Razvoj i implementacija ADAS algoritma za upozorenje o napuštanju vozne trake na ugradbenu računalnu platformu

Dragaš, Srđan

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:521693

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

Sveučilišni studij

RAZVOJ I IMPLEMENTACIJA ADAS ALGORITMA ZA UPOZORENJE O NAPUŠTANJU VOZNE TRAKE NA UGRADBENU RAČUNALNU PLATFORMU

Diplomski rad

Srđan Dragaš

Osijek, 2020.

SADRŽAJ

1. UVOD

U zadnje vrijeme je napredak automobilske industrije fokusiran na razvoj autonomnih vozila zbog poboljšanja kvalitete vožnje te sigurnosti svih sudionika u prometu. Ljudska greška odgovorna je za otprilike 95% svih prometnih nesreća u Europskoj Uniji u 2019. godini. Te godine poginulo je 22,800 osoba, a teško ozlijeđeno 120,000 osoba. Statistika Europskog Parlamenta [1] navodi da bi razvoj autonomne vožnje mogao spasiti više od 25,000 života te izbjeći barem 140,000 težih ozljeda u odnosu na prošle godine.

Porast autonomnosti vozila u današnje vrijeme postiže se ugradnjom različitih sustava za pomoć vozaču (engl. *Advanced Driving Assistance System*) koji čine vožnju sigurnijom i udobnijom. ADAS se sastoji od odgovarajuće računalne platforme koja vrši obradu informacija koje se konstantno prikupljaju pomoću raznih vrsta senzora (kamera, radar, lidar, ultrazvučni senzori i sl.) te upozorava vozača na potencijalne opasnosti ili samostalno djeluje na upravljački sustav (npr. automatsko kočenje). Neki od ADAS sustava su prilagodljivi tempomat, sustav protiv blokiranja kotača, sustav za upozoravanje na sudar i slični sustavi.

Automatizirani sustavi vožnje se dijele na šest razina automatizacije. Na nultoj razini u današnje vrijeme se nalazi velika većina vozila, naime to je razina gdje je vozač zadužen za kompletno upravljanje vozila. Prva razina je ustvari najniža razina automatizacije gdje je sustav odgovoran za pomoć vozaču pri npr. upravljanju volanom ili ubrzavanjem, dok su na drugoj razini sustavi osposobljeni za upravljanje tim dijelovima, ali vozač u svakom trenutku može isključiti sustav te nastaviti sam upravljati. U trećoj razini sustav je u mogućnosti sam donositi odluke npr. u pretjecanju vozila koje se nalazi ispred, ali još uvijek vozač mora sudjelovati u vožnji te ako je potrebno preuzeti upravljanje nad vozilom. Pri četvrtoj razini autonomne vožnje, vozač ne mora pratiti sustav, a sustav može upravljati cjelokupnim procesom vožnje, no samo u određenim scenarijima. Na petoj razini računalni sustavi unutar vozila su potpuno samostalni, nije im potrebna intervencija vozača, a samom vozilu neće biti potrebne papučice za davanje brzine i kočenje. Autonomni sustav vožnje bit će u mogućnosti raditi sve što i iskusan vozač može raditi. Vozila na četvrtoj i petoj razini još uvijek su u svojoj testnoj fazi [2].

Sam proces razvijanja sustava autonomne vožnje sadrži razna ograničenja, kao što su na primjer razvoj i projektiranje uređaja, ograničenost fizičkog prostora i slično. Izvršavanje algoritama u stvarnom vremenu također ograničava razvitak budući da algoritmi zaduženi za neku od sigurnosnih funkcija moraju točno u određenom trenutku prikupiti potrebne informacije i reagirati, a ako se to ne ispuni može se dogoditi nesreća. Nadalje, ponegdje će biti potrebno obaviti

i izmjene na infrastrukturama kako bi se same prometnice prilagodile za autonomnu vožnju kao bi omogućile preciznije i brže izvođenje operacija. Također, baš kao i samim vozačima i sustavima će biti otežano upravljati vozilom na siguran način prilikom nepovoljnih vremenskih uvjeta koje utječu na informacije koje se prikupljaju putem različitih senzora pa je primjerice LIDAR vrlo osjetljiv na vremenske uvjete zbog svog principa rada, dok je u slučaju radara osjetljivost na vremenske uvjete puno manja. Međutim, informacije prikupljene putem radara su značajno niže razlučivosti stoga se na modernim vozilima koristi kombinacija različitih senzora kako bi se dobilo na robusnosti ADAS sustava. Kamere su također jako važan senzor zbog svoje cijene koja je niska, a ujedno pruža veliki izvor informacija o okolini vozila, ali i o vremenskim uvjetima. No, postoje situacije gdje slike koje dolaze s kamere jednostavno nisu dovoljne kako bi algoritam mogao donijeti sigurnu procjenu [3].

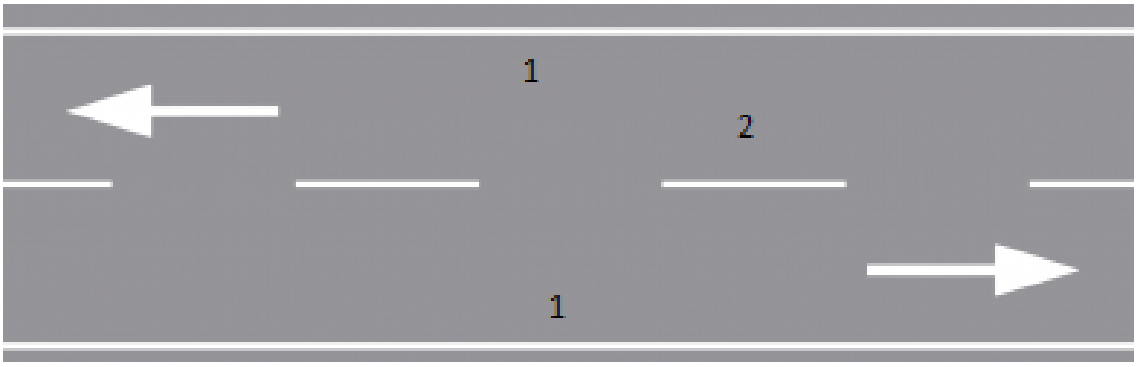
Jedan od vrlo čestih ADAS sustava je sustav koji detektira voznu traku na temelju uzdužnih kolničkih oznaka i sustav koji upozorava vozača kada vozilo počinje napuštati voznu traku. Takvi sustavi temelje se na obradi slike dobivene s kamere montirane na prednjoj strani vozila (engl. *Front view camera*). U ovom radu predložen je jedan takav algoritam koji detektira voznu traku i upozorava vozača kada nastupa napuštanje vozne trake. Algoritam se temelji na metodama razvijenim u području računalne obrade slike poput detekcije rubova, *Hough*-ove transformacije, filtera boja i sl. Za razvoj algoritma korišten je C++ programski jezik uz pomoć *OpenCV* biblioteke, a rješenje je implementirano na ADAS Alpha ploči.

Rad je strukturiran na sljedeći način. U drugom poglavlju predstavljen je opis problema detekcije voznih traka, dan je pregled postojećih rješenja te su istaknute prednosti i nedostaci pojedinih rješenja. Treće poglavlje sadrži opis vlastitog rješenja za detekciju vozne trake i upozoravanje vozača o napuštanju vozne trake, korištene tehnologije, proces razvijanja te svi međukoraci. U četvrtom poglavlju dana je evaluacija predloženog rješenja. Dan je opis baze signala na kojoj je provedeno testiranje, sam način testiranja te koje su mjere korištene za procjenu učinkovitosti predloženog algoritma. Na kraju rada dan je zaključak.

2. DETEKCIJA VOZNE TRAKE I UPOZORAVANJE VOZAČA O NAPUŠTANJU VOZNE TRAKE

2.1. Problem detekcije vozne trake

Detekcija vozne trake je svakako jedna od temeljnih zadaća koju bi autonomna vozila trebala moći raditi jednako dobro kao i iskusni vozači. Sama detekcija vozne trake je osnova za mnoge ostale ADAS algoritme kao npr. prilagođavanje brzine, ubrzavanje ili kočenje. Prvi korak kod svih algoritama za detekciju vozne trake je isti, a to je kontinuirano dohvaćanje slike koja se dobiva s kamere montirane na prednjoj strani vozila. Nakon toga je potrebno prepoznati regiju od interesa na ulaznoj slici gdje se nalazi vozna traka. Na temelju dobivene regije od interesa može se zaključiti gdje se točno nalazi vozilo kako bi se moglo aktivirati upozorenje vozača u slučaju napuštanja vozne trake. Sama detekcija se zasniva na detekciji uzdužnih kolničkih oznaka koje omeđuju voznu traku. Vozna traka je obilježeni uzdužni dio kolnika koji mora imati dovoljnu širinu za neometan prolazak vozila u jednom smjeru. Najčešće se nalaze dvije vozne trake za dva smjera, no u većim gradovima zbog gušćeg prometa postoji i više takvih traka. Vozne trake su u većini slučajeva označene žutom ili bijelom bojom, a dijele se na razdjelne i rubne. Razdjelne trake su oznake na kolniku koje označavaju podjelu vozne trake za jedan odnosno za drugi smjer prometa, dok rubne trake označavaju krajnji lijevi ili krajnji desni rub kolnika. Primjer oznaka na kolniku se nalazi na slici 2.1. Na slici razdjelna traka je označena s brojem 2, a rubna s brojem 1.



Slika 2.1. Oznake na kolniku.

Prilikom kretanja kolnikom najbliža lijeva i najbliža desna uzdužna kolnička oznaka omeđuju voznu traku u kojoj se vozilo nalazi. Zbog različitih uvjeta vožnje te zbog kolničke infrastrukture i drugih sudionika u prometu, slike primljene s kamere montirane na prednjoj strani vozila sadrže i veliki broj nepotrebnih informacija koje najčešće otežavaju detekciju uzdužnih kolničkih oznaka. Nadalje, oznaka za razdjelnu traku osim može biti puna ili isprekidana linija, te je potrebno da

algoritam obje linije detektira kao jednu uzdužnu kolničku oznaku. Nakon određivanja krajnje lijeve i krajnje desne uzdužne kolničke oznake, prostor omeđen detektiranim oznakama naziva se vozna traka. Također, potrebno je ovisno o ulaznoj slici zaključiti prelazi li vozilo u drugu voznu traku kako bi se moglo obavijestiti vozača da napušta voznu traku i ulazi u novu. S obzirom na to da se uzdužne kolničke oznake najčešće nalaze pod određenim kutom u dohvaćenoj slici, detekcija oznake koja je pod kutom koji nije u uobičajenom rasponu rezultira detekcijom napuštanja vozne trake.

2.2. Pregled postojećih rješenja za detekciju vozne trake i upozoravanje vozača o napuštanju vozne trake

U radu [4] novi pristup detekciji vozne trake uvodi se kao značajan dodatak za polu/ potpuno autonomni sustav pomoći vozaču. Način detekcije vozne trake uključuje naprednu detekciju pravaca pomoću višeslojne frakcijske *Fourier*-ove transformacije (engl. *Multi layered fractional Fourier transform*) i naprednog detektora uzdužnih kolničkih oznaka (engl. *Advanced lane detector*). Ulazna slika dolazi u prethodno navedenu višeslojnu frakcionalnu *Fourier*-ovu transformaciju gdje dobiveni frekvencijski odziv slike ima više frekvencijskih uzoraka u usporedbi s tradicionalnom jednoslojnom transformacijom. Stoga, dobiva se više informacija o promjeni intenziteta piksela što povećava performanse i točnost. Algoritam je testiran na slikama pruženima od strane autora na *Dell* računalu s *Intel Core i5 2.4 GHz*. U testiranju je predloženo rješenje uspoređeno s tradicionalnim rješenjem koje kombinira detekciju rubova i *Hough*-ovu transformaciju. Utvrđeno je da je predloženo rješenje u stanju obraditi 0.8 sekundi po okviru, a tradicionalno rješenje 0.12 sekundi po okviru što znači da novo rješenje obavlja algoritam više od 6 puta brže od tradicionalnog. Zbog velikog broja proračuna unutar ove metode ona se teško može primijeniti u ugradbenim računalnim sustavima.

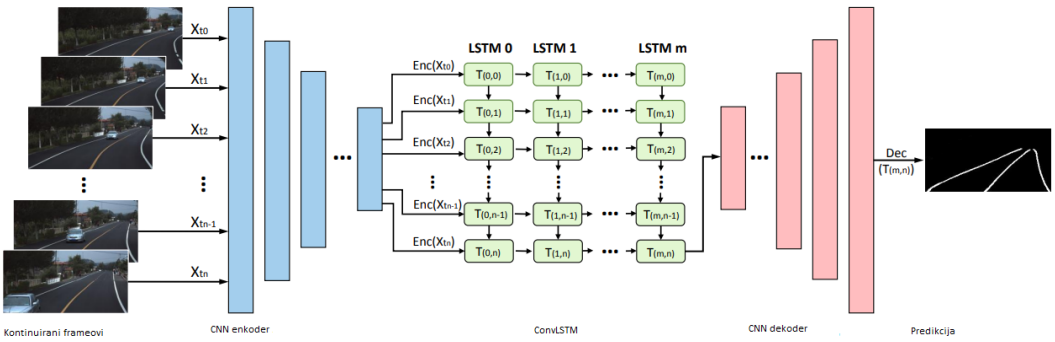
Rad [5] predlaže algoritam detekcije vozne trake i praćenje vozne trake za upozoravanje vozača o napuštanju vozne trake pomoću slika dobivenih s kamere montirane na prednjoj strani vozila. Algoritam prati dvije najbliže uzdužne kolničke oznake, detektira odvojeno lijevu i desnu uzdužnu kolničku oznaku kako bi preciznije javio upozorenje vozaču zbog mogućnosti spajanja i razdvajanja voznih traka. Algoritam ulaznu sliku pretvara u sliku sa sivim tonovima iz koje pomoću *Canny* detektora rubova traži rubove na slici. Nakon toga je provedeno pretraživanje slike od dolje prema gore gdje je cilj pronaći uzdužne kolničke oznake. Kako bi bili sigurni da su pronađeni pravci ustvari uzdužne kolničke oznake, prikladno je tražiti jednu točku od sredine linije

na jednu stranu, a zatim pretraživati drugu točku na drugu stranu. Potrebno je pronaći i intenzitet i širinu pravca kako bi bili sigurni da se radi o uzdužnoj kolničkoj oznaci, a ne o nekoj smetnji. Nakon pronađene točke, postavljen je određeni raspon za pretraživanje druge točke. Raspon služi kako bi se osiguralo da će se obavljeno pretraživanje izvršiti u drugom djelu slike. Nakon ove obrade slike, vozna traka je pronađena i označena na slici. Testiranje je provedeno na slikama koje su snimljene od strane autora. Algoritam je implementiran na *Pentium(R) Dual-Core CPU E5400 2.70 Ghz* koristeći *matlab* 7.1. Krajnji rezultati testiranja su brzina obrade od 0.12 sekunde po okviru te 91% točnost detekcije. Velika prednost ovog rada je njegova inovativnost, dakle nisu korištene tradicionalne tehnike, no problem nastaje kada se na kolniku pojavljuje više smetnji gdje se onda obrada slike značajno usporava budući da se rade dodatne provjere.

U radu [6] predstavljen je jednostavan, brz, robustan i učinkovit pristup rješavanju ovog problema. Temelji se na transformiranju pogleda u ptičju perspektivu nakon čega se dobivena slika filtrira pomoću selektivnih *Gauss*-ovih prostornih filtera koji su optimizirani za pronalaženje vertikalnih pravaca. Tada se dobivena slika dodatno filtrira kako bi ostale samo najviše vrijednosti, zatim se pravci detektiraju pojednostavljenom *Hough*-ovom transformacijom, nakon čega slijedi RANSAC (engl. *Random sample consensus*), a potom novi RANSAC dizajniran za pročišćavanje otkrivenih ravnih pravaca te za detektiranje zakrivljenih uzdužnih kolničkih oznaka. Konačno slijedi čišćenje i lokalizacija koja se provodi na ulaznoj slici. Prvi korak je dakle pretvorba u ptičju perspektivu. Ovim korakom dobiveni su potencijalni pravci i fokus samo na regiju od interesa. Zatim se filtrira slika pomoću 2D *Gauss*-ovog filtera. Nakon toga slijedi pojednostavljena *Hough*-ova transformacija gdje je samo potrebno izbrojati koliko pravaca je pronađeno na slici. Kasnije se radi RANSAC pomoću kojega se traži što više točaka na pravcu kako bi se osiguralo da se radi o uzdužnoj kolničkoj oznaci. Prethodni korak je dao potencijalne pravce na slici koji se zatim usavršavaju idućim korakom gdje se koristi napredna verzija RANSAC-a. Za svaki pravac uzima se prostor oko njega na slici na kojem se pokreće RANSAC. Nakon toga se primjenjuje *Bezier*-ova krivulja te se traži kojim pravcima najbolje odgovara pomoću metode najmanjeg kvadrata. U standardnom RANSAC-u bi se računala udaljenost od svake točke do trećeg stupnja krivulje kako bi se odlučilo dali ta krivulja odgovara, no to bi zahtijevalo rješavanje jednadžbe petog stupnja za svaku točku. Umjesto toga se izračunava rezultat rascjepa (engl. *Spline*) pomoću rasterizacije koristeći učinkovit iterativan način, a nakon toga se broje vrijednosti elementa slike koje se nalaze na krivulji. Algoritam je implementiran koristeći C++ i *OpenCV* biblioteku te je korištena *caltech* baza slika i videa. Algoritam je pokrenut na dva načina. U prvom je potrebno detektirati samo dvije uzdužne kolničke oznake koji radi s preciznošću od 96.34%, a drugi način je detekcija svih

uzdužnih kolničkih oznaka na slici gdje je preciznost 90.89%. Rezultati testiranja su pokazali da nastaje problem prilikom vožnje desnom stranom, a ne postoji desna rubna uzdužna kolnička oznaka te algoritam detektira lažni pravac na mjestu gdje bi ona trebala biti. Unatoč velikoj preciznosti algoritma, zbog više različitih izračuna algoritam ne bi zadovoljavao brzinu procesiranja koja je potrebna za većinu ADAS Alpha ploči.

Metoda predložena u radu [7] kombinira Konvolucijsku neuronsku mreža (*engl. Convolutional neural network* - CNN) i Povratnu neuronsku mreža (*engl. Recurrent neural network* - RNN) za detekciju vozne trake s nizom kontinuiranih video okvira iz vožnje. Tijekom vožnje, okviri snimljeni prednjom kamerom su uzastopni, te vozna traka u jednom i vozna traka u prethodnom okviru obično se preklapaju što omogućava detekciju vozne trake u okviru predviđanja vremenske serije pomoću RNN. Arhitektura mreže nalazi se na slici 2.2. CNN koder i CNN dekoder su dvije potpune konvolucijske mreže. Uz niz kontinuiranih okvira kao ulaz, CNN koder obrađuje svaki od okvira i dobiva vremensku seriju mapa značajki. Zatim se te značajke unose u mrežu dugoročne memorije (engl. *Long short-term memory* - LSTM) za informacije o voznoj traci. Izlaz iz LSTM-a predaje ih u CNN dekoder kako bi se izradila mapa vjerojatnosti za predviđanje vozne trake. Mapa vjerojatnosti ima istu veličinu kao i ulazna slika. Testiranje je provedeno na *Intel Core Xeon E5-2630, 2.3GHz, 64GB Ram* i dvije *GeForce GTX TITAN-X*, slike za treniranje su preuzete iz *TUSimple* baze te vlastite slike snimljene s kamerom na vozilu. Testiranje je provedeno u različitim uvjetima vožnje, s dvije ili više uzdužnih kolničkih oznaka koje su isprekidane ili kontinuirane te koje su zakrivljene ili ravne. Predloženi model pokazao je veće performanse s relativno većom preciznošću i točnošću u odnosu na druge modele. Pored toga, zbog različitih uvjeta vožnje je pokazao i dobru robusnost, no u usporedbi s drugim modelima potrebna mu je puno veća procesorska moć za obradu i treniranje kao što je i sama implementacija zahtjevnija.



Slika 2.2. Arhitektura mreže za detekciju vozne trake.

Glavna metoda kojom se detektira vozna traka u radu [8] je metoda raspoređivanja koordinatnih točaka. Prije glavne metode potrebno je sliku dobivenu direktno s kamere postavljene na vozilu, pretvoriti u HSV prostor boja (engl. *Hue, saturation, value*). Slika se pretvara u HSV prostor boja zbog lakše identifikacije objekta na slici u usporedbi s ostalim prostorima. Sljedeći korak je filtriranje slike pomoću konturnog filtra koji je dizajniran za segmentiranje objekata s unaprijed definiranom veličinom, dakle korišten je primarno u radu za uklanjanje predmeta koji su sigurno izvan raspona veličine uzdužnih kolničkih oznaka. Nakon toga slijedi uklanjanje prostora na slici gdje se sigurno ne nalazi vozna traka. Uklanjanje je obavljeno pomoću regije od interesa (engl. *Region of interest* - ROI). Veličina regije od interesa je postavljena na prosjek širina trake 2.5 metara s prosječnom duljinom od 1-2 metra. Nakon toga slijedi glavni dio algoritma. Prvi korak je utvrđivanje vrijednosti točke uzorka. Točka uzorka će se koristiti kao mjesto za provjeru pravca, a njen broj utječe na razinu preciznosti otkrivanja pravca. Nakon određivanja točke uzorka, postupak skeniranja se izvodi okomito i vodoravno od početnog piksela. Ako je točka uzorka otkrivena kao točka pravca, vrijednosti koordinate će biti pohranjene. Sljedeći korak je uspoređivanje unutar dozvoljenog raspona pohranjenih koordinata s referentnom točkom koja je ustvari prva točka s najvećim brojem stupaca. Ako je izvan raspona, vrijednost koordinate će se pomjeriti te se ponovo provodi usporedba a vrijednost prethodne točke postaje nula. Nakon ovoga, pomoću jednadžbe pravca se predviđaju ostale točke koordinate. Za kraj slijedi proračun položaja vozila kako bi vozilo uvijek bilo u sredini vozne trake koji se računa u ovisnosti koliko je pravaca pronađeno. Testiranje je provedeno snimajući kamerom *Logitech* C270. U testiranju je cilj bio provjeriti koliko točno algoritam procjenjuje položaj vozila u odnosu na sredini vozne trake, čiji je uspjeh od 0.0992 metara s brzinom obrade od.0625 okvira po sekundi. U ovome radu je predložene su različite ideje koje se mogu koristiti u sličnim algoritmima, no sam algoritam radi dosta loše u lošijim uvjetima zbog nemogućnosti brzog traženja referentne točke.

3. PREDLOŽENO RJEŠENJE ZA UPOZORAVANJE VOZAČA O NAPUŠTANJU VOZNE TRAKE

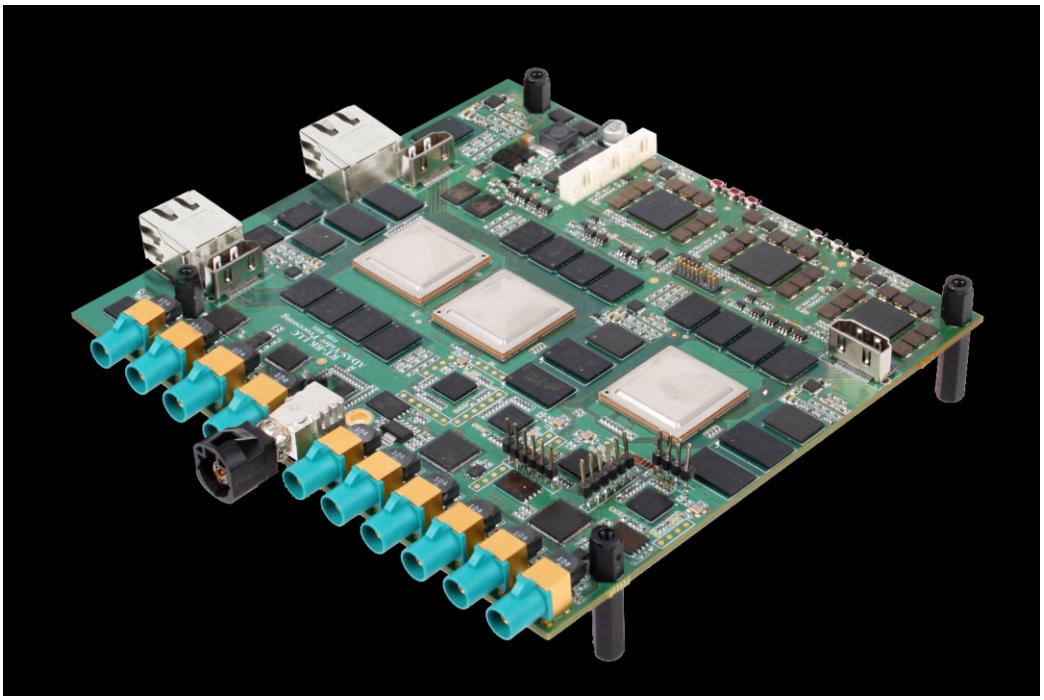
U ovom poglavlju predloženo je rješenje za detekciju vozne trake i upozoravanje vozača o napuštanju vozne trake. Koncept rješenja razvijen je na osobnom računalu (engl. *Personal Computer* – PC) u programskom jeziku C++ uz pomoć biblioteke *OpenCV* (engl. *Open Source Computer Vision library*), a zatim je rješenje implementirano na ADAS Alpha ploču u razvojnom okruženju *VisionSDK* uz pomoć makete vozila i kamera.

Algoritam za upozoravanje vozača o napuštanju vozne trake podijeljen je u više dijelova. Kao ulaz u algoritam koristi se slika dobivena s kamere montirane na prednjoj strani vozila. Na ulaznu sliku se najprije primjenjuju metode predobrade slike kao što su izdvajanje regije od interesa, *Gauss*-ovo filtriranje, filtriranje po boji te *Sobel* detekcija rubova. Nakon primjene navedenih metoda predobrade dobiva se binarna slika koja je ulaz u *Hough*-ovu transformaciju koja služi za detekciju pravaca na slici. Pomoću *Hough*-ove transformacije na slici se detektiraju uzdužne kolničke oznake koje pripadaju voznoj traci u kojoj se nalazi vozilo. Dobiveni pravci koji predstavljaju uzdužne kolničke oznake koje omeđuju voznu traku. Pomoću dobivenih kutova detektiranih pravaca moguće je upozoriti vozača o napuštanju vozne trake.

3.1. Opis ADAS Alpha ploče i *VisionSDK* razvojnog alata

ADAS algoritmi implementiraju se na odgovarajuće ugradbene računalne platforme koje su pogodne za ugradnju u vozilo. Jedan od takvih razvojnih platformi je i ADAS Alpha ploča s pripadnim razvojnim softverom koja je prikazana na slici 3.1. Ploča je izgrađena od strane *Texas Instruments*-a u suradnji s institutom RT-RK. Razvijena je kako bi mogla pružati podršku za osnovne i napredne sustave upozoravanja, aktivne sustave za upravljanje te polu-autonomne operacije. Alpha ploča sadrži tri TDA2X sustava na čipu (engl. *Systems on Chip* – SoC):

- SoC #1 – TDA2SX #1 – SC (*Surround Camera*) SoC
- SoC #2 – TDA2SX #2 – FFN (*Front view camera near angle stereoscopic view, Front view camera wide angle, Night vision camera*) SoC
- SoC #3 – TDA2SX #3 – FUS (*Fusion*) SoC

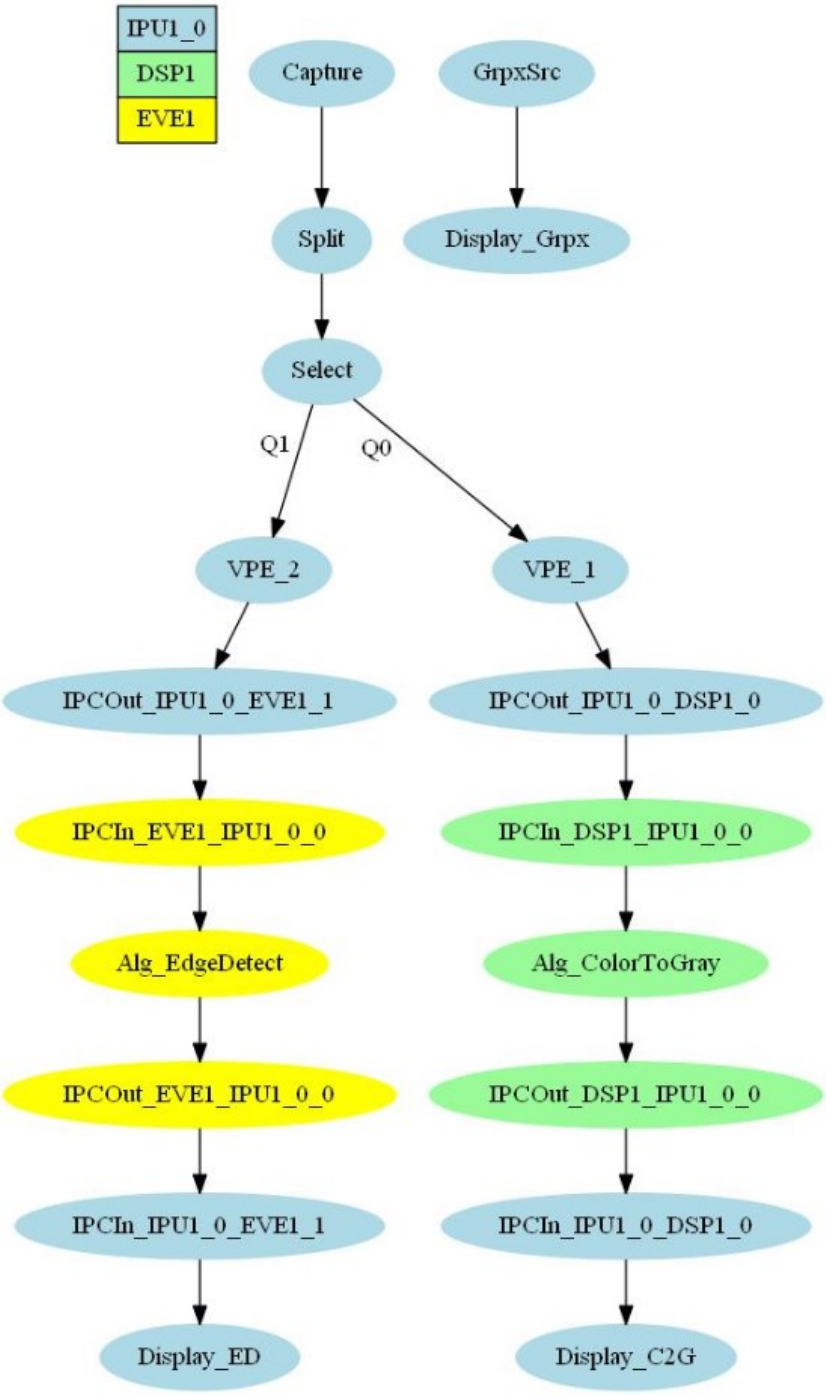


Slika 3.1. ADAS Alpha ploča.

Svaki SoC sadrži nekoliko različitih procesorskih jezgri (dva DSP, dva A15, dva M4 i četiri EVE procesora), a korisnik je u mogućnosti raspoređivati zadatke između njih, pri čemu mora uzeti u obzir razlike u radnim frekvencijama i namjeni samih procesorskih jezgri. Osim toga ploča na sebi sadrži dva *Ethernet* priključka, deset priključaka za kamere (6 za SC Soc te 4 za FFN Soc), tri utora za *microSD* kartice gdje svaka kartica odgovara jednom od SoC-ova, a služe za pohranu programskog koda kojeg je potrebno izvršavati. Također na ploči se nalaze HDMI izlaz, UART i JTAG priključak.

Za potrebu razvoja algoritama tvrtka *Texas Instruments* razvila je više-procesorsko programsko razvojno okruženje pod nazivom *VisionSDK*. Unutar navedenog okruženja programerima je omogućeno stvaranje korisničkih slučajeva (engl. *Use-case*) putem kojih se definira tok podataka. Okruženje dolazi s unaprijed kreiranim korisničkim slučajevima putem kojih se demonstriraju mogućnosti korištenja ADAS Alpha ploče te samog programskog okruženja. Nadalje, *VisionSDK* omogućava snimanje/ preuzimanje videa, algoritme za kompresiju te prikaz i analizu videa. Okvir (engl. *Framework*) na kojem je zasnovan *VisionSDK* su veze i lanci (engl. *Links and chains*), a okvir se može sastojati od više veza povezanih u jednu cjelinu. Svaka veza ima svoj dio posla koji izvršava, npr. detekcija rubova, različita filtriranja, a pomoću lanaca, veze se spajaju te tvore naprednije algoritme. Na slici 3.2. prikazan je jedan korisnički slučaj gdje

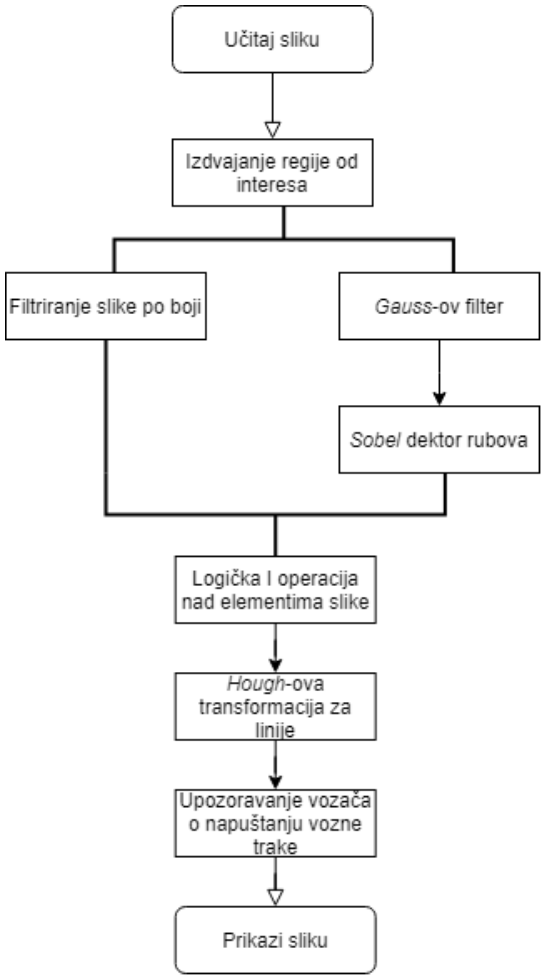
su korištene sljedeće veze: *Capture* služi za dohvaćanje slike s kamere, *Split* za razdvajanje obrade ulazne slike, *Select* pomoću kojega se odabire na koju vezu se predaje slika, *VPE* koji služi za promjenu veličine slike, izdvajanje regije od interesa na slici ili za obavljanje pretvorbe formata boje, *Display* pomoću kojega se prikazuje obrađena slika, *Edge Detection* za detekciju rubova te *ColorToGray* koji pretvara sliku u nijanse sive. Ovako povezane veze čine jedan lanac.



Slika 3.2. Primjer korisničkog slučaja u *VisionSDK*.

3.2. Koncept algoritma za upozoravanje vozača o napuštanju vozne trake

Iterativni postupak razvoja programske podrške nezaobilazan je kako za ugradbene računalne sustave tako i za sve ostale. No, kod ugradbenih računalnih sustava, svaka izmjena u kodu zahtjeva veliki vremenski trošak. Zbog učestalosti izmjena u kodu za vrijeme razvoja rješenja na ADAS Alpha ploči, prevođenje jednog takvog koda može iziskivati veliko vrijeme. U tu svrhu razvijen je prototip rješenja koristeći C++ programski jezik zajedno s *OpenCV* bibliotekom koja je otvorenog koda, a u sebi sadrži preko 2500 algoritama korištenih za obradu slike. Pomoću već ugrađenih algoritama, značajno se ubrzava dolazak do konačnog rješenja. Na slici 3.3. dan je blok prikaz algoritma za upozoravanje vozača o napuštanju vozne trake.



Slika 3.3. Blok prikaz algoritma za upozoravanje vozača o napuštanju vozne trake.

Prvi korak algoritma predstavlja učitavanje slike na kojoj je potrebno detektirati uzdužne kolničke oznake. Slike korištene u ovom radu su prikupljene pomoću kamere montirane na prednjoj strani vozila čija je rezolucija 480*640 elemenata slike. Učitavanje ulazne slike izvodi se

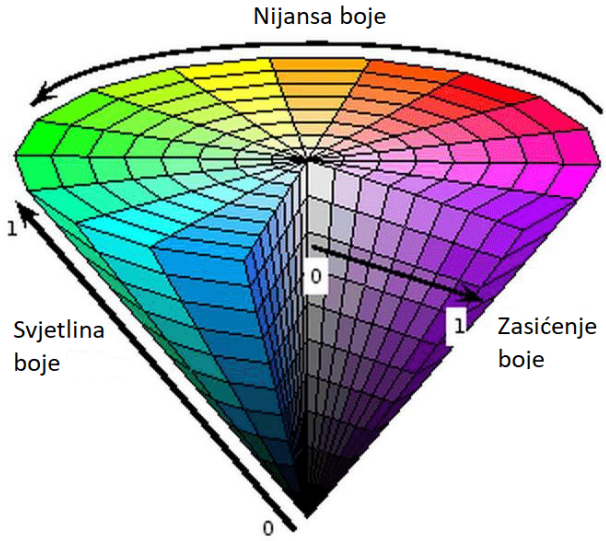
pomoću *OpenCV* funkcije *imread*(), a rezultat se prikazuje korištenjem funkcije *imshow*(). Slika 3.4. prikazuje jednu takvu ulaznu sliku u algoritam iz Kitti skupa slika.

Slika 3.4. Ulazna slika u algoritam iz Kitti skupa slika.

Nakon učitavanje se iz slike izdvaja regija od interesa. S obzirom na to da se kolnik uvijek nalazi u donjem dijelu slike potrebno je taj dio izdvojiti kao regiju od interesa pomoću funkcije *Rect*. Slika 3.5. prikazuje izdvojenu regiju od interesa s ulazne slike.

Slika 3.5. Izdvojena regija od interesa s ulazne slike.

Nakon izdvajanja regije od interesa obrada se grana u dvije grane od kojih će jedna izvršiti filtriranje po boji, a druga *Gauss*-ovo filtriranje i detekciju rubova pomoću *Sobel* operatora. Sljedeći objašnjeni korak algoritma je filtriranje po boji. Zbog robusnost na različito osvjetljenje objekata, filtriranje slike po boji se odvija u HSV prostoru boja prikazanim na slici 3.6. Ovaj prostor boja sastoji se od tri komponente: nijanse boje H, zasićenja boje S i svjetline boje V. Raspon nijanse boja kreće se unutar 0 i 180, dok se zasićenje boje i svjetlina boje nalaze u rasponu od 0 do 255 [9].

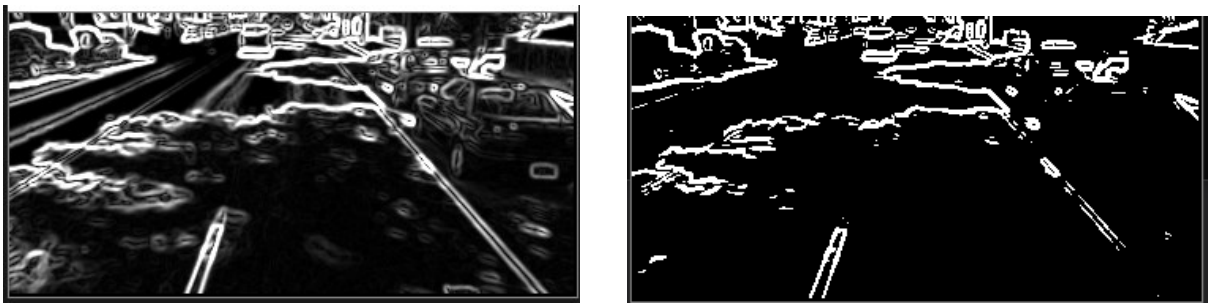


Slika 3.6. Prikaz HSV prostora boja.

Filtriranje se vrši unutar HSV prostora boja jer navedeni prostor razlikuje intenzitet svjetlosti od komponente boje. Elementi slike koji zadovoljavaju prikladan raspon vrijednosti imaju vrijednost 255, dok oni koji ne zadovoljavaju isti uvjet imaju vrijednost 0, dakle dobiva se binarna slika. U ovom radu filtriranje po boji je provedeno za žutu i bijelu boju jer su uzdužne kolničke oznake najčešće označene s tim bojama. Korišteni raspon H komponente za žutu boju u radu je 15-45, S komponenta žute boje je u rasponu 30–255 a V komponenta žute boje je u rasponu 50-255. Korišteni raspon H komponente za bijelu boju u radu je 0-150, S komponenta bijele boje je u rasponu 0-150 a V komponenta bijele boje je u rasponu 100-255. Pomoću *OpenCV* naredbe *inRange* koja prima ulaznu sliku te donje i gornje vrijednosti pragova svih kanala određenog modela boja izvršava se filtriranje po boji. Na slici 3.7. prikazan je rezultat primjene filtera boja na ulaznu sliku. Na slici se vidi da ovakav način filtriranja „propušta“ uzdužne kolničke oznake u izlaznoj slici no također i druge objekte koji zadovoljavaju granice filtera.

Slika 3.7. Rezultat primjene filtera boja na ulaznu sliku.

Detekcijom rubova pronalaze se dijelovi slike na kojima postoji nagla promjena intenziteta svjetline koja često odgovara konturama objekta ili granicama između različitih segmenata objekta. Za početak potrebno je ulaznu sliku pretvoriti u nijanse sive boje kako bi se na njoj mogla obaviti detekcija rubova. Također, prije same detekcije bitno je obaviti *Gauss*-ov filter kako bi se smanjio broj detalja na slici što ujedno umanjuje šum prisutan na slici. *Gauss*-ovo filtriranje obavljeno je pomoću *OpenCV* funkcije *GaussianBlur* gdje je potrebno odrediti vrijednost standardne devijacije te dimenzije konvolucijske matrice. U radu korištena je 3x3 konvolucijska matrica. U radu su rubovi detektirani pomoću *Sobel* detektora rubova. *Sobel* detektor rubova radi na principu gradijenta svjetline. Naime, dva objekta u pravilu bi trebala imati naglašenu razliku u svjetlini. Pomoću *Sobel*-ovog operatora određuje se usmjerena promjena intenziteta elementa slike u horizontalnom i vertikalnom smjeru te ukupna jakost ruba. Operator je implementiran izračunavanjem horizontalnog i vertikalnog gradijenta gdje se dobiju dvije matrice koje se koriste za izračun ukupne jakosti ruba u promatranom elementu slike. Nakon toga, potrebno je zbog prelijevanja vrijednost van raspona (engl. *Overflow*) postaviti sve vrijednosti koje se prelijevaju na 255 te na kraju spremiti rezultat. U *Sobel*-ov detektor rubova dodan je prag pomoću kojega su na izlaznoj slici prikazane samo vrijednosti elementa slike koje su veće od 180 što je eksperimentalno utvrđeno kao najbolja granica praga.. Rezultat ove operacije je binarna slika s istaknutim rubovima. Na slici 3.8. prikazana je usporedba rezultata primjene *Sobel* detektora rubova s i bez primijenjenog praga.



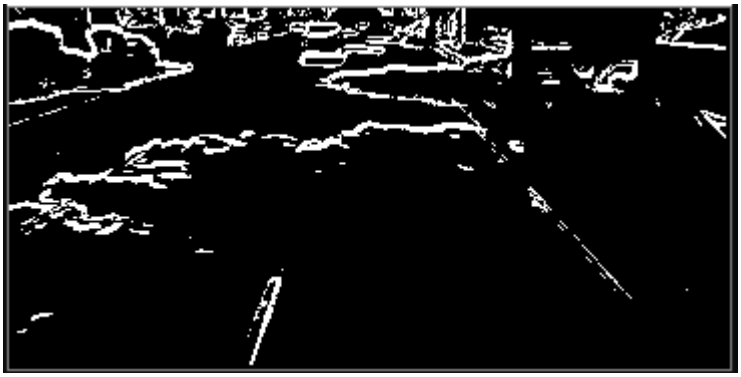
a) b)

Slika 3.8. Usporedba rezultata primjene *Sobel* detektora rubova:

a) Bez primijenjenog praga

b) S primijenjenim pragom

Sljedeći korak je proces spajanja dviju binarnih slika od kojih je prva slika filtrirana po boji, a druga slika je slika s istaknutim rubovima. Slika se dobiva korištenjem *bitwise_and()* operacije. Ovo je logička operacija I koja se odvija između odgovarajućih elemenata slike dvaju rezultantnih slika iz prethodnog koraka obrade. Ovaj korak je potreban kako bi na rezultantnoj slici ostali rubovi objekata na slici, no samo oni koji odgovaraju bijeloj odnosno žutoj boji. Na slici 3.9. prikazana je rezultat primjene *bitwise_and()* operacije.



Slika 3.9. Rezultat primjene *bitwise_and()* operacije.

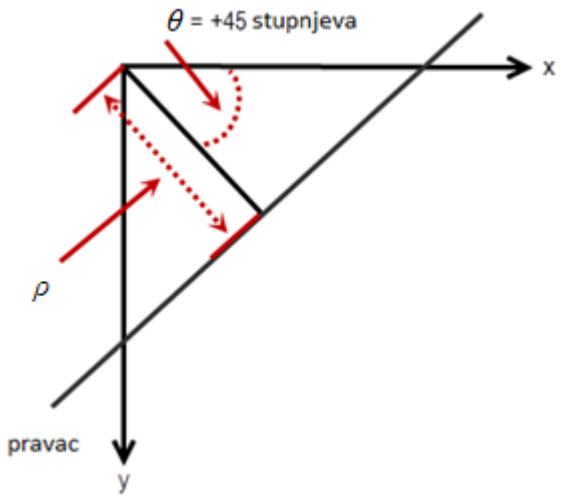
Rezultanta slika iz prethodnog koraka je binarna slika pa je na njoj moguće sada provesti *Hough* - ovu transformaciju s ciljem detektiranja pravaca koji su prisutni na ulaznoj slici poput uzdužnih kolničkih oznaka. Kako bi se *Hough*-ova transformacije obavila potrebno je prikazati elemente slike iz prethodnog koraka u *Hough*-ovom prostoru (engl. *Hough Space*). Poznato je da se pravac može generalno opisati s izrazom (3-1) gdje su y i x koordinate jedne točke, a je nagib pravca, a b odsječak na y odnosno na x osi.

$$y = ax + b \tag{3-1}$$

Postoji jedna mana s predstavljanjem pravca u gore navedenom obliku. Algoritam neće moći detektirati okomite pravce jer je nagib a beskonačan za okomite pravce. Kako bi se izbjegao ovaj problem, pravac se predstavlja u polarnom koordinatnom sustavu prema (3-2) gdje je θ (*theta*) kut između x osi i linije koja povezuje ishodište s tom najbližom točkom, a ρ (*rho*) je udaljenost od ishodišta do najbliže točke na ravnom pravcu [10].

$$\rho(x, y) = x \cos\theta + y \sin\theta \quad (0 < \theta < \pi) \tag{3-2}$$

Na slici 3.10. prikazan je pravac u polarnom koordinatnom sustavu. Ishodište koordinatnog sustava prilikom implementacije nalazi se u gornjem lijevom kutu koji je prikazan na slici.



Slika 3.10. Pravac u polarnom koordinatnom sustavu.

Za izvođenje *Hough*-ove transformacije potrebno je odrediti veličinu akumulatora u obliku matrice čije se dimenzije računaju prema izrazu (3-3) gdje je ρ_{max} ustvari dvostruka dijagonala ulazne slike jer je potrebno omogućiti spremanje i negativnih vrijednosti za ρ koje se postižu u slučajevima kada je pravac pod kutom većim od 90°. Kreirana matrica popunjava se inicijalno s nulama, a tijekom izvođenja algoritma vrijednosti matrice povećavaju se za jedan što označava jedan glas (engl. *Vote*) za određeni pravac. Dakle, vrijednosti ρ_{max} imaju raspon od 0 do dvostruke vrijednosti dijagonale, a θ_{max} predstavlja broj vrijednosti kutova koje pravac može poprimiti, a u

radu je to 180°. Dakle, algoritam za svaki element binarne slike izračunava vrijednost ρ prema izrazu (3-2) a zatim povećava vrijednost θ za 2° prilikom svakog idućeg izračuna.

$$\rho_{max} = 2 * \sqrt{{a_{max}}^2 + {b_{max}}^2}$$
$$\theta_{max} = \pi = 180° \rightarrow 180 \tag{3-3}$$

Važno je napomenuti kako je prije izračuna vrijednosti ρ potrebno pretvoriti svaku vrijednost kuta u radijane. Osim toga, prilikom izračuna na vrijednost ρ koja je dobivena za određeni element slike potrebno je zbrojiti vrijednost dijagonale ulazne slike kako bi se vrijednost ρ postavila u raspon između 0 i dvostruke vrijednosti dijagonale [11]. U radu, veličina regije od interesa je 410 *120 elemenata slike što znači da je dijagonala slike jednaka 428 te se ta vrijednost dijagonale dodaje na izračunatu vrijednost ρ. Nakon što se navedeni proces izvede za svaki element slike koji ima vrijednost 255, potrebno je pronaći u popunjenom akumulatoru dvije vrijednosti koje imaju najviše glasova, smatrajući da su to pravci koji bi trebali odgovarati uzdužnim kolničkim oznakama. Prilikom pretraživanja *Hough*-ovog akumulatora glasova potrebno je postaviti uvjet da se apsolutna razlika kutova prvog i drugog pravca nalaze unutar određenog intervala kako bi algoritam pronašao dva pravca koji su međusobno udaljeni za postavljenu apsolutnu razliku. Ovaj korak potreban je zato što dvije najveće vrijednosti mogu biti jedna blizu druge u akumulatoru što odgovara pravcima iste uzdužne kolničke oznake. U ovome radu je empirijski određeno da je ta apsolutna razlika interval od 40 do 110. Za detekciju vozne trake uzimaju se vrijednosti iz akumulatora koje su iznad nekog praga čime se postiže detekcija pravca koji odgovara rubovima uzdužnih kolničkih oznaka. Nakon pronalaska dvaju najvećih vrijednosti u akumulatoru, vrijednosti ρ i θ *se* predaju funkciji *polarToCartesian*() zajedno s točkama *p1* i *p2* u koje će biti spremljena vrijednost *x* i *y* koordinata pomoću kojih je moguće nacrtati pravac. Prikaz koda koji obavlja vraćanje u kartezijev koordinatni sustav dan je na slici 3.11. Pomoću *OpenCV* funkcije *line*() prikazuje se pravac koji odgovara uzdužnim kolničkim oznakama na izlaznoj slici.

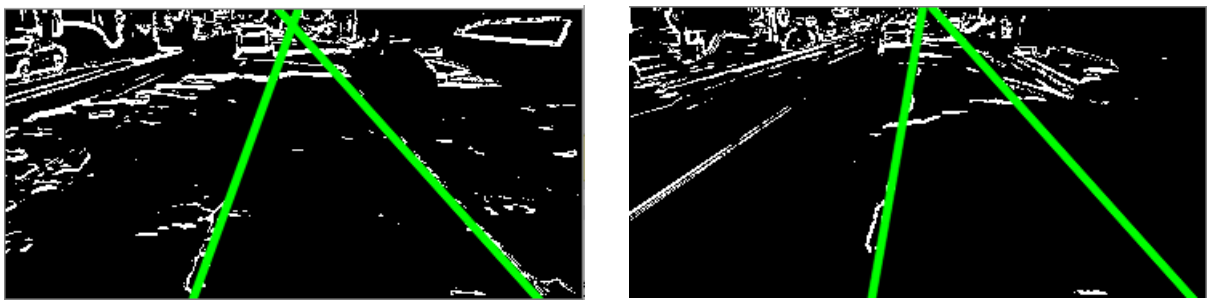
Linija ***Kod***

```
1:      void polarToCartesian(double rho, int theta, Point& p1, Point& p2){
2:           double theta1 = 0.0;
3:           theta1 = ((double)theta - 90) * 3.1415) / 180
4:           int x0 = cvRound(rho * cos(theta1));
5:           int y0 = cvRound(rho * sin(theta1));
6:           p1.x = cvRound(x0 + 1000 * (-sin(theta1)));
```

```
7:            p1.y = cvRound(y0 + 1000 * (cos (theta1)));
8:            p2.x = cvRound(x0 - 1000 * (-sin (theta1)));
9:            p2.y = cvRound(y0 - 1000 * (cos(theta1)));
10:     }
```

Slika 3.11. Pretvorba u kartezijev koordinatni sustav.

Zadnji korak algoritma predstavlja dio za upozoravanje vozača o napuštanju vozne trake. Na slici 3.12. prikazan je položaj uzdužnih kolničkih oznaka vozne trake. Na slici *a*) vozilo se nalazi unutar vlastite vozne trake. U slučaju kada se vozilo nalazi unutar vlastite vozne trake, empirijski je utvrđeno da pravac koji odgovara lijevoj uzdužnoj kolničkoj oznaci ima vrijednosti θ u rasponu od 110 do 130, a pravac koji odgovara desnoj uzdužnoj kolničkoj oznaci nalazi se u rasponu od 46 do 62. Na slici *b* prikazano je vozilo koje napušta voznu traku. Kako vozilo napušta voznu traku tako detektirani pravac koji odgovara lijevoj uzdužnoj kolničkoj oznaci počinje imati sve manju vrijednost θ dok praktički ne postane okomit, a vrijednost ρ se približava centru slike. Desna uzdužna kolnička oznaka za to vrijeme povećava svoju vrijednost ρ, a smanjuje vrijednost kuta θ pod kojim se nalazi. Kako bi se detektiralo napuštanje voze trake postavljena su dva uvjeta od kojih jedan provjerava u kojem djelu slike se nalaze pravci koji označavaju uzdužne kolničke oznake, a drugi uvjet provjerava pod kojim kutom se nalaze pravci koji označavaju uzdužne kolničke oznake. U slučaju kada su oba uvjeta ispunjena ispisuje se poruka „*Prestrojavanje*“. U prilogu P.3.1. dan je odgovarajući kod.



a) b)

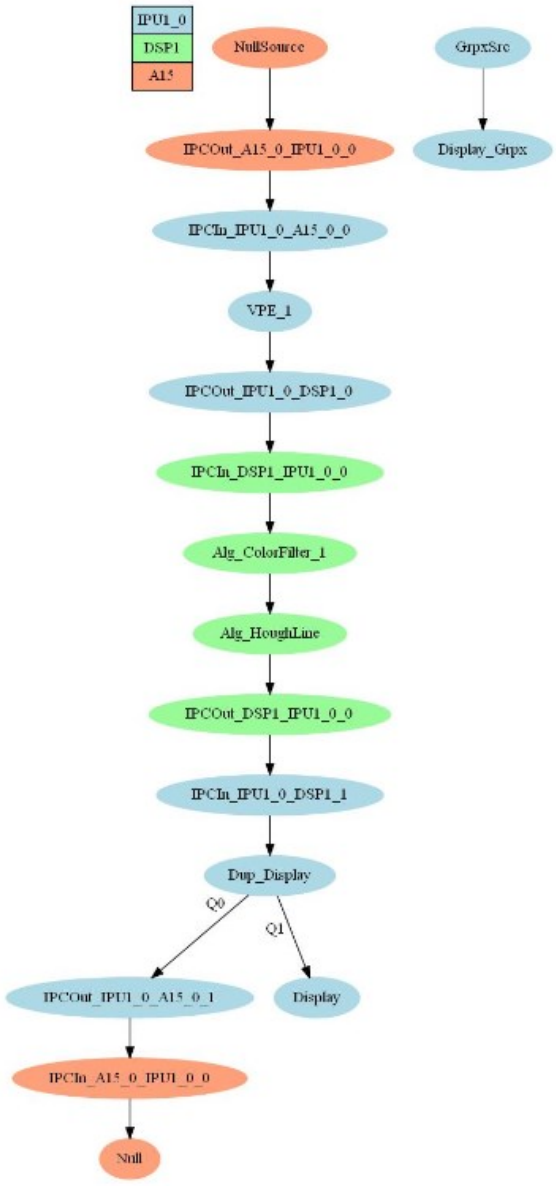
Slika 3.12. Položaj uzdužnih kolničkih oznaka vozne trake:

a) Položaj vozila unutar svoje vozne trake

b) Položaj vozila pri prelasku u lijevu voznu traku

3.3. Razvoj i implementacija rješenja na ADAS Alpha ploču

Nakon što je rješenje implementirano na osobnom računalu koristeći *OpenCV* biblioteku te je njegova ispravnost provjerena, pristupilo se implementaciji rješenja na ADAS Alpha ploči. U ovom potpoglavlju opisana je izrada algoritma u *VisionSDK* okruženju te način pokretanja rješenja na ADAS Alpha ploči. Na slici 3.13. prikazan je izgled korisničkog slučaja implementiranog algoritma. Razvojem algoritma implementirana su dva *link*-a. Prvi je *ColorFilter* link u kojem se obavlja filtriranje po boji, *Gauss*-ov filter te detekcija rubova *Sobel*-ovim operatorom. Drugi link je *HoughLine* u kojem se obavlja *Hough*-ova transformacija za linije te upozoravanje vozača o napuštanju vozne trake.



Slika 3.13. Izrađeni korisnički slučaj implementiranog algoritma.

Prema osmišljenom konceptu prvi korak je učitavanje slike. Za učitavanje slike u *VisionSDK* korišten je već postojeći *link* koji omogućava učitavanje slike s mreže. S obzirom na to da *link* za učitavanje slika s mreže prima kao ulazni format samo YUV 4:2:0 format, potrebno je prije same obrade slike format YUV 4:2:0 pretvoriti u RGB prostor boja. YUV 4:2:0 format komprimira U i V kromatske komponente slike, samim time smanjuje količinu informacije o boji unutar slike pa se zbog ovoga može očekivati razlika u vrijednostima elemenata slike između koncepta rješenja te rješenja na ADAS Alpha ploči. Sljedeći korišteni *link* je VPE *link*. VPE *link* podržava pretvorbu formata boja u YUV 4:2:2 ili YUV 4:4:4. Pomoću ovoga *link*-a format boja je pretvoren iz YUV 4:2:0 u YUV 4:2:2 radi lakše obrade i prijenosa slike.

YUV 4:2:2 format boja sastoji se od jedne U i V kromatske komponente na koje dolaze dvije Y luminantne komponente. Sve tri komponente elemenata slike prikazane su kao 8-bitni cijeli broj u memoriji računala [12]. Nakon pretvorbe YUV 4:2:0 formata u YUV 4:2:2 potrebno je pretvoriti sliku u RGB prostor boja. RGB prostor boja sastoji se od tri kanala boje: crveni (R), zeleni (G) i plavi (B), gdje se za svaki element slike pohranjuju sve tri vrijednosti dok se ostale boje dobivanju pomoću tri osnovne [13]. Pretvorba boja iz YUV 4:2:2 u RGB, a zatim iz RGB u HSV prostor boja prikazana je na slici 3.14. Proračuni za pretvorbu YUV 4:2:2 u RGB nalaze se na linijama 4-9, dok se pretvorba RGB u HSV te potrebni proračuni nalaze na linijama 10-25. Prilikom dobivanja H komponente, vrijednosti komponente mogu biti u intervalu od 0 do 360. Budući da su komponente zapisane kao 8-bitni cijeli broj gdje vrijednost može biti u rasponu od 0 do 255 potrebno je podijeliti H komponentu s 2 kako bi se mogla zapisati. Samim time, H komponenta sada može poprimiti vrijednost u intervalu od 0 do 180, dakle u istom rasponu kao i prilikom izrade koncepta.

Linija ***Kod***

```
1:      Void YUV2HSV(UInt8* YUV, UInt8* HSV){
2:         UInt8 R, G, B;
3:         float Cmin, Cmax, delta, _R, _G, _B;
4:         R = YUV[0] + 1.19389 * (Int8)(YUV[2] - 128);
           G = YUV[0] - 0.39465 * (Int8)(YUV[1] - 128) - 0.58060 *
5:      (Int8)(YUV[2] - 128);
6:         B = YUV[0] + 2.03211 * (Int8)(YUV[1] - 128);
7:         _R = R / (float)255;
8:         _G = G / (float)255;
9:         _B = B / (float)255;
10:        Cmin = min(_R, _G, _B);
11:        Cmax = max(_R, _G, _B);
12:        delta = Cmax - Cmin;
```

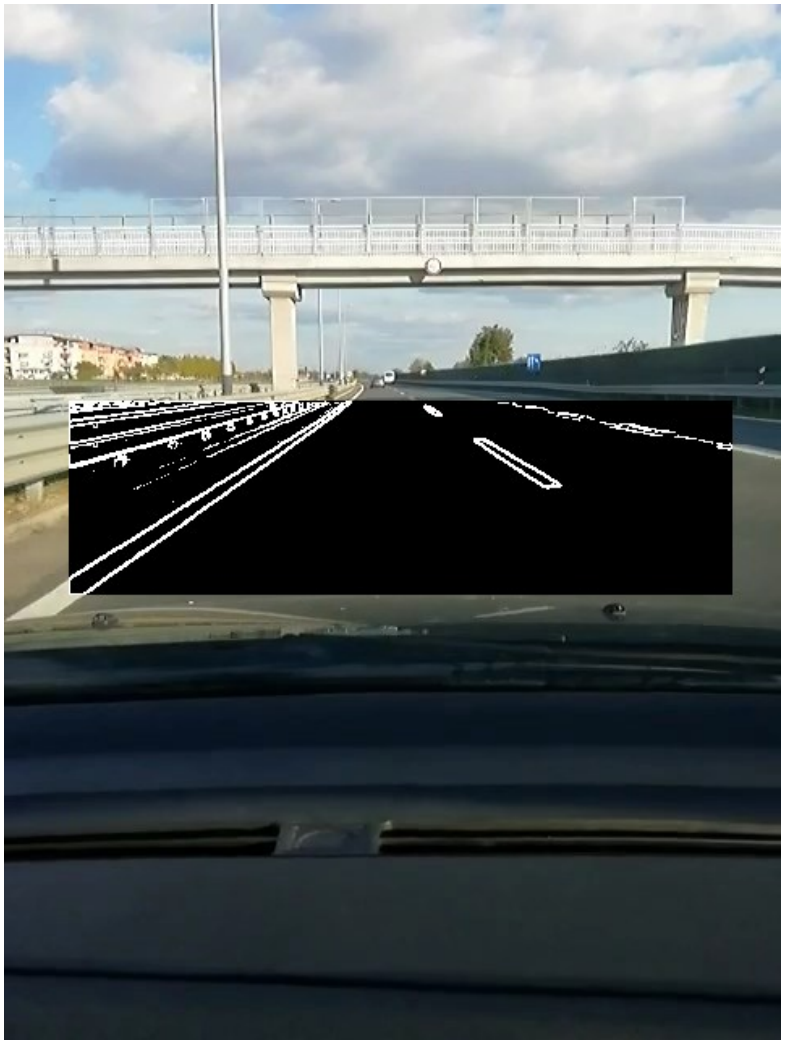
```
13:         if(Cmax == Cmin){
               HSV[0] = (UInt8)0;
14:         }
15:         else if(Cmax == _R){
               HSV[0] = (UInt8)round( (60 * ((_G - _B) / delta)) / 2);
16:         }
17:         else if(Cmax == _G){
               HSV[0] = (UInt8)round(((60 * ((_B - _R) / delta)) + 120) / 2);
18:         }
19:         else if(Cmax == _B){
               HSV[0] = (UInt8)round(((60 * ((_R - _G) / delta)) + 240) / 2);
20:         }
21:         if(Cmax != 0){
               HSV[1] = (UInt8)round((delta / Cmax) * 255);
22:         }
23:         else{
               HSV[1] = (UInt8)0;
24:         }
25:         HSV[2] = (UInt8)round(Cmax * 255);
26:     }
```

Slika 3.14. Pretvorba formata boja.

Nakon pretvorbe zapisa slike u HSV prostor boja, potrebno je obaviti filtriranje boja. Prilikom filtriranja korištene su iste vrijednosti H, S i V komponente kao i u konceptnom rješenju implementiranom na osobnom računalu. Konceptno rješenje u ovome koraku vraća binarnu sliku, no u ovome slučaju potrebno je postaviti vrijednosti Y komponente na 255 ako je element slike unutar zadanih graničnih vrijednosti odnosno, 0 ako nije unutar graničnih vrijednosti. Unutar istog *link*-a, obavlja se i *Gauss*-ov filter i *Sobel* detektor rubova na isti način kao i u *OpenCV* -u. Kako bi se prikazala rezultantna slika na zaslonu, vrijednosti U i V komponente postavljaju se na vrijednost 128 koja odgovara vrijednosti crne boje, dok se Y komponenta postavlja na vrijednost 255.

Izdvajanje regije od interesa putem VPE *link*-a ne smanjuje brzinu izvođenja algoritma zato što se obrada događa na cijeloj slici. Kako bi se izbjegla obrada cijele slike, a samim time i smanjila brzina izvođenja algoritma, potrebno je obrađivati samo regiju od interesa koja je definirana svojom visinom i širinom. Visina i širina regije od interesa je jednaka kao i u konceptnom rješenju. Nakon toga, slika se na izlazu prikazuje u izvornoj veličini, dok se regija od interesa prikazuje u obrađenom obliku, odnosno kao binarna slika. Pomoću logičke operacije I sada je potrebno spojiti rezultantnu sliku koja se dobiva filtriranjem po boji te rezultantnu sliku koja se dobiva *Sobel* detektorom rubova. Prethodno se moralo osigurati da su slike iste veličine kako bi se mogli uspoređivati elementi slika na istim pozicijama. Ako su vrijednosti na istoj poziciji obiju slika 255, izlazna slika će na tom mjestu imati vrijednost 255, u suprotnom izlazna slika će na toj poziciji imati vrijednost 0. Sada se na sliku dobivenu Sobel detektorom rubova primjenjuje prag pomoću

kojega na slici ostaju vrijednosti elementa slike koje su veće od njega. Vrijednost praga empirijski je utvrđena i iznosi 200. Vrijednost praga nije ista kao i u implementaciji na osobnom računalu što je i za očekivati zbog gubitaka dobivenih prilikom pretvorbe ulazne slike u YUV 4:2:0 format boja. Na slici 3.15. prikazana je vozna traka dobivena nakon *ColorFilter link*-a, unutar kojeg se obavlja filtriranje po boji, *Gauss*-ovo filtriranje te izdvajanje rubova putem *Sobel* operatora.



Slika 3.15. Prikaz vozne trake nakon *ColorFilter link* – a.

Prema izgrađenom konceptu sljedeći korak je *Hough*-ova transformacija koja je implementirana u *HoughLine link*-u koji se nalazi u prilogu P.3.2. Prilikom razvoja *Hough*-ove transformacije na ADAS Alpha ploči, potrebno je napraviti akumulator za spremanje glasova. Razlika u dimenzijama akumulatora u odnosu na *OpenCV* je zbog YUV(4:2:2) formata slike pa se mora uzeti u obzir da je širina slike dvostruko veća pri izračunima jer za svaku vrijednost elementa slike postoji Y i U ili Y i V komponenta. Način na koji su prikazane linije na ADAS Alpha ploči nalazi se na slici 3.16. Prva linija koda predstavlja prolazak kroz širinu slike tako da se uvećava

vrijednost varijable *colIdx* za 2. Uvećavanje varijable *colIdx* za 2 znači prolazak kroz svaki drugi element slike što ustvari predstavlja Y komponentu ulazne slike. Ulaskom u petlju, vrijednost θ preračunava se u radijane te se izračunava pripadni sinus i kosinus dobivene vrijednosti θ. Nakon toga s dobivenim vrijednostima u prethodnom koraku izračunava se vrijednost ρ. Zatim, pokazivač koji pokazuje na određenu memorijsku adresu gdje je pohranjena određena komponenta elementa slika postavlja se na memorijsku adresu izračunate vrijednosti ρ te se toj memorijskoj adresi mijenja U i V komponenta elementa slike u 9 i 10 liniji koda. Nakon svega, potrebno je vratiti pokazivač na početnu poziciju te se petlja dalje izvršava.

Linija ***Kod***

```
1:      for(colIdx= 80; colIdx < (inPitch[0] - 60); colIdx+=2){
2:         theta1 = ((Double)tht * 3.1415)/180.0;
3:         temp1 = cos(theta1);
4:         temp2 = sin(theta1);
5:         y = (int)(((Double)rhoo - ((Double)colIdx * temp1))/ temp2);
6:         if((y > 245 && y < 365)) {
7:            acc1[colIdx] = y;
8:            inputPtr += (inPitch[0]) * acc1[colIdx];
9:            *(inputPtr + colIdx-1) = 255;
10:           *(inputPtr + colIdx+1) = 255;
11:           inputPtr = inPtr[0];
12:        }
13:     }
```

Slika 3.16. Prikaz linija na ADAS Alpha ploči

Kao i u konceptu, zadnji korak algoritma je upozoravanje vozača o napuštanju vozne trake. Slično kao i kod implementacije na osobnom računalu, potrebno je odrediti interval vrijednosti ρ i θ u kojima se nalazi vozna traka kada vozač napušta voznu traku. Vrijednosti su eksperimentalno određene a nalaze se u intervalu od 620 do 660 te u intervalu -515 i -480 za ρ, a za θ intervali su u rasponu od 1 do 20 i 151 do 176. U prilogu P 3.2. dan je korisnički slučaj zajedno sa svim korištenim *link*-ovima.

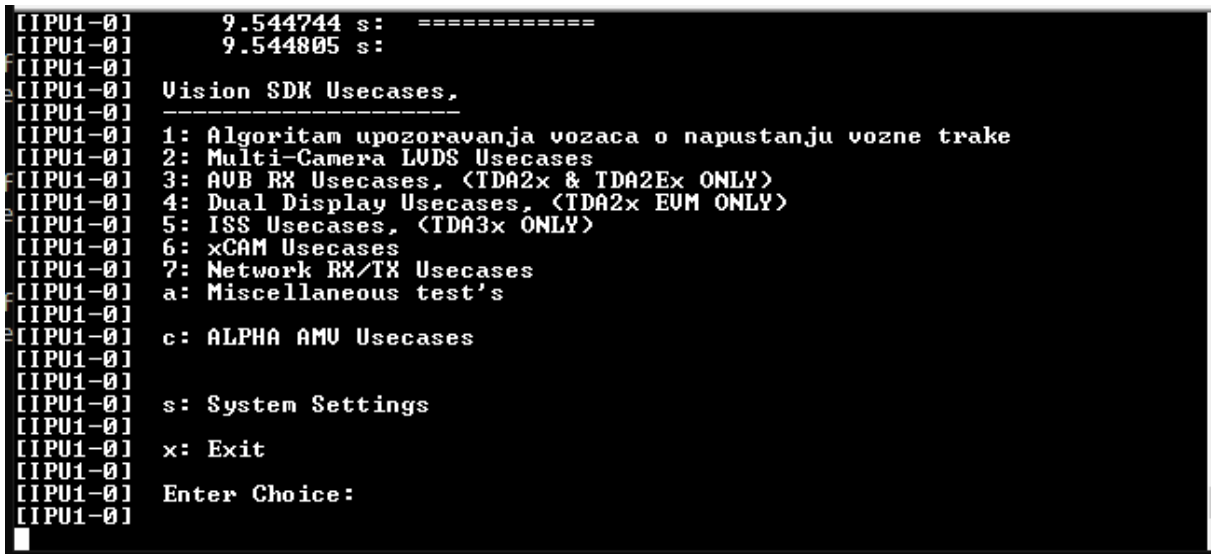
3.4. Pokretanje algoritma na ADAS Alpha ploči

Nakon implementacije korisničkog slučaja te algoritama koji su uključeni u njega, potrebno je dovršiti izgradnju programskog rješenja te ga pokrenuti na ADAS Alpha ploči. Prilikom izgradnje algoritama potrebno se pozicionirati u komadnoj liniji na sljedeću mapu: *C:\VISION_SDK_02_12_01_00\vision_sdk* . Nakon toga naredbom *gmake -s -j depend* izgrađuju

se niže razine *VisionSDK* dok se *bootloader* izgrađuje pomoću naredbe *gmake -s -j sbl_sd*. Sljedeća naredba *gmake –s –j* potrebna je kako bi se izgradili *VisionSDK* dodatci te algoritmi, a naposljetku naredba *gmake –s appimage* omogućava izgradnju *image* datoteke. Izgrađenu *image* datoteku zajedno s *bootloader*-om potrebno je pohraniti na SD karticu, a zatim karticu ubaciti u odgovarajući utor za SoC.

Budući da je u radu korišten *link* za slanje slika preko mreže potrebno je omogućiti statičku IP adresu ploče te postaviti IP adresu ploče i osobnog računala kako bi se mogli povezati. Nakon toga potrebno je povezati osobno računalo i ploču *ethernet* kablom. Osim toga, potrebno je SC SoC UART povezati s USB priključkom na računalu te SC SoC HDMI izlaz povezati na zaslon.

Za komunikaciju računala i ADAS Alpha ploče u radu korišten je *TeraTerm* alat otvorenog koda. Pomoću ovoga alata, omogućeno je upravljanje ADAS Alpha pločom slanjem ASCII znakova očitanih s tipkovnice putem serijske komunikacije te prikaz povratnih informacija dobivenih s ADAS Alpha ploče u prozoru *TeraTerm* alata. Uspješnim povezivanjem računala i ploče, u *TeraTerm* aplikaciji prikazuje se izbornik na kojem je moguće odabrati više korisničkih slučajeva koji je prikazan na slici 3.17. Pritiskom na tipku 1 pokreće se korisnički slučaj izgrađen u okviru ovog rada.



Slika 3.17. *TeraTerm* izbornik korisničkih slučajeva.

Sada je moguće poslati slike koje će se obrađivati izgrađenim korisničkim slučajem. Za slanje podataka putem mreže unutar *VisionSDK* nalazi se skup alata pod nazivom *network_tx* i *network_rx*. *Network_tx* je alat koji služi za slanje MJPEG komprimiranih okvira i RAW/YUV okvira s računala na ADAS Alpha ploču. *Network_rx* alat služi za primanje MJPEG komprimiranih okvira i RAW/YUV/meta-podatkovnih okvira na računalu. Za slanje podataka na

ADAS Alpha ploču najprije je potrebno pretvoriti ulazne slike ili video u odgovarajući YUV 4:2:0 format. U ovome radu ta pretvorba odrađena je putem *ffmpeg* alata otvorenog koda. Nakon toga potrebno je pozicionirati se te kopirati slike ili video u sljedeći direktorij: *C:\VISION_SDK_02_12_01_00\vision_sdk\tools\network_tools\bin*. Slike obrađene algoritmom moguće je pregledati putem *yuvplayer* alata otvorenog koda koji omogućava prikazivanje slika ili video signala spremljenih u različitim YUV formatima.

4. EVALUACIJA PREDLOŽENOG RJEŠENJA ZA UPOZORAVANJE VOZAČA O NAPUŠTANJU VOZNE TRAKE

Ovo poglavlje sadrži evaluaciju implementacije predloženog rješenja na osobnom računalu i na ADAS Alpha ploči. Potpoglavlje 4.1. opisuje način evaluacije, korištene metode i korišteni podatkovni skup. U potpoglavlju 4.2. prikazani su kvantitativni rezultati evaluacije, a potpoglavlje 4.3. sadrži kvalitativne rezultate evaluacije. Evaluacija koncepta rješenja provedena je putem osobnog računala i biblioteke *OpenCV*, a nakon toga razvijeni algoritam na ADAS Alpha ploči evaluiran je putem osobnog računala i *TeraTerm* alata.

4.1. Način evaluacije

Za evaluaciju predloženog rješenja za upozoravanje vozača o napuštanju vozne trake koriste se točnost (engl. *Accuracy*), preciznost (engl. *Precision*), odziv (engl. *Recall*) te *F1* mjera. Pri izračunavanju ovih mjera potrebno je poznavati broj istinito pozitivnih (engl. *True positive*, *TP*), istinito negativnih (engl. *True negative*, *TN*), lažno pozitivnih (engl. *False positive*, *FP*) te lažno negativnih (engl. *False negative*, *FN*) rezultata. Točnost označava udio točno klasificiranih primjera u skupu svih primjera te se dobiva izrazom (4-1).

$$točnost = \frac{TP + TN}{TP + TN + FP + FN} \tag{4-1}$$

Preciznost označava udio točno klasificiranih primjera u cjelokupnom skupu pozitivno klasificiranih primjera te se dobiva izrazom (4-2).

$$preciznost = \frac{TP}{TP + FP} \tag{4-2}$$

Odziv označava udio točno klasificiranih primjera u skupu svih pozitivnih primjera te se dobiva izrazom (4-3).

$$odziv = \frac{TP}{TP + FN} \tag{4-3}$$

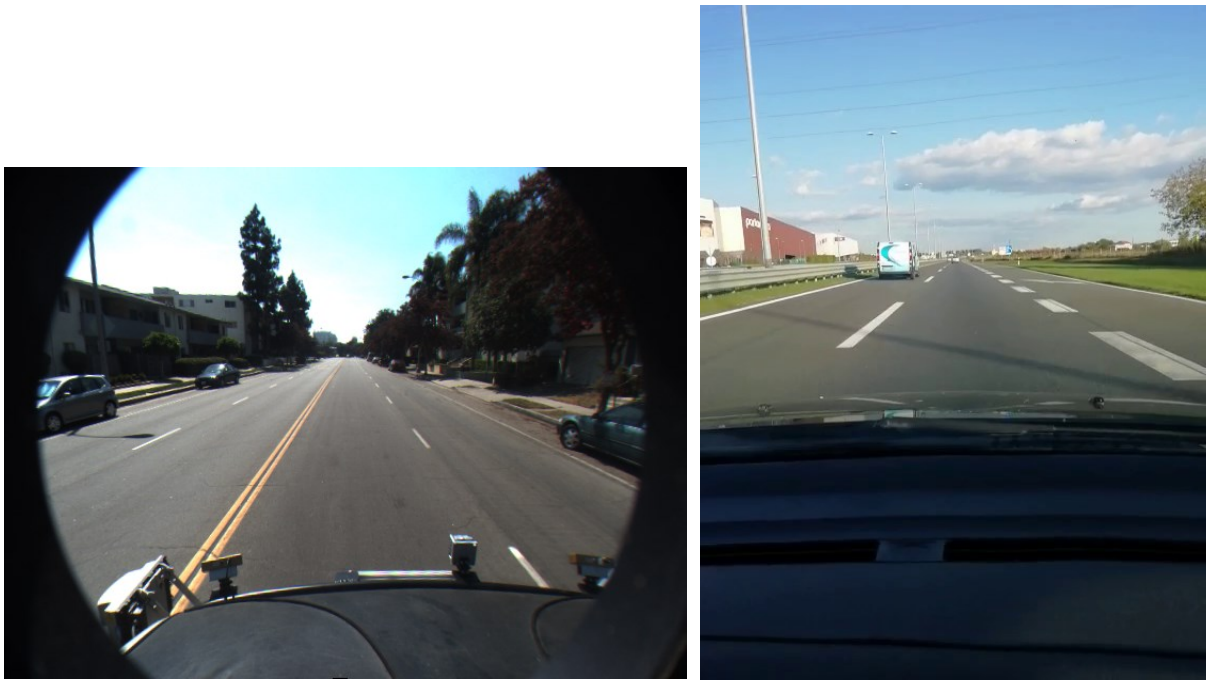
F1 mjera je ponderirani prosjek preciznosti i odziva. Ovaj rezultat uzima u obzir i *FP* i *FN* rezultate. Pogodan je u situacijama kada je greška prouzrokovana *FP* i *FN* vrlo različita. Izračunava se pomoću izraza (4-4) [14].

$$F1 = 2 * \frac{preciznost * odziv}{preciznost + odziv} \tag{4-4}$$

Rješenje za upozoravanje vozača o napuštanju vozne trake kvantitativno je evaluirano koristeći izraze (4-1) do (4-4). Procjena je li je rezultat za neku testnu sliku *TP*, *FP*, *FN* ili *TN* provedena je subjektivno budući da ne postoje stvarne pozicije uzdužnih kolničkih oznaka (engl. *Ground truth*) za navedene slike. Kako bi se utvrdio broj *TP* rezultata za svaku sliku uvedena je mjera od dvije uzdužne kolničke oznake koje se moraju pronaći kako bi se moglo reći da je algoritam uspješno detektirao voznu traku, ali samo u slučaju da uzdužne kolničke oznake zaista postoje na slici. Rezultat se označava kao *TP* u slučaju kada je detektiran pravac koji odgovara uzdužnoj kolničkoj oznaci. U slučaju kada uzdužne kolničke oznake ne postoje na ulaznoj slici, a implementacijom nisu detektirani pravci, rezultat se označava kao *TN*. U slučaju da uzdužne kolničke oznake postoje na ulaznoj slici, a algoritam ne detektira pravce rezultat se označava kao *FN*. Ako na ulaznoj slici postoje uzdužne kolničke oznake, a algoritam detektira pravce koji ne odgovaraju uzdužnim kolničkim oznakama rezultat se označava kao *FP*, ali i kao *FN* s obzirom na to da nije detektirao postojeće uzdužne kolničke oznake. U slučaju kada algoritam detektira samo jedan pravac koji odgovara uzdužnoj kolničkoj oznaci, a drugi pravac ne odgovara uzdužnoj kolničkoj oznaci rezultat se označava kao *TP* jer je detektiran pravac koji odgovara jednoj uzdužnoj kolničkoj oznaci te *FP* i *FN* jer drugi pravac ne odgovara uzdužnoj kolničkoj oznaci. Nadalje, u slučaju kada na ulaznoj slici ne postoje uzdužne kolničke oznake, a algoritam detektira pravce rezultat se označava kao *FP*. Prilikom evaluacije napuštanja vozne trake, algoritam mora uspješno detektirati napuštanje vozne trake ako dolazi do njega kako bi se rezultat označio s *TP*, u slučaju da dolazi do napuštanja vozne trake, a algoritam nije detektirao napuštanje vozne trake rezultat se označava kao *FN*. Ako ne dolazi do napuštanja vozne trake, a algoritam detektira napuštanje vozne trake, rezultat se upisuje kao *FP*.

Evaluacija je provedena na dva skupa podataka. Prvi skup podataka je Kitti [15]. Preuzete slike snimljene su u različitim uvjetima , a rezolucija slika je 640*480 elemenata slike. Prilikom testiranja koncepta rješenja na osobnom računalu slike su učitane u izvornom formatu dok je kod testiranja implementacije rješenja na ADAS Alpha ploči slike su najprije pretvorene u YUV 4:2:0 format korištenjem *ffmpeg* alata. Korišteno je 250 slika gdje postoji 464 jasno vidljivih uzdužnih kolničkih oznaka za vlastitu traku, a 18 slika nema uzdužne kolničke oznake za vlastitu voznu traku. Svaka slika pojedinačno je obrađena koristeći algoritam napisan u C++ programskom jeziku uz korištenje *OpenCV* biblioteke, a obrađene slike se nalaze u prilogu P.4.1. Slike obrađene u algoritmu napisanom za ADAS Alpha ploču nalaze se u prilogu P.4.2. Prilikom evaluacije rješenja zapisivane su vrijednosti za ρ i θ za detektirane pravce na svakoj slici. Evaluacija provedena na osobnom računalu provedena je u više iteracija u kojima su se sve slike evaluirale s različitim vrijednostima praga s obzirom na broj glasova u *Hough*-ovom akumulatoru glasova. Evaluacija je provedena za sljedeće pragove glasova: 40, 35, 30, 25, 20 i 15, a nakon toga se odredio optimalni prag koji je onda korišten u evaluaciji na ADAS Alpha ploči jer evaluacija na njoj iziskuje znatno više vremena.

Vlastiti skup slika korišten se sastoji od 702 slike gdje postoji 1404 jasno vidljive uzdužne kolničke oznake za vlastitu traku koje su spremljene pomoću VLC *player*-a i koje predstavljaju svaki deseti okvir videa snimljenog za potrebe evaluacije algoritma za upozoravanje vozača o napuštanju vozne trake te se nalaze u prilogu P.4.3. Evaluacija nad vlastitim skupom slika provedena je na potpuno isti način kao i nad Kiti skupom slika, a rezultati evaluacije nalaze se u prilozima P.4.4. za evaluaciju koncepta i P.4.5. za evaluaciju rješenja na ADAS Alpha ploči. U svrhu evaluacije algoritma upozoravanje vozača o napuštanju vozne trake snimljen je video zapis vožnje nad kojim je provedena evaluacija. Video snimljen za evaluaciju algoritma za upozoravanje vozača o napuštanju vozne trake snimljen je s kamerom mobitela fiksiranom u okomiti položaj koristeći držač za mobitele. Držač je pozicioniran ispod vjetrobranskog stakla te centriran između dva kotača vozila a nalazi se u prilogu P.4.6. Video zapis traje 9 minuta i 19 sekundi, no s obzirom na to da se tijekom snimanja videa nalaze dijelovi gdje nema prestrojavanja video je podijeljen na 13 manjih dijelova u kojem se vozilo prestrojava. Snimljeni video sadrži ukupno 21 prestrojavanje. Video je snimljen u rezoluciji 480*640 elementa slike a sadrži 28 okvira po sekundi. Prilikom evaluacije je mjereno i vrijeme izvođenja algoritma. Primjeri slika korištenih u evaluaciji rada algoritma prikazani su 4.1.



a) b)

Slika 4.1. Primjer slika korištenih u evaluaciji rada algoritma:

a) Primjer slike iz Kitti skupa

b) Primjer slike iz vlastitog skupa

4.2. Rezultati kvantitativne evaluacije

U tablici 4.1. prikazani su rezultati evaluacije algoritma za detekciju vozne trake provedene nad Kitti skupom slika uz različite vrijednosti praga s obzirom na broj glasova u *Hough*-ovom akumulatoru glasova te evaluacija rješenja algoritma na ADAS Alpha ploči. Evaluacija provedena za prag 40 dala je najbolje rezultate što se tiče *FP* rezultata, a najlošije za *TP* i *FN*. Takvi rezultati su i očekivani s obzirom na to da sve pronađene linije ispod tog praga neće biti detektirane, a akumulator za slike gdje nema uzdužnih kolničkih oznaka generalno ima malo glasova. Smanjujući vrijednost praga s obzirom na broj glasova prilikom pretraživanja *Hough*-ovog akumulatora *FN* rezultati se raspodjeljuju u *TP* rezultate, no ujedno se povećava i broj *FP* rezultata. Nadalje, smanjujući prag za glasove vidljivo je da se broj *TN* rezultata smanjuje jer sada algoritam često detektira i dijelove koji nisu uzdužne kolničke oznake. Uspoređivanjem rezultata implementacije na osobnom računalu te implementacije na ADAS Alpha ploči vidljive su razlike. One se najviše očituju zbog pretvorbe zapisa slike u YUV 4:2:0 format kod testiranja rješenja razvijenog na ADAS Alpha ploči. U ovim koracima dolazi do gubitka informacija o U i V

kromatskoj komponenti koja je nositelj boje. Tablica 4.1. također sadrži mjere točnosti, preciznosti, odziva i *F1* provedene evaluacije nad Kitti skupom slika. Vidljivo je da točnost i odziv rastu sa smanjenjem praga za glasove što je i za očekivati zbog sve većeg broja *TP* rezultata, a manjeg broja *FN* rezultata. S druge strane preciznost opada zbog povećanja *FP* rezultata koji se sve više pojavljuju kada se prag smanjuje. Prema prikazanim rezultatima za evaluaciju algoritma za detekciju uzdužnih kolničkih oznaka na ADAS Alpha ploči odabrana je varijanta s 20 glasova koja pruža optimalne rezultate. Kao i u tablici 4.1. vidljive su razlike u rezultatima uspoređujući postignute rezultate implementacije na osobnom računalu te implementacije na ADAS Alpha ploči.

Tablica 4.1. Rezultati evaluacije algoritma za detekciju vozne trake nad Kitti skupom slika

PRAG	***TP***	***FP***	***FN***	***TN***	**Točnost**	**Preciznost**	**Odziv**	***F1***
Implementacija na PC-u								
40	387	10	77	31	83%	97%	83%	90%
35	407	15	57	30	86%	96%	88%	92%
30	419	26	43	26	86%	94%	90%	92%
25	427	32	37	21	87%	93%	92%	93%
20	432	37	32	19	87%	92%	93%	93%
15	433	43	31	18	86%	91%	93%	92%
ADAS implementacija								
20	434	32	32	24	88%	93%	93%	93%

U tablici 4.2. prikazani su rezultati evaluacije koncepta rješenja algoritma za detekciju vozne trake proveden nad vlastitim skupom slika uz različite vrijednosti praga s obzirom na broj glasova u *Hough*-ovom akumulatoru glasova te evaluacija rješenja na ADAS Alpha ploči. Povećanje broja *FN* te *FP* rezultata u odnosu na rezultate postignute evaluacijom nad Kitti skupom slika je zbog toga što je prilikom testiranja ovog skupa slika povećan mogući opseg vrijednosti za ρ i θ. Opseg je povećan jer prilikom napuštanja vozne trake vrijednosti za ρ i θ pravaca koji predstavljaju uzdužne kolničke oznake poprimaju više različitih vrijednosti odnosno sam pravac kojeg je potrebno detektirati mijenja svoju poziciju i kut u većem opsegu u odnosu na prethodnu evaluaciju. Zbog povećanja opsega, za očekivati je da će doći do većeg broja pogreški prilikom detekcije uzdužnih kolničkih oznaka. Uspoređujući rezultate koji su dobiveni implementacijom na osobnom računalu s rezultatima koji su dobiveni implementacijom na ADAS Alpha ploči vidljive su razlike. Vidljiv je porast *FP* i *FN* rezultata u odnosu na evaluaciju istog skupa na osobnom računalu što je rezultat gubitaka informacije tijekom obrade slike pa se na slikama u nekim situacijama nalazi više

elemenata slike koji nose više glasova, a u nekim slikama manje elemenata slike koji nose manje glasova za pravce. Tablica 4.2. također sadrži mjere točnosti, preciznosti, odziva i *F1* provedene evaluacije nad vlastitim skupom slika. Povećanjem opsega vrijednosti za ρ i θ povećavaju se i greške prilikom detekcije uzdužnih kolničkih oznaka. Vidljivo je da je to povećanje utjecalo na rezultate mjera u odnosu dobivene rezultate s manjim opsegom. Za evaluaciju rješenja algoritma na ADAS Alpha ploči odabrana je varijanta s 15 glasova kao optimalno rješenje.

Tablica 4.2. Rezultati evaluacije algoritma za detekciju vozne trake nad vlastitim skupom slika

PRAG	***TP***	***FP***	***FN***	***TN***	**Točnost**	**Preciznost**	**Odziv**	***F1***
Implementacija na PC-u								
40	959	140	455	0	62%	87%	68%	76%
35	999	171	405	0	63%	85%	71%	78%
30	1053	209	351	0	65%	83%	75%	79%
25	1084	242	320	0	66%	82%	77%	79%
20	1108	256	296	0	67%	81%	79%	80%
15	1121	267	283	0	67%	81%	80%	80%
ADAS implementacija								
15	1085	303	319	0	64%	78%	77%	78%

Tablica 4.3. razmatra uspješnost detekcije napuštanja vozne trake provedene na osobnom računalu. Rezultati su prikazani uz različite vrijednosti praga s obzirom na broj glasova u *Hough*-ovom akumulatoru glasova. Smanjujući vrijednosti praga s obzirom na broj glasova tijekom evaluacije smanjuje se i broj *FN* rezultata što je očekivano jer algoritam s velikim brojem glasova ne pronalazi niti jedan pravac kako bi obavijestio da vozač napušta voznu traku, odnosno pravac nema dovoljan broj glasova da se detektira. S druge strane, smanjujući prag s obzirom na broj glasova u *Hough*-ovom akumulatoru glasova povećava se broj *FP* rezultata. Tablica 4.3. također sadrži mjere točnosti, preciznosti, odziva i *F1*. Vidljivo je da su najbolji rezultati dobiveni postavljajući vrijednost praga s obzirom na broj glasova u *Hough*-ovom akumulatoru glasova na vrijednost 30. Razlog tome je što se prilikom tog praga glasova pojavi ukupno najmanje *FP* i *FN* rezultata, dok svi ostali rezultati daju više ili *FP* ili *FN* rezultata. Također je vidljivo da implementacija ima visok odziv što znači da ako se napuštanje vozne trake uistinu događa, algoritam to jako dobro detektira, no s druge strane, implementacija ima nižu preciznost pa algoritam često javlja lažna upozorenja o napuštanju vozne trake.

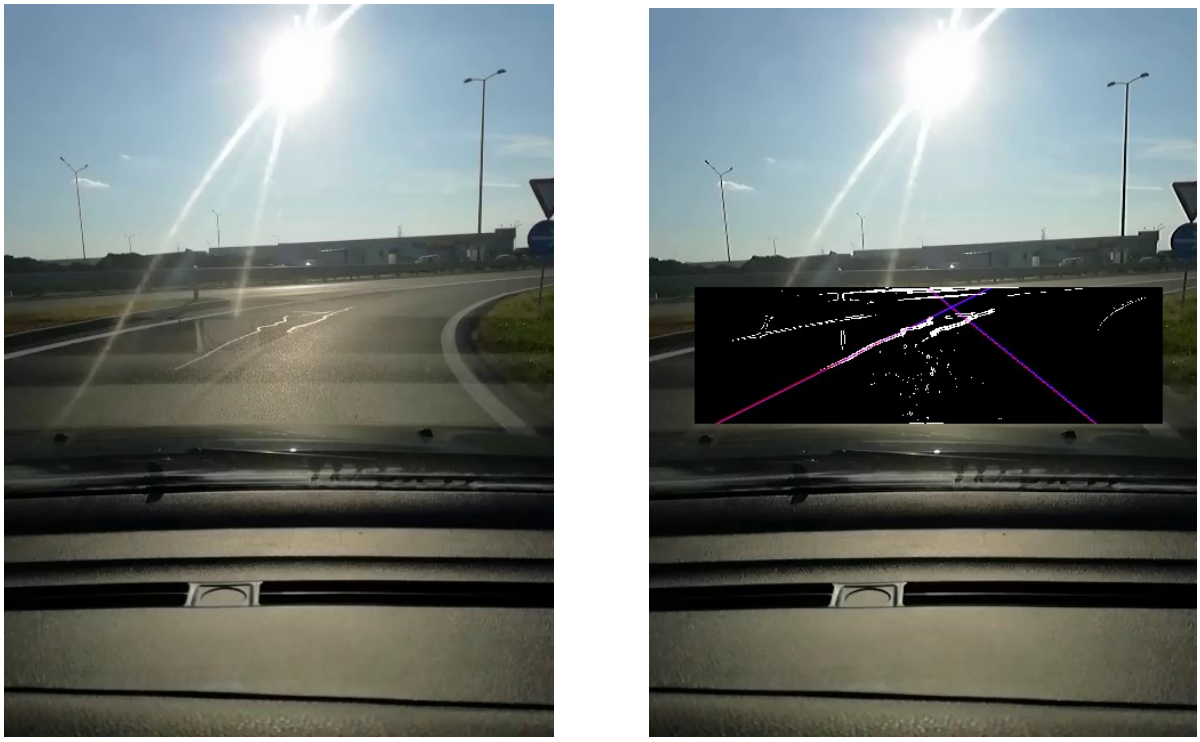
Tablica 4.3. Rezultati evaluacije algoritma za upozoravanje vozača o napuštanju vozne trake

PRAG	***TP***	***FP***	***FN***	***TN***	**Točnost**	**Preciznost**	**Odziv**	***F1***
Implementacija na PC-u								
40	18	9	3	0	60%	67%	86%	75%
35	19	10	2	0	61%	66%	90%	76%
30	20	10	1	0	65%	67%	95%	78%
25	20	13	1	0	59%	61%	95%	74%
20	20	14	1	0	57%	59%	95%	73%
15	21	14	0	0	60%	60%	100%	75%

Obrada ulazne slike prilikom evaluacije algoritma na osobnom računalu provedena na Kitti skupu slika te na vlastitom skupu slika je u rasponu od 170 do 300 milisekundi ovisno o sceni, dok obrada video zapisa snimljenog tijekom vožnje nad kojim je provedena evaluacija algoritma za upozoravanje vozača o napuštanju vozne trake je u rasponu od 190 do 350 milisekundi. Obrada slika na ADAS Alpha ploči je u rasponu od 400 do 600 milisekundi što znači da pokretanje ovog algoritma u stvarnom vremenu nije moguće budući da bi se trebalo moći obraditi 25 okvira u jednoj sekundi.

4.3. Kvalitativna analiza

U ovom potpoglavlju dano je nekoliko primjera detekcije vozne trake. Rezultati su postignuti na ADAS Alpha ploči. Na slici 4.2. prikazane su ulazna slika i izlazna slika nakon obrade algoritma gdje su rozom bojom označene detektirane uzdužne kolničke oznake. Vidljiva je pogrešna detekcija uzdužnih kolničkih oznaka nastala zbog utjecaja infrastrukture kolnika. Budući da kolničke oznake zbog trošenja postaju sive, potrebno je proširiti raspon filtera bijele boje, no zbog tog proširenja kroz filtar prolaze i oštećenja na kolniku koje algoritam potencijalno može detektirati kao uzdužne kolničke oznake. U slučaju da se vrijednosti praga s obzirom na broj glasova u *Hough*-ovom akumulatoru za glasove poveća, algoritam neće detektirati ni jedan pravac što je također greška.



a) b)

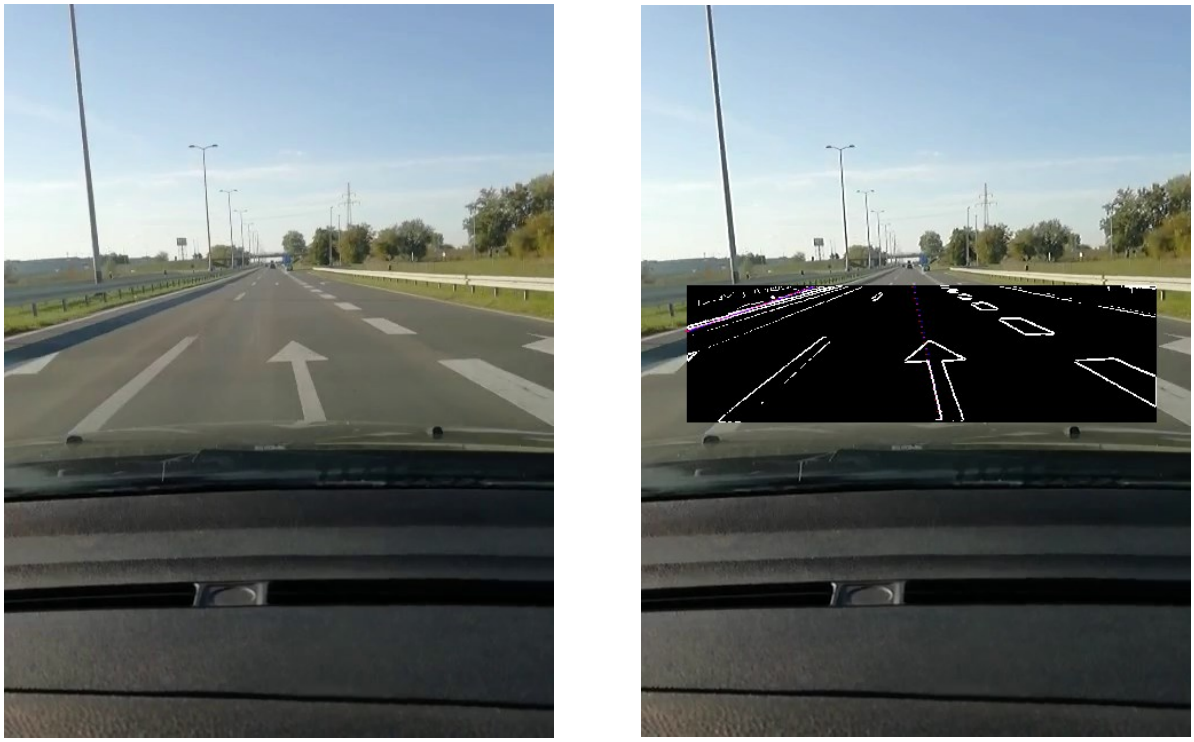
Slika 4.2. Prikaz utjecaja infrastrukture na detekciju uzdužnih kolničkih oznaka:

a) Ulazna slika u algoritam

b) Ulazna slika obrađena algoritmom

Slika 4.3. prikazuje utjecaj drugih kolničkih oznaka na detekciju vozne trake te upozoravanja vozača o napuštanju vozne trake. U ovom slučaju algoritam je prilikom detekcije uzdužnih kolničkih oznaka dobio više glasova za kolničku oznaku nego za desnu uzdužnu kolničku oznaku koja se nalazi pored. Također na slici je prikazan i utjecaj okoline s obzirom na to da je algoritam detektirao ogradu kao uzdužnu kolničku oznaku. Drugi problem bi se mogao riješiti daljnjim ograničavanjem regije od interesa, no u ovome skupu slika to nije bilo moguće jer se na nekim dijelovima vožnje u tom dijelu nalazi uzdužna kolnička oznaka. Osim ovog problema, problem također nastaje prilikom upozoravanja vozača o napuštanju vozne trake s obzirom na to da se ova kolnička oznaka nalazi u djelu slike gdje bi trebala biti vozna traka tijekom napuštanja vozne trake pa ju algoritam detektira kao napuštanje vozne trake. Ovaj problem bi se mogao ispraviti s uspoređivanjem prethodnih okvira slike s trenutnim okvirom slike pa bi se ova greška mogla ispraviti jer algoritam u prethodnim okvirima nije detektirao ovu kolničku oznaku kao liniju. No problem nastaje kada algoritam u prethodnim okvirima također pronađe neke druge objekte, kao

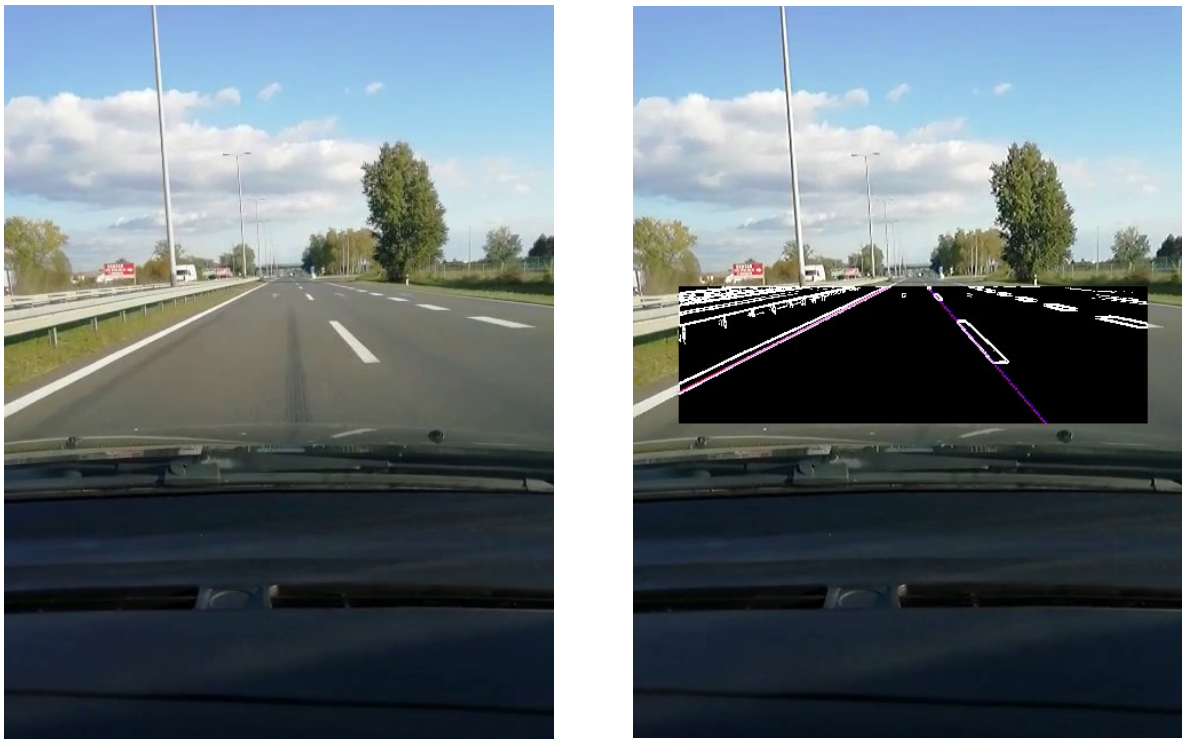
što je na slici prikazana ograda pa bi se upozoravanje vozaču opet izvršilo jer su vozne trake pomjerile nagib za određeni kut koji bi algoritam onda detektirao kao napuštanje vozne trake. Osim toga, uspoređivanje prošlih okvira bi još više usporilo već sporu obradu slika.



a) b)

Slika 4.3. Prikaz utjecaja kolničkih oznaka na detekciju vozne trake:
a) Ulazna slika u algoritam
b) Ulazna slika obrađena algoritmom

Slika 4.4. prikazuje ispravan rad algoritma prilikom pogodnih uvjeta. Uzdužne kolničke oznake su označene na mjestima gdje bi trebale biti te je vozač započeo napuštanje vozne trake te će biti upozoren o napuštanju kada detektirani pravci koji odgovaraju uzdužnim kolničkim oznakama dođe u raspon pokrivenog kuta za napuštanje vozne trake



a) b)

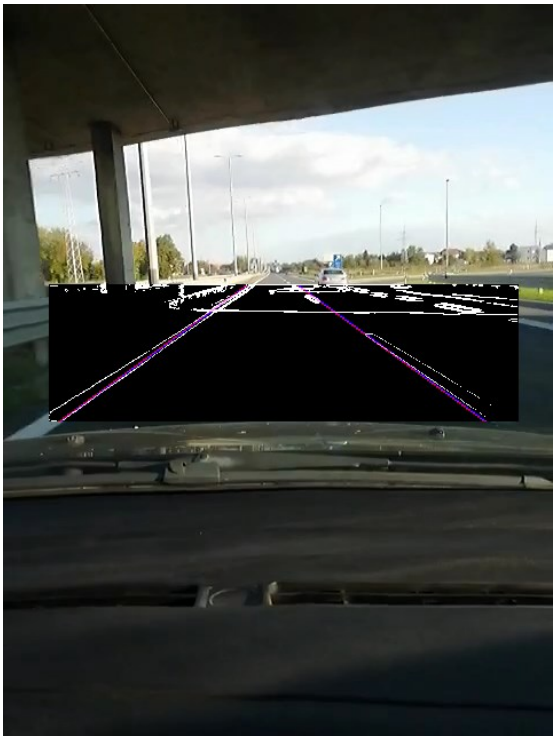
Slika 4.4. Prikaz ispravnog rada algoritma prilikom pogodnih uvjeta:

a) Ulazna slika u algoritam

b) Ulazna slika obrađena algoritmom

Na slici 4.5. prikazan je ispravan rad algoritma prilikom utjecaja sjene. Zbog ovakvih uvjeta gdje sjena prelazi preko uzdužnih kolničkih oznaka potreban je filter boja.

a)



b)

Slika 4.5. Prikaz ispravnog rada algoritma prilikom utjecaja sjene:

a) Ulazna slika u algoritam

b) Ulazna slika obrađena algoritmom

.

5. ZAKLJUČAK

U ovom radu osmišljen je i implementiran algoritam za upozoravanje vozača o napuštanju vozne trake. Cilj ovakvog algoritma je smanjenje nesreća uzrokovanih pogreškama vozača, ometanja tijekom vožnje te umora. Za implementaciju korišteni su alati: programski jezik C++, biblioteka *OpenCV*, programski jezik C, *VisionSDK* programsko okruženje, automotiv kamere, maketa vozila te ADAS Alpha ploča.

U okviru rada najprije je izgrađeno konceptno rješenje na osobnom računalu koristeći programski jezik C++ i biblioteku *OpenCV*. Koncept je izgrađen zbog bržeg eksperimentiranja s potencijalnim načinima rješavanja problema detekcije vozne trake. Nakon koncepta, izgrađeno je predloženo rješenje u *VisionSDK* programskom okruženju koje se sastoji od jednog korisničkog slučaja te dva algoritamska linka od kojih jedan izvršava *Gauss*-ovo filtriranje, filtriranje po boji i *Sobel* detekciju rubova, a drugi *Hough*-ovu transformaciju. Testiranje algoritma obavljeno je na Kitti skupu slika te vlastitom skupu slika. Rezultati evaluacije pokazali su da algoritam za detekciju vozne trake implementiran na ADAS Alpha ploči dostiže točnost od 64%, preciznost 78%, odziv 77% te *F1* 78% nad vlastitim skupom slika. Mjerenja su pokazala kako navedena implementacija obrađuje slike unutar raspona od 400 do 600 milisekundi. Stoga, navedena implementacija ne postiže rad u stvarnom vremenu s obzirom na aplikaciju i primjenu u stvarnim ADAS sustavima. Rad u stvarnom vremenu bi mogao biti moguć putem dodatnog upoznavanja s platformom što bi rezultiralo boljom optimizacijom regije od interesa, boljom optimizacijom izvršavanja algoritma i raspoređivanjem zadataka na različite procesorske jezgre.

LITERATURA

[1] „Road fatality statistics in the EU (infographic) | News | European Parliament“. https://www.europarl.europa.eu/news/en/headlines/society/20190410STO36615/road-fatality-statistics-in-the-eu-infographic (pristupljeno ruj. 24, 2020).

[2] „The 6 Levels of Vehicle Autonomy Explained | Synopsys Automotive“. https://www.synopsys.com/automotive/autonomous-driving-levels.html (pristupljeno ruj. 24, 2020).

[3] „3 types of autonomous vehicle sensors in self-driving cars“. https://www.itransition.com/blog/autonomous-vehicle-sensors (pristupljeno ruj. 24, 2020).

[4] P. S. Rahmdel, D. Shi, i R. Comley, „Lane detection using Fourier-based line detector“, u *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, kol. 2013, str. 1282–1285, doi: 10.1109/MWSCAS.2013.6674889.

[5] Yue Dong, Jintao Xiong, Liangchao Li, i Jianyu Yang, „Robust lane detection and tracking for lane departure warning“, u *2012 International Conference on Computational Problem-Solving (ICCP)*, lis. 2012, str. 461–464, doi: 10.1109/ICCPS.2012.6384266.

[6] M. Aly, „Real time detection of lane markers in urban streets“, u *2008 IEEE Intelligent Vehicles Symposium*, lip. 2008, str. 7–12, doi: 10.1109/IVS.2008.4621152.

[7] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, i Q. Wang, „Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks“, *IEEE Trans. Veh. Technol.*, sv. 69, izd. 1, str. 41–54, sij. 2020, doi: 10.1109/TVT.2019.2949603.

[8] R. Ramadhani, A. S. Rohman, i Y. W. Hadi, „Line Detection Using Arranging Coordinate Point Method“, u *2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, ruj. 2019, str. 338–343, doi: 10.23919/EECSI48112.2019.8977053.

[9] „Hue, Value, Saturation | learn.“ http://learn.leighcotnoir.com/artspeak/elements-color/hue-value-saturation/ (pristupljeno stu. 16, 2020).

[10] S. Lee, „Lines Detection with Hough Transform“, *Medium*, lip. 06, 2020. https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549 (pristupljeno stu. 09, 2020).

[11] „Why Radians?“, *Teaching Calculus*, lis. 12, 2012. https://teachingcalculus.com/2012/10/12/951/ (pristupljeno stu. 16, 2020).

[12] „YUV - VideoLAN Wiki“. https://wiki.videolan.org/YUV (pristupljeno stu. 13, 2020).

[13] „RGB Color Codes Chart“. https://www.rapidtables.com/web/color/RGB_Color.html (pristupljeno stu. 16, 2020).

[14] K. P. Shung, „Accuracy, Precision, Recall or F1?“, *Medium*, tra. 10, 2020. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (pristupljeno stu. 11, 2020).

[15] „The KITTI Vision Benchmark Suite“. http://www.cvlibs.net/datasets/kitti/eval_road.php (pristupljeno stu. 11, 2020).

SAŽETAK

U ovome radu opisan je razvoj algoritma za upozoravanje vozača o napuštanju vozne trake pri čemu su korištene razne metode obrade razvijene na području računalnog vida. Prvo je dan pregled nekoliko postojećih rješenja detekcije vozne trake koja su poslužila kao inspiracija predloženog algoritma u okviru ovog rada. Nakon toga u radu je izrađen koncept algoritma za upozoravanje vozača o napuštanju vozne trake koji je izrađen korištenjem osobnog računala i biblioteke *OpenCV*. Izrađeni koncept poslužio je kao polazna točka za implementaciju algoritma na ADAS Alpha ploču. Na kraju je provedena evaluacija i usporedba koncepta i rješenja algoritma na dva skupa slika. Algoritam za detekciju vozne trake postiže točnost od 64%, preciznost 78%, odziv 77% te *F1* 78%, dok algoritam za upozoravanje vozača o napuštanju vlastite vozne trake nije moguće pokrenuti u stvarnom vremenu.

Ključne riječi : ADAS, TDA2xx, VISION SDK, detekcija vozne trake, obrada slike

ABSTRACT

This master's thesis describes the development of the lane departure warning system using various methods of computer image processing. Firstly, an overview of several existing lane detection solutions that served as the inspiration for the proposed algorithm. After that, the concept of the lane departure warning system was developed using personal computer and OpenCV library. Such a developed concept served as a starting point for the implementation on the ADAS Alpha board. Finnaly, an evaluation and comparison of the concept and solution of the algorithm was performed on two set of images. The lane detection algorithm achieves an accuracy of 64%, an accuracy of 78%, a response of 77% and an F1 of 78%, while the lane departure warning system cannot be started in real time.

Keywords: ADAS, TDA2xx, VISION SDK, lane detection, image processing

ŽIVOTOPIS

Srđan Dragaš rođen je 4.travnja 1996. godine u Somboru. Od 2002. do 2010. godine pohađa Osnovnu školu Šećerana u Šećerani. Godine 2010. upisuje Prvu srednju školu Beli Manastir, koju završava 2014. polaganjem državne mature. Iste godine upisuje stručni studij Informatike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Na istom fakultetu 2017. godine upisuje razlikovnu godinu. Godine 2018. upisuje Diplomski studij računarstva, smjer Programsko inženjerstvo.

_____
Potpis autora

PRILOZI

- P.3.1. - Konceptni kod razvijen na osobnom računalu pomoću *OpenCV* biblioteke
- P.3.2. - Razvijeni korisnički slučaj na ADAS Alpha ploči
- P.4.1. - Obrađene slike iz Kitti skupa slika na osobnom računalu
- P.4.2. - Obrađene slike iz Kitti skupa slika na ADAS Alpha ploči
- P.4.3. - Vlastiti skup slika
- P.4.4. - Obrađene slike iz vlastitog skupa slika na osobnom računalu
- P.4.5. - Obrađene slike iz vlastitog skupa slika na ADAS Alpha ploči
- P.4.6. - Video snimljen za evaluaciju algoritma za upozoravanje vozača o napuštanju vozne trake