# Razvoj rješenja za prepoznavanje svjetlosne prometne signalizacije i implementacija razvijenog rješenja na realnu ADAS platformu

**Kakuk, Antun**

**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:200:509870

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-11-27**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

**Sveučilišni diplomski studij Računarstva**

# RAZVOJ RJEŠENJA ZA PREPOZNAVANJE SVJETLOSNE PROMETNE SIGNALIZACIJE I IMPLEMENTACIJA RAZVIJENOG RJEŠENJA NA REALNU ADAS PLATFORMU

**Diplomski rad**

**Antun Kakuk**

**Osijek, 2020.**

# SADRŽAJ

# 1. UVOD

Naglim porastom računalne moći i smanjenjem troškova izrade mikročipova u zadnjem desetljeću dolazi do naglog razvoja u području računalnog vida i umjetne inteligencije. Kamere na mobilnim telefonima koriste metode računalnog vida i umjetne inteligencije kako bi svaka fotografija ispala što bolje, vrši se detekcija lica u nadzornim sustavima, autonomni inteligentni roboti sortiraju masivna skladišta [1] i patroliraju bolnicama [2], prediktivne sposobnosti umjetne inteligencije koriste se na Wall Street-u i zamjenjuju financijske analitičare [3] pa čak i predviđaju smrt osoba na temelju dugogodišnjih praćenja nalaza s nevjerojatnih 74% preciznosti [4].

Ovakav ubrzan napredak tehnologije nije zaobišao ni automobilsku industriju u kojoj se razvijaju napredni sustavi za pomoć pri vožnji (engl. *Advanced Driver-Assistance Systems - ADAS*). Ovi sustavi koriste metode računalnog vida i strojnog učenja kako bi vozačima olakšale upravljanje vozilom. SAE J3016 standard [5] predlaže šest razina autonomije vozila. Vozila s nultom razinom posjeduju osnovne oblike automatizacije kao što je na primjer ABS sustav (engl. *anti-lock braking system*). Vozila prve razine posjeduju vrlo jednostavne sustave koji pomažu vozaču ostati u traci ili kontrolirati brzinu vozila. Vozila druge razine sadrže sve sustave za pomoć pri upravljanju kao na primjer pomoć pri zaobilaženju, pomoć pri uključivanju na autoput ili silaženju s autoputa. Vozila treće razine su sposobna upravljati vozilom u specifičnim situacijama kao što je recimo upravljanje vozilom prilikom zastoja u koloni, ali je i dalje potrebno da vozač bude spreman preuzeti upravljanje. Vozila četvrte razine imaju mogućnost potpunog samostalnog upravljanja unutar određene domene za koju su dizajnirani, dok vozila pete razine moraju biti sposobna samostalno upravljati u svim uvjetima. Trenutna tehnološka razina je negdje između drugog i trećeg stupnja.

Kako bi napredni sustavi za pomoć u vožnji uspješno upravljali vozilom ili pomagali u vožnji potrebne su informacije o događajima iz neposredne okoline vozila. Ove informacije, putem senzora i računalne obrade, prikupljaju ADAS podsustavi zaduženi za specifične zadatke. Jedan od tih zadataka je i detekcija svjetlosne prometne signalizacije kojom se detektira stanje svjetlosne prometne signalizacije u vožnji kako bi ADAS sustav mogao donositi ispravne odluke u vožnji.

U ovom radu će se opisati izrada rješenja za detekciju svjetlosne prometne signalizacije koja će se implementirati na realnu ugradbenu ADAS platformu. Platforma koja će se koristiti je ALPHA ploča instituta RT-RK [6] dizajnirana u suradnji s tvrtkom Texas Instruments. Ploča koristi vlastiti SDK (engl. *Software Development Kit*) naziva *VisionSDK* [7].

Ostatak rada je strukturiran na sljedeći način. U drugom poglavlju dan je opći pregled i podjela metoda za detekciju svjetlosne prometne signalizacije, a potom je napravljen pregled i usporedba nekolicine novijih radova na temu detekcije svjetlosne prometne signalizacije. U trećem poglavlju opisano je predloženo rješenje za detekciju svjetlosne prometne signalizacije. Četvrto poglavlje bavi se evaluacijom izrađenog rješenja, dok je u petom poglavlju dan uvid u zaključke ovoga rada.

# 2. PROBLEM PREPOZNAVANJA SVJETLOSNE PROMETNE SIGNALIZACIJE U ADAS SUSTAVIMA NA REALNOJ PLATFORMI

Trenutno se mogu razlučiti dva različita pristupa u prepoznavanju svjetlosne prometne signalizacije. Prvi pristup svodi se na komunikaciju o stanju svjetlosne prometne signalizacije između vozila i infrastrukture, tzv. V2I (engl. *vehicle-to-infrastructure*) komunikaciju, dok se drugi pristup oslanja na detekciju stanja svjetlosne prometne signalizacije putem senzora [8]. Budući da se sustavi za pomoć pri upravljanju vozilom trenutno oslanjaju na senzore, u radu će se govoriti samo o drugom pristupu.

## 2.1. Opća podjela i značajke

U današnje vrijeme postoje mnoga rješenja za prepoznavanje svjetlosne prometne signalizacije putem senzora, no samo je mali broj rješenja testiran na realnim ADAS platformama. Rješenja je moguće podijeliti s obzirom na vrstu korištenog senzora, što izravno utječe na vrstu signala koji se obrađuje te s obzirom na konkretan način obrade signala dobivenog iz senzora.

Senzori koji se koriste u sustavima za detekciju svjetlosne prometne signalizacije su:

- Kamera
- Sustav s više kamera [9]
- GPS (engl. *Global Positioning System*) [10]
- LiDAR (engl. *Light Detecting and Ranging*) [11]

Metode koje se koriste za obradu dobivenog signala su:

- Metode računalne obrade slike (engl. *Image processing*)
- Metode strojnog učenja (engl. *Machine learning*)

Kamera je zbog svoje pristupačnosti i robusnosti trenutno najzastupljeniji senzor te se koristi u gotovo svim sustavima za prepoznavanje svjetlosne prometne signalizacije, a posebice u ugradbenim sustavima koji imaju ograničene računalne resurse. Postavljanjem većeg broja kamera i njihovom kalibracijom realizira se sustav s više kamera. Iz sustava s više kamera moguće je dobiti informaciju o dubini svakog pojedinog piksela te se time postavlja temelj za naprednije algoritme koji omogućuju još veću točnost detekcije. Problem sustava s više kamera sa stajališta ugradbenih sustava je što porastom broja kamera raste i složenost algoritama te količina informacija koje je

potrebno obraditi, što dodatno opterećuje ionako ograničene računalne kapacitete ugradbenih sustava. GPS se rjeđe koristi kao glavni senzor budući da nije u stanju detektirati stanje semafora, već služi kao pomoćni senzor koji signalizira kada se vozilo nalazi u blizini svjetlosne signalizacije [8]. LiDAR se, kao i GPS, u pravilu koristi kao pomoćni senzor. Vrlo je skup, što ograničava komercijalnu uporabu te budući da kao izlaz daje trodimenzionalni oblak točaka, potrebni su računalno vrlo zahtjevni algoritmi kako bi se iz oblaka točaka dobila tražena informacija.

Računalni vid interdisciplinarno je područje koje se bavi mogućnostima računala da iz digitalne slike ili videa dobije informaciju više razine, npr. položaj objekta od interesa, dok se iz perspektive inženjerstva metodama računalnog vida putem senzora i računala pokušava rekreirati mogućnosti ljudskog osjetila vida [12]. Svojim opsegom obuhvaća metode računalne obrade slike i metode strojnog učenja. Međutim, iako su obje metode obrađivanja signala u domeni računalnog vida, međusobno su vrlo različite.

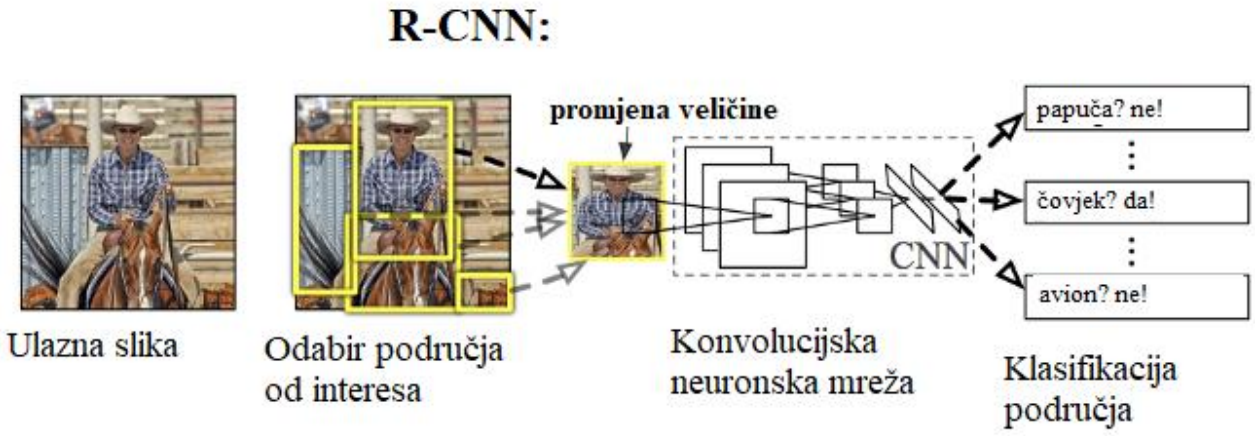
Područje tradicionalne računalne obrade slike koristi jasno definirane algoritme i metode koji programeru pružaju potpuni uvid u njihovo unutrašnje djelovanje, a zbog transparentnosti nazivaju se još i modeli bijele kutije. Ovi su algoritmi u pravilu jednostavni za izvođenje. Nasuprot tome, metode strojnog učenja, od kojih su u domeni računalnog vida najzastupljenije konvolucijske neuronske mreže (engl. *Convolutional Neural Networks, CNN*), najčešće se sastoje od više milijuna parametara koji se (samo)podešavaju tijekom treniranja, stoga je vrlo teško razaznati koji parametar kako utječe na krajnji rezultat pa se ovakvi modeli još nazivaju i modeli crne kutije. Zbog količine parametara s kojima je potrebno raditi ova metoda zahtijeva izrazito mnogo računalnih resursa i enormne skupove podataka. Tradicionalni algoritmi ne oslanjaju se na skupove podataka već na programerovo shvaćanje algoritama i problema koji se rješava, jer je upravo programer taj koji podešava parametre kod tradicionalnih metoda računalne obrade slike [13]. Obje su metode relevantne i koriste se u svrhu prepoznavanja svjetlosne prometne signalizacije. Kratka usporedba metoda dana je u tablici 2.1.

Tablica 2.1. Usporedba metoda obrade slike i strojnog učenja.

| **Obrada slike** | **Strojno učenje** |
|---|---|
| Jednostavnije | Složenije |
| Parametre podešava programer | Samopodešavanje |
| Transparentnost unutrašnjeg rada (model bijele kutije) | Netransparentnost unutrašnjeg rada (model crne kutije) |
| Zahtijeva programerovo poznavanje algoritama i problema koji rješava | Zahtjeva ogromne količine podataka |

## 2.2. Pregled znanstvenih radova

U ovomu poglavlju ukratko je obrađena i uspoređena nekolicina recentnijih znanstvenih radova na ovu temu. U radu [14] predloženo je korištenje R-CNN (engl. *Region-based Convolutional Neural Networks*) algoritma u svrhu detekcije svjetlosne prometne signalizacije. Za razliku od klasičnih konvolucijskih neuronskih mreža koje izvode operacije konvolucije i sažimanja na cijeloj slici te potom vrše klasifikaciju putem potpuno povezanog sloja na kraju, u ovom radu predloženo je da se arhitektura modificira na način da se prvo na slici predlože područja od interesa putem neke od metoda za detekciju područja od interesa kao što je recimo *Edge boxes* metoda [15]. Područje od interesa se potom sažima ili razvlači (engl. *warping*) na predefinirani oblik koji se predaje konvolucijskoj neuronskoj mreži. Konvolucijska neuronska mreža izdvaja značajke te ih šalje stroju s potpornim vektorima (engl. *Support Vector Machine*) [16] koji će naposljetku izvršiti klasifikaciju [17]. Za R-CNN algoritam može se reći da je hibridni jer koristi i računalnu obradu slike i strojno učenje. Vizualni prikaz ovog procesa vidljiv je na slici 2.1. Za razliku od klasičnih konvolucijskih neuronskih mreža u kojima izlazni sloj ne može biti varijabilan, što uzrokoje problem kod slika koje sadrže različiti broj objekata od interesa koje je

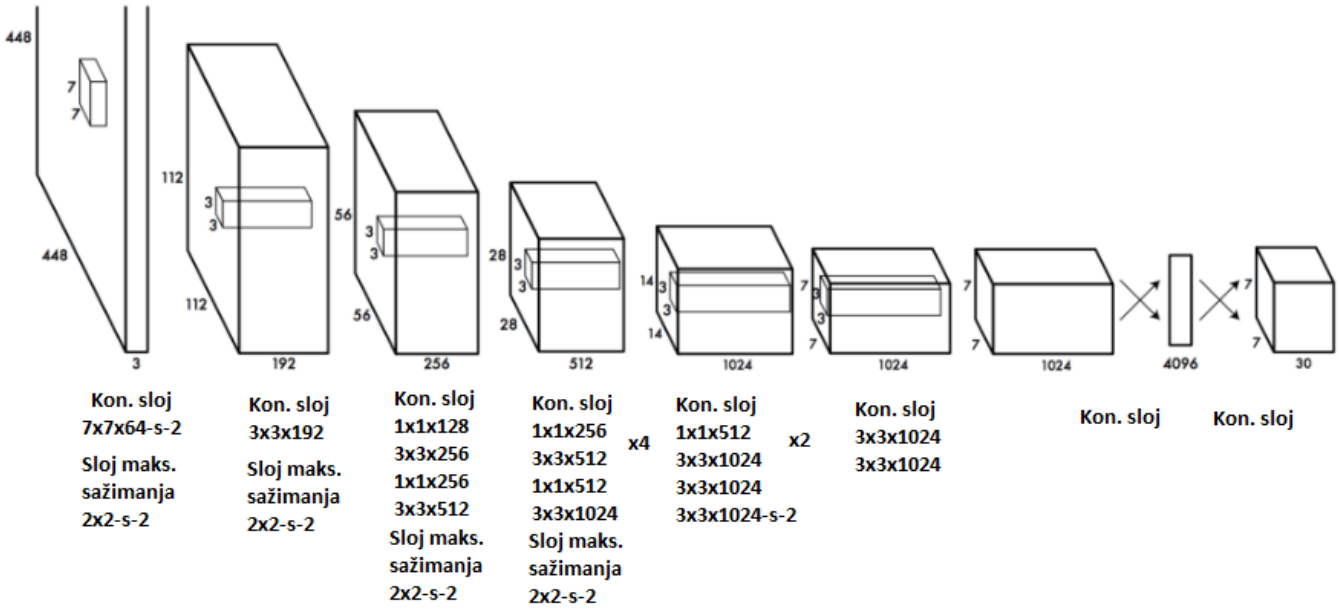



Slika 2.1. Prikaz rada R-CNN.

potrebno detektirati, R-CNN ima sposobnost detektirati varijabilan broj objekata na slikama. Zbog ovih mogućnosti jedan od problema koji se javlja prilikom korištenja R-CNN je broj predloženih područja od interesa koji iznosi 2000 što značajno utječe na performanse, jer je svako područje potrebno analizirati putem konvolucijske neuronske mreže i stroja s potpornim vektorima. Prema [18], R-CNN ostvaruje točnost od 53.5% uz brzinu izvođenja od 7 video okvira po sekundi. Performanse su testirane na *Nvidia TitanX* grafičkoj kartici.

U radu [19], napravljena je evaluaciju različitih verzija YOLO (engl. *You Only Look Once*) modela neuronskih mreža na *LISA Traffic Light* skupu podataka [20], koji sadrži slike svjetlosne prometne signalizacije s pripadajućom temeljnom istinom (engl. *ground truth*). YOLO neuronska mreža kao ulaz uzima 2D sliku koju potom provlači kroz niz konvolucijskih filtera i slojeva maksimalnog sažimanja. Potom, idu dva potpuno povezana sloja, a kao izlaz dobije se tenzor veličine 7 x 7 x 30. Prikaz arhitekture YOLO neuronske mreže nalazi se na slici 2.2. [21]. Na ulazu se slika dijeli na *S* x *S* mrežu ćelija. Prilikom korištenja YOLO algoritma za svaku ćeliju se računa vektor vrijednosti **C**, koji daje postotak vjerojatnosti da ćelija sadrži određeni objekt, i određeni broj graničnih okvira od kojih svaki ima vrijednosti *x*, *y*, *w*, *h* i *c*. Vrijednosti *x* i *y* određuju središte, dok vrijednosti w i h određuju širinu i visinu dobivenog graničnog okvira. Vrijednost *c* daje vjerojatnost da se određena klasa nalazi unutar okvira. Zbog svoje izvedbe YOLO neuronska mreža ponajviše problema ima s detekcijom malih objekata ili nakupina malih objekata, kao što je na primjer jato ptica u letu. Ovo je rezultat ograničavanja mogućih graničnih okvira unutar jedne ćelije.U radu je za evaluaciju detekcije svjetlosne prometne signalizacije dobivena vrijednost 90.49% za YOLO v3.1. algoritam. U radu nije navedeno koje su performanse testa te na kojem

sklopovlju je rješenje testirano, dok se u radu [22] navodi da općeniti YOLO algoritam radi na 45 okvira po sekundi sa srednjom prosječnom preciznošću od 63,4%, a cijeli algoritam se izvodi na *TitanX* grafičkoj kartici.




Slika 2.2 Prikaz arhitekture YOLO mreže.

U radu [23] koriste se klasične metode obrade slike za detekciju svjetlosne prometne signalizacije. Ovaj pristup za identifikaciju područja od interesa koristi sliku transformiranu u HSV (engl. *Hue, Saturation, Value*) format boja iz koje stvara binarnu sliku putem filtracije po boji (engl. *Color thresholding*) [24]. Nakon toga provodi se morfološka analiza slike gdje se nakupine piksela, takozvane *blobove*, ispituje zadovoljavaju li određene geometrijske uvjete. Neki od uvjeta koji se provjeravaju su: je li radijus *bloba* unutar zadanih vrijednosti, je li mjera visine i širine *bloba* približno unutar zadanih vrijednosti, je li omjer površine elemenata slike (engl. *pixel*) koji pripadaju *blobu* i površine koju zatvara kontura oko *bloba* veći od 0.8. Nakon što se morfološkom analizom utvrdi da *blobovi* zadovoljavaju sve uvjete i da su potencijalna svjetlosna prometna signalizacija, detektira se boja. Ovaj rad predlaže novi pristup detekciji boje metodom gustoće boje (engl. *Color density*). Boja s najvećom gustoćom uzima se kao boja *bloba*, uz uvjet da je prešla određenu granicu. Performanse rada ovog algoritma testirane su na *Intel core i7 3.60GHz* procesoru sa *16GB* RAM. Pri tome je ostvarena mogućnost obrade od 60 slika po sekundi uz točnost od 95.29% za vangradsku vožnju i 84.99% za vožnju unutar grada.

U radu [25] predloženo je rješenje koje je u potpunosti unutar domene računalne obrade slike te se odlikuje jednostavnošću izvedbe. Kao ulaz se uzima RGB slika te se pomoću (2-4)

pretvara u normaliziranu RGB sliku. Koristeći novonastalu normaliziranu RGB sliku pomoću operacije filtriranja dolazi do stvaranja binarne slike, čime se izdvajaju kandidatna područja (engl. *Candidate region extraction*). Na binarnu sliku područja kandidata primjenjuje se Sobelov detektor rubova (engl. *Sobel edge detector*) [24] kako bi se detektirali rubovi područja kandidata. Naposljetku, slika rubova područja kandidata prolazi kroz Houghovu transformaciju za kružnice (engl. *Hough circle transform*) [26], čime se utvrđuje je li određeno područje kandidat kružnog oblika te ujedno time i svjetlosna prometna signalizacija. Prednosti su ovakvog načina detekcije jednostavnost izvedbe i mali zahtjevi na računalne resurse. Za evaluaciju rada korištene su vlastite fotografije autora rada slikane kroz vjetrobransko staklo automobila. Slikano je 30 fotografija, a na 26 je algoritam pronašao svjetlosnu prometnu signalizaciju uz performanse od 0.347 sekundi po slici. Sklopovlje na kojemu je rad testiran nije poznato. Nepostojanje mogućnosti detekcije stanja svjetlosne prometne signalizacije jedan je od problema koje ovaj rad nije riješio, a drugi je mali uzorak testnog skupa bez slika nastalih u različitim vremenskim uvjetima.

U radu [27] predloženo rješenje je bazirano na brzoj radijalnoj simetriji (engl. *Fast radial symmetry*) [28] te vremensko-prostornoj perzistenciji (engl. *Temporospatial persistency*), kako bi se poboljšala detekcija svjetlosne prometne signalizacije u noćnim uvjetima ili u ekstremnim vremenskim uvjetima poput kiše ili magle. Na ulaz se dovodi slika u RGB formatu. Potom se RGB format boja transformira u CIE L*a*b* format boja [29]. Kako bi se dodatno pojačala razlika između crvene i zelene boje stvara se novi kanal nazvan RG. Kako bi se izbjegao *blooming* efekt koji ponekad stvara svjetlosna prometna signalizacija, stvara se drugi kanal naziva YB te se potom RG i YB kanal zbrajaju u RGYB kanal. Nakon toga slika se razdvaja u dvije: jednu s vrijednostima RGYB većima od 0 i jednu s manjima, pri čemu se praznine popunjavaju *grayscale 4-connected neighbourhood* metodom [30] kako bi se izbjegli problemi prilikom noćne detekcije svjetlosne prometne signalizacije. Na ovakvim slikama svjetlosna prometna signalizacija prikazana je kao vrlo taman krug u slučaju zelenog semafora ili kao vrlo svijetao krug u slučaju crvenog. Koristeći metodu brze radijalne simetrije u središtu ovih krugova pronalaze se maksimumi odnosno minimumi, ovisno o boji, koji predstavljaju središta krugova. Traži se globalni maksimum/minimum te još četiri lokalna maksimuma/minimuma ukoliko su veći od polovine globalnog maksimuma/minimuma. Korak verifikacije kandidata koristi inherentnu vremensko-prostornu perzistenciju kretanja stvarnih objekata kako bi se izbacili lažni pozitivi (engl. *false positive*), koja kaže da je objekt u pitanju semafor ako se u tri od četiri sličice videa u nizu objekt nalazi unutar radijusa od 20 elemenata slike. Ovaj rad testiran je na vlastitom skupu snimljenom u različitim situacijama, a to su urbana vožnja, noćna urbana vožnja i vožnja po kiši. Sklopovlje na

kojemu je testiran rad je Intel Core 2 Quad 2.83GHz s 4GB RAM. Za urbanu vožnju rješenje je postiglo preciznost 61.22% te odziv od 93.75%. Za noćnu urbanu vožnju i vožnju po kiši ponuđena je samo kvalitativna analiza u kojoj je utvrđeno da algoritam radi zadovoljavajuće.

Uzimajući u obzir gore spomenute radove, iako su se metode strojnog učenja pokazale kao efektivnije prilikom konačnih evaluacija, za ovaj rad odabran je pristup računalne obrade slike zbog svoje jednostavnosti implementacije i relativno male računske zahtjevnosti koju valja uzeti u obzir prilikom razmatranja algoritma za ugradbenu ADAS platformu.

# 3. PRIJEDLOG RJEŠENJA ZA PREPOZNAVANJE SVJETLOSNE PROMETNE SIGNALIZACIJE ZA PRIMJENU U NAPREDNIM ADAS ALGORITMIMA

U ovom poglavlju dat je prikaz i detaljan opis procesa izrade rješenja za detekciju svjetlosne prometne signalizacije. Opisano je sklopovlje za koje je algoritam izrađen i set razvojnih alata, pod nazivom *VisionSDK*, koji omogućuju rad vlastito napisanih algoritama na tom sklopovlju. Nadalje, opisana je izrada koncepta rješenja u programskom jeziku *Python* koje je potom rekreirano na realnoj ADAS platformi s prethodno navedenim setom alata. U posljednjem poglavlju prikazan je proces pokretanja programske podrške razvijenog rješenja na realnoj ADAS platformi.

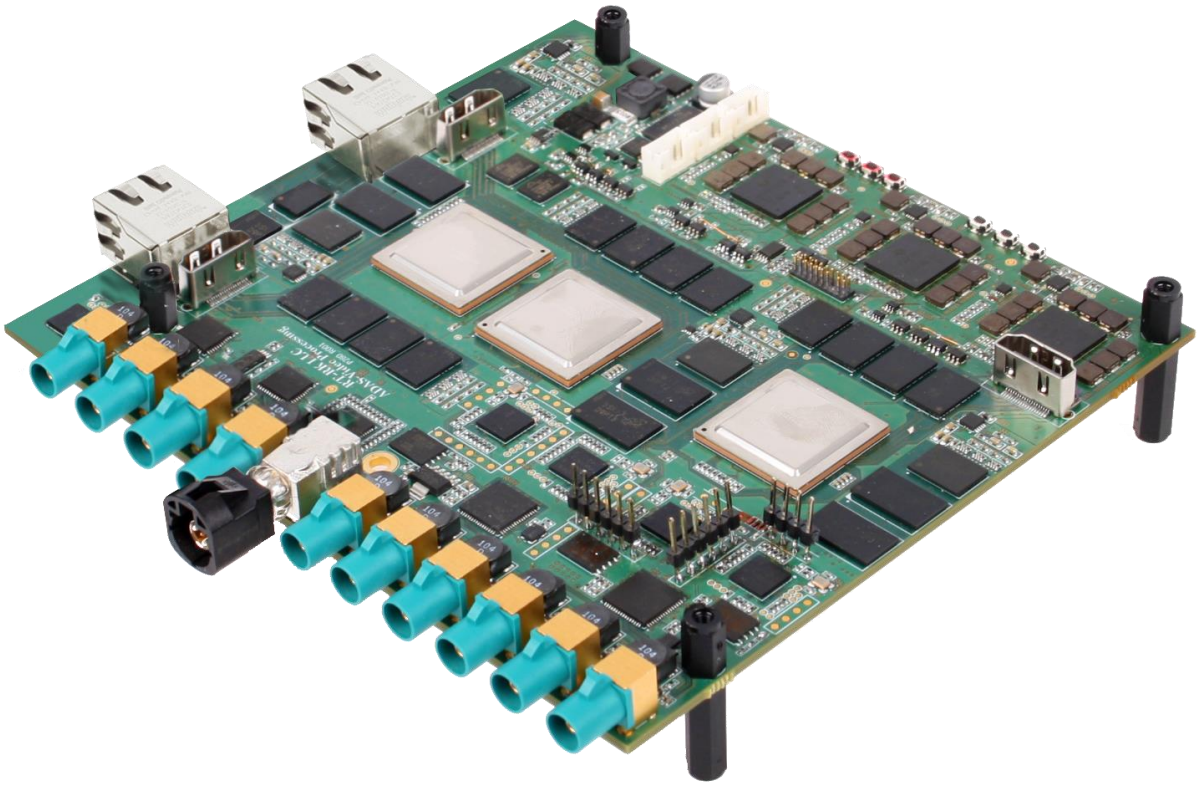
Pri izradi rješenja korištene su razne metode računalne obrade slike: konverzija iz jednog formata boje u drugi, segmentacija slike po boji, detekcija rubova, transformacija, binarna slika i logičke operacije nad slikom. Kombiniranjem ovih metoda određena su područja od interesa koja predstavljaju potencijalnu svjetlosnu prometnu signalizaciju.

## 3.1. ADAS sklopovlje i VisionSDK set razvojnih alata

ADAS sustavi sastoje se od sklopovlja i pripadajućih razvojnih alata namijenjenih razvoju programske podrške. ADAS platforma korištena u izradi ovog rada je ALPHA ploča, prikazana na slici 3.1., dizajnirana od strane Texas Instrumentsa u suradnji s institutom RT-RK. Ploča se sastoji od ulaznog sučelja, izlaznog sučelja i sustava na čipu (engl. *Systems on Chip - SoC*). Od ulaznih i izlaznih sučelja ploča sadrži dva Ethernet konektora, deset ulaza za kamere i konektor za video izlaz. Na ploči se nalaze tri sustava na čipu oznake TDA2X [31]:

1. SC (engl. *Surround Camera*)
2. FFN (engl. *Front view camera near angle stereoscopic view, Front view camera wide angle, Night vision camera*)
3. FUS (engl. *Fusion*)

Svaki od ovih sustava na čipu ima svoju posebnu namjenu. SC SoC koristi se za obradu slike i informacija koje dolaze sa šest kamera zaduženih za opažanje u potpunoj okolini vozila. FFN SoC koristi preostala četiri ulaza za kamere kako bi procesuirao slike i informacije o stanju

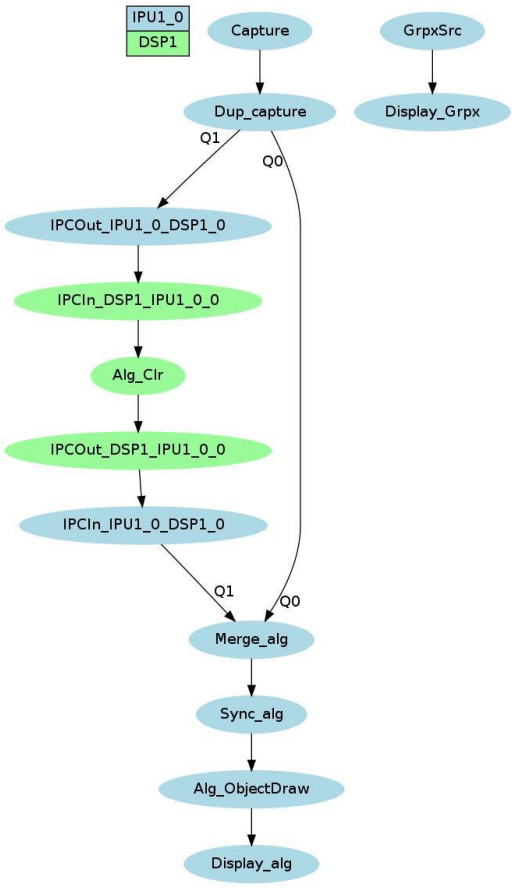


Slika 3.1 Prikaz ALPHA ADAS ploče.

okoline ispred vozila. Ove kamere su u pravilu različitih namjena tako da jedna može biti noćna, druga širokokutna, a treća obična, čime se osigurava robusnost sustava. Svaki od tri sustava na čipu opremljen je s dvije ARM Cortex A15 jezgre frekvencije 1150 MHz, dvije ARM Cortex M4 jezgre frekvencije 200 MHz, dvije DSP (engl. *Digital Signal Processor*) jezgre frekvencije 750 MHz, 1.5 GB DDR3 RAM te raznim akceleratorima. Također svaki SoC posjeduje i vlastiti set ulazno-izlaznih sučelja od kojih su za ovaj rad važni: microSD čitač, HDMI izlaz, JTAG konektor i UART.

Kako bi se iskoristio potencijal navedenog sklopovlja programerima je potrebno dati set razvojnih alata koji omogućuju kompilaciju programskog koda za navedeno sklopovlje. Set razvojnih alata za ALPHA ploču naziva se VisionSDK. VisionSDK omogućuje korisniku stvaranje korisničkih slučajeva (engl. *use-case*) putem kojih korisnik definira tokove podataka. *Use-caseovi* koriste takozvani „Karike i lanci“ (engl. „*Links and Chains“*) okvir (engl. *Framework)* koji se sastoji od više karika (engl. *link*) povezanih u jednu cjelinu. Svaki *link* ima određenu zadaću koju izvršava, na primjer konverzija boja ili učitavanje toka podataka s mreže, a više *linkova* ulančava se u svrhu izvedbe naprednih algoritama tvoreći *use-case*. *Linkove* se može

rasporediti po različitim procesorima, no to mora uraditi sam korisnik jer ne postoji ugrađeni mehanizam automatske raspodjele opterećenja. Postoji nekolicina ugrađenih *linkova* koje je moguće koristiti za učitavanje ili slanje podataka preko mreže, za promjenu veličine slike, za prikaz slike preko HDMI sučelja, za snimanje slike kamerom i drugi. Postojeći *linkovi* pokrivaju samo osnovne radnje, a za kompleksnije algoritme korisniku je dana mogućnost izrade vlastitih *linkova*. Svaki *link* posjeduje ulaz i izlaz, a ukoliko su dva linka povezana potrebno je osigurati da međuspremnik (engl. *buffer*) izlaza prvog i međuspremnik ulaza drugoga budu istovjetne veličine. *Link* može posjedovati više ulaza od više različitih *linkova*. VisionSDK dolazi zajedno sa svim potrebnim bibliotekama i alatima za izgradnju aplikacije. Na slici 3.2. dan je prikaz jednog *use-casea* sastavljenog od više povezanih *linkova*.




Slika 3.2. Prikaz VisionSDK generiranog *use-case* grafa.

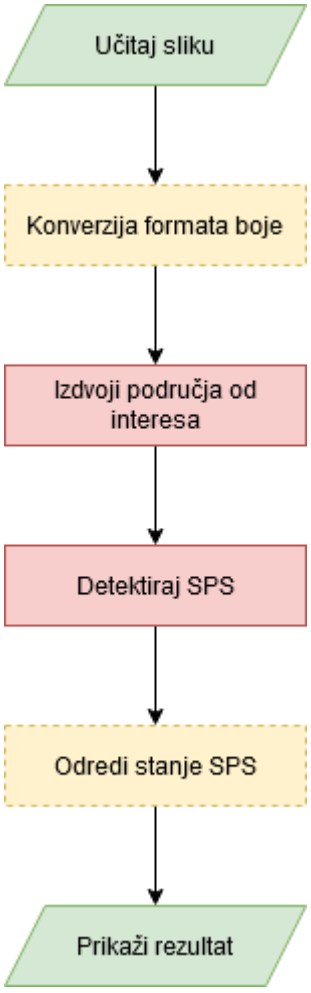
## 3.2. Koncept rješenja za detekciju svjetlosne prometne signalizacije

Razvoj programske podrške za ugradbene računalne sustave, kao i za sve ostale računalne sustave, iterativan je postupak, no ono što je specifično za ugradbene računalne sustave jest da svaka, pa i najmanja izmjena, zahtijeva visok vremenski trošak jer se sastoji od ponovne kompilacije koda, prebacivanja izvršne datoteke na ugradbenu platformu i pokretanje aplikacije. Ovaj proces na ALPHA ploči može trajati od tri minute za minorne izmjene u izvornom kodu kao što je izmjena nekoliko vrijednosti jednog *linka*, pa i do desetak minuta ako se mijenja više različitih *linkova* ili izmjenjuje osnovna konfiguracija. U fazi izrade koncepta rješenja ovaj problem je najizraženiji jer su izmjene izvornog koda učestale zbog isprobavanja više različitih pristupa rješenju te jer ponekad trajanje izmjene u izvornom kodu traje nekoliko sekundi, nakon čega slijedi nekoliko minuta čekanja. Kako bi se ubrzao proces izrade koncepta rješenja i kako bi se što prije došlo do nekakvog prototipa, koncept rješenja u ovom radu izrađen je na osobnom računalu koristeći *Python* programski jezik čime je vrijeme jedne iteracije znatno smanjeno jer je izvođenje koda gotovo trenutno.

Za još brži razvoj prototipa korištena je *OpenCV* biblioteka otvorenog koda. *OpenCV* biblioteka sastoji se od preko 2500 algoritama za računalni vid i strojno učenje [32]. Korištenjem *OpenCV* biblioteke značajno se ubrzava dolazak do konačnog koncepta rješenja jer sadrži sve potrebne funkcije i algoritme za obradu slike, ali potrebno je imati na umu da se većina algoritama sadržanih u *OpenCVu* ne nalazi *out-of-the-box* na ALPHA ploči te će ih biti potrebno samostalno izraditi unutar „*Links and Chains*“ okvira.

Proučavajući radove predstavljene u prethodnom poglavlju utvrđeni su općeniti koraci koje svi algoritmi prepoznavanja svjetlosne prometne signalizacije izvode. Na početku se učitava slika s kojom će se raditi, a koja može biti s kamere ili kao datoteka te može doći u više različitih formata. Ukoliko je potrebno konvertirati sliku u format boja pogodan za određivanje područja od interesa to se izvodi u drugom koraku. Treći korak algoritma određuje područja od interesa nekom od metoda računalne obrade slike, kao što su izrezivanje, segmentacija, detekcija rubova ili transformacije. Ovaj korak je u pravilu jednostavan i računalno nezahtjevan, a rezultira izbacivanjem svih nebitnih informacija kako bi se idući korak koji se odnosi na detekciju svjetlosne prometne signalizacije, a koji je računalno zahtjevniji, izvodio na značajno manje podataka te se tako uštedilo na računalnim resursima i smanjio broj lažno pozitivnih (engl. *false positive*) detekcija. Korak detekcije svjetlosne prometne signalizacije prema postavljenom kriteriju određuje što jest, a što nije svjetlosna prometna signalizacija. Ovi kriteriji sežu od vrlo

jednostavnih morfoloških kriterija pa sve do kompleksnih algoritama. Nadalje, slijedi detekcija stanja svjetlosne prometne signalizacije kojom se dobiva informacija o trenutnom stanju svjetlosne prometne signalizacije. Naposljetku, rezultat obrade prikazuje se na ekranu ili šalje dalje višim upravljačkim algoritmima. Na slici 3.3. moguće je vidjeti grafički prikaz toka rješenja za detekciju svjetlosne prometne signalizacije. Zeleno su obojani ulazi i izlazi, žuto su opcionalne funkcije, a crveno osnovne.




Slika 3.3. Grafički prikaz općenitog algoritma za detekciju SPS (Svjetlosna Prometna Signalizacija).

Koristeći prethodno navedene spoznaje izrađen je prvi koncept rješenja za detekciju svjetlosne prometne signalizacije. Inicijalna slika učitava se korištenjem *OpenCV* funkcije *imread()*, dok se rezultat prikazuje korištenjem funkcije *imshow()*. Za izdvajanje područja od interesa prvo se rabi metoda izrezivanja jer se svjetlosna prometna signalizacija u pravilu nalazi u gornjoj polovici slike, a potom metoda segmentacije slike po boji. U tu svrhu koristi se HSV format

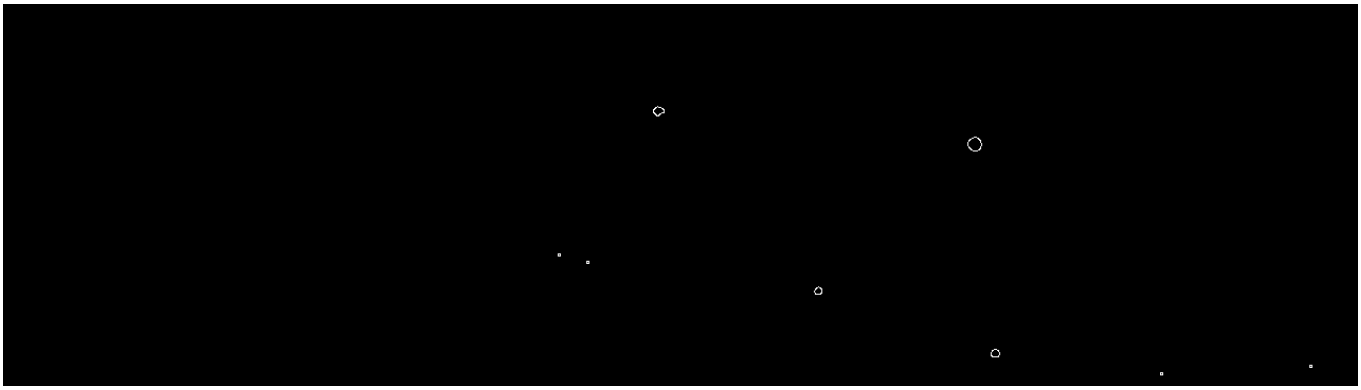
boja [33]. Ovaj format sastoji se od tri komponente: nijanse boje H, zasićenja boje S i svjetline boje V, a posebno je pogodan za segmentaciju po nijansi boje. Budući da je svjetlosna prometna signalizacija u crvenoj i zelenoj nijansi te izrazitog zasićenja i svjetline, parametri po kojima će se segmentirati postavljeni su prema jednadžbi 3-1 gdje je $z$ vrijednost elementa binarne slike s koordinatama $x$ i $y$, a H, S i V kanali su elementa originalne slike na koordinatama $x$ i $y$.

$$z_{(x,y)} = \begin{cases} 255, & if\,(0 < H_{(x,y)} < 25\ and\ 100 < S_{(x,y)} < 255\ and\ 100 < V_{(x,y)} < 255) \\ 255, & if\,(50 < H_{(x,y)} < 80\ and\ 100 < S_{(x,y)} < 255\ and\ 100 < V_{(x,y)} < 255) \\ 0, & u\ suprotnom \end{cases} \quad (3\text{-}1)$$

Segmentacija po boji daje binarnu sliku s vrijednostima 255 ako se na tom polju nalazi element slike s traženim vrijednostima i 0 ako element slike nema tražene vrijednosti, drugi naziv za ovakvu sliku je maska. Ovako segmentirana binarna slika na područjima gdje se nalazi svjetlosna prometna signalizacija ima nakupine visokih vrijednosti (bijela područja) što je vidljivo na slici 3.4.


Slika 3.4. Prikaz originalne slike (gore) i slike nakon segmentacije po boji (dolje).

Zahvaljujući činjenici da je svjetlosna prometna signalizacija kružnog oblika, dio zadužen za detekciju svjetlosne prometne signalizacije iskorištava ovu informaciju i koristi *Houghovu* transformaciju za kružnice. Također postoji svjetlosna prometna signalizacija streličastog oblika, no u ovom radu se ne bavi ovom problematikom već se rad fokusira isključivo na detekciju svjetlosne prometne signalizacije kružnog oblika. Ako se *Houghova* transformacija provede na slici dobivenoj segmentacijom po boji doći će do detekcije više krugova unutar jedne nakupine visokih vrijednosti. Kako bi se ovo spriječilo, na slici segmentiranoj po boji provodi se postupak detekcije rubova *Canny* detektorom rubova [34], pa umjesto cijele površine kruga ostane samo vanjska kružnica prikazana na slici 3.5.



Slika 3.5. Prikaz rezultata *Canny* detekcije rubova na slici segmentiranoj po boji.

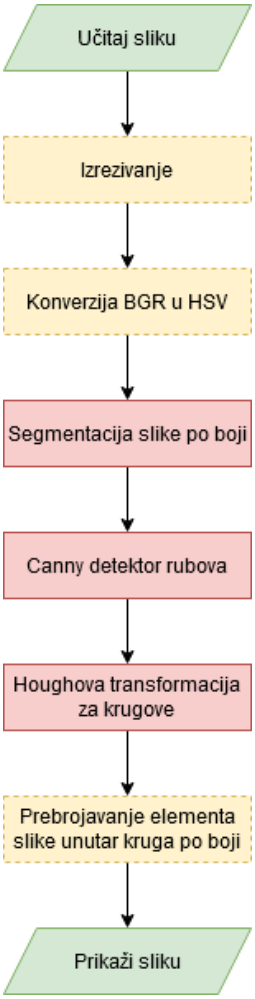
Ovako obrađena slika dodatno smanjuje broj elemenata slike koji se obrađuju *Houghovom* transformacijom te ju čine robusnijom. *Houghova* transformacija za kružnice omogućuje pronalazak krugova na slici. Parametri koje je moguće podešavati su minimalna udaljenost između dva kruga, minimalan i maksimalan radijus kruga te granična vrijednost (engl. *threshold*) detekcije kruga u akumulatoru. Analizom fotografija svjetlosne prometne signalizacije ustanovljeno je da su pojedinačne instance svjetlosne prometne signalizacije razmaknute barem 30 elemenata slike, pri čemu su instance veličine kruga radijusa od 2 do 15 elemenata slike. Ako je granična vrijednost premala, detektirat će se mnogo lažno pozitivnih krugova, dok za prevelike granične vrijednosti stvarni krugovi neće biti detektirani. Eksperimentalno je utvrđeno da je optimalno postaviti graničnu vrijednost za detekciju kruga unutar akumulatora na 7. Prebrojavajući elemente slike određene boje unutar pronađenog kruga određuje se stanje detektirane svjetlosne prometne signalizacije. Na originalno učitanoj slici iscrtava se crveni ili zeleni krug oko detektirane

svjetlosne prometne signalizacije u ovisnosti o detektiranom stanju. Konačna slika prikazuje se na zaslonu, a rezultat se može vidjeti na slici 3.6.


Slika 3.6. Prikaz izlazne slike iz algoritma za detekciju svjetlosne prometne signalizacije.

## 3.3. Razvoj i implementacija rješenja na realnu ADAS platformu

Razvojem koncepta rješenja, svojevrsnog prototipa, dobiven je nacrt prema kojemu je razvijeno rješenje za ALPHA ploču u sklopu „*Links and Chains*“ okvira. Na slici 3.7. prikazan je graf toka za prvi koncept rješenja. U ovom potpoglavlju opisana je izrada rješenja koje će se pokretati na ALPHA ploči. Za izradu rješenja korišteno je nekoliko *linkova* koji dolaze s *VisionSDK* okvirom, dok su ostali samostalno izrađeni.




Slika 3.7. Graf toka prvog koncepta rješenja.

Prema konceptu rješenja prvi *link* koji se koristi je *link* za učitavanje slike. Budući da *VisionSDK* ima već gotov *link* za učitavanje slike on je iskorišten za potrebe ovog rada, no već ovdje nailazi se na prvu razliku. *Link* za učitavanje slike s mreže ili kamere kao ulazni format

prima samo YUV(4:2:0) format [35], stoga je potrebno provesti konverziju iz YUV(4:2:0) u HSV. Međutim, s obzirom na to da ne postoji izravna konverzija iz YUV(4:2:0) u HSV, slika je prvo pretvorena u RGB zapis te potom iz RGB zapisa konvertirana u HSV. Također, valja spomenuti kako YUV(4:2:0) kompresira U i V kromatske komponente slike, takozvano kromatsko poduzorkovanje boja. Ova kompresija smanjuje količinu informacije o boji unutar slike za 75%, stoga se može očekivati da će rezultati segmentacije po boji biti nešto lošiji.

Za drugi *link* korišten je ugrađeni VPE *link* pomoću kojega se može izrezati, smanjiti i uvećati sliku ili promijeniti format boja na slici, no samo u YUV(4:2:2) ili YUV(4:4:4). Ovaj je *link* prvenstveno korišten da bi se izrezala gornja polovica slike u kojoj se nalazi svjetlosna prometna signalizacija, ali iskorištena je i mogućnost promjene formata boja pa je format promijenjen iz YUV(4:2:0) u YUV(4:2:2) koji je mnogo jednostavniji za uporabu.

YUV(4:2:2) format boja pogodan je za kompresiju i prijenos slike, no za segmentaciju po nijansi boje potrebno je koristiti HSV format boja. YUV(4:2:2) format boja sažet je tako da na jednu U i V komponentu dolaze dvije Y komponente, a zapis komponenti u memoriji je $Y_1U_1Y_2V_1Y_3U_3Y_4V_3$…, pri čemu $Y_1$, $U_1$ i $V_1$ tvore jedan element slike; $Y_2$, $U_1$ i $V_1$ tvore drugi element slike; a $Y_3$, $U_3$ i $V_3$ treći element slike. Y, U i V komponente elemenata slike su u memoriji računala prikazane kao 8-bitni cijeli broj. S obzirom na to da ne postoji izravna konverzija iz YUV(4:2:2) u HSV format boja, za međukorak odabrana je konverzija u RGB format boja kao standardni prikaz boja, a potom iz RGB formata boja u HSV. RGB prikaz standardni je prikaz slike gdje svaka komponenta nosi informaciju o količini određene boje: R nosi informaciju o crvenoj, G o zelenoj, a B o plavoj boji. R, G i B komponente elemenata slike u memoriji računala prikazane su kao 8-bitni cijeli broj. Za razliku od YUV(4:2:2) prikaza, u ovom prikazu boja svaki element slike ima vlastitu R, G i B komponentu. Transformacija u međukorak odvija se prema jednadžbi (3-2) [36] gdje su R, G i B komponente RGB elementa slike, a Y, U i V komponente YUV(4:2:2) elementa slike.

$$\begin{aligned} R &= Y + 1.403(U - 128) \\ G &= Y - 0.714(U - 128) - 0.344(V - 128) \\ B &= Y + 1.773(V - 128) \end{aligned} \tag{3-2}$$

Konverzija iz RGB formata boja HSV format boja nešto je složenija, što se može vidjeti u jednadžbi (3-3) u kojoj su R, G i B komponente elemenata slike RGB formata slike, a H, S i V komponente elemenata slike HSV formata slike.

$$
\begin{aligned}
V &= max(R,G,B) \\
S &= \begin{cases} \dfrac{V - min(R,G,B)}{V}, & if(V \neq 0) \\ 0, & if(V = 0) \end{cases} \\
H &= \begin{cases} \dfrac{60(G-B)}{V - min(R,G,B)}, & if(V = R) \\ \dfrac{120 + 60(B-R)}{V - min(R,G,B)}, & if(V = G) \\ \dfrac{240 + 60(R-G)}{V - min(R,G,B)}, & if(V = B) \end{cases}
\end{aligned}
\tag{3-3}
$$

Budući da se, koristeći jednadžbu (3-3), za H komponentu dobiju vrijednosti od 0 do 360, kako bi se mogle zapisati kao 8-bitni cijeli broj kojemu je raspon od 0 do 255, H komponenta dijeli se s 2, dakle može poprimiti vrijednost u rasponu od 0 do 180, što zapravo predstavlja još jedan gubitak informacije o nijansi boje. Budući da ALPHA ploča nije u mogućnosti prikazati HSV format, koristeći jednadžbe (3-2) i (3-3) napisane su funkcije koje za ulaz primaju vektor Y, U i V komponenti elemenata slike, a za izlaz daju vektor H, S i V komponenti elemenata slike. Ove funkcije korištene su prilikom segmentacije kako bi se odredilo što je potrebno segmentirati, ali se ulazna slika ne sprema u HSV format, već ostaje u YUV(4:2:2), zaobilazeći potrebu za ponovnom konverzijom natrag u YUV(4:2:2) format.

*Link* za segmentaciju po boji, nazvan *ColorFilter*, radi vrlo slično kao što je prikazano u konceptu rješenja koristeći iste granice za segmentaciju prikazane u formuli (3-1), ali s nešto drugačijim izlazom. Prolaskom po cijeloj slici u YUV(4:2:2) formatu uzimaju se Y, U i V komponente svakog pojedinačnog elementa slike te se predaju funkciji za konverziju iz YUV(4:2:2) u HSV format boja. Ova funkcija vrati konvertirane pojedinačne H, S i V komponente koje se potom koriste za segmentaciju po boji. No za razliku od koncepta rješenja ovdje se ne kreira binarna slika već se vrijednosti Y komponente originalne YUV slike postavljaju na 255 ako je element slike unutar zadanih graničnih vrijednosti te na 0 ako nije. Ovo je moguće zbog toga što je Y komponenta tzv. *luma* komponenta koja nosi informaciju o količini svjetla pa je prikaz izlaza identičan binarnoj slici samo što su još uvijek prisutne U i V komponente kako bi se rezultat mogao prikazati na zaslonu.

Za detekciju svjetlosne prometne signalizacije u konceptu korišten je *Canny* detektor rubova. Zbog visoke računalne zahtjevnosti pri izradi konačnog rješenja *Canny* detektor rubova zamijenjen je *Sobel* detektorom rubova [37] bez značajnog gubitka u kvaliteti detekcije svjetlosne prometne signalizacije. *Sobel* detektor rubova metoda je detekcije rubova zasnovana na gradijentu svjetline. Metode zasnovane na gradijentu svjetline imaju vrlo dobre mogućnosti raspoznavanja

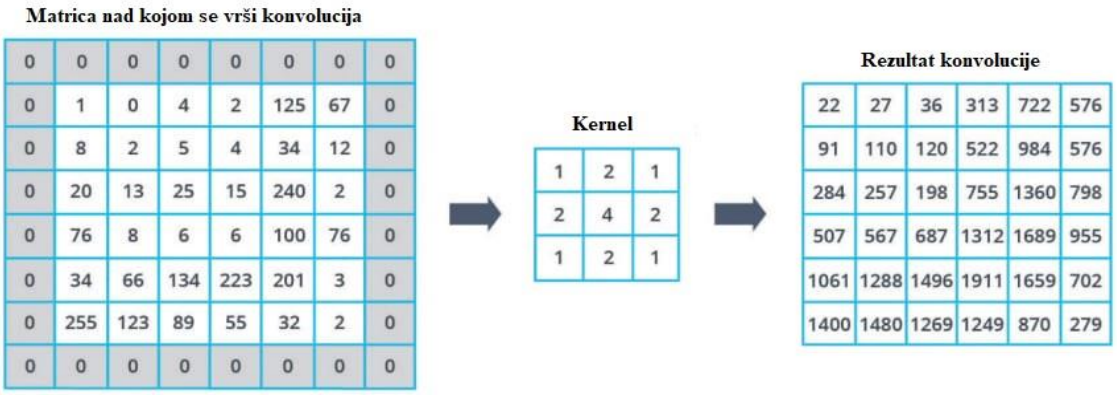
rubova objekata budući da dva objekta u pravilu imaju naglašenu razliku u svjetlini. *Sobel* detektor rubova koristi dvije matrice i metodu konvolucije za određivanje gradijenta svjetline u horizontalnom i vertikalnom smjeru kao što je prikazano u jednadžbama (3-4) i (3-5). Kao rezultat ovog postupka dobiju se dvije matrice: $\mathbf{G_x}$ i $\mathbf{G_y}$, koje korištenjem jednadžbe (3-6) rezultiraju **G** matricom koja je prikaz rubova. Svi rubovi iznad granične vrijednosti smatraju se jakim rubovima. Eksperimentalno je utvrđeno da granična vrijednost od 230 daje zadovoljavajuće rezultate. Matrica **A** je matrica slike na kojoj se izvodi detekcija rubova, a * operator je operator konvolucije.

$$\boldsymbol{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \boldsymbol{A} \tag{3-4}$$

$$\boldsymbol{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \boldsymbol{A} \tag{3-5}$$

$$\boldsymbol{G} = \sqrt{\boldsymbol{G_x}^2 + \boldsymbol{G_y}^2} \tag{3-6}$$

Konvolucija je matematička operacija između dvije matrice korištena u diskretnim sustavima [38]2. *Kernel* matrica sadrži težine i ima dimenzije *N*x*N*. Najčešće veličine *kernela* su 3x3 ili 5x5. Konvolucija se izvodi tako da se središte *kernel* matrice postavi iznad početnog elementa matrice nad kojom se konvolucija izvodi. Elementi koji se poklapaju pomnože se i potom se svi međusobno zbroje i spremaju u novu matricu na isto mjesto, a zatim se *kernel* pomiče za određeni broj koraka (engl. *stride*) te se operacija ponavlja dok se ne dođe do kraja matrice nad kojom se izvodi konvolucija. Ako se *kernel* matrica veličine 3x3 postavi u prvi element vidljivo je kako se prvi redak i prvi stupac *kernel* matrice ne poklapaju ni s čim. Kako bi se doskočilo ovom problemu moguće je započeti konvoluciju u prvom elementu u kojem se *kernel* preklapa s matricom nad kojom se izvodi konvolucija, u ovom slučaju element u drugom redu i drugom stupcu, ili se uvodi *padding*. *Padding* podrazumijeva proširenje dimenzija matrice za (*N*-1)/2 u visinu i u širinu sa svake strane te ispunjavanje novostvorenih elemenata određenim vrijednostima, najčešće nulama. Za *Sobel* detektor rubova matrica nad kojom se izvodi konvolucija ulazna je slika načinjena od Y komponenti ulazne slike. *Kernel* matrica veličine je 3x3, a *stride* je 1. Zbog potrebe za ponovnom alokacijom memorije i kopiranjem sadržaja u slučaju *paddinga*, on nije korišten, već se s konvolucijom započinje od elementa u drugom stupcu i drugom retku. Prikaz konvolucije vidljiv je na slici 3.8. gdje sivi obrub predstavlja *padding*.



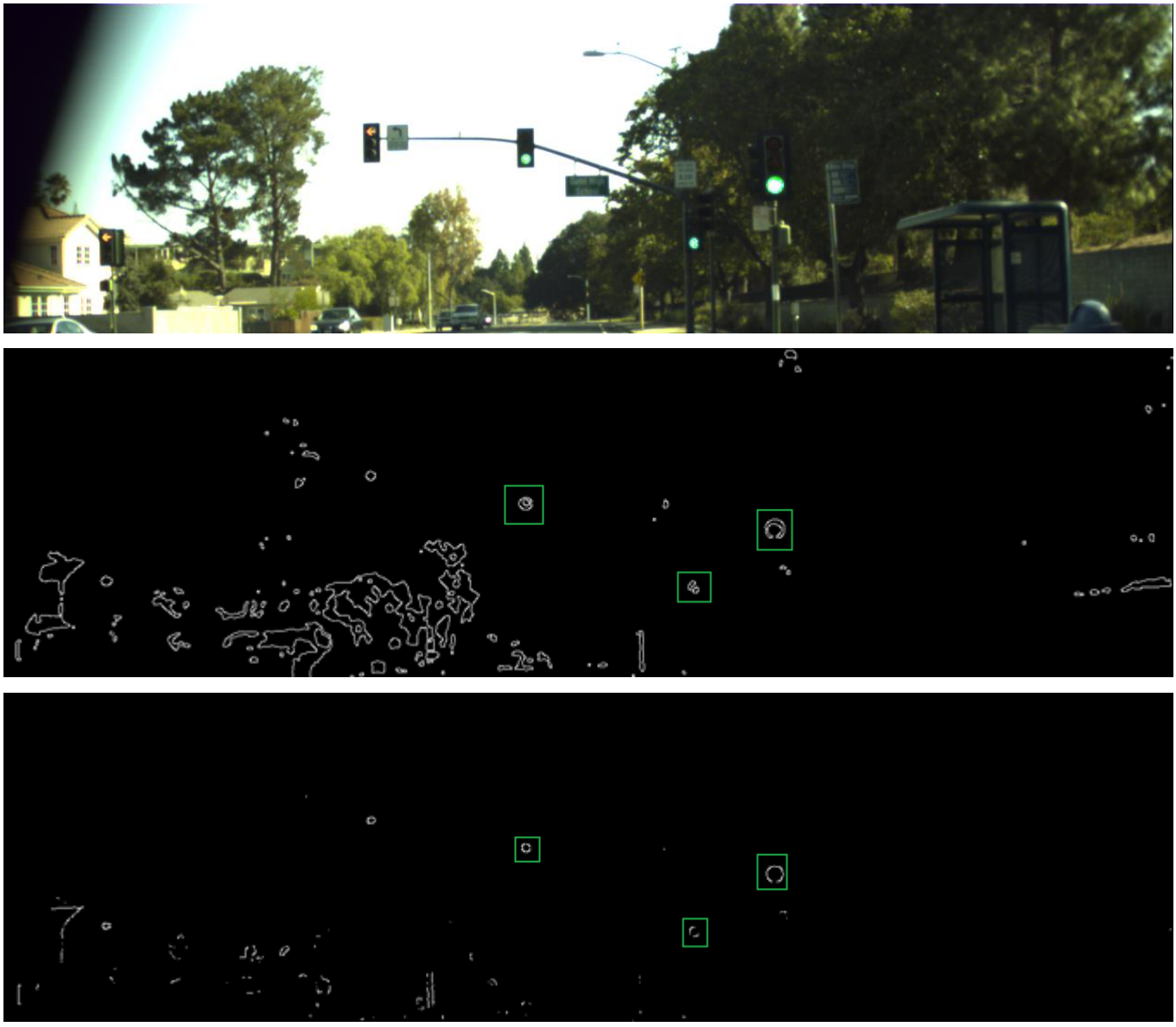

Slika 3.8. Prikaz konvolucije.

Kraćim testiranjem rješenja u trenutnoj fazi razvoja utvrđeno je da nakon segmentacije slike po boji i *Sobel* detektora rubova u pojedinim situacijama još uvijek ostaje previše područja od interesa, koja su zapravo šum, nad kojima bi se trebala provesti *Houghova* transformacija za kružnice rezultirajući lažnim pozitivima. Također, budući da je *Houghova* transformacija za kružnice računalno najzahtjevniji dio rješenja potrebno je vrlo dobro izolirati područja od interesa kako bi se vrijeme izvođenja svelo na minimum. Izmjene se prvo provode na konceptu rješenja da bi se ubrzao iterativni proces. Novopredloženo rješenje umjesto sekvencijalno postavljenog algoritma za segmentaciju po boji sa *Sobel* detektorom rubova iz izvorne slike zasebno vrši segmentaciju slike po boji i zasebno čini *Sobel* detekciju rubova. Rezultati obje operacije binarne su slike pa se između obje slike provodi logička operacija I [39].

Logička operacija I između dviju slika za ulaz uzima dvije slike iste širine i visine i uspoređuje vrijednosti na istim mjestima slike. Ako su obje vrijednosti 255, izlaz nove slike na tom mjestu je 255. U svim ostalim slučajevima rezultat na istom mjestu nove slike je 0, što je vidljivo u jednadžbi (3-7). Oznaka A predstavlja element izlazne slike na mjestu ($x$, $y$), dok B i C predstavljaju element iz matričnog zapisa ulaznih slika na istim tim mjestima ($x$, $y$).

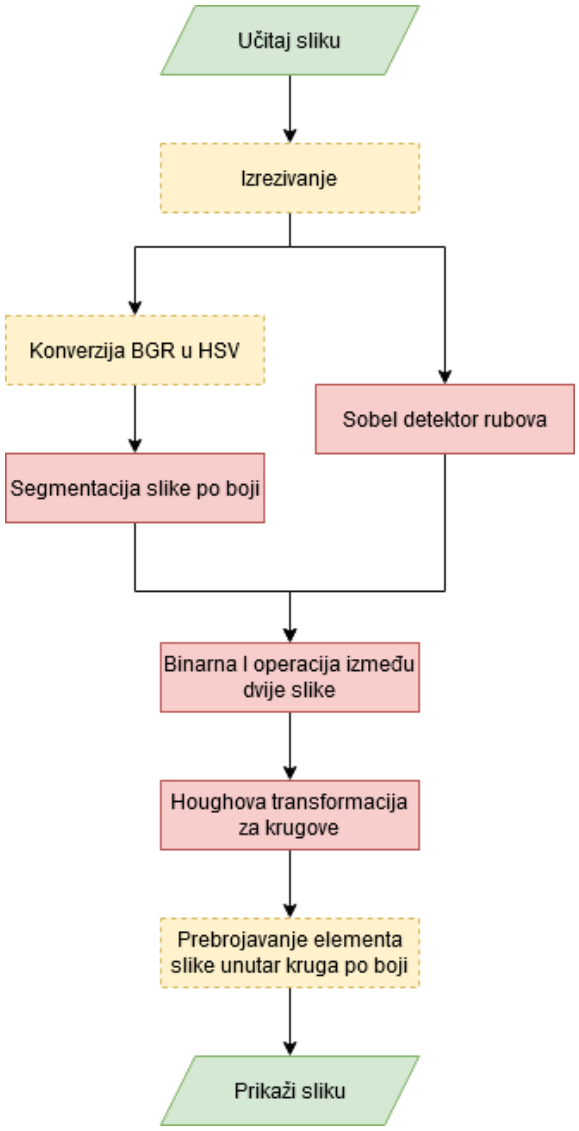
$$A_{(x,y)} = \begin{cases} 255, & if\,(B_{(x,y)} = 255\ AND\ C_{(x,y)} = 255) \\ 0, & u\ suprotnom \end{cases} \tag{3-7}$$

Usporedba novog algoritma dana je na slici 3.9., gdje je vidljivo da donja slika koja predstavlja izlaz iz novog algoritma, sadrži puno manje neželjenog šuma u usporedbi sa srednjom slikom koja je izlaz iz starog algoritma. Gornja slika je polazna slika za oba algoritma. Na srednjoj i donjoj slici zelenim kvadratom su označeni dijelovi koje je potrebno detektirati. Graf toka novog koncepta rješenja prikazan je na slici 3.10.



Slika 3.9. Prikaz ulazne slike (gore), izlazne slike starog algoritma (sredina) i izlazne slike novog algoritma (dolje).

*Houghova* transformacija za kružnice je među osnovnim alatima za detekciju značajki (engl. *feature extraction*) i služi za pronalaženje kružnica na slikama. Budući da je krug definiran kružnicom pomoću *Houghove* transformacije za kružnice moguće je pronaći i krugove. Kružnice




Slika 3.10. Graf toka novog algoritma.

koje se traže ne moraju biti potpune kružnice. Kao i ostale *Houghove* transformacije, *Houghova* transformacija za kružnice koristi se metodom glasanja u parametarskom prostoru. Parametarski prostor predstavljen je akumulatorom s onoliko dimenzija koliko ima nepoznatih parametara koje je potrebno pronaći. Jednadžba (3-8) predstavlja jednadžbu kružnice u dvodimenzionalnom prostoru gdje $a$ i $b$ predstavljaju koordinate središta kruga, $r$ predstavlja radijus kruga, a $x$ i $y$ su

$$(x-a)^2 + (y-b)^2 = r^2 \tag{3-8}$$

uređeni par točaka koje zadovoljavaju jednadžbu kružnice i nalaze se na kružnici. Ako su nam iz

jednadžbe poznati radijus *r* i ako je poznata točka s koordinatama *x* i *y* za koju je pretpostavljeno da se nalazi na kružnici preostali nedefinirani parametri su *a* i *b* koji predstavljaju središte kruga.

Dakle, za pronalazak kružnice s poznatim radijusom *r* potreban je akumulator s dvije dimenzije. Ako je radijus *r* nepoznat, akumulator postaje trodimenzionalan jer je potrebno pronaći i radijus što znatno povećava kompleksnost algoritma. Za detekciju svjetlosne prometne signalizacije eksperimentalno je određen minimalan radijus od 3 elementa slike i maksimalan radijus od 15 elemenata slike jer vrijednosti manje od 3 elementa slike uzrokuju previše lažnih pozitiva, a radijusi veći od 15 elemenata slike se u pravilu na slikama nastalim iz vozila ne pojavljuju. što znači da je korišten trodimenzionalan akumulator s dimenzijama visina slike *a*, širina slike *b* i razlika maksimalnog radijusa i minimalnog radijusa *r*. U početku su sva polja akumulatora postavljena na 0. Akumulator se puni na način da se ide po slici te se za svaki element slike s koordinatama (*x, y*), kojemu je vrijednost Y komponente 255, u akumulatoru na mjestu (*a*=*x*, b=*y*, *r*) za svaki radijus *r* „crta“ kružnica koja zadovoljava formulu (3-8) metodom opisanom od četvrte do sedme linije koda na slici 3.11. te se svim elementima akumulatora koji leže na kružnici daje jedan glas, odnosno vrijednost uvećava za 1.

***Linija*** ***Kod***

```
1:      /*Pseudokod za punjenje akumulatora*/
2:      Za svaki elementSlike(x,y):
3:          Za svaki radijus r od 2 do 15:
4:              Za svaki kut theta od 0 do 360
5:                  a = x – r * cos(theta * pi / 180)
6:                  b = y – r * sin(theta * pi / 180)
7:                  A[a,b,r]++
```

Sl. 3.11. Prikaz pseudokoda za punjenje akumulatora

Nakon što je akumulator dimenzija (*a, b, r*) napunjen kroz njega se prolazi od dimenzije većeg radijusa *r* prema manjem, element po element, te se provjerava vrijednost svakog elementa. Ukoliko je vrijednost veća od zadane granične vrijednosti provjerava se je li to najveća vrijednost u lokalnoj okolini. Lokalna okolina elementa akumulatora je definirana kao svi elementi akumulatora trenutno promatrane *r* dimenzije akumulatora koji se nalaze unutar radijusa *r* uvećanog za vrijednost minimalne udaljenosti između krugova. Ako je najveća lokalna vrijednost pronađena na mjestu (*a, b, r*), na ulaznoj slici se promatra kružnica sa središtem u (*a, b*) i radijusa

$r$ te se promatra omjer broja elemenata slike koji leže na kružnici da imaju vrijednost Y komponente elementa slike jednaku 255 i omjer broja elemenata slike koji leže na kružnici. Ako je omjer veći od zadane granične vrijednosti krug je pronađen i ima parametre: središte u ($a$, $b$) i radijus $r$. U akumulatoru se brišu sve vrijednosti unutar lokalnog radijusa pronađenog središta za sve dimenzije radijusa. Za detekciju svjetlosne prometne signalizacije granična vrijednost akumulatora je 80, granična vrijednost omjera je 0.4, a minimalna udaljenost između krugova je 30 elemenata slike. Navede vrijednosti su eksperimentalno utvrđene.

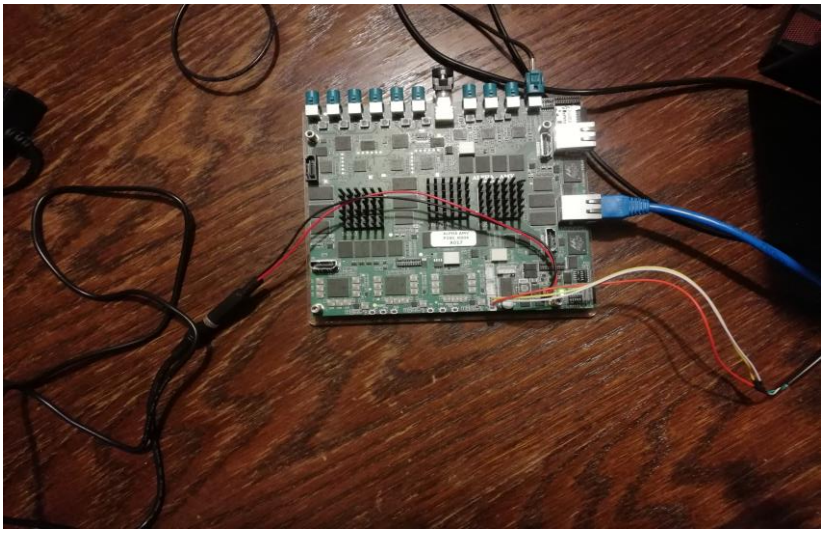
Nakon što su krugovi koji predstavljaju svjetlosnu prometnu signalizaciju detektirani preostaje detektirati kojeg je stanja svjetlosna prometna signalizacija. Ovaj korak provodi se tako da se svi elementi slike koji predstavljaju površinu pronađenog kruga segmentiraju po zelenoj i crvenoj boji prema formuli (3-9). Ukoliko je unutar detektiranog kruga više zelenih elemenata

$$stanje = \begin{cases} stani, & if(0 < H_{(x,y)} < 25\ and\ 100 < S_{(x,y)} < 255\ and\ 100 < V_{(x,y)} < 255) \\ kreni, & if(50 < H_{(x,y)} < 80\ and\ 100 < S_{(x,y)} < 255\ and\ 100 < V_{(x,y)} < 255) \end{cases} \quad (3\text{-}9)$$

slike stanje svjetlosne prometne signalizacije je „kreni“, a ako je više crvenih elemenata slike stanje svjetlosne prometne signalizacije je „stani“. Rješenje za izlaz daje središta i radijus svjetlosne prometne signalizacije te stanje na njima. U prilogu P.3.1. dan je *use-case* zajedno sa svim izrađenim algoritmima.

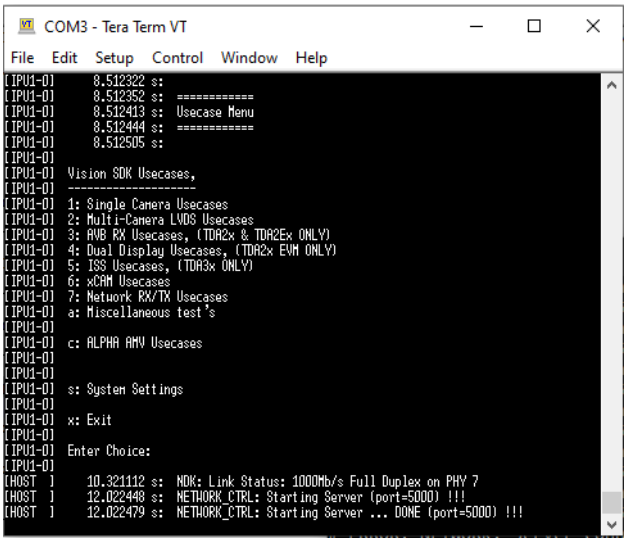
## 3.4. Pokretanje programskog rješenja na realnoj ADAS platformi

Pokretanje rješenja na realnoj ADAS platformi zahtijeva fizičko povezivanje ALPHA ploče s ulazno-izlaznim uređajima i računalom. Kako bi rješenje moglo biti pokrenuto potrebno je: povezati SCV SoC HDMI izlaz na zaslon, povezati SCV SoC UART s USB portom na računalu, SCV SoC ethernet port povezati na mrežu i ubaciti *MicroSD* karticu u slot za SCV SoC na kojoj se nalaze *appimage* i *sbl_sd* slike koje sadrže kompilirano rješenje. ALPHA ploča spojena sa svom periferijom i spremna za rad prikazana je na slici 3.12.



Slika 3.12. Spojena ALPHA ploča.

Za komunikaciju računala s pločom putem UART komunikacije korišten je *TeraTerm* [40] alat otvorenog koda. Ovaj alat omogućuje kontrolu ALPHA ploče slanjem ASCII znakova očitanih s tipkovnice putem serijske komunikacije i prikaz povratne informacije s ALPHA ploče u prozoru *TeraTerm* aplikacije. Nakon povezivanja *TeraTerma* s ALPHA pločom prikazan je skup opcija u aplikaciji *TeraTerma*. Razvijeno rješenje pokrenuto je pritiskom na tipku „c“, što dovodi do idućeg izbornika unutar kojeg je potrebno pritisnuti i odabrati opciju „9“. Na slici 3.13. vidi se prozor aplikacije *TeraTerma* s početnim izbornikom.




Slika 3.13. Prikaz glavnog izbornika aplikacije za ALPHA ploču u *TeraTermu*.

Ako je sve napravljeno prema uputama, na ploči je pokrenut *use-case* za detekciju svjetlosne prometne signalizacije i očekuje slanje slike putem mreže. Kako bi se podatci putem mreže poslali ALPHA ploči, unutar *VisionSDK* nalazi se set alata kojima je moguće slati i primati podatke. Ovi alati zovu se *network_tx* i *network_rx*. ALPHA ploči dodijeljena je statička IP adresa 192.168.0.69. Za slanje podataka na ALPHA ploču potrebno je kopirati željene datoteke u mapu s navedenim programima za slanje i primanje podataka, pozicionirati se u tu istu mapu putem *command prompta* te iz njega pokrenuti aplikaciju za slanje ili primanje podataka sa željenim parametrima među kojima su obvezni IP adresa pošiljatelja, IP adresa primatelja i ime datoteke koja se šalje. Ovim postupkom slika je učitana u *use-case* te se kroz nekoliko sekundi slika prikazuje na zaslonu povezanom s ALPHA pločom.

# 4. EVALUACIJA RADA PREDLOŽENOG RJEŠENJA ZA PREPOZNAVANJE SVJETLOSNE PROMETNE SIGNALIZACIJE U NAPREDNIM ADAS ALGORITMIMA

U ovom poglavlju prikazana je evaluacija rada rješenja za detekciju svjetlosne prometne signalizacije na SCV sustavu na čipu ALPHA ploče. U potpoglavlju 4.1. objašnjen je način evaluacije, korištene metode i korišteni podatkovni skup. U potpoglavlju 4.2. prikazani su rezultati kvantitativne metode evaluacije. U potpoglavlju 4.3. dana je kvalitativna evaluacija.

## 4.1. Način evaluacije

Za kvantitativnu evaluaciju rada rješenja za detekciju svjetlosne prometne signalizacije koriste se preciznost (engl. *precision*) i odziv (engl. *recall*). Preciznost je definirana kao udio točno klasificiranih primjera u skupu pozitivno klasificiranih primjera i računa se prema jednadžbi (4-1). Odziv je definiran kao udio točno klasificiranih primjera u skupu svih pozitivnih primjera. Za izračun preciznosti i odziva potrebno je poznavati broj istinito pozitivnih (engl. *True Positive, TP*), lažno pozitivnih (engl. *False Positive, FP*) i lažno negativnih (engl. *False Negative, FN*) rezultata detekcije. Preciznost se računa prema jednadžbi (4-1), a odziv prema jednadžbi (4-2).

$$preciznost = \frac{TP}{TP + FP} \tag{4-1}$$

$$odziv = \frac{TP}{TP + FN} \tag{4-2}$$

Rješenje za detekciju svjetlosne prometne signalizacije kvantitativno je evaluirano koristeći preciznost i odziv. Kvalitativna evaluacija izvršena je uspoređujući stvarne izlaze s predviđenim izlazima algoritma za detekciju svjetlosne prometne signalizacije.

Kako bi se utvrdio broj istinito pozitivnih rezultata potrebno je uvesti nekakvu mjeru prema kojoj će se rezultat smatrati istinito pozitivnim. U ovu svrhu korištene su metoda vlastite procjene i presjek nad unijom (engl. *Intersection Over Union, IOU*). Prvobitno je korištena metoda vlastite procjene rezultata detekcije. Ovom metodom vizualno se analizira dobiveni rezultat detekcije i prema vlastitoj procjeni određuje TP, FP i FN. Dakle, ako se vizualno detekcija približno nalazi na području sa svjetlosnom prometnom signalizacijom tada se to označava kao TP. Ako je recimo primijećeno da negdje svjetlosna prometna signalizacije nije detektirana tada je to FN, a ako se očita detekcija svjetlosne prometne signalizacije tamo gdje je nema označava se s FP. Ovakva

evaluacija rezultata nije u mogućnosti proizvesti reproducibilne rezultate stoga je dodana metoda evaluacije korištenjem presjeka nad unijom.

Korištenjem presjeka nad unijom rezultati evaluacije su označeni kao istinito pozitivni ukoliko im je presjek nad unijom veći od 50%. Presjek nad unijom računa se prema (4-3) gdje je IOU presjek nad unijom, $B_p$ površina predviđenog graničnog okvira, a $B_i$ površina stvarnog graničnog okvira. Algoritam za računanje presjeka nad unijom također u sebi sadrži i brojač za FP

$$IOU = \frac{površina(B_p \cap B_i)}{površina(B_p \cup B_i)} \tag{4-3}$$

i FN. Detekcija se označava sa FP ako je učitana s liste predviđenih graničnih okvira, a nije joj pronađen par u listi *ground truth* okvira, odnosno nema presjek ni s jednim *ground truth* graničnim okvirom. Slično ovomu detekcija se označava s FN ako je učitana s liste *ground truth* graničnih okvira i nema presjek ni s jednim predviđenim graničnim okvirom. Skripta koja sadrži algoritam za evaluaciju je napisana u *Python* programskom jeziku i priložena je kao prilog P.4.1.

Problem koji nastaje iz korištenja presjeka nad unijom je što funkcija evaluacije na jednaki način penalizira greške na velikim i malim graničnim okvirima, iako mala greška na malom okviru puno jače utječe na presjek nad unijom. Za primjer su uzeti granični okvir veličine 3x3 i granični okvir veličine 10x10. Ako je predviđeni granični okvir pomaknut za jedan stupac udesno za granični okvir veličine 3x3 ta pogreška iznosi 50%, a za granični okvir veličine 10x10 pogreška iznosi 18%.

Evaluacija je izvršena na reduciranom *Boschovom* skupu podataka za svjetlosnu prometnu signalizaciju (*Bosch Traffic Lights Dataset*) [41]. *Boschov* skup sadrži fotografije nastale vožnjom po danu u različitim vremenskim uvjetima. *Boschov* skup podataka je reduciran na način da je iz njega nasumično odabrano 60 fotografija koje sadrže svjetlosnu prometnu signalizaciju. Ovako izrađen skup nalazi se u prilogu P.4.2. Svaka fotografija je provučena kroz izrađeni algoritam za detekciju svjetlosne prometne signalizacije nakon čega je dobivena datoteka s predviđenim graničnim okvirima. Skup ovih datoteka dan je u prilogu P.4.3. Za označavanje *ground truth* graničnih okvira koristio se *openlabeling* alat [42] čime se dobije tekstualna datoteka s informacijama o *ground truth* graničnim okvirima danima u prilogu P.4.4.

## 4.2. Kvantitativna evaluacija

U tablici 4.1. prikazani su rezultati evaluacije rješenja za detekciju svjetlosne prometne signalizacije metodom vlastite procjene za stanje kada je na svjetlosnoj prometnoj signalizaciji zeleno i za stanje kada je crveno. Iz tablice 4.1. vidljiv je nesrazmjer u *Boschovom* skupu podataka između ukupnog broja slučajeva kada je svjetlosna prometna signalizacija indicirala stanje zeleno i kada je svjetlosna prometna signalizacija indicirala stanje crveno. Na cijelom korištenom skupu moguće je pronaći 124 stanja zeleno i 42 stanje crveno.

Tablica 4.1. Rezultati evaluacije algoritma za detekciju svjetlosne prometne signalizacije metodom vlastite procjene.

| | TP | FP | FN |
|---|---|---|---|
| Zeleno | 63 | 10 | 61 |
| Crveno | 21 | 52 | 21 |

U tablici 4.2. prikazane su mjere preciznosti i odziva za metodu slobodne procjene za svako detektirano stanje. Iz tablice 4.2. vidljivo je da je preciznost detekcije stanja crveno vrlo niska. Lažni pozitivi koji uzrokuju ovako nisku preciznost za stanje crveno su zadnja svjetla drugih vozila

Tablica 4.2. Preciznost i odziv za stanje zeleno i crveno dobiveni metodom slobodne procjene.

| | Preciznost | Odziv |
|---|---|---|
| Zeleno | 86.30% | 50.81% |
| Crveno | 30.88% | 50.00% |

u prometu. Algoritam za detekciju svjetlosne prometne signalizacije stražnja svjetla vrlo lako zamijeni za svjetlosnu prometnu signalizaciju, pogotovo ako su kružnog oblika. Stanje zeleno ima visoku preciznost što se može pripisati činjenici da u prirodi nema puno zelenih objekata s visokim vrijednostima svjetline i saturacije pa rijetko dolazi do lažnih pozitiva. Vrijednosti odziva za oba stanja su približno iste, pri čemu je odziv za crveno ipak malo veći.

U tablici 4.3. prikazani su rezultati evaluacije kada je korišten presjek nad unijom.

Tablica 4.3. Rezultati evaluacije algoritma za detekciju svjetlosne prometne signalizacije pri čemu je korišten presjek nad unijom.

| | TP | FP | FN |
|---|---|---|---|
| Zeleno | 58 | 10 | 66 |
| Crveno | 16 | 52 | 26 |

Usporedbom tablice 4.3. s tablicom 4.1. moguće je vidjeti da se korištenjem presjeka nad unijom broj istinito pozitivnih rezultata umanjio, dok se broj lažno negativnih rezultata uvećao. Ovo je rezultat toga što presjek nad unijom kao istinito pozitivne rezultate uzima samo one rezultate s vrijednošću iznad 50%, dok ostale smatra nedetektiranima, odnosno lažno negativnim rezultatima. Broj lažno pozitivnih rezultata ostao je nepromijenjen u odnosu na metodu vlastite procjene. U tablici 4.4. dane su vrijednosti za preciznost i odziv presjeka nad unijom. Analizom i usporedbom

Tablica 4.4. Preciznost i odziv za stanje zeleno i crveno dobiveni metodom presjeka nad unijom.

| | Preciznost | Odziv |
|---|---|---|
| Zeleno | 85.30% | 46.77% |
| Crveno | 25.40% | 38.10% |

tablice 4.4 i tablice 4.2. vidljivo je kako se korištenjem metode presjeka nad unijom značajno smanjio odziv za detekciju stanja crveno. Ostale vrijednosti su nešto manje u odnosu na metodu slobodne procjene. Prosječno vrijeme obrade jedne ulazne slike je približno četiri sekunde, što znači da rješenje u trenutnom stanju nije sposobno raditi na ALPHA ploči u stvarnom vremenu.

## 4.3. Kvalitativna analiza

Na slici 4.1. vidljiva je detekcija streličaste svjetlosne prometne signalizacije.

Slika 4.1. Prikaz detekcije streličaste svjetlosne prometne signalizacije kao lažno pozitivnih detekcija.

Donja slika je ulazna slika u algoritam, a gornja slika predstavlja ulaznu sliku s detektiranom svjetlosnom prometnom signalizacijom. Zeleni krugovi označavaju detektirano zeleno stanje, a crveni detektirano crveno stanje. Budući da se u ovom radu ne nastoji detektirati ovu vrstu svjetlosne prometne signalizacije, u evaluaciji su ovakvi rezultati predstavljeni kao lažni pozitivi za crvenu boju, čime je znatno umanjena preciznost rješenja za detekciju svjetlosne prometne signalizacije. Do detekcije lažnih pozitiva dolazi jer streličasta svjetlosna prometna signalizacija na većim udaljenostima, zbog nesavršenosti kamere, poprima kružni oblik te se detektira kao kružna. Na slici 4.2. moguće je vidjeti streličastu svjetlosnu prometnu signalizaciju na manjoj udaljenosti. Ovdje je vidljivo da pri manjim udaljenostima streličasta svjetlosna prometna

signalizacija ima svoj izvorni oblik te se ona ne detektira, što je ispravno.


Slika 4.2. Prikaz streličaste svjetlosne prometne signalizacije na kraćoj udaljenosti.

Na slici 4.3. prikazan je rad rješenja u uvjetima intenzivnog osvjetljenja.

Slika 4.3. Prikaz detekcije svjetlosne prometne signalizacije u uvjetima visoke osvijetljenosti.

Na slici je vidljivo da su ispravno detektirane dvije od tri instance svjetlosne prometne signalizacije. Kao suprotnost slici 4.3. postavljena je slika 4.4. koja prikazuje detekciju svjetlosne prometne signalizacije u uvjetima smanjenog osvjetljenja.


Slika 4.4. Prikaz detekcije svjetlosne prometne signalizacije u uvjetima niske osvijetljenosti.

Na ovoj slici nije detektirana niti jedna instanca svjetlosne prometne signalizacije jer su zasićenost i svjetlina signalizacije vrlo niskih vrijednosti pa ne prolaze uvjete segmentacije po boji. Proširenjem granica za segmentaciju omogućila bi se detekcija ovakvih slučajeva, no tada bi se pojavili i mnogi lažni pozitivi nastali detekcijom okolnog raslinja.

Na slici 4.5. vidljiva je detekcija stražnjih svjetala drugih vozila kao lažnih pozitiva.


Slika 4.5. Prikaz detekcije stražnjih svjetala kao lažnih pozitiva.

Do detekcije dolazi zbog sličnosti u nijansi boje čime prolazi uvjet segmentacije po boji dok

*Houghovu* transformaciju za kružnice prolazi zbog veće udaljenosti kamere od točkastog izvora svijetlosti čime izvor svijetla poprima kružni oblik.

Na slici 4.6. nalazi se prikaz utjecaja udaljenosti na sposobnost detekcije svjetlosne prometne signalizacije. Na prvom raskrižju nalaze se tri instance svjetlosne prometne signalizacije


Slika 4.6. Prikaz utjecaja udaljenosti na detekciju svjetlosne prometne signalizacije.

koje je rješenje detektiralo, a na idućem, znatno udaljenijem raskrižju, nalaze se još tri instance koje nisu detektirane umanjujući ukupni odziv. Udaljene instance svjetlosne prometne signalizacije su ispod zadanog minimalnog radijusa *Houghove* transformacije za kružnice. Ako bi se minimalni radijus za detekciju dodatno smanjio, gotovo svaka nakupina od nekoliko elemenata slike detektirala bi se kao krug.

# 5. ZAKLJUČAK

Konstantne inovacije u području računalnog vida i umjetne inteligencije omogućuju razvoj sve sofisticiranijih sustava za pomoć u vožnji. Svi napredni sustavi za pomoć u vožnji sastoje se od mnoštva podsustava zaduženih za analizu stanja u okolini, kontrolu vozila i donošenje odluka u vožnji. U ovom diplomskom radu prikazan je postupak izrade jednog takvog podsustava za detekciju svjetlosne prometne signalizacije i njegova implementacije na realnu ADAS platformu naziva ALPHA ploča.

Prilikom izrade rješenja analizirana je nekolicina znanstvenih radova na temu detekcije svjetlosne prometne signalizacije što je uvelike olakšalo daljnje korake izrade koncepta rješenja za detekciju svjetlosne prometne signalizacije i implementacije rješenja na ALPHA ploču. Izradom koncepta rješenja u *Python* programskom jeziku omogućeno je brzo eksperimentiranje s potencijalnim načinima rješavanja problema detekcije svjetlosne prometne signalizacije bez da se bori sa specifičnostima ugradbene platforme. Implementiranje rješenja za detekciju svjetlosne prometne signalizacije na ALPHA ploču oduzelo je najviše vremena prilikom izrade ovog rada, jer svaka ugradbena platforma ima svojih specifičnosti s kojima je potrebno biti vrlo dobro upoznat kako bi se iz platforme izvukle najbolje performanse. Rezultati evaluacije metodom koja koristi presjek nad unijom su pokazali preciznost od 55.35% pri detekciji stanja svjetlosne prometne signalizacije dok je odziv nešto manji s 43.44%. Izrađeno rješenje nije moguće pokrenuti u stvarnom vremenu na ALPHA ploči. Ovo bi moglo biti ispravljeno dodatnim upoznavanjem s platformom što bi rezultiralo i boljom optimizacijom koda za samu platformu.

# LITERATURA


[1] *Watch an army of robots efficiently sorting hundreds of parcels per hour*. 2017.

[2] 'Moxi', *Diligent Robotics*. https://diligentrobots.com/moxi (accessed Sep. 25, 2020).

[3] J. Kelly, 'Artificial Intelligence Is Superseding Well-Paying Wall Street Jobs', *Forbes*. https://www.forbes.com/sites/jackkelly/2019/12/10/artificial-intelligence-is-superseding-well-paying-wall-street-jobs/ (accessed Sep. 25, 2020).

[4] M. W. March 27 and 2019, 'AI Is Good (Perhaps Too Good) at Predicting Who Will Die Prematurely', *livescience.com*. https://www.livescience.com/65087-ai-premature-death-prediction.html (accessed Sep. 25, 2020).

[5] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. 2018.

[6] 'RT-RK - Automotive'. https://www.rt-rk.com/services/automotive (accessed Oct. 01, 2020).

[7] 'TI Vision SDK, Optimized Vision Libraries for ADAS Systems', p. 12, 2014.

[8] V. John, L. Zheming, and S. Mita, 'Robust traffic light and arrow detection using optimal camera parameters and GPS-based priors', in *2016 Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, Jul. 2016, pp. 204–208, doi: 10.1109/ACIRS.2016.7556213.

[9] J. Müller, A. Fregin, and K. Dietmayer, 'Multi-camera system for traffic light detection: About camera setup and mapping of detections', in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2017, pp. 165–172, doi: 10.1109/ITSC.2017.8317946.

[10] M. Munoz-Organero, R. Ruiz-Blaquez, and L. Sánchez-Fernández, 'Automatic detection of traffic lights, street crossings and urban roundabouts combining outlier detection and deep learning classification techniques based on GPS traces while driving', *Comput. Environ. Urban Syst.*, vol. 68, pp. 1–8, Mar. 2018, doi: 10.1016/j.compenvurbsys.2017.09.005.

[11] L. C. Possatti *et al.*, 'Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars', *2019 Int. Jt. Conf. Neural Netw. IJCNN*, pp. 1–8, Jul. 2019, doi: 10.1109/IJCNN.2019.8851927.

[12] D. H. Ballard, *Computer vision*. Englewood Cliffs, N.J: Prentice-Hall, 1982.

[13] N. O'Mahony *et al.*, 'Deep Learning vs. Traditional Computer Vision', in *Advances in Computer Vision*, vol. 943, K. Arai and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 128–144.

[14] R. Kulkarni, S. Dhavalikar, and S. Bangar, 'Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning', in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, Aug. 2018, pp. 1–4, doi: 10.1109/ICCUBEA.2018.8697819.

[15] C. L. Zitnick and P. Dollár, 'Edge Boxes: Locating Object Proposals from Edges', in *Computer Vision – ECCV 2014*, vol. 8693, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 391–405.

[16] Y. Zhang, 'Support Vector Machine Classification Algorithm and Its Application', in *Information Computing and Applications*, Berlin, Heidelberg, 2012, pp. 179–186, doi: 10.1007/978-3-642-34041-3_27.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, 'Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation', in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[19] M. B. Jensen, K. Nasrollahi, and T. B. Moeslund, 'Evaluating State-of-the-Art Object Detector on Challenging Traffic Light Data', in *2017 IEEE Conference on Computer Vision*

*and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, USA, Jul. 2017, pp. 882–888, doi: 10.1109/CVPRW.2017.122.
[20] 'LISA Traffic Light Dataset'. https://kaggle.com/mbornoe/lisa-traffic-light-dataset (accessed Sep. 04, 2020).
[21] M. Gluhaković, 'Detekcija vozila u okruženju autonomnog vozila u svrhu upozorenja na potencijalnu koliziju', master's thesis, Josip Juraj Strossmayer University of Osijek. Faculty of Electrical Engineering, Computer Science and Information Technology Osijek. Department of Communications. Chair of Electronics and Microelectronics., 2019.
[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', *ArXiv150602640 Cs*, May 2016, Accessed: Sep. 03, 2020. [Online]. Available: http://arxiv.org/abs/1506.02640.
[23] T. H. Tran, C. C. Pham, T. P. Nguyen, T. T. Duong, and J. W. Jeon, 'Real-time traffic light detection using color density', in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Oct. 2016, pp. 1–4, doi: 10.1109/ICCE-Asia.2016.7804791.
[24] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2016.
[25] Masako Omachi and Shinichiro Omachi, 'Traffic light detection with color and edge information', in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Aug. 2009, pp. 284–287, doi: 10.1109/ICCSIT.2009.5234518.
[26] B. Jähne, *Digital image processing*, 5., rev.Extended ed. Berlin: Springer, 2002.
[27] G. Siogkas and E. Skodras, 'Traffic Lights Detection in Adverse Conditions Using Color, Symmetry and Spatiotemporal Information', in *VISAPP 2012 - Proceedings of the International Conference on Computer Vision Theory and Applications*, 2012, vol. 1.
[28] G. Loy and A. Zelinsky, 'Fast radial symmetry for detecting points of interest', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 959–973, Aug. 2003, doi: 10.1109/TPAMI.2003.1217601.
[29] M. R. Luo, 'CIELAB', in *Encyclopedia of Color Science and Technology*, R. Luo, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1–7.
[30] P. Soille, *Morphological Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.
[31] 'Materijali za laboratorijske vježbe iz predmeta Digitalna obrada slike i videa za autonomna vozila'. .
[32] 'About'. https://opencv.org/about/ (accessed Sep. 18, 2020).
[33] D. Li, Ed., 'HSV Color Space', in *Encyclopedia of Microfluidics and Nanofluidics*, Boston, MA: Springer US, 2008, pp. 793–793.
[34] J. Canny, 'A Computational Approach to Edge Detection', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986, doi: 10.1109/TPAMI.1986.4767851.
[35] 'YUV pixel formats'. http://www.fourcc.org/yuv.php (accessed Sep. 19, 2020).
[36] 'OpenCV: Color conversions'. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html (accessed Sep. 24, 2020).
[37] I. Sobel, 'An Isotropic 3x3 Image Gradient Operator', *Present. Stanf. AI Proj. 1968*, 2014.
[38] V. Dumoulin and F. Visin, 'A guide to convolution arithmetic for deep learning', 2016.
[39] G. Anbarjafari, *7. Arithmetic and logic operations*. .
[40] 'Tera Term - Terminal Emulator for Windows'. https://ttssh2.osdn.jp/ (accessed Sep. 25, 2020).
[41] K. Behrendt, L. Novak, and R. Botros, 'A deep learning approach to traffic lights: Detection, tracking, and classification', in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1370–1377, doi: 10.1109/ICRA.2017.7989163.

[42] ‘GitHub - Cartucho/OpenLabeling: Label images and video for Computer Vision applications’. https://github.com/Cartucho/OpenLabeling (accessed Oct. 01, 2020).

# SAŽETAK

U ovom radu razvijeno je rješenje za detekciju svjetlosnu prometnu signalizaciju pri čemu su korištene metode računalne obrade slike. Prvo je izrađen koncept rješenja u programskom jeziku *Python,* korištenjem *OpenCV* biblioteke. Konceptno rješenje je poslužilo kao polazna točka za razvoj rješenja unutar *VisionSDK* razvojnog alata kako bi se olakšala implementacija na ALPHA ploču. U potpunosti razvijeno rješenje za detekciju svjetlosne prometne signalizacije pokrenuto je na ALPHA ploči kako bi se provela evaluacija rezultata. Rezultati su pokazali kako razvijeno rješenje u trenutnom obliku nudi preciznost od 55.35%, odziv od 43.44% i da nije u mogućnosti raditi u stvarnom vremenu.

Ključne riječi: *ADAS, detekcija svjetlosne prometne signalizacije, računalna obrada slike, VisionSDK*

# Development of a solution for the recognition of light traffic signaling and implementation of the developed solution on a real ADAS platform


## ABSTRACT

Subject of this master's thesis is development of a solution for traffic light recognition for implementation on real ADAS platform using image processing methods. First stepo was development of concept solution, created using *Python* and *OpenCV* library Concept solution was used as a strating point for *VisionSDK* solution development. *VisionSDK* solution was developed so it can be easly implemented on ALPHA board. Once completed, solution was run on ALPHA board to be evaluated. Evaluation was done using intersection over union method. Results of evalutaion showed precision score of 55.35% and recall score of 43.44%.



Keywords: *ADAS, Image processing, traffic light recognition, VisionSDK*

# ŽIVOTOPIS

Antun Kakuk rođen je 10. ožujka 1993. godine u Požegi. Od 1999. do 2007. pohađa Osnovnu školu fra Kaje Adžića u Pleternici. Nakon završetka osnovne škole upisuje Katoličku klasičnu gimnaziju s pravom javnosti u Požegi od 2007. do 2011. godine. Te iste 2011. godine upisuje prvi studij, preddiplomski studij biologije na Odjelu za biologiju sveučilišta Josipa Jurja Strossmayera u Osijeku kojeg napušta nakon godinu dana studiranja. Nakon dvije godine upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku kojeg završava 2018. godine. Iste godine upisuje diplomski studij računarstva smjer Robotika i umjetna inteligencija.

# PRILOZI

- P.3.1. Zip datoteka s *use-caseom* rješenja i izrađenim algoritmima (priloženo na DVD-u uz rad)
- P4.1. *Python* skripta za evaluaciju rješenja. (priloženo na DVD-u uz rad)
- P.4.2. Slike korištene za evaluaciju rada rješenja (priloženo na DVD-u uz rad)
- P.4.3. Tekstualne datoteke s koordinatama predviđenih graničnih okvira (priloženo na DVD-u uz rad)
- P.4.4. Tekstualne datoteke s koordinatama graničnih okvira sa *ground truthom* (priloženo na DVD-u uz rad)