svaki okvir u kojem je linija ispravno detektirana. Ako linija u određenom okviru nije detektirana ili je detektirana neispravno, oduzima se 10% od vrijednosti pouzdanosti detekcije te linije. Ako je sveukupna vrijednost pouzdanosti detekcije pojedine linije u određenom okviru manja od 20%, to se interpretira na način da liniju nije pouzdano označiti te se ne crta na izvornoj slici. Time je omogućeno kratkoročno pamćenje položaja linije u slučajevima da oznake voznih linija na cesti nisu jasno vidljive u svakom okviru. Na slici 3.14 je u gornjem desnom uglu vidljiv prikaz postotka pouzdanosti detekcije pojedine linije za trenutni okvir.

Radi jednostavnosti programiranja i otklanjanja pogreška, u programu je kreirana mogućnost dijagnostičkog prikaza s kombiniranim slikama rezultata svih koraka (slika 3.15). Kompletni prikaz sastoji se od ukupno šest kombiniranih prikaza koji redom prikazuju tok razvijenog algoritma i obradu izvorne slike sa svim koracima definiranim u ovom potpoglavlju. U prilogu P.3.1. na elektroničkom mediju priloženom uz ovaj rad nalazi se kompletan programski kod razvijen za potrebe rada algoritma sa svim potrebnim datotekama.

Slika. 3.13. Prikaz detektiranih linija na binarnoj slici s pripadnim koeficijentima krivulja polinoma koje označavaju linije vozne trake.

Slika. 3.14. Prikaz detektiranih linija i obojanih voznih traka na izvornoj slici.



Slika. 3.15. Izgled dijagnostičkog prikaza rada kompletnog algoritma za detekciju linija voznih traka na jednom okviru snimke.

3.2.7. Upozoravanje o napuštanju vlastite vozne trake

Nakon što su linije vozne trake uspješno detektirane, potrebno je bilo osmisliti način detekcije napuštanja vozne trake i na određen način upozoriti vozača ukoliko vozilo krene napuštati vlastitu voznu traku. Detekcija vlastite vozne trake je ostvarena na vrlo jednostavan način. Iskorištavaju se informacije dobivene pomoću histograma binarne slike (opisano u dijelu 3.2.4.) te se određuje X koordinata središta histograma po X-osi. Za širinu slike od 1280 elemenata, ta vrijednost je 640. Nakon toga određuju se dvije nove X koordinate koje su za 100 elemenata slike pomaknute u desno (prva X koordinata) i pomaknute u lijevo (druga X koordinata) od X koordinate središta histograma binarne slike. Za širinu slike od 1280 elemenata, taj raspon je od 540 do 740 elemenata slike po X-osi. Time je zapravo određen raspon vrijednosti provjere koordinata oko središta histograma slike. Taj raspon predstavlja prostor unutar kojeg X koordinate položaja lijeve i desne linije vlastite vozne trake trebaju biti ako vozilo napušta vlastitu voznu traku. Ako se unutar tog raspona (540-740 po X osi) nalazi jedna od detektiranih linija vlastite vozne trake, pretpostavlja se da se vozilo previše približilo toj liniji i da dolazi do napuštanja vlastite vozne trake. Ako se položaj lijeve linije vlastite vozne trake nalazi između 540 i 640 elemenata slike po X osi, to predstavlja prestrojavanje vozila u lijevu voznu traku. Isto tako ako se položaj desne linije vlastite vozne trake nalazi između 640 i 740 elemenata slike po X osi, to predstavlja prestrojavanje vozila u desnu voznu traku. Nakon detekcije napuštanja vlastite vozne trake ispisuje se poruka „Alert! Lane Departure.“ na izvornoj slici. Za detekciju napuštanja vlastite vozne trake i ispisivanje poruke na izvornoj slici kreirana je funkcija *write_alert()* koja uspoređuje navedene X koordinate linija vlastite vozne trake s rasponom vrijednosti X koordinata oko središta histograma binarne slike. Slika 3.16. prikazuje okvir snimke gdje se vozilo prestrojava iz vlastite vozne trake u lijevu voznu traku te izgled ispisa upozorenja o napuštanju vlastite vozne trake na izvornoj slici snimke.

Slika. 3.16. Prikaz izgleda ispisa upozorenja o napuštanju vlastite vozne trake na izvornoj slici snimke.

4. EVALUACIJA PERFORMANSI PREDLOŽENOG ALGORITMA ZA DETEKCIJU VOZNIH TRAKA I UPOZORENJE O NAPUŠTANJU VOZNE TRAKE

Evaluacija performansi algoritama za detekciju voznih traka u praksi općenito nije jednostavan zadatak [2]. Odabir baze podataka za testiranje algoritma, odabir mjerenja kojim se procjenjuje stopa uspješnosti rada algoritma te čimbenici okoline u sekvencama kojima se testira algoritam čine evaluaciju performansi iznimno kompliciranom. Okolišni čimbenici se uglavnom odnose na sadržaj video sekvenci kojima se testira algoritam (uvjeti vidljivosti voznih traka zbog raznih vremenskih uvjeta (kiša, magla, snijeg) ili doba dana (vožnja po danu, noćna vožnja)). Različiti uvjeti okoline mogu utjecati na postotak uspješnosti rada razvijenog algoritma zbog čega je odabir skupa uvjeta okoline važan.

Pod bazom podataka smatra se skup sekvenci prikaza ceste prednjom kamerom na automobilu u obliku slika ili videa te neki način reprezentacije oznaka linija na tim sekvencama koje zapravo predstavljaju temeljnu istinu (*engl. ground truth*) položaja linije na okvirima sekvenca. Stvarne linije na okvirima sekvenca trebaju prije evaluacije rada algoritma biti označene i to ispravno na položajima gdje se one stvarno nalaze na slikama, što nije jednostavan i kratkotrajan zadatak. Općenito je potrebno ručno označiti linije vozne trake na svakom okviru sekvence kako bi baza podataka bila uspješno kreirana i pripremljena za evaluaciju.

Postoje dva načina evaluacije uspješnosti rada algoritma na određenoj bazi podataka, a to su *online* i *offline* način evaluacije [2]. Pod *online* način evaluacije smatra se pokretanje algoritma na sklopovlju vozila gdje se ulazni podaci generiraju u trenutku. *Online* evaluacija se rijetko koristi jer je tehnički zahtjevnija naspram *offline* evaluacije. *Offline* način evaluacije je zapravo izvanmrežni način testiranja gdje se obično pokreću testiranja na stolnom računalu određene konfiguracije pomoću unaprijed pripremljene baze podataka. Takav način evaluacije se najčešće koristi jer omogućuje promjenu algoritma i brže testiranje rezultata rada algoritma na unaprijed pripremljenoj bazi podataka. Uz to, sama baza podataka može biti javna ili privatna. Uglavnom se za testiranje funkcionalnosti koriste javne baze podataka jer se time olakšava razvoj algoritma na način da se vrijeme i trud ne ulaže u kreiranje vlastite, privatne, baze podataka. Osim toga, ukoliko su dostupni rezultati već postojećih algoritama na javnom bazama u odgovarajućem obliku (što često i nije slučaj), onda je moguće vlastiti algoritam i usporediti s tim postojećim. Za potrebe ovog rada kreiran je skup sekvenci koji je zapravo skup javnih

video sekvenci snimljenih kamerom na prednjem kraju automobila koji sadrže različite prikaze scena na cesti i vremenskih uvjeta.

Kod odabira mjera kojima se procjenjuje stopa uspješnosti rada algoritma, mogu se koristiti različite veličine. Uglavnom se uspoređuju položaji detektiranih linija voznih traka koje je algoritam generirao sa stvarnim položajima linija voznih traka koje se nalaze na ulaznoj slici [2]. Sjecište preko unije (*engl. Intersection over Union - IoU*) je jedna od popularnijih metoda uspoređivanja položaja detektiranih linija voznih traka [22]. IoU radi na principu izračuna dviju vrijednosti koje se stavljaju međusobno u omjer: količina elemenata slike koji je zajednički liniji temeljne istine i detektirane linije (prostor preklapanja dviju linija) te izračuna prostora tj. količine elemenata slike koje obje linije zajedno sadrže (prostor unije tih linija). Izračunom omjera između tih dviju vrijednosti dobiva se rezultat točnosti položaja detektirane linije koji je u rasponu od 0 do 1, gdje 1 predstavlja da su elementi slike detektirane linije i linije temeljne istine potpuno jednake, a vrijednost 0 pokazuje da linije nemaju niti jedan element slike zajednički.

Evaluacija performansi rada razvijenog algoritma obavljena na način da se uspoređivao broj detektiranih linija voznih traka od strane razvijenog algoritma s točnim brojem linija voznih traka na cesti (temeljna istina) i to za svaki okvir. Uz to je evaluirano upozorenje o puštanju vlastite vozne trake na sekvencama koje sadrže napuštanje vlastite vozne trake na način da su se prije rada algoritma označili trenuci napuštanja vlastite vozne trake u sekvencama i kasnije usporedili s rezultatima kreiranih upozorenja o napuštanju vlastite vozne trake. Također, mjerila se i brzina rada samog algoritma na stolnom računalu, mjereći broj okvira koji algoritam stiže obraditi u jednoj sekundi (FPS).

4.1. Evaluacija točnosti broja detektiranih linija vozne trake

Prvim testom mjerila se točnost broja detektiranih linija voznih traka u odnosu na stvarni broj linija voznih traka na cesti, za svaki okvir ulazne video sekvence. Pripremljeno je 12 video snimki koje sadrže različite scene na cesti (razni vremenski uvjeti i doba dana) kojima se ujedno testira i prilagodljivost rada razvijenog algoritma različitim uvjetima okoline. U prilogu P.4.1. na elektroničkom mediju priloženom uz ovaj rad nalazi se kompletni skup video sekvenci. U tablici 4.1. prikazan je popis svih snimki s dodatnim informacijama o tipu linija voznih traka i vremenskim uvjetima na cesti za pojedinu sekvencu. Sve snimke u tablici su rezolucije 1280x720. Na snimkama se mogu pronaći vožnje po sunčanom, maglovitom, kišovitom i

snježnom vremenu, kao i vožnje po danu i noći. Slika 4.1. prikazuje po jedan izdvojeni okvir iz svake od 12 video sekvenci korištenih u ovom testu.

Prije detekcije linija razvijenim algoritmom, za snimke iz tablice 4.1 za svaki je okvir zabilježen broj linija voznih traka, a to će služiti za ispitivanje performansi algoritma. U ovom testu pod „ispravno detektiranom linijom“ smatrala se ona linija koja je subjektivnim promatranjem bila blizu stvarne linije i njezina se oznaka zaista odnosila na stvarnu liniju, a ne na nešto drugo. Razlog zašto se tako odabrao način testiranja je taj što su navedene video snimke pronađene na internetu zasebno te nisu sa sobom sadržavale oznake linija koje predstavljaju temeljnu istinu. Da bi se oznake linija koje predstavljaju temeljnu istinu kreirale za pripremljene video snimke, to bi zahtijevalo veliku količinu vremena, čime se izlazi iz okvira ovog diplomskog rada. Tablica 4.2. prikazuje rezultate obavljenog testa. U tablici se nalaze i komentari vezani za pojedinu snimku gdje je sažeto objašnjen razlog neispravnog broja detektiranih linija u pojedinoj snimci

Tablica 4.1. Popis video sekvenci korištenih u testiranju točnosti broja detektiranih linija voznih traka u odnosu na stvarni broj linija voznih traka na cesti

Naziv video snimke	Vremenski uvjeti	Tipovi linija voznih traka	Način vožnje
Sunny_YW_Straight_NT_1.mp4	Sunčano	Žuta puna linija, bijela isprekidana linija	Ravno bez prestrojavanja
Sunny_YW_Curved_NT_1.mp4	Sunčano	Žuta puna linija, bijela isprekidana linija	Krivina bez prestrojavanja
Sunny_YW_Straight_T_1.mp4	Sunčano	Žuta puna linija, bijela isprekidana linija, bijela puna linija	Ravno s prestrojavanjem
SunnyExtreme_YW_Curved_NT_1.mp4	Izrazito sunčano	Žuta dvostruka puna linija, bijela puna linija	Krivina bez prestrojavanja

Night_YW_Straight_Curved_NT_1.mp4	Noć	Bijela puna linija, žuta dvostruka puna linija	Ravno i krivina bez prestrojavanja
Foggy_YW_Straight_NT_1.mp4	Maglovito, dan	Bijela puna linija, žuta dvostruka puna linija, bijela isprekidana linija	Ravno bez prestrojavanja
Rainy_YW_Straight_T_Curved_NT_1.mp4	Kišovito, dan	Bijela puna linija, žuta dvostruka puna linija, bijela isprekidana linija	Ravno s prestrojavanjem, krivina bez prestrojavanja
Sunny_YW_Straight_Curved_NT_1.mp4	Sunčano	Bijela puna linija, žuta dvostruka puna linija, bijela isprekidana linija	Ravno i krivina bez prestrojavanja
Snow_W_Straight_Curved_NT.mp4	Snijeg, dan	Bijela puna linija, bijela isprekidana linija	Ravno i krivina bez prestrojavanja
Sunny_W_Straight_NT_1.mp4	Sunčano	Bijela dvostruko puna linija, bijela puna linija	Ravno bez prestrojavanja
Sunny_YW_Straight_Curved_NT_2.mp4	Sunčano	Bijela puna linija, žuta dvostruka puna linija, bijela isprekidana linija	Ravno i krivina bez prestrojavanja
Rainy_YW_Straight_Curved_NT_2.mp4	Kišovito	Žuta puna linija bijela puna linija bijela isprekidana linija	Ravno i krivina bez prestrojavanja

Slika. 4.1. Prikaz jednog izdvojenog okvira korištenih u testiranju točnosti broja detektiranih linija voznih traka u odnosu na stvarni broj linija voznih traka na cesti za video sekvencu (a) Sunny_YW_Straight_NT_1.mp4, (b) Sunny_YW_Curved_NT_1.mp4, (c) Sunny_YW_Straight_T_1.mp4, (d) SunnyExtreme_YW_Curved_NT_1.mp4, (e) Night_YW_Straight_Curved_NT_1.mp4, (f) Foggy_YW_Straight_NT_1.mp4, (g) Rainy_YW_Straight_T_Curved_NT_1.mp4, (h) Sunny_YW_Straight_Curved_NT_1.mp4, (i) Snow_W_Straight_Curved_NT.mp4, (j) Sunny_W_Straight_NT_1.mp4, (k) Sunny_YW_Straight_Curved_NT_2.mp4, (l) Rainy_YW_Straight_Curved_NT_2.mp4

Tablica 4.2. Rezultati testiranju točnosti broja detektiranih linija voznih traka u odnosu na stvarni broj linija voznih traka na cesti

Naziv video snimke	**Broj okvira u zapisu**	**Broj okvira s točnim brojem detekcija linija**	**Postotak okvira s točnim brojem detekcija**	**Komentar**
Sunny_YW_Straight_NT_1.mp4	469	439	93.60%	Kod prolaska kroz nadvožnjak ne detektira linije
Sunny_YW_Curved_NT_1.mp4	602	601	99.83%	Jasno vidljive oznake na cesti zbog čega algoritam uspješno detektira linije voznih traka
Sunny_YW_Straight_T_1.mp4	1806	1671	92.52%	Detektirana žuta linija suprotnog voznog traka u nekoliko okvira
SunnyExtreme_YW_Curved_NT_1.mp4	1190	365	30.67%	Izrazito izazovna snimka koja sadrži nagla skretanja i promjene osvjetljenja kroz snimku
Night_YW_Straight_Curved_NT_1.mp4	853	844	99.94%	Jasno vidljive oznake na cesti pod osvjetljenjem zbog čega algoritam uspješno detektira linije voznih traka
Foggy_YW_Straight_NT_1.mp4	1192	812	68.12%	Snimka sadrži skretanje na raskrižju gdje je potrebna nova kalibracija binarnih filtera kako bi se algoritam prilagodio novim linijama voznih traka
Rainy_YW_Straight_T_Curved_NT_1.mp4	3200	2565	80.15%	Slabija vidljivost linija vozne trake zbog koje algoritam nekad propusti rubne linije susjednih voznih traka

Sunny_YW_Straight_Curved_NT_1.mp4	2202	1670	75.84%	Neispravna detekcija linije vozne trake na prostoru tla pored ceste u određenom trenutku
Snow_W_Straight_Curved_NT.mp4	1056	518	49.05%	Detekcija linije na području tla pokrivenog snijegom Detektirane linije koje se ne vide jer su pod snijegom
Sunny_W_Straight_NT_1.mp4	1053	955	90.69%	Neispravna detekcija linije na prostoru tla pored ceste u određenom trenutku
Sunny_YW_Straight_Curved_NT_2.mp4	2355	2142	90.95%	Nagla skretanja vozila, neuspješna detekcija linija pri skretanju
Rainy_YW_Straight_Curved_NT_2.mp4	1574	846	53.74%	Slabo vidljive oznake linija voznih traka uzrokovane lošijom kvalitetom snimke zbog kojih algoritam ne detektira linije voznih traka
Sve sekvence zajedno	17552	13428	76.50%	

Iz rezultata testa u tablici 4.2. vidljivo je da razvijeni algoritam radi ispravno s pretežito malim postotkom neispravnih detekcija (5-10%) z za 6 od 12 sekvenci.. Kod snimki gdje su prisutni snijeg ili kiša, algoritam teže detektira linije zbog povećane pojave smetnji na obrađenoj binarnoj slici, uzrokovanih vremenskim uvjetima koji su bili za vrijeme nastajanja snimke. Zbog toga za te snimke postotak točnosti detekcije broja linija je manji naspram ostalih snimki. Kod većine snimki algoritam uspješno prilagodi binarne filtre za novonastalu scenu na cesti kroz idućih 150 okvira. U prilogu P.4.2. na elektroničkom mediju priloženom uz ovaj rad nalaze se snimke koje prikazuju rezultat rada razvijenog algoritma na svim snimkama opisanim u tablici 4.1.

Slika 4.2 prikazuje primjer točne detekcije linija na cesti na 49. okviru iz snimke *Foggy_YW_Straight_NT_1.mp4.* Slika 4.3. prikazuje primjer netočne detekcije linija na cesti na 24. okviru iz iste snimke. Kod netočne detekcije na slici 4.3. algoritam je detektirao tri od četiri linije na izvornoj slici. Detektirane linije su bijele boje i na ispravnom su položaju. Razlog zašto algoritam nije detektirao četvrtu, žutu, liniju je taj što se ta žuta linija tek pojavila u snimci te su dva odabrana binarna filtra na početku snimke određena samo za obradu bijelih linija jer su se u početnim okvirima snimke nalazile samo linije bijele boje. Slika 4.2. sadrži okvir snimke (49. okvir) koji je obrađen nakon okvira neispravne detekcije žute linije (24. okvir) i na njemu je vidljivo da je algoritam prepoznao žutu liniju na cesti i prilagodio svoje binarne filtre da prepoznaju i linije žute boje.

Slika 4.4. prikazuje primjer točne detekcije linija na cesti na 206. okviru iz snimke *Snow_W_Straight_Curved_NT.mp4.* Slika 4.5. prikazuje primjer netočne detekcije linija na cesti na 106. okviru iz iste snimke. Kod netočne detekcije na slici 4.5. algoritam je detektirao tri od četiri linije na izvornoj slici. Detektirane linije su bijele boje i na ispravnom su položaju. Razlog zašto algoritam nije detektirao četvrtu liniju, je taj što je linija na rubu okvira snimke te je zbog toga intenzitet svjetline bijele boje linije jako malen, zbog čega se linija ne propusti s izračunatim binarnim filtrima u ostatku algoritma. Nakon nekog vremena algoritam pronađe binarne filtre koji bolje detektiraju linije prisutne na cesti i samim time se detektira i linija koja nije detektirana na desnoj strani ceste u sljedećim okvirima, što je vidljivo i na slici 4.4.

Slika 4.6. prikazuje primjer točne detekcije linija na cesti na 374. okviru iz snimke *Sunny_YW_Straight_Curved_NT_1.mp4.* Slika 4.7. prikazuje primjer netočne detekcije linija na cesti na 37. okviru iz iste snimke. Kod netočne detekcije na slici 4.7. algoritam je detektirao četiri od četiri prisutne linije na izvornoj slici, ali je zadnja detektirana linija na krivom položaju i zbog toga nije odabrana kao točna detekcija priliko ovog testiranja. Neispravno detektirana linija se nalazi na slici u poziciji gdje nije prostor ceste. Razlog zašto je algoritam detektirao liniju na prostoru gdje nije cesta jest taj što je na tom prostoru prisutan šum koji se pojavio zbog sličnih karakteristika elemenata slike na tom prostoru kao i karakteristika elemenata slike linija zbog kojeg izračunati binarni filtri prosljeđuju šum dalje kao potencijalne linije i algoritam na tom položaju detektira liniju. Nakon nekog vremena odabrani filtri postaju bolji u detekciji linija i propuštaju manje šuma na binarnu sliku, te se time ova pogreška eliminira.

Osim objašnjenih razloga netočnih detekcija algoritam neispravno detektira linije vozne trake na način da detektira manji broj linija od onoga koji stvarno postoji na izvornoj slici (1 od 3 linije, 2 od 3 linije, 1 od 4 linije, 2 od 4 linije, 3 od 4 linije itd.). Najčešći razlog zašto algoritam detektira manji broj linija je taj da se neispravno odrede filtri za kreiranje binarne slike, zbog čega algoritam tik nakon pojave tih linija „ne vidi“ liniju vozne trake određene boje. Taj problem se naknadno riješi implementiranim kalibracijskim procesom određivanja parametara filtracije binarne slike (dio 3.2.3.). Nakon 5 do 10 sekundi rada algoritma na video snimci algoritam se prilagodi novoj sceni na cesti i samim time uspješno detektira sve linije vozne trake. Osim toga algoritam ponekad detektira liniju vozne trake na području tla pored ceste, gdje pomoćni proces za određivanje dva najbolja filtra za obradu binarne slike (dio 3.2.3.) nije odabrao ispravne filtre za obradu binarne slike.

Slika. 4.2. Primjer točne detekcije linija na cesti na 49. okviru iz snimke Foggy_YW_Straight_NT_1.mp4

Slika. 4.3. Prikaz okvira netočne detekcije linija na cesti na 24. okviru iz snimke „Foggy_YW_Straight_NT_1.mp4“

Slika. 4.4. Prikaz okvira točne detekcije linija na cesti na 206. okviru iz snimke „Snow_W_Straight_Curved_NT.mp4“

Slika. 4.5. Prikaz okvira netočne detekcije linija na cesti na 106. okviru iz snimke „Snow_W_Straight_Curved_NT.mp4“

Slika. 4.6. Prikaz okvira točne detekcije linija na cesti na 374. okviru iz snimke „Sunny_YW_Straight_Curved_NT_1.mp4“

Slika. 4.7. Prikaz okvira netočne detekcije linija na cesti na 37. okviru iz snimke „Sunny_YW_Straight_Curved_NT_1.mp4“

4.2. Mjerenje brzine rada razvijenog algoritma

U ovom je testu izvršeno mjerenje brzine rada samog algoritma na računalu s Intel Core i7-4790 procesorom i 16GB RAM memorije. Mjerenje izračunava broj FPS-a koje algoritam može obraditi. Testiranje je obavljeno na snimkama iz tablice 4.1 koje sadrže različite scene na cesti poput dnevne vožnje, noćne vožnje, vožnje po kišovitom ili maglovitom vremenu i sl. (dio 4.1.). Vrijeme obrade jednog okvira ovisi o odabranim filtrima koji obrađuju izvornu sliku (dio 3.2.3) i zbog toga brzina rada algoritma nije ista za različite snimku niti za različite okvire jedne snimke. Tablica 4.4. prikazuje brzinu rada algoritma za pojedinu snimku pri trima različitim rezolucijama ulazne slike. Za snimke razlučivosti 1280x720 elemenata slike prosječna brzina rada algoritma je 2.79 FPS, sa standardnom devijacijom od 0.078 FPS . Za iste snimke smanjene na rezoluciju 640x360 elemenata slike brzina algoritma se povećava te poprima prosječnu vrijednost 11.40 FPS sa standardnom devijacijom od 0.036 FPS. Kada se iste snimke smanje na rezoluciju 320x180 elemenata slike, algoritam ima prosječnu brzinu obrade 28.15 FPS sa standardnom devijacijom od 0.869 FPS te je kao takav prikladan za rad u stvarnom vremenu, uz pretpostavku ulaznog videa s 25 FPS-a.

Tablica 4.4. Rezultati mjerenja brzine rada algoritma

Naziv video snimke	**Rezolucija testnog video zapisa**	**Minimalna brzina rada algoritma [FPS]**	**Maksimalna brzina rada algoritma [FPS]**	**Srednja brzina rada algoritma [FPS]**	**Standardna devijacija [FPS]**
Sunny_YW_Straight_NT_1.mp4	1280x720	2.47	2.81	2.64	0.24
	640x360	11.23	11,59	11.41	0.18
	320x180	27.8	29.77	28.78	0.80
Sunny_YW_Curved_NT_1.mp4	1280x720	2.53	2.93	2.73	0.28

	640x360	11.29	11.64	11.46	0.14
	320x180	26,83	27.55	27.19	0.29
Sunny_YW_Straight_T_1.mp4	1280x720	2.54	2.84	2.69	0.12
	640x360	11.35	11.45	11.4	0.04
	320x180	29.15	29.44	29.29	0.12
SunnyExtreme_YW_Curved_NT_1.mp4	1280x720	2.60	2.78	2.69	0.07
	640x360	11.31	11.58	11.45	0.11
	320x180	26.5	28.33	27.42	0.74
Night_YW_Straight_Curved_NT_1.mp4	1280x720	2.69	2.89	2.79	0.08
	640x360	11.27	11.5	11.38	0.09
	320x180	28.35	28.87	28.61	0.22
Rainy_YW_Straight_T_Curved_NT_1.mp4	1280x720	2.57	2.95	2.76	0.16
	640x360	11.28	11.49	11.39	0.08
	320x180	26.35	26.9	26.62	0.22
Rainy_YW_Straight_Curved_NT_2.mp4	1280x720	2.48	2.61	2.54	0.05
	640x360	11.3	11.39	11.35	0.03
	320x180	27.3	29.48	28.39	0.88
Snow_W_Straight_Curved_NT.mp4	1280x720	2.73	3.05	2.89	0.13
	640x360	11,31	11,61	11.46	0.12
	320x180	28.3	28.7	28.5	0.16
Foggy_YW_Straight_NT_1.mp4	1280x720	2.70	2.85	2.8	0.06
	640x360	11,27	11,46	11.36	0.08
	320x180	27.8	28.4	28.1	0.24
Sunny_YW_Straight_Curved_NT_1.mp4	1280x720	2.72	2.87	2.79	0.06
	640x360	11.44	11.53	11.49	0.04
	320x180	26.3	27.8	27.05	0.61
Sunny_W_Straight_NT_1.mp4	1280x720	2.68	2.83	2.75	0.06
	640x360	11.18	11.60	11.39	0.17
	320x180	25.9	26.78	26.34	0.36
Sunny_YW_Straight_Curved_NT_2.mp4	1280x720	2.70	2.79	2.74	0.04
	640x360	11.38	11.68	11.53	0.13
	320x180	24.53	26.87	25.7	0.95

Ovdje treba napomenuti da su performanse algoritma mjerene u potpoglavlju 4.1 mjerene na rezoluciji 1280x720 elemenata slike, budući da je to široko korištena rezolucija u praksi. Iste nisu mjerene za rezolucije 640x360 i 320x180 elemenata slike. Ono što je sigurno je da bi se za te manje rezolucije algoritam trebao dodatno usmjeriti i bilo bi potrebno korigirati u određenoj mjeri neke njegove korake. No to u ovome trenutku izlazi van okvira ovog rada i ostaje kao zadataka potencijalnog budućeg rada. Ideja je ovog testa bila vidjeti na koliko bi bilo potrebno smanjiti rezoluciju ulaznog videa da bi se postigla obrada od 25 FPS uz trenutnu složenost razvijenog algoritma.

4.3. Testiranje mehanizma upozoravanja o napuštanju vozne trake

Zadnje testiranje koje je potrebo obaviti je provjera upozorenja o napuštanju vlastite vozne trake. Testiranje se izvodi na video snimkama u pripremljenom skupu podataka (tablica 4.1.).

Kako bi se izvelo ovo testiranje, potrebno je izbrojati broj promjena vozne trake za svaku snimku. Tri od 12 snimki u kreiranom skupu podataka sadrže prestrojavanje iz jednog voznog traka u drugi i samim time napuštanje vlastite vozne trake. Potrebno je obaviti testiranje na svim video snimkama, neovisno sadrže li promjenu vlastite vozne trake ili ne. U snimkama gdje nema promjene vlastite vozne trake algoritam ne treba ispisati upozorenje o promjeni vlastite vozne trake. Nakon što je broj promjena vozne trake za svaku snimku izbrojan, provjerile su se snimke koju je razvijeni algoritam kreirao s oznakama linija voznih traka i pripadnim upozorenjima o napuštanju vozne trake. Zatim je izbrojeno koliko se upozorenja o napuštanju vozne trake ispisalo na snimci i uz to kod ispisanih upozorenja je provjereno radi li se o stvarnim promjenama vozne trake. Tablica 4.5. prikazuje popis odabranih video snimki kao i broj promjena voznih traka u svakoj snimci i broja ispisanih upozorenja o promjeni vlastite vozne trake u svakoj snimci.

Tablica 4.5. Rezultati broja upozorenja o napuštanju vlastite vozne trake u odnosu na stvaran broj napuštanja vlastite vozne trake

Naziv video snimke	**Broj napuštanja vlastite vozne trake**	**Broj upozorenja o napuštanju vlastite vozne trake**
Sunny_YW_Straight_NT_1.mp4	0	0
Sunny_YW_Curved_NT_1.mp4	0	0
Sunny_YW_Straight_T_1.mp4	3	3
SunnyExtreme_YW_Curved_NT_1.mp4	0	14
Night_YW_Straight_Curved_NT_1.mp4	0	6
Foggy_YW_Straight_NT_1.mp4	0	0
Rainy_YW_Straight_T_Curved_NT_1.mp4	1	4
Sunny_YW_Straight_Curved_NT_1.mp4	0	0
Snow_W_Straight_Curved_NT.mp4	0	3
Sunny_W_Straight_NT_1.mp4	0	0
Sunny_YW_Straight_Curved_NT_2.mp4	0	5
Rainy_YW_Straight_Curved_NT_2.mp4	1	14

U tablici 4.5. vidljivo je da broj upozorenja kod video snimki koje ne sadrže promjenu vlastite vozne trake nije uvijek jednak nuli. Snimke *SunnyExtreme_YW_Curved_NT_1.mp4* i *Sunny_YW_Straight_Curved_NT_2.mp4* sadrže nagla skretanja vozila na cesti zbog kojeg algoritam ne detektira točno linije voznih traka tj. detektirane su linije voznih traka ali na netočnim položajima među kojima je jedna detektirana linija jako blizu središta slike i samim time se ta linija detektira kao da se napušta (dio 3.2.7.) i algoritam ispiše upozorenje o napuštanju vlastite vozne trake. Zbog toga algoritam u navedenim snimkama ispiše krivo upozorenje o napuštanju vlastite vozne trake čak 14 puta. Isto tako u snimci *Snow_W_Straight_Curved_NT.mp4* područje interesa izvorne slike (dio 3.2.2) je kreirano na način da su u slici transformacije perspektive linije vozne trake puno bliže jedna drugoj te su koncentrirane oko središta slike, čime algoritam u tri instance netočno upozori vozača na napuštanje vozne trake, iako u cijeloj snimci nema napuštanja vlastite vozne trake. Od 12 snimki kod kojih je testirano upozoravanje o napuštanju vlastite vozne trake, pet snimki ne sadrži napuštanje vlastite vozne trake kao i niti jedan prikaz poruke upozorenja o napuštanju vozne trake. Kod *Sunny_YW_Straight_T_1.mp4* snimke algoritam jasno detektira vlastite vozne trake i samim time uspješno detektira sva tri prestrojavanja koja se dogode u snimci te upozori vozača porukom na snimci u sva tri slučaja. Potrebno je naglasiti da se na ovoj snimci jasno vide linije voznih traka zbog čega algoritam nema problema s detekcijom napuštanja vozne trake. Kod *Rainy_YW_Straight_T_Curved_NT_1.mp4* snimke linije voznih traka nisu jasno vidljive na cesti zbog kiše te algoritam u određenim trenucima krivo detektira položaj linija (tablica 4.2.). Zbog toga se dogode trenuci u kojima algoritam prepozna liniju vozne trake na neispravnom mjestu u samo jednom okviru i detektira to kao napuštanje vlastite vozne trake, zbog čega se poruka upozorenja pojavi u trenucima kada nema napuštanja vlastite vozne trake. Provjera upozorenja o napuštanju vozne trake se izvršava za svaki okvir zasebno zbog čega algoritam ispiše upozorenje o napuštanju vlastite vozne trake netočno tri puta u tri okvira.

.

4.4. Osvrt na performanse algoritma

Nakon obavljenog testiranja i evaluacije rada i performansi razvijenog algoritma, zaključeno je da algoritam radi ispravno na snimkama gdje su linije voznih traka jasno vidljive. Dinamična promjena filtera obrade binarne slike pomaže algoritmu da se prilagodi novim promjenama scene ceste poput ulaska vozila u tunel, pojave žute linije na cesti ili promjeni vremenskih uvjeta na cesti za samo 5 do 10 sekundi, čime se osigurava ispravna detekcija linija voznih traka

neovisno o promjenama na ulaznoj slici. Na snimkama gdje je kvaliteta prikaza ceste lošija i gdje se linije vozne trake ne mogu jednostavno uočiti, algoritam ne detektira linije vozne trake ispravno. Isto tako najčešća pogreška koja se kod razvijenog algoritma dogodi je detekcija samo dijela linija voznih traka prisutnih na slici. Potencijalno poboljšanje jest optimiziranje algroritma za rad sa slikama manje rezolucije te moguće dodavanje neuronske mreže koja kreira binarnu sliku s označenim linijama vozne trake umjesto dosadašnjeg postupka kreiranja binarne slike, čime bi poboljšala točnost detekcije linija voznih traka na snimkama vožnje na cestama gdje je to manje uočljivo. Upozorenje o napuštanju vlastite vozne trake funkcionira u uvjetima kada su linije voznih traka jasno vidljive te kada algoritam ispravno prepoznaje vozne trake. U situacijama kada linije voznih traka nisu ispravno detektirane ili broj detektiranih linija nije točan, algoritam ne može znati položaj linija vlastite vozne trake i samim time prepoznati radi li se o napuštanju vlastite vozne trake ili ne. Potencijalna poboljšanja za upozoravanje o napuštanju vlastite vozne trake uključuju praćenje napuštanja vlastite vozne trake kroz više okvira naspram svakog okvira zasebno zbog čega algoritam neće ispisati poruku upozorenja na samo jednom okviru. Kada algoritam detektira napuštanje vozne trake, idućih nekoliko okvira bi se pratilo događa li se isto i tek onda upozorilo vozača.

5. ZAKLJUČAK

Cilj rada bio je izraditi algoritam za detekciju voznih traka pomoću prednje kamere na vozilu i dati upozorenje vozaču ukoliko vozilo kreće napuštati svoju voznu traku. Rad se zasniva na metodama zasnovanim na značajkama u slici i paraboličnom modeliranju linija. Kompletni se algoritam sastoji od osam koraka obrade izvorne slike čime se dobivaju informacije o broju linija voznih traka te njihovom položaju na slici. Rješenje je uspješno razvijeno u Python programskom jeziku uz pomoć OpenCV i NumPy biblioteka. Kompletan Python program uz glavni proces koji vrši obradu i detekciju linija sadrži i dodatni proces koji obrađuje izvornu sliku i pronalazi najbolje filtre za daljnju obradu u glavnom procesu. Nakon provedenog testiranja, vidljivo je da su rezultati zadovoljavajući u uvjetima gdje su jasno vidljive linije voznih traka na snimci te da algoritam uspješno može detektirati do četiri linije. Uz to, zbog brzine rada samog programa, vidljivo je da algoritam u svojoj sadašnjoj izvedbi nije prikladan za implementaciju u sustave stvarnog vremena. Algoritam može na originalno zamišljenoj rezoluciji (1280x720 elemenata slike) raditi s 2.8 FPS-a, ali uz potencijalna poboljšanja i optimizaciju koda njegov bi se rad sigurno mogao ubrzati. Također, moguće je obrađivati i snimke niže rezolucije, što bi svakako dovelo do manje količine potrebnih računalnih resursa, pa samim time i do povećanja brzine rada algoritma. Algoritam u određenim situacijama na cesti, gdje linije nisu potpuno vidljive, nailazi na poteškoće te ne detektira sve linije voznih traka na cesti. Zbog tih problema algoritam ne detektira ispravno vozne trake u svim mogućim situacijama pa zbog toga nije prikladan za komercijalnu upotrebu. Dodavanjem neuronske mreže u algoritam, koja bi služila za stvaranje binarne slike linija voznih traka iz izvorne slike, dodatno bi se ubrzao rad algoritma te povećala točnost detekcije voznih traka.

LITERATURA

[1] K. Kluge, S. Lakshmanan. A Deformable-Template Approach to Lane Detection. In Intelligent Vehicles '95 Symposium, Sep 1995

[2] D. Vajak, M. Vranješ, R. Grbić, D. Vranješ, Recent Advances in Vision-Based Lane Detection, Solutions for Automotive Applications, Faculty of Electrical Engineering, Computer Science and Information Technology, dostupno na: file:///C:/Users/Spoky/OneDrive/Fakultet/Diplomski/Ostali%20seminari/Elmar2019_final_ver_v3.pdf

[3] Y. Xing, C. Lv, Advances in Vision-Based Lane Detection: Algorithms, Integration, Assessment, and Perspectives on ACP-Based Parallel Vision, IEEE, Journal of Automatica Sinica, vol. 5, no. 3, Svibanj 2018

[4] G. Welch, G. Bishop, An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[5] C. Low, H. Zamzuri, S. A. Mazlan, "Simple robust road lane detection algorithm,", Proc. Int. Conf. Intell. Adv. Syst., 2014

[6] T. Tan, S. Y. Yin, and L. B. Liu, "Efficient lane detection system based on monocular camera," in Proc. IEEE Int. Conf. Consum. Electron., Mar. 2015

[7] Q. Chen and H. Wang, "A real-time lane detection algorithm based on a hyperbola-pair model," in Proc. Intell. Veh. Symp., Sep. 2006

[8] M. Aly, „Real time Detection of Lane Markers in Urban Streets", California Institute of Technology, CA, USA, Nov. 2014, dostupno na: https://arxiv.org/pdf/1411.7113.pdf

[9] M. A. Fischler and R. C. Bolles, „Random Sample Consenus, A Paradigm For Model Fitting With Applications To Image Analysis And Automated Cartography", in SRI International, Mar 1980

[10] K H. Lim et al., "Lane detection and Kalman-based linear parabolic lane tracking," in Proc. Int. Conf. Intell. Human-Mach. Syst. Cybern., 2009,

[11] A. Mammeri, A. Boukerche, and G. Lu, "Lane detection and tracking system based on the MSER algorithm, Hough transform and Kalman filter," in Proc. ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst., 2014

[12] C. Ma et al., "Lane detection using heuristic search methods based on color clustering," in Proc. Int. Conf. Commun., Circuits Syst., 2010

[13] G. Liu, S. Li, and W. Liu, "Lane detection algorithm based on local feature extraction,"

[14] P. Wu, C. Chang, and C. Lin, "Lane-mark extraction for automobiles under complex conditions," Aug. 2014.

[15] J. Huang et al., "Lane marking detection based on adaptive threshold segmentation and road classification," 2014

[16] Y. Wang, X. Wang, and C. Wen, “Gradient-pair constraint for structure lane detection,” Jun. 2012

[17] T. Y. Sun, S. J. Tsai and V. Chan, „HSI Color Model Based Lane-Marking Detection“, Sep 2006, dostupno na: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1707380

[18] J. J. Hopfield, „Neural networks and physical systems with emergent collective computational abilities“, in Proc. NatL Acad. Sci. USA, Apr 1982, dostupno na: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC346238/

[19] J. Li, X. Mei, D. Prokhorov and D. Tao, „Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene“, in IEEE, Jan 2016, dostupno na http://islab.ulsan.ac.kr/files/announcement/552/Deep%20Neural%20Network%20for%20Structural%20Prediction%20and%20lane%20detection%20in%20traffic%20scene.pdf

[20] G. Hinton et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” IEEE Signal Process., Nov. 2012

[21] I. Sutskever, J. Martens, and G. E. Hinton, “Generating text with recurrent neural networks,” 2011

[22] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection“, University of Washington ,Lipanj 2015, dostupno na: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

[23] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans and L. V. Gool, „Towards End-to-End Lane Detection: an Instance Segmentation Approach“, Feb 2018, dostupno na: https://arxiv.org/pdf/1802.05591.pdf

[24] The tuSimple lane challange, http://benchmark.tusimple.ai/

[25] OpenCv, About, https://opencv.org/about/

[26] NumPy, https://www.numpy.org/

[27] N. Mansurov, „What is Lens Distortion?“, https://photographylife.com/what-is-distortion

[28] M. S. Kurečić, Kolegij: Primjena digitalne fotografije u reprodukcijskim medijima, Nastavni materijali, dostupno na: http://repro.grf.unizg.hr/media/download_gallery/skripta%20za%20web.pdf

[29] OpenCV; Camera Calibration, https://docs.opencv.org/3.4.3/dc/dbb/tutorial_py_calibration.html

[30] Definition of perspective in English dictionary , https://www.lexico.com/en/definition/perspective

[31] Binary Images, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html

[32] I. Sobel, History and Definition of the Sobel Operator, 2014

[33] Colorizer, http://colorizer.org/

[34] Histograms, https://statistics.laerd.com/statistical-guides/understanding-histograms.php

[35] Parabola, https://www.mathsisfun.com/geometry/parabola.html

[36] S. Gupta and S. G. Mazumdar, „Sobel Edge Detection Algorithm“, , Feb 2013, dostupno na: https://pdfs.semanticscholar.org/6bca/fdf33445585966ee6fb3371dd1ce15241a62.pdf

SAŽETAK

U sklopu ovog diplomskog rada izrađen je računalni algoritma za detekciju voznih traka na cesti ispred vozila u slici dobivenoj s kamere na prednjem kraju vozila i upozorenje vozača na napuštanje vozne trake. Kompletan računalni program sastoji se od osam koraka obrade izvorne slike, koji daju informacije o broju detektiranih linija voznih traka u slici te njihovom položaju na slici. Obrada slike se vrši za svaki okvir u snimci zasebno. Algoritam koristi metode računalnog vida pomoću implementacije OpenCV biblioteke. Kompletan algoritam je izrađen u Python programskom okruženju s dodatkom OpenCV i NumPy biblioteka. Testiranjem izrađenog algoritma na raznim snimkama snimljenih pomoću kamere na prednjem kraju vozila u različitim vremenskim uvjetima i s pojavama raznih vrsta linija voznih traka, zaključeno je da je algoritam prilagodljiv na razne vremenske uvjete, doba dana ili izgled prometnih linija. Također, rezultati testiranja pokazali su i određene manjkavosti algoritma i prostor za potencijalna buduća poboljšanja.

Ključne riječi: detekcija voznih linija, obrada slike, OpenCV, računalni vid, ADAS

LANE DETECTION AND LANE DEPARTURE WARNING USING FRONT VIEW CAMERA IN VEHICLE

ABSTRACT

As part of this diploma paper, a computer algorithm was developed for the lane detection on the road with the image obtained from the camera at the front end of the vehicle with addition to warn the driver against unwanted lane departure. The complete computer program consists of seven steps of processing the original image, which provide information on the number of detected lane lines in the image and their position on the image. Image processing is done for each frame in the video separately. The algorithm uses computer vision methods using an OpenCV library implementation. The complete algorithm was created in a Python programming environment with the addition of OpenCV and NumPy libraries. Testing the developed algorithm on various shots taken at the front end of the vehicle in different weather conditions and with the appearance of different types of lanes, it was concluded that the algorithm is adaptable to different weather conditions, time of day or the appearance of traffic lines recorded by the camera at the front end of the car. Also, the test results showed some weaknesses in the algorithm and room for potential future improvements.

Keywords: lane detection, image processing, OpenCV, computer vision, ADAS

ŽIVOTOPIS

Domagoj Špoljar rođen je 26.3.1996. u Đakovu. Završio je opću gimnaziju A. G. Matoša u Đakovu s vrlo dobrim uspjehom. Nakon srednje škole upisuje Elektrotehnički fakultet u Osijeku, smjer Računarstvo. Godine 2017. stječe akademski naziv sveučilišni prvostupnik (baccalaureus) inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij računarstva, smjer računalno inženjerstvo na istom fakultetu i uz to postaje stipendist instituta RT-RK u Osijeku što se nastavilo i sljedeće godine. Aktivno se služi engleskim jezikom u govoru i pismu.

PRILOZI

Prilog P.3.1. Kompletan programski kod predloženog algoritma sa svim potrebnim datotekama za ispravan rad (elektronički prilog).

Prilog P.4.1. Skup video snimki korišten za ispitivanje performansi rada algoritme (elektronički prilog).

Prilog P.4.2. Video sekvence generirane nakon obrade predloženim algoritmom (električni prilog).