Spajanje slika dobivenih s više kamera u jedinstvenu sliku i korekcija osvjetljenja dobivene slike za primjenu u ADAS algoritmima

Kundid, Josip

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

Sveučilišni studij

SPAJANJE SLIKA DOBIVENIH S VIŠE KAMERA U JEDINSTVENU SLIKU I KOREKCIJA OSVJETLJENJA DOBIVENE SLIKE ZA PRIMJENU U ADAS ALGORITMIMA

Diplomski rad

Josip Kundid

Osijek, 2020.

SADRŽAJ

1. UVOD

Razvoj algoritama u području automobilske industrije značajno doprinosi sigurnosti vozača i drugih sudionika u prometu. Algoritmi za obradu slike, kontrolu kretanja vozila, prikupljanje podataka sa senzora, obradu prikupljenih podataka i sl., usavršavaju se i implementiraju u vozila s ciljem postizanja što veće sigurnosti vozača i ostalih sudionika u prometu. Sustavi za pomoć vozaču prilikom vožnje naziva ADAS (engl. *Advanced Driver Assistance Systems*), omogućuju implementaciju navedenih algoritama, čime se vozaču vozila obradom prikupljenih podataka iz okoline sa senzora vozila, olakšava kontrola vozila i povećava sigurnost prometa u pogledu: postupne zamjene kontrole upravljanja vozilom u određenom broju scenarija bez potrebe za intervencijom vozača, upozoravanja na nailazeće opasnosti i prepreke te pregleda okoline vozila. Postizanje visoke razine sigurnosti prometovanja vozila omogućeno je sve preciznijim i povoljnijim senzorima, ograničenom, ali dovoljno velikom računalnom moći ugradbenih procesora kao i razvojem algoritama za rad u stvarnom vremenu. Jedan od osnovnih algoritama za obradu slike je spajanje više slika dobivenih s kamera u jedinstvenu sliku i naknadna obrada (engl. *post process)* dobivene slike. Spajanjem više slika omogućen je pregled većeg dijela okoline vozila, nego što je omogućeno jednom širokokutnom kamerom, poput panoramskog pregleda okoline iza vozila (engl. *Rear Stitched View)* za kontrolu mrtvih kutova kao i kružnog pregleda oko vozila (engl. *Surround view*), za detekciju nailazećih objekata u blizini vozila, itd.

Algoritam za spajanje slika dobivenih s više kamera u jedinstvenu sliku i korekciju osvjetljenja dobivene slike omogućava prikaz veće površine okoline vozila, čime se unutar svakog dobivenog okvira slike videa (engl. *frame*) snimljenog u stvarnom vremenu kamerama vozila, dobiva veća količina bitnih informacija. Kreiranjem spojenog prikaza pojedinih okvira, za potrebe pregleda ispred te iza vozila kao i prikaz kružnog pregleda vozila, omogućena je obrada dobivene slike koja sadrži sve objekte okoline snimljene s više kamera. Obuhvaćanjem većeg dijela okoline i prisutnih objekata olakšava se prepoznavanje objekata za algoritme detekcije jer su objekti u sceni jasniji, vidljivi, pregledniji te se objekti mogu pratiti u odnosu na položaj vozila. Potreba za pregledom šire okoline očituje se u algoritmima za detekciju nailazećih objekata u mrtvom kutu, gdje se preglednost mrtvog kuta povećava u odnosu na vidno područje vozača, detekciju prepreka i nailazećih opasnosti ispred vozila te praćenje i pozicioniranje objekata u blizini vozila. Iako je spajanje više slika s područjem preklapanja već razvijeno u industriji računalnog vida, algoritmi za spajanje više slika za potrebe izvođenja na ugradbenim računalima vozila susreću se s većim preprekama prilikom implementacije. Problemi prilikom spajanja slika za ugradbena računala

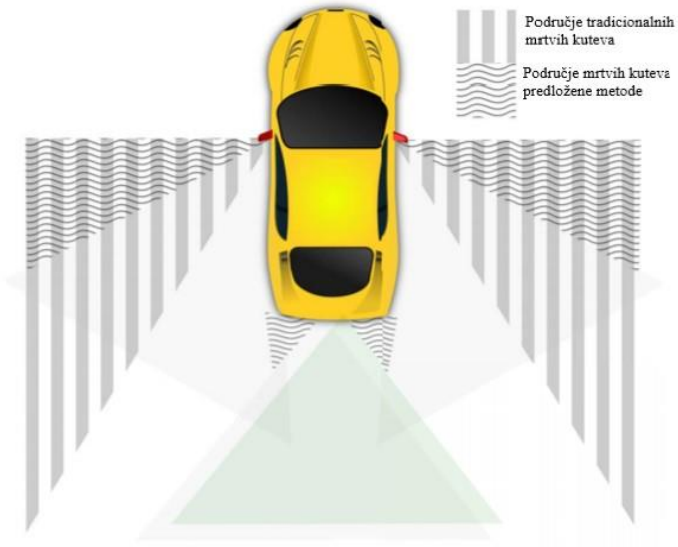
unutar vozila znatno ovise o ograničenoj moći ugradbenih računala, većoj potrebi za izvršavanjem u stvarnom vremenu te dinamičnim promjenama okoline i osvjetljenja prilikom kretanja vozila. Cilj ovog algoritma je u stvarnom vremenu spojiti više dobivenih okvira s različitih kamera, a da se pri tome očuva kvaliteta ulaznih slika, ne izgube značajne informacije, područje prijelaza između spojenih slika bude neprimjetno te da se ispravi različiti intenzitet osvjetljenja na slikama snimljenim s više kamera, korištenih za spajanje u jedinstvenu sliku. Navedeni ciljevi se, u osnovi, uspješno rješavaju razdvajanjem problema na segmente: kalibracija kamera, prepoznavanje ključnih točaka na dobivenim okvirima, kreiranje deskriptora ključnih točaka, povezivanje istih ključnih točaka na različitim slikama, proračun matrice homografije, spajanje slika pomoću izračunate matrice homografije te korekcija različitih intenziteta osvjetljenja spojene slike.

Ovim diplomskim radom proučene su i uspoređene postojeće metode za: detekciju ključnih točaka, spajanje jednakih ključnih točaka na različitim slikama u dijeljenom području te metode za proračun matrice homografije. Također, istražene su metode za korekciju osvjetljenja nad spojenim slikama te odlukom o primjeni određenih metoda razvijen je algoritam za rad u stvarnom vremenu. Za potrebe testiranja i ispitivanja kvalitete dobivene slike praćeni su parametri ulaznih slika kroz cjelokupni proces, zabilježeni su parametri dobivene slike te su uspoređena vremena izvođenja različitih metoda ključnih za rad algoritma. Također, vizualno su prikazani rezultati eksperimentalnih testiranja kao i rezultati provedene ankete na skupu neovisnih ispitanika za potrebe evaluacije rezultata provedenih eksperimentalnih testova.

Ovaj diplomski rad u drugom dijelu obuhvaća opis izazova implementacije navedenog algoritma za primjenu u ADAS algoritmima, kroz analizu postojećih metoda za detekciju ključnih točaka, osvrt na postojeće algoritme za spajanje slika u stvarnom vremenu te korekciju dobivene slike. U trećem poglavlju prikazana je i objašnjena procedura razvoja vlastitog algoritma za spajanje slika i korekciju osvjetljenja, čiji su rezultati evaluirani u četvrtom poglavlju. Na kraju rada izneseni su zaključci na temelju rezultata testiranja.

2. PROBLEM SPAJANJA SLIKA DOBIVENIH S VIŠE KAMERA U JEDINSTVENU SLIKU

Kako je već naglašeno, algoritmi za spajanje slika dobivenih s više kamera u jedinstvenu sliku razvijeni su u području računalnog vida, a s vremenom su usavršavani te su različitim metodama za rješavanje pojedinih problema sveli algoritam spajanja slika na implementiranje postojećih algoritama i postavljanje pripadajućih kontrolnih parametara implementiranih funkcija. Jedan od primjera važne potrebe za ovakvim algoritmom očituje se u broju prometnih nesreća uzrokovanim nepreglednim mrtvim kutom iza vozila. Povezivanjem slika dobivenih od kamera lijevog i desnog zrcala vozila te lijeve kamere stereo para na stražnjem dijelu vozila, povećava se preglednost mrtvog kuta naspram vidnog područja vozača vozila, kao što je pokazano u radu [1] s prikazom usporedbe pokrivene površine na slici 2.1.



Sl. 2.1. Usporedba pokrivenosti mrtvih kutova predložene metode iz [1] s tradicionalnim pregledom [1]

Usporedbom pokrivenosti površine mrtvih kutova, vidljivo je kako metoda iz [1] povećava preglednost okoline iza vozila, tako što kamerama zrcala obuhvaća šire područje nego bočna zrcala vozila te lijevom kamerom stereo para na stražnjem dijelu vozila obuhvaća širu i pregledniju okolinu nego unutarnje zrcalo vozila.

Sama aktivnost vozača prilikom okretanja glave preko ramena s ciljem pregleda mrtvog kuta, dovodi do toga da dio vremena vozač nema pregled ključnog područja ispred vozila, čime dolazi do vrlo visoke mogućnosti prometne nesreće. Spajanje navedenih slika dobivenih s kamera vozila, osim što osigurava veće područje preglednosti, također uklanja kritično vrijeme za klasičnu

provjeru mrtvog kuta („pogled preko ramena“). Za implementaciju ovakvog algoritma za rad u stvarnom vremenu na ADAS razvojnim pločama, potrebno je osmisliti rješenje za sljedeće probleme: kalibracija kamera te algoritam ispravljanja izobličenja slike, detekcija ključnih točaka i stvaranje deskriptora istih, povezivanje jednakih ključnih točaka na različitim okvirima, proračun matrice homografije te spajanje okvira na temelju dobivenog proračuna i korekcija osvjetljenja na dobivenoj spojenoj slici. Problem detekcije i stvaranja deskriptora ključnih točaka te povezivanje istih na različitim okvirima detaljnije će biti objašnjeno te će se naglasiti dodatni problemi uzrokovani ograničenom računalnom moći ugradbenih računala. Stoga, u radu će se obuhvatiti i drugačiji pristup spajanju slika za primjenu algoritma u ADAS-u.

Kako implementacija algoritma zahtjeva rješavanje više složenih zadataka, u sljedećim podpoglavljima obrađeni su dostupni radovi i postojeće metode koje se bave određenim zadatkom potrebnim za realizaciju cjelovitog zadatka ovog rada, pri čemu je naglašena važnost pojedinog zadatka. Obrađeni su radovi u području računalnog vida te radovi vezani za ADAS algoritme, iz razloga što su potrebna znanja obaju područja, kako bi u potpunosti shvatili probleme i moguća rješenja te razlike u implementaciji navedenog algoritma u području računalnog vida te u području *automotiva*.

2.1. Kalibracija kamere i ispravljanje izobličenja slike

Za potrebe snimanja okoline vozila pomoću senzora za ugradbena računala koristi se CMOS (engl. *Complementary Metal Oxide Semiconductor*) kamera. Prikvačena foto dioda (engl. *Pinned photodiode*) kao struktura foto detektora CMOS kamera omogućava manju količinu šuma, visoku kvantnu učinkovitost, visoku brzinu snimanja, manju potrošnju energije i male iznose tamne struje (engl. *Dark current*) koja protječe kroz foto osjetljivi materijal unutar senzora [2]. Također, u radu [2] objašnjen je proces generiranja elemenata potrebnih za stvaranje slike te razvoj arhitekture CMOS kamera kroz povijest. Nadalje, autor rada [2] upućuje na probleme izrade arhitekture CMOS kamera s prikvačenom foto diodom kao što su: količina naboja koju pojedinačni element može zadržati prije zasićenja (engl. *full-well capacity*) i potpuni prijenos naboja između kondenzatora. Iako u radu nije razrađena selekcija valne duljine osvjetljenja za elemente slike pomoću mikro leća, svjetlosnih vodiča, niza filtera boja i ostalih mikro optičkih struktura, za potrebe razvoja ovog projekta dovoljno je poznavanje rada CMOS kamera i uzrok stvaranja radijalnih i tangencijalnih distorzija na dobivenim slikama.

Ukratko, radijalna distorzija uzrokovana je zakrivljenošću leća kamera, dok tangencijalna distorzija nastaje kada leća kamere i ravnina slike nisu paralelne. Upravo zbog pojave ovakvih

izobličenja, algoritmima za kalibraciju kamere mogu se pohraniti parametri korištene kamere potrebni za ispravljanje navedenih izobličenja. Proračunom intrinzičnih, ekstrinzičnih i distorzijskih koeficijenata, dobivenu sliku sa senzora vozila moguće je ispraviti te ostvariti prikaz točnije aproksimacije stvarne okoline. Metode za ispravljanje izobličenja kamera dijele se u dvije skupine: iterativni proces nelinearne optimizacije te ispravljanje izobličenja pomoću detekcije ključnih točaka [3]. U radu [3] objašnjene su metode te navedene prednosti i mane procedure kalibracije i proračuna distorzijskih parametara, s prioritetom na izvođenje u stvarnom vremenu. Iako navode da ispravljanje izobličenja pomoću detekcije ključnih točaka zahtjeva veći broj kalibracijskih slika i otežava kalibraciju kamera te ispravljanje izobličenja u stvarnom vremenu, za potrebe zadatka diplomskog rada, proračun kalibracijskih parametara potrebno je provesti samo jednome te pohraniti kalibracijske, parametre kako bi jedino ispravljanje izobličenja kamere proveli u stvarnom vremenu. Važnost ključnih točaka i detaljnija diskusija o njima opisani su je u sljedećem podpoglavlju. Nadalje, rad [3] predlaže poboljšanu adaptivnu metodu za korekciju izobličenja kamera s kraćim vremenom proračuna i visokom točnošću. Proračun kalibracijskih parametara osmišljen za rad u stvarnom vremenu temelji se na adaptivnom iterativnom procesu na distorzijskom modelu, gdje ispravlja svaku ključnu točku zasebno te računa distorzijske koeficijente detektirane ključne točke. Zatim bilinearnom interpolacijom i primjenom proračunatog ispravljenog modela rješava problem izobličenja kamera. Provedbom simulacije na 49 ključnih točaka autori su prikazali grafički kalibracijske pogreške i tablicom usporedbe distorzijske pogreške prednosti predložene metode u odnosu na postojeće. Nedostatak se očituje u potrebi za detekcijom ključnih točaka, što zahtjeva idealne koordinate ključnih točaka prilikom stvarne primjene, što autori radom nisu obuhvatili.

2.2. Metode detekcije ključnih točaka i spajanje slike

Osnova algoritma spajanja više slika je pronalaženje i povezivanje jednakih elemenata na slikama za spajanje. Drugim riječima potrebno je odraditi detekciju ključnih točaka svakog *frame*-a te pronaći parove jednakih ključnih točaka na željenim *frame*-ovima za spajanje. Ključne točke su prostorne lokacije ili točke na slici koje se ističu u odnosu na ostatak slike. Njihova važnost očituje se u mogućnošću detekcije iste ključne točke neovisno o afinim transformacijama, poput rotacije slike, izmjene veličine slike, translacije slike te ispravljanja izobličenja matricama homografije i drugim euklidskim transformacijama. Dobro su lokalizirane u prostornoj i frekvencijskoj domeni, čime se smanjuje vjerojatnost poremećaja uzrokovanog izostavljanjem elemenata ili šumom [4]. Za potrebe povezivanja istih ključnih točaka u algoritmima spajanja slika i detekcije objekata, za svaku ključnu točku stvara se i njen deskriptor. Deskriptorima se ključna

točka opisuje na iznimno karakterističan način, čime se ostvaruje mogućnost pronalaženja iste ključne točke u velikoj bazi podataka ključnih točaka unatoč značajnim promjenama uzrokovanim izobličenjem i različitim intenzitetom osvjetljenja. Tri važne metode za detekciju ključnih točaka te izradu njihovih deskriptora su: SIFT (engl. *Scale-Invariant Feature Transform*)[4], SURF (engl. *Speeded Up Robust Features*)[5] i ORB (engl. *Oriented Fast and Rotated Brief*)[6].

Godine 2004-te izdan je rad pod nazivom „*Distinctive Image Features from Scale-Invariant Keypoints*“[4]. Autori predstavljaju metodu SIFT koja uzima zamah razvojem računalnog vida te je do danas usavršena u mnogo segmenata i temelj je gotovo svih radova vezanih za ključne točke, detekciju različitih objekata i sl. Prepoznavanje ključnih točaka i stvaranje deskriptora podijeljeno je u četiri faze: skalarno-prostorna detekcija ekstrema, lokalizacija ključnih točaka, dodjeljivanje orijentacije te stvaranje deskriptora ključnih točaka. U prvoj fazi, primjenom kaskadnog filtriranja i funkcijom Gaussove razlike na različitim veličinama slike, odnosno skalama slike, identificiraju potencijalne ključne točke koje su neovisne o orijentaciji i razlikama u veličini. Zatim detaljnim modelom određuju lokaciju i skalu, gdje se odabiru stabilne ključne točke, a odbacuju one osjetljive na šum. U trećoj fazi ključnim točkama dodjeljuje se orijentacija pomoću orijentacijskog histograma ostvarenog matricom s 36 elemenata (razreda), kojom se obuhvaća 360° okoline ključne točke. Orijentacija pojedinog elementa matrice, koja se dodaje u orijentacijski histogram, određena je veličinom magnitude i kružno-težinskim Gaussovim prozorom. Orijentacija ključne točke određuje se najvišim vrhom histograma orijentacije koji odgovara dominantnim orijentacijama te uz ostale lokalne vrhove koji su iznad 80% visine najvišeg vrha. Završna faza obuhvaća stvaranje deskriptora mjerenjem lokalnih gradijenata slike na odabranoj skali u području svake ključne točke. Rezultati testiranja metode prikazali su značajan napredak u odnosu na ostale algoritme, a s vremenom metoda je usavršena kako od strane prvobitnog autora metode Lowe-a tako i velikog broja članova znanstvene zajednice. Također, u radu je predložena i upotreba metode za implementaciju u algoritmima detekcije objekata, što je omogućilo znatan napredak algoritama za rad u stvarnom vremenu. Detekcijom ključnih točaka na slici snimljenog objekta pomoću kamere i povezivanjem detektiranih ključnih točaka s odgovarajućim ključnim točkama unutar kreirane baze ključnih točaka objekata za prepoznavanje, ostvarili su jednostavnije prepoznavanje objekata i njihovu klasifikaciju.

Na temelju rada [4], kao mjerilo za usporedbu uspješnosti metode detekcije ključnih točaka i stvaranje deskriptora, objavljene je rad pod nazivom „*SURF: Speeded Up Roboust Features*“ [5]. S ciljem ubrzavanja pronalaženja ključnih točaka i stvaranja deskriptora, usredotočili su se na veličinu i kompleksnost samoga deskriptora, a da pri tome svaki deskriptor i dalje ostane

karakterističan u odnosu na ostale. Stavljanjem prioriteta, detekcije ključnih točaka i stvaranje deskriptora, na skaliranje i rotiranje slike, ostvarili su kompromis u pogledu kompleksnosti ključne točke i robusnosti na postojeća izobličenja slike. Isticanjem tvrdnje autora SIFT-a Lowe-a da složenost značajki potpuno neovisnih o afinim transformacijama negativno utječe na robusnost [4], predstavljaju inačicu svoje metode naziva U-SURF (engl. *Upright SURF),* kojom izostavljaju različitost u rotaciji te ostvaruju deskriptor zasnovan jedino na neovisnosti o skaliranju. Odbacivanjem razlike rotacije prilikom stvaranja deskriptora, ostvarili su kraću brzinu izvođenja te veću mogućnost razlikovanja ključnih točaka. SURF-128, kao inačica SURF-a, ostavlja mogućnost proširenja deskriptora za potrebe orijentacije ključnih točaka neovisne o afinim transformacijama. U pogledu fotometrijskih izobličenja, predlažu jednostavni linearni model sa skalarnim faktorom i *offset*-om, što je dovelo do toga da detektor i deskriptor ključnih točaka budu bez boje. Vremenski kraći period za detekciju ključnih točaka u odnosu na SIFT postigli su korištenjem determinante Hessian matrice za odabir i lokacije i skale ključnih točaka. Također, predložili su alternativnu metodu za prostornu i veličinsku analizu primjenom blok filtera u zamjenu za Gaussov filter. Rezultatima testiranja prikazali su napredak u odnosu na tadašnji SIFT te je danas uz dorađeni SIFT, SURF jedan od suvremenih algoritama za detektore i deskriptore ključnih točaka.

Kao alternativna metoda za SIFT i SURF koji su objavljeni kao patentirani algoritmi, godine 2011. u sklopu laboratorija OpenCV (engl. *Open Source Computer Vision Library*) objavljen je rad pod nazivom „*ORB: an efficient alternative to SIFT or SURF“* [6] za slobodnu upotrebu predloženog algoritma. Metoda je zasnovana na FAST detektoru (engl. *Features from Accelerated Segment Test*) [7] i BRIEF (engl. *Binary Robust Independent Elementary Features)* deskriptoru ključnih točaka [8] koji su korišteni zbog odličnih performansi i male potrošnje. Nadogradnjom navedenih metoda ostvarili su puno veću brzinu detekcije ključnih točaka u odnosu na SIFT i SURF. Metodu FAST unaprijedili su dodavanjem točne orijentacije koja prvobitno nije postojala te provođenje detekcije FAST-a s Harris korner mjerom za ključne točke u piramidalnoj skali različitih veličina slike. Izmijenjena metoda s nazivom oFAST (engl. *oriented FAST*) određuje orijentaciju pomoću kutova intenziteta centralnog dijela ključne točke, tako što dodjeljuju vektorsku orijentaciju ključnoj točki na temelju pomaka (eng. o*ffset)* intenziteta kutova centroida. Testiranjem rada metode BRIEF, istaknuli su zašto detektor BRIEF-a loše radi s rotacijama slike te su predložili sljedeće korake kojima su postigli inačicu BRIEF-a naziva rBRIEF (engl. *rotated BRIEF*): pokretanje testova na svim trening zakrpama, kreiranje vektora $\vec{T}$ pomoću sortiranja

testova prema njihovoj udaljenosti u odnosu na proračunati prosjek 0,5 te pohlepnu potragu (engl. *Greedy search*). Algoritam *greedy search* obuhvaća korake prikazane pseudokodom na slici 2.2.

Linija ***Kod***

```
1:  prvi test postaviti u R⃗ i maknuti ga iz T⃗
2:  Sve dok je ukupan broj testova u R⃗ manji od 256 činiti:
3:     Uzeti sljedeći test iz T⃗ i usporediti ga sa svim testovima u R⃗
4:     Ako je apsolutna korelacija testa veća od zadanog praga
5:              Odbaciti test
6:     U suprotnom
7:              Pridodati ga u R⃗
8:     Ako su uzeti svi testovi iz T⃗ i ako R⃗ sadrži manje od 256 testova
9:              Povećati vrijednost praga i ponoviti postupak
```

Sl. 2.2. Pseudokod algoritma *greedy search* [6]

Implementacijom ovih koraka postigli su značajan napredak u pogledu korelacije u odnosu na algoritam *steered* BRIEF. Iako su proveli FAST detektor s Harris korner mjerom na piramidalnoj skali različitih veličina slika, nisu obuhvatili neovisnost ključne točke o navedenoj skali u pogledu dubine znakova i sl. Također, kao i SIFT i SURF, u svome radu predložili su i upotrebu metode za algoritme detekcije s naglaskom na izvođenje u stvarnom vremenu te se uz SIFT i SURF, ORB smatra suvremenom tehnologijom i danas.

Razumijevanjem problema detekcije ključnih točaka s prioritetom na snagu računalne moći i potrebnog vremena za provedbu detekcije te stvaranja deskriptora, jasna je potreba za alternativnim rješenjima prilikom spajanja slika unutar ADAS algoritama. U spomenutom radu [1] koji implementira algoritam spajanja slika u ADAS, predloženo je rješenje pomoću spajanja slika kamera iz zrcala vozila i lijeve kamere stereo para kamera na stražnjem dijelu vozila. Spajanjem dobivenih *frame*-ova s kamera vozila, ostvaruje se dojam u kojemu dobivena slike prikazuje okolinu snimljenu s „virtualnom“ kamerom iza vozila koja obuhvaća veće područje okoline nego što jedna stvarna širokokutna kamera može postići. Proračunom intrinzičnih i ekstrinzičnih parametara svake kamere, transformirali su prikaz svake zrcalne kamere te lijeve kamere stereo para kao virtualne kamere na ravnu površinu udaljenu iza vozila. Također, mapu dispariteta dobivenu pomoću stereo para, primjenom iste transformacije kao na lijevoj kameri stereo para, ostvarili su prikaz mape dispariteta na istoj ravnoj površini iza vozila. Mapom dispariteta određuje se dubina prostora iza vozila te omogućuje pozicioniranje elemenata slike s

preciznijom aproksimacijom koordinata u odnosu na stvarnu snimljenu okolinu. Algoritmom detekcije šavova i mapom dispariteta uspjeli su odrediti optimalne minimalno zahtjevne linije šavova kojima spajaju susjedne prikaze. Nedostatak ovakvog algoritma očituje se u gubitku značajnog dijela slike jer prikaz s virtualne kamere izostavlja dijelove okoline koji su obuhvaćeni kamerama u bočnim zrcalima vozila. Izostavljeni dio odnosi se na područje od vanjskih zrcala do stražnjeg dijela vozila, jer upravo ti dijelovi su izostavljeni kako bi se prikaz tih kamera reproducirao kao prikaz virtualne kamere iza vozila. Algoritam je razvijen za upotrebu na TDA3x SoC-u (engl. *TDA3x System on Chip)* te pomoću četiri sirova Bayer senzora slika (engl. *raw Bayer image sensor*) rezolucije 1280 x 720 elemenata slike i brzine snimanja 30 FPS (engl. *frames per second*). Rezultati testiranja implementiranog algoritma predloženog u [1], prikazali su napredak u odnosu na klasične prikaze kamera iza vozila u pogledu pokrivenosti i uglađenosti krajnje slike rezolucije 1929 x 480 te očuvanju brzine snimanja u iznosu 30 FPS. Kako je korištena tehnologija za razvoj algoritma predloženog u [1] naprednija od korištene tehnologije prilikom razvoja zadatka diplomskog rada, objasniti će se razlika korištenih tehnologija u trećem poglavlju. Nadalje, kako su predloženim algoritmom u [1] postigli zavidne rezultate te zbog razlike u korištenim tehnologijama za razvoj algoritma, diplomskim radom veći prioritet staviti će se na usporedbu navedenog algoritma u području računalnog vida i *automotiva*. Također objasniti će se zašto implementacija detekcije ključnih točaka s deskriptorima nije isplativa za ugradbena računala vozila. U trećem poglavlju objasniti će se i različita rješenja za spajanje više slika dobivenih sa senzora vozila u jedinstvenu sliku te korekciju intenziteta osvjetljenja dobivene spojene slike.

Pristup spajanja više slika dobivenih s kamera vozila u jedinstvenu sliku objašnjen u radu [9] predlaže prosljeđivanje prikupljenih slika kamera kontrolnome centru za obradu podataka te primanje obrađene spojene slike za potrebe prikaza vozaču. Ističu probleme algoritma spajanja slika s pokretnim kamerama, razlike u intenzitetu osvjetljenja kamera i dinamično kretanje objekata u snimanoj sceni. Metoda spajanja slika predložena za implementaciju na računala s grafičkim karticama i visokim performansama, započinje sa SURF detektorom ključnih točaka. Dobivene deskriptore koriste za povezivanje istih ključnih točaka KNN (engl. *K-nearest neighbors*) metodom. Spojenim parovima ključnih točaka i RANSAC (engl. *RANdom SAmple Consensus*) metodom proračunavaju matricu homografije u prostornoj domeni, a modelom pomičnog prosjeka (engl. *moving average model*) na prijašnjim i trenutnoj matrici homografije ostvaruju uglađenost matrice homografije u vremenskoj domeni. KNN i RANSAC metode detaljnije će biti objašnjene u nastavku rada. Spajanje slika izvršeno je cilindričnim zavrtanjem (engl. *cylindrical warping*). Brzinu izvođenja za potrebe rada u stvarnom vremenu postižu

spajanjem slika blok po blok, umjesto element po element slike, korištenjem GPU SURF-a sa zadanim kernelom radi ubrzavanja detekcije i izvršavanja KNN metode te ubrzavanjem rada sustava paralelnim programiranjem pomoću CUDA (engl. *Compute Unified Device Architecture*) platforme. Testiranjem algoritma na GTX 970 *mini* grafičkoj kartici prikazali su uglađeniji prijelaz na šavovima spojenih slika. Problem algoritma očituje se u potrebi za visokom računalnom moći koja je neophodna za obradu prikupljenih podataka u stvarnome vremenu. Visoka računalna moć potrebna za izvođenje predloženog algoritma iz [9], idejno je omogućena pomoću komunikacije vozila s infrastrukturom (engl. *Vehicle to Infrastructure*, V2I), pri čemu vozilo prosljeđuje prikupljene podatke centru za obradu podatak, a zatim obrađene podatke vozilo prima od centra za obradu podataka. Predložena komunikacija između vozila i infrastrukture trenutno je u začetku razvoja i susreće se s mnogo problema pri komercijalnoj implementaciji u prometu. Navedeni razlozi stoga onemogućuju implementaciju predloženog algoritma iz [9], u trenutnoj fazi automobilske industrije, za potrebe obrade prikupljenih podataka sa senzora vozila u stvarnome vremenu pomoću ugradbenih računala.

2.3. Korekcija osvjetljenja

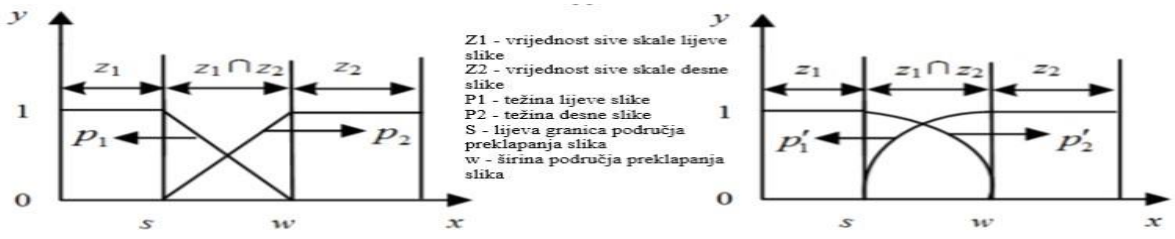
Iako korekcija osvjetljenja dobivene slike ne predstavlja veliku važnost te ne pridonosi značajnome poboljšanju performansi ostalih algoritama, implementira se povodom potrebe za što kvalitetnijim prikazom spojene slike vozaču tijekom vožnje. Slikama 2.3. preuzetim iz [10], prikazan je rezultat spajanja slika s područjem preklapanja, pri čemu se različiti intenzitet osvjetljenja slika za spajanje nije ispravio.

a) b)

c)

Sl. 2.3. Prikaz rezultata spajanja slika bez korekcije osvjetljenja slika za spajanje: (a) lijeva slika s vlastitim osvjetljenjem (b) desna slika s vlastitim osvjetljenjem (c) spojena lijeva i desna slika bez korekcije osvjetljenja [10]

Poboljšanjem vizualne kvalitete osigurava se koncentracija vozača, bez distrakcija na značajne razlike intenziteta osvjetljenja u spojenoj slici, čime se umanjuje rizik krive procjene vozača. Primjena korekcije osvjetljenja na spojenim slikama prvobitno obuhvaća uglađeni prijelaz intenziteta osvjetljenja na šavovima spojenih slika, odnosno na području preklapanja okoline snimljene različitim kamerama. Postojeće algoritme za korekciju osvjetljenja na šavovima slike poput algoritma postupne fuzije (engl. *gradual fusion algorithm*) i algoritma prosječno težinske fuzije (engl. *weighted average fusion*), detaljnije se može proučiti u radu [10]. U radu predlažu poboljšanu inačicu *gradual fusion algoritma.* Naglašavaju nedostatke linearnog *weighted average fusion* algoritma u segmentu lošeg prijelaza intenziteta osvjetljenja na šavovima spojene slike, jer se prijelaz temelji na direktnoj primjeni prosjeka težina elemenata odabrane referentne slike na spojeni dio slike. Odabiru *gradual fusion* algoritam kao osnovu za njihov prijedlog te navedeni algoritam detaljnije opisuju objašnjavanjem matematičkih formula potrebnih za ostvarivanje parametara korekcije osvjetljenja. Predlažu novi proračun istih parametara drugačijim pristupom. G*radual fusion* algoritam računa intenzitet osvjetljenja sivih elemenata slike na području preklapanja spojene slike pomoću linearnih težinskih formula te parametara širine područja preklapanja spojene slike i udaljenosti elementa slike za koji se računa u odnosu na rubove područja preklapanja. Za razliku od *Gradual fusion* algoritma, poboljšani algoritam iz [10] određuje navedeni intenzitet u prvom kvadrantu jednadžbe kružnice, čime trend težine osvjetljenja na području preklapanja ostvaruje uglađeniji prijelaz. Grafičkim prikazom trenda težine osvjetljenja na spojenoj slici *gradual fusion* algoritma i njihove predložene inačice, teorijski su prikazali i objasnili poboljšanje, što je jasno vidljivo i na slikama 2.4 i 2.5. preuzetim iz [10].



a) b)

Sl. 2.4. Usporedba prijelaza trenda težine osvjetljenja: (a) prijelaz trenda težine osvjetljenja ostvaren *gradual fusion* algoritmom (b) prijelaz trenda težine osvjetljenja ostvaren *improved gradual fusion* algoritam [10]

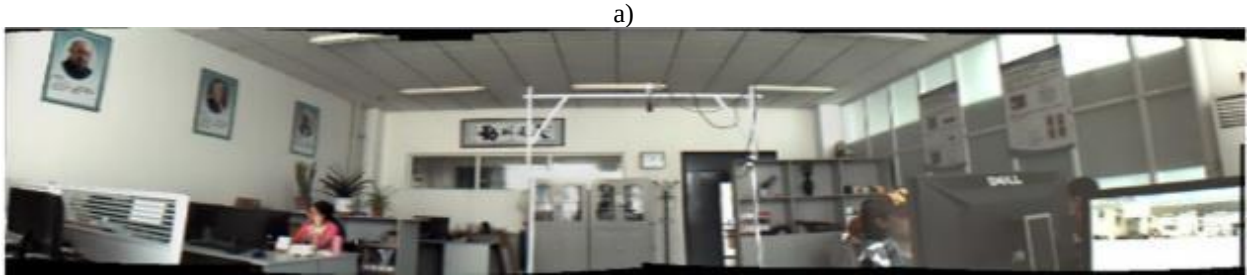
a) b)

Sl. 2.5. Usporedba rezultata ispravljanja osvjetljenja na spojenoj slici: (a) ispravljanje osvjetljenja spojene slike *gradual fusion* algoritmom (b) ispravljanje osvjetljenja spojene slike *improved gradual fusion* algoritmom [10]

Testiranjem oba algoritma na spojenim slikama sličnog intenziteta te spojenim slikama s većom razlikom intenziteta osvjetljenja, tabličnim prikazom usporedili su dobivene rezultate težinskih gradijenata. Težinski gradijenti predstavljaju vrijednosti težine osvjetljenja u sredini područja preklapanja slika, odnosno vrijednosti na točki sjecišta težina osvjetljenja p_1 i p_2 pravaca, prikazanih u grafovima na slici 2.4. Dobiveni gradijenti predloženog algoritma u testnom slučaju sa sličnim intenzitetom osvjetljenja spojenih slika neznatno su manjeg iznosa, što predstavlja uglađeniji ali i dalje gotovo jednaki prijelaz kao u *gradual fusion* algoritmu. U testnom slučaju s većom razlikom intenziteta osvjetljenja ostvarili su značajno manje vrijednosti težinskih gradijenata, što je omogućilo kvalitetniji i uglađeniji prijelaz intenziteta na šavovima spojenih slika u odnosu na *gradual fusion* algoritam. Nedostatak predloženog algoritma odnosi se na drugo shvaćanje algoritama korekcije intenziteta osvjetljenja spojenih slika, što je korekcija cijele slike u odnosu na odabranu referentu sliku iz skupa slika za spajanje. Ovo shvaćanje podrazumijeva ispravljanje osvjetljenja cjelokupne slike koja se spaja s referentnom, a ne samo u području preklapanja slika, odnosno na šavovima spojene slike.

Radom [11] predlaže se metoda korekcije osvjetljenja cjelokupne spojene slike na osvjetljenje odabrane referentne slike iz skupa slika za spajanje. Osvrću se na različite metode za proračun razlike intenziteta osvjetljenja poput *edge blending* i *weighted average*. Predlažu metodu u kojoj se intenzitet osvjetljenja svakog elementa slike za spajanje računa kao suma trenutne vrijednosti elementa s prosječnom razlikom intenziteta osvjetljenja parova spojenih ključnih točaka u području preklapanja slika za spajanje. Prosječna razlika intenziteta osvjetljenja parova ključnih točaka cjelobrojni je koeficijent s vrijednošću u rasponu [-255, 255] za predloženi proračun razlike komponente V u formatu boje HSV (engl. *Hue, Saturation, Value*). Raspon vrijednosti razlike intenziteta u negativnom dijelu skupa vrijednosti predstavlja slučaj u kojemu je slika za spajanje većeg intenziteta osvjetljenja nego referentna slika s kojom se spaja. Sukladno tome, pozitivni raspon vrijednosti predstavlja slučaj u kojemu je slika za spajanje nižeg intenziteta osvjetljenja nego referentna slika s kojom se spaja. Ovisno o slučaju upotrebe (engl. *UseCase*), slici za spajanje potrebno je povećati ili smanjiti vrijednost komponente V, pazeći na prekoračenje vrijednosti komponente V, čija vrijednost obuhvaća raspon [0, 255]. Vizualnim prikazom eksperimentalnih rezultata priloženih na kraju rada, može se ustanoviti da su jednostavnim pristupom riješili problem korekcije intenziteta osvjetljenja na šavovima spojenih slika, ujednačili intenzitet osvjetljenja cjelokupne spojene slike te ostvarili oku ugodniji prikaz dobivenog rezultata, kao što je prikazano slikama 2.6. preuzetim iz [11].

a)



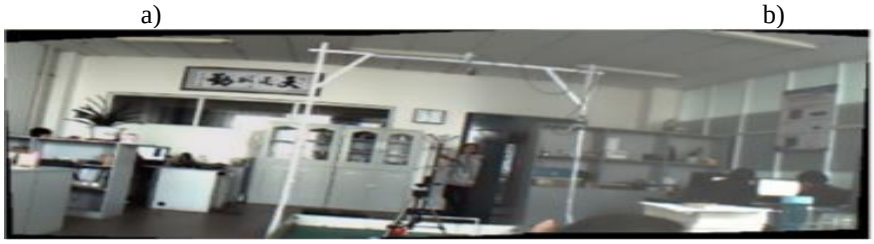
b)

Sl. 2.6. Usporedba spojenih slika: (a) spojena slika bez korekcije osvjetljenja (b) spojena slika s korekcijom osvjetljenja primjenom metode iz [11]

Nedostatak predložene metode očituje se u slučaju upotrebe u kojemu su razlike intenziteta osvjetljenja spojenih parova ključnih točaka visoke, čime proračunati koeficijent s visokom apsolutnom vrijednošću, dodavanjem na svaki element slike za spajanje na većem području dosežu vrijednosti 0 ili 255. Prikaz dobivene slike navedenog slučaja upotrebe razotkriva nekvalitetno osvjetljenje cjelokupne slike, jer visokim apsolutnim vrijednostima proračunatog koeficijenta svaki element slike za spajanje ne prihvati potpunu razliku intenziteta osvjetljenja već se ograničava na raspon komponente V, čime se svaki element slike ne promjeni za jednaku vrijednost. Slikom 2.7. prikazan je slučaj u kojemu su slike za spajanje visoke razlike intenziteta osvjetljenja, što uzrokuje prividno zamućenje na slici za spajanje, kojoj se osvjetljenje ispravlja.

a)

b)



c)

Sl. 2.7. Rezultat spajanja slika visoke razlike intenziteta slika za spajanje: (a) slika za spajanje s visokim intenzitetom osvjetljenja (b) slika za spajanje s niskim intenzitetom osvjetljenja (c) spojena slika s ispravljenim osvjetljenjem metodom iz [11]

2.4. Diskusija o postojećim metodama spajanja slika za primjenu u ADAS algoritmima

Osvrtom na radove u prijašnjem poglavlju obuhvaćene su različite predložene ideje kojima implementacija algoritma spajanja više slika snimljenih pomoću kamera i korekcija osvjetljenja može ostvariti visoke rezultate, kako u pogledu detekcije i povezivanja istih ključnih točaka, tako i u pogledu uglađenog i kvalitetnog prikaza dobivene spojene slike.

Predložene metode za detekciju i stvaranje deskriptora ključnih točaka omogućuju efikasno spajanje više slika, ali su ograničene potrebom za visokom računalnom moću i vremenom izvođenja na ugradbenim sustavima. Radom [9] predloženo je rješenje implementacije algoritma za ugradbena računala vozila, u kojemu je problem izvođenja detekcije ključnih točaka i stvaranje deskriptora u stvarnom vremenu eliminiran izvođenjem na računalima visoke računalne moći. Idejni prijedlog komunikacije sustava vozila s navedenim računalima, ostvaren pomoću „*Vehicle to infrastructure"* komunikacije, otvara mogućnost prosljeđivanja prikupljenih podataka sa senzora vozila, centrima za obradu podataka visoke računalne moći. Implementacija idejne komunikacije vozila s infrastrukturom, omogućila bi izvršavanje kompleksnih algoritama za obradu prikupljenih podataka sa senzora vozila u stvarnome vremenu. Problem predložene ideje očituje se u trenutnoj nemogućnosti komercijalne implementacije takve infrastrukture i ograničenoj brzini komunikacije vozila s infrastrukturom što ograničava izvođenje u stvarnome vremenu. Stoga metode detekcije ključnih točaka i stvaranja deskriptora trenutno nije efikasno implementirati za ugradbena računala vozila.

Također, više različitih algoritama predloženo je za korekciju osvjetljenja koji ostvaruju željeni rezultat, ali ne obuhvaćaju svaki smisao algoritma korekcije osvjetljenja. Nedostatak visoke računalne moći ugradbenih računala vozila, za detekciju i deskripciju ključnih točaka prilikom spajanja slika, može se nadomjestiti koristeći algoritme namijenjene ugradbenim računalima vozila. Algoritmima za primjenu u ADAS, može se odrediti položaj elemenata primljene slike s kamera vozila pomoću mape dispariteta te intrinzičnih, ekstrinzičnih i distorzijskih parametara kamera. Poznavanjem lokacije elemenata slike, metodama transformacije mogu se ispraviti slike odnosno položaji elemenata slike, čime se ostvaruje preciznija aproksimacija stvarne snimljene okoline pomoću senzora vozila.

Iako diplomski rad cilja na implementaciju algoritma za primjenu u ADAS, spajanje slika provest će se spomenutim metodama za detekciju ključnih točaka i spajanje istih, kako bi prikazali potrebno vrijeme izvođenja na razvojnom računalu. Nadalje, algoritam korekcije osvjetljenja

osmišljen za rad u ADAS algoritmima primjenjuje ideju rada [11]. Vizualnim prikazom eksperimentalnih testova te provedbom ankete, na skupu neovisnih ispitanika, za potrebe ocjenjivanja kvalitete spojenih slika i korekcije osvjetljenja rezultata eksperimentalnih testova, prezentirane su razlike u kvaliteti rezultata implementiranih algoritama.

3. ALGORITAM SPAJANJA VIŠE SLIKA DOBIVENIH S KAMERA U JEDINSTVENU SLIKU I KOREKCIJA OSVJETLJENJA DOBIVENE SLIKE

Zadatak diplomskog rada, s ciljem razvoja ADAS algoritma za spajanje više slika dobivenih s kamera u jedinstvenu sliku i korekciju osvjetljenja dobivene slike, obuhvaća sljedeće korake:

- upotreba ADAS razvojne ploče i pripadajućeg programskog okruženja za razvoja rješenja;
- implementacija algoritma spajanja slika dobivenih s kamera vozila u jedinstvenu sliku;
- implementacija algoritma korekcije osvjetljenja dobivene spojene slike;
- proračun vremena izvođenja pojedinih algoritama te evaluacija rezultata testiranja.

Vrijeme izvođenja pojedinih algoritama detekcije ključnih točaka, prikazat će se s ciljem potvrde tvrdnje da algoritmi detekcije ključnih točaka nisu isplativi za potrebe ADAS algoritama, kako je spomenuto u radu. Realizacijom i detaljnim objašnjenjem koraka pri implementaciji rješenja u sljedećim podpoglavljima te prikazom potrebnog vremena izvođenja pojedinih algoritama, usporediti će se implementacija SIFT, SURF i ORB algoritama za detekciju i stvaranje deskriptora ključnih točaka. Nadalje, objasnit će se ograničena upotreba navedenih algoritama za implementaciju na ugradbene sustave vozila. Objasniti će se i prikazati implementacija algoritma povezivanja istih ključnih točaka pomoću predložene metode autora SIFT-a Lowe-a i KNN algoritma. Također, ispraviti će se različiti intenzitet osvjetljenja na okvirima za spajanje primjenom predložene metode u [11] za formate boja YUV i HSV te će se prikazati i objasniti dobiveni rezultati. Sljedećim podpoglavljima objasniti će se svaki korak implementacije rješenja, uz detaljnije teorijsko i praktično objašnjenje. Također, opisana je arhitektura i programsko okruženje potrebno za razvoj algoritma.

3.1. ADAS ALPHA razvojna ploča

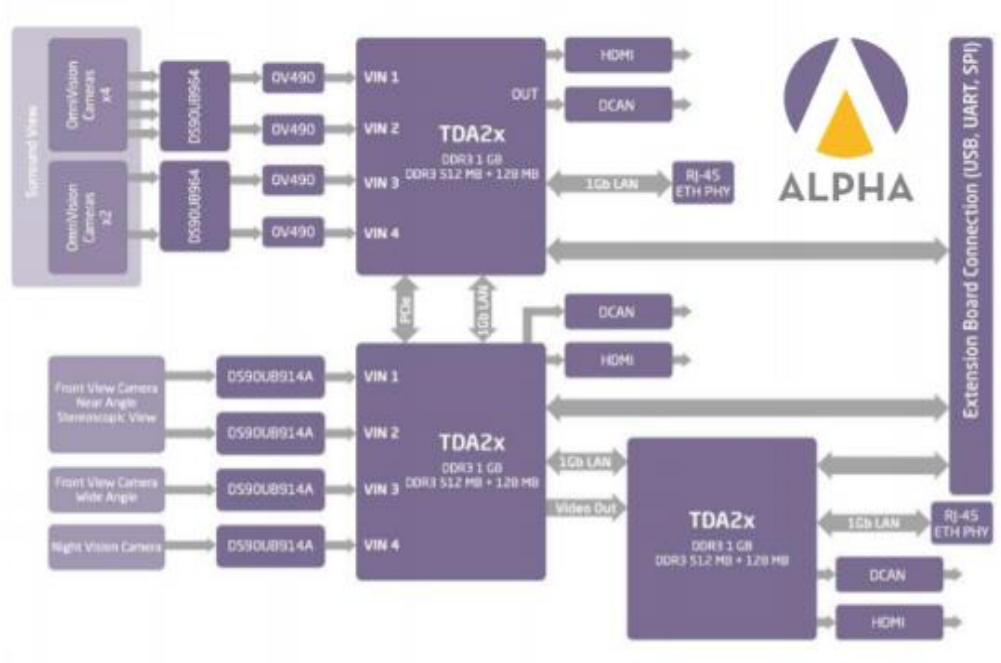
ADAS elektronički sustavi dizajnirani su za pomoć vozaču prilikom vožnje. Raspon pomoći vozaču proteže se od prikaza podataka okoline dobivenih sa senzora vozila, do preuzimanja kontrole nad vozilom u kritičnim situacijama. Razlika ADAS u odnosu na pasivne sustave sigurnosti očituje se u direktnoj intervenciji na aktivnosti vožnje obradom podataka iz okoline vozila [12]. ADAS pruža dodatne informacije o okolini vozaču, upozorava na prepreke i opasnosti u prometu, preuzima kontrolu u kritičnim situacijama te omogućava međusobnu komunikaciju

vozila kao i komunikaciju vozila s infrastrukturom. Potrebu za ADAS uzrokovao je sve veći broj nesreća u prometu uzrokovan pogrešnom reakcijom vozača ili krivom procjenom okoline. ADAS osigurava veću kontrolu prometa senzorima poput radara, lidara, kamera, ultra soničkim senzorima i sl. Tvrtke Bosch, Texas Instruments, Continental i druge, pružaju različite inačice komercijalnih i razvojnih ADAS ploča. Komercijalne ADAS ploče postaju standardna oprema vozila jer značajno povećavaju sigurnost prometa te omogućuju korak bliže autonomnoj vožnji, što je stvorilo potrebu za kontrolu kvalitete razvijenih ADAS [13]. Također, za potrebe kontrole razvijenih algoritama ustanovljene su različite norme i standardi, poput ISO 26262 standarda za kontrolu sigurnosti kritičnih komponenti vozila.

ADAS razvojna ploča korištena za razvoj rješenja zadatka diplomskog rada je ADAS ALPHA. Razvijena ploča od strane istraživačko-razvojnog instituta i tvrtke RT-RK s Texas Instruments-om, pruža podršku za osnovne i napredne sustaveupozorenja, aktivne sustave upravljanja i polu-autonomne operacije. ADAS ALPHA ploča pruža podršku pomoću triju TDA2SX sustava na čipu (engl. *System on a Chip, SoC*), razvijenih od tvrtke Texas Instruments. SCV SoC (engl. *Surround Camera View*) podržava do šest ulaza za uskokutne i širokokutne kamere. SCV SoC koristi se za prikaz okoline vozila, detekciju objekata u okolini vozila i upozorenje na opasnost iz okoline. FFN SoC (engl. *Front view camera near angle stereoscopic view, Front view camera wide angle, Night vision camera*) podržava do četiri ulaza za uskokutne i širokokutne kamere i koristi se za snimanje stanja ispred vozila, u svrhu procjene udaljenosti vozila i drugih objekata ispred, procjene kretanja vozila ispred i detekcije prepreka na putu. FUS SoC (engl. *Fusion*), ne pruža podršku za sustav kamera, već se koristi za obradu informacija navedenih SoC-ova pomoću algoritama za točnost podataka. Svaki SoC sadrži četiri dvojezgrena procesora opće namjene, od kojih dva procesora imaju ARM Cortex M4, te ARM Cortex A15 arhitekturu jezgre. Dva procesora su Texas Instruments C66x DSP (engl. *Digital Signal Processor*) arhitekture jezgre za digitalnu obradu video signala. Nadalje, ploča sadrži i četiri procesorske jedinice EVE (engl. *Embedded Vision Engine*) za obradu video signala niskog i srednjeg opterećenja. Ploča sadrži i VPE (engl. *Video Processing Engine*) odjeljak za obradu video sadržaja, poput skaliranja veličine i promjene formata boje. Za međusobnu komunikaciju SoC-ova ploča sadrži sabirnice za brzu komunikaciju: Ethernet, USB, CAN, SPI i sl. Za komunikaciju s razvojnim računalom, ploča pruža UART (engl. *Universal Asynchronous Receiver/Transmitter*) i *Ethernet* sučelja. Komunikacija sa sustavom kamera te prihvaćanje i obrada pristiglih podataka omogućeni su pomoću I2C (engl. *Inter-Integrated Circuit*) sučelja. Prikaz video sadržaja omogućen je pomoću tri HDMI izlaza, po jednim za svaki SoC. Također, za svaki SoC na ploči nalaze se i MicroSD konektori, korišteni s

microSD karticom za pokretanje sustava određenog SoC-a te kao spremnik za pohranu. Na slici 3.1. prikazana je korištena ADAS ALPHA razvojna ploča te arhitektura ADAS ALPHA ploče kao i veze između pojedinih SoC-ova [14].

a)



b)

Sl. 3.1. ADAS ALPHA razvojna ploča: (a) fizički izgled ADAS ALPHA razvojne ploče (b) arhitektura ADAS ALPHA razvojne ploče s prikazanim vezama između SoC-ova [14]

3.2. Programsko okruženje i biblioteke korištene za razvoj rješenja

Za potrebe razvoja rješenja zadatka diplomskog rada korištene su različite tehnologije. Razvoj rješenja za primjenu na ADAS ALPHA, omogućen je pomoću *Code Composer Studio*-a. *Code Composer Studio* razvijen od tvrtke Texas Instruments, integrirano je razvojno okruženje (engl. *integrated development environment,* IDE) koje podržava portfolio *Texas Instruments*-ovih mikrokontrolera i ugrađenih procesora. *Code Composer Studio* sadrži niz alata koji se koriste za razvoj i uklanjanje pogrešaka ugrađenih aplikacija. Uključuje optimizacijski C/C ++ kompajler, uređivač izvornog koda, okruženje za izgradnju projekata, program za ispravljanje pogrešaka i mnoge druge značajke. Intuitivni IDE pruža jedno korisničko sučelje koje korake toka razvoja aplikacija. *Code Composer Studio* kombinira prednosti softverskog okvira *Eclipse* s naprednim ugrađenim mogućnostima otklanjanja pogrešaka, što rezultira uvjerljivim razvojnim okruženjem bogatim značajkama za ugrađene programe [15]. *Code Composer Studio* nadograđen s VisionSDK programskim razvojnim okruženjem objašnjenim u sljedećem podpoglavlju.

Nadalje, za potrebe pisanja izvršnih skripti u *Python* programskom jeziku, pomoću *Python install*-era osposobljeno je radno okruženje s *Python 3.8.5* inačicom, pripadajućim integrirano-razvojnim i edukacijskim okruženjem (engl. *integrated development and learning environment,* IDLE) te upraviteljem *Python* paketa (engl. *PIP Install Packages,* PIP)[16]. Također, koristeći PIP, nadograđen je *Python* s paketima biblioteke *OpenCV, numPy, PIL i imutils,* čije detaljnije objašnjenje slijedi u sljedećem podpoglavlju.

3.2.1. VisionSDK

Za implementaciju algoritama i *UseCase*-ova kao funkcionalnosti ploče, koristi se VisionSDK (engl. *Software Development Kit*) programsko razvojno okruženje. Kao više-procesorsko programsko razvojno okruženje, omogućuje kreiranje različitih tokova podataka ADAS aplikacija, poput obrada snimljenog ili preuzetog video sadržaja, analize video sadržaja i prikaz video sadržaja. Podržava istovremeno izvršavanje na različitim procesorskim jedinicama. Zasnovan je na okviru pod nazivom „*Links and Chains*“, generira strukturu *UseCase*-a i uspostavlja komunikaciju i tok podataka za definirani *UseCase*. Pruža i korisničko sučelje „*Link API*“ (engl. *Application Programming Interface*) okvira.

Links and Chains okvir koristi se pri definiranju *UseCase* -a unutar VisionSDK. Svaki *Link* izvršava se kao zasebna nit i posjeduje jednu ili više ulaznih/izlaznih konekcija te vlastiti poštanski sandučić pomoću kojega komunicira s ostalim *Linkovima*. Jednoznačno određuje algoritam, hardversku vezu na ploči, obradu signala i sl. Izvođenje pojedinog *Linka* omogućeno je

povezivanjem u *Chain*, pri čemu broj izlaznih konekcija i tip spremnika jednog *Linka* moraju odgovarati broju ulaznih konekcija i tipu spremnika sljedećeg *Linka*. Glavna namjena ove arhitekture je omogućiti lakše stvaranje *UseCase*-a i znatno ubrzavanje razvijanja algoritma. Također VisionSDK pruža i *UseCase* generator kojim se generira skup potrebnih datoteka nužnih za rad *UseCase*-a, čime se skraćuje potrebno vrijeme za razvoj. Nadalje, generator detektira i izvještava o pogreškama prilikom razvoja *UseCase*-a te omogućuje jednostavniju promjenu toka podataka i veza tijekom razvoja.

3.2.2. Programske biblioteke

OpenCV (engl. *Open Source Computer Vision Library*) softverska je biblioteka za računalni vid i strojno učenje otvorenog koda. *OpenCV* je izgrađen kako bi pružio zajedničku infrastrukturu za programe računalnog vida i ubrzao upotrebu percepcije strojeva u komercijalnim proizvodima. *OpenCV* biblioteka ima više od 2500 optimiziranih algoritama, što uključuje sveobuhvatan skup klasičnih i najsuvremenijih algoritama računalnog vida i strojnog učenja. Ovi se algoritmi mogu koristiti za otkrivanje i prepoznavanje lica, prepoznavanje objekata, klasificiranje ljudskih radnji u videozapisima, praćenje kretanja kamere, praćenje objekata u pokretu, izvlačenje 3D modela predmeta, izradu 3D oblaka točaka iz stereo kamera, spajanje slika i sl. *OpenCV* ima više od 47 tisuća korisnika i procijenjeni ukupni broj preuzimanja od trenutka nastanka koji premašuje 18 milijuna. *OpenCV* biblioteka intenzivno se koristi u tvrtkama, istraživačkim skupinama i sl. [17] .

NumPy je projekt s bibliotekom otvorenog koda čiji je cilj omogućiti numeričko računanje unutar *Python* programskog jezika. Razvijena biblioteka 2005. godine, nadovezuje se na rani rad numeričkih biblioteka i biblioteka numeričkih polja. *NumPy* je razvijen i omogućen putem *GitHub*-a, kroz konsenzus *NumPy*-a i šire znanstvene zajednice *Python*-a [18].

Imutils biblioteka niz je praktičnih funkcija za olakšavanje osnovnih funkcija obrade slika, kao što su translacija, rotacija, promjena veličine slike, prikazivanje *Matplotlib* slika, sortiranje kontura, otkrivanje rubova i još mnogo toga [19].

PIL (engl. *Python Imaging Library*) omogućuje obradu slika unutar *Python* interpretera. Biblioteka pruža opsežnu podršku različitih formata slika, učinkovitu reprezentaciju slika te visoku moć obrade slika. Biblioteka je dizajnirana za brzi pristup podatcima pohranjenim u različitim formatima boja te pruža čvrst temelj alatima za obradu slika [20].

3.3. Razvoj programskog rješenja za spajanje okvira dobivenih s više kamera u jedinstveni okvir te korekciju osvjetljenja dobivenog okvira

U ovome je podpoglavlju detaljnije objašnjen svaki korak prilikom razvoja rješenja za spajanje više okvira s područjem preklapanja dobivenih s kamera razvojnog sustava, pri čemu se opisuju korištene funkcije, formati boja i potrebne matematičke formule pojedinih algoritama.

3.3.1. Postupak kalibracije kamera

Iako VisionSDK pruža alat za kalibraciju kamera, odnosno proračun kalibracijskih parametara kamere, prilikom korištenja alata susreće se problem ispravljanja slika postojećom funkcijom *remap*, čije izvođenje je potrebno provesti kako bi se ispravilo izobličenje slike. Problem se očituje prilikom interpolacije lokacija elemenata slike, pri čemu funkcija *remap* može uspješno ispraviti izobličenje te ostvariti ispravljenu sliku prikazanu u razinama sive boje, ali ne može izvršiti interpolaciju elemenata u boji te ostvariti ispravljenu sliku u boji, a što je potrebno za zadatak diplomskog rada. Nemogućnost interpolacije elemenata u boji vidljiva je nedostatkom kontrole U i V kanala formata boje YUV420 unutar funkcije *remap*, što je u istoimenoj funkciji novijih inačica VisionSDK implementirano. Kako VisionSDK svojim inačicama prati razvoj TDA SoC-a te pri razvoju zadatka diplomskog rada nije moguće koristiti inačice VisionSDK u kojima je funkcijom *remap* omogućeno ispravljanje slika u YUV420 formatu boje, postupak kalibracije kamera izvršiti će se pomoću *Python* skripte. Također, implementacija razlike pojedinih funkcija *remap* nije moguća jer se izvršavanje navedene funkcije ostvaruje preko izvršnih datoteka VisionSDK, čiju nadogradnju programskog okruženja (engl. *rebuild*) nije moguće realizirati zbog nedostatka licence potrebnih programa za nadogradnju. Nadalje, kako nije moguća interpolacija elemenata slike u boji funkcijom *remap*, prilikom razvoja rješenja implementirat će se algoritam korekcije osvjetljenja slika za korištenje u ADAS algoritmima, dok će se spajanje slika provesti korištenjem algoritama računalnog vida.

Postupak kalibracije kamera započinje implementacijom *UseCase*-a unutar VisionSDK programskog razvojnog okruženja. Kreira se novi direktorij te se u tekstualnoj datoteci definira tok i veze *UseCase*-a između pojedinih procesora. Pozicioniranjem unutar terminala (engl. *Command Prompt*, CMD) u direktoriji novo stvorenog *UseCase*-a i pokretanjem naredbe „*vsdk_win64.exe -file -img -path*“, s parametrima putanje novo stvorenog direktorija te naziva tekstualne datoteke, generiraju se potrebne datoteke za implementaciju *UseCase*-a. Nakon dobivanja datoteka za implementaciju *UseCase*-a, sljedeći korak podrazumijeva pisanje samog

algoritma *UseCase*-a. Kako *UseCase*-om treba pohraniti videozapis na razvojno računalo putem *Ethernet* modula, u kojemu se kamerom snima kalibracijski panel, potrebno je omogućiti:

- mrežnu komunikaciju ADAS ALPHA razvojne ploče i razvojnog računala;
- tok podataka snimljenih s jednom uskokutnom kamerom sustava;
- kompresiju videozapisa za pohranu na razvojno računalo;
- prikaz snimanog područja za kontrolu procedure snimanja kalibracijskog panela;

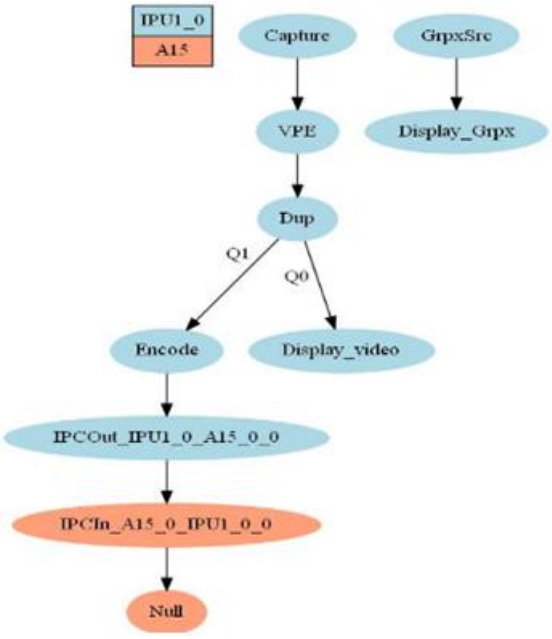
Stoga, tekstualna datoteka potrebna za generiranje datoteka ovog *UseCase*-a napisana je kao na slici 3.2. Rezultat pokretanja naredbe za generiranje datoteka *UseCase*-a skup je potrebnih datoteka nužnih za implementaciju *UseCase*-a na razvojnu ploču, te prikaz toka podataka i veza, prikazan na slici 3.3.

```
UseCase: chains_ffn_camCalib

Capture ->  VPE -> Dup -> Display_video
                   Dup -> Encode -> Null (A15)


GrpxSrc -> Display_Grpx
```

Sl. 3.2. Prikaz tekstualne datoteke potrebne za generiranje *UseCase*-a za pohranu snimanog videozapisa kamerom ADAS ALPHA razvojne ploče na razvojno računalo



Sl. 3.3. Prikaz toka podataka i veza *UseCase*-a za pohranu snimanog videozapisa kamerom ADAS ALPHA razvojne ploče na razvojno računalo

Capture linkom prihvaćaju se podaci prikupljeni pomoću senzora, odnosno CMOS kamere spojene na ADAS ALPHA razvojnu ploču. Podaci se prosljeđuju pomoću jednog reda s jednim kanalom u *VPE* link, kojim se postavlja format boje na YUV420SP. Format boje YUV420SP s prikazom pod-uzorkovanja na slici 3.4. [21], omogućuje jednostavniju i bržu pohranu te prosljeđivanje parametara između procesora ugradbenih sustava, u odnosu na ostale formate kao što su RGB i HSV.

Sl. 3.4. Prikaz pod-uzorkovanja YUV420SP formata boje [21]

Promjenom formata boje, tok podataka prosljeđuje se na *DUP* link, kojim se primljeni red s jednim kanalom duplicira u dva reda, svaki s po jednim identičnim kanalom. Jedan se dupliciran red prosljeđuje na *Display_video* link za potrebe prikaza aktivne snimane okoline s kamerom sustava, dok se podaci drugog dupliciranog red s kanalom prosljeđuju na *Encode* link. *Encode* linkom komprimiraju se podaci svakog okvira, kako bi se omogućila jednostavnija i brža pohrana podataka snimanog videozapisa. *Encode* linkom određuju se parametri komprimiranog videozapisa, poput formata videozapisa, ciljane brzine komprimiranja i sl. Komprimirani podaci zatim se prosljeđuju na *Null* link kojim je omogućena pohrana podataka na razvojno računalo putem mrežnog modula. Na slici 3.53.5 prikazane su potrebne funkcije za postavljanje parametara senzora kamere te inicijalizaciju kamere za rad. Druga linija koda postavlja identifikacijsku oznaku senzora na prvi *port* kamere ADAS ALPHA razvojne ploče FFN SoC-a. Nadalje, trećom se linijom inicijalizira korištena kamera za rad, postavljajući parametre visine i širine videozapisa te identifikacijske oznake korištenog senzora kamere. Linijom sedam, dodatno se opisuju parametri korištene kamere u pogledu moguće najveće rezolucije snimanja te željene rezolucije snimanja. Globalnim konstantama *CAPTURE_SENSOR_WIDTH* i *CAPTURE_SENSOR_HEIGHT* definirane su vrijednosti koje predstavljaju rezoluciju slike kojom senzor kamere snima okolinu, a odgovaraju rezoluciji 1280 x 720.

Linija ***Kod***

```
1:      /*Parametri senzora kamere te inicijalizacija kamere za rad*/
2:      pObj->sensorId = FFN_VIDEO_SENSOR_1;
3:      ChainsCommon_Amv_StartFFNCaptureDevice(pObj->chainsCfg->captureSrc,
4:                                             CAPTURE_SENSOR_WIDTH,
5:                                             CAPTURE_SENSOR_HEIGHT,
6:                                             pObj->sensorId);
7:      ChainsCommon_Amv_SingleCam_SetFFNCapturePrms(&pUcObj->CapturePrm,
8:                                                   CAPTURE_SENSOR_WIDTH,
9:                                                   CAPTURE_SENSOR_HEIGHT,
10:                                                  CAPTURE_SENSOR_WIDTH,
11:                                                  CAPTURE_SENSOR_HEIGHT,
12:                                                 pObj->chainsCfg->captureSrc,
13:                                                  pObj->sensorId);
```

Sl. 3.5. Programski kod za inicijalizacije kamere ADAS ALPHA razvojne ploče

Slikom 3.6. prikazan je poziv funkcija potrebnih za postavljanje parametara *VPE, Encode* te *Nul* linkova Funkcijom *chains_ffn_camCalib_SetVpePrm* postavljaju se parametri *VPE* za izmjenu zapisa formata boje primljenih podataka s *Capture* linka u YUV420SP format boje. Linijom šest postavljaju se parametri *Null* linka, postavljanjem pripadajuće identifikacijske oznake te prosljeđivanjem postavljenih parametara funkciji *chains_camCalib_SetNullPrms* postavlja se način rada mrežnog modula na *NETWORK_TX_SERVER_PORT*. Postavljanjem načina rada mrežnog modula omogućena je transmisija podataka s razvojne ploče pomoću *Null* linka i mrežnog modula, na razvojno računalo. U liniji devet, drugim argumentom funkcije *chains_ffn_cam_Calib_SetEncPrms* odabire se format videozapisa na MJPEG, čija se globalna varijabla kojom je format definiran naziva *SYSTEM_IVIDEO_MJPEG*.

Linija ***Kod***

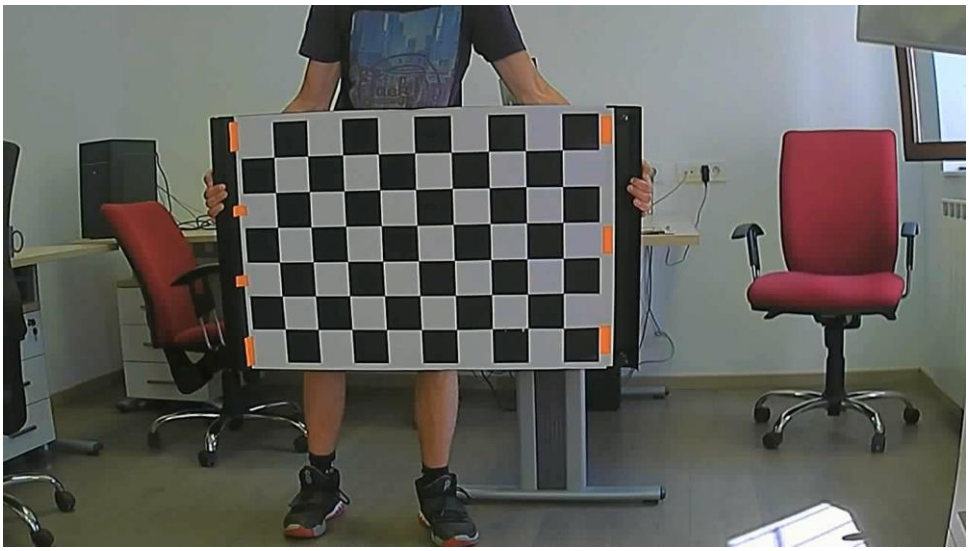
```
1:      chains_ffn_camCalib_SetVpePrm(&pUcObj->VPEPrm,
2:                                    CAPTURE_SENSOR_WIDTH,
3:                                    CAPTURE_SENSOR_HEIGHT,
4:                                    SYSTEM_DF_YUV420SP_UV );
5:       pUcObj->NullLinkID = SYSTEM_MAKE_LINK_ID(pObj->netProcId,
6:                                                pUcObj->NullLinkID);
7:      chains_camCalib_SetNullPrms(pObj, &pUcObj->NullPrm);
8:      chains_ffn_cam_Calib_SetEncPrms(&pUcObj->EncodePrm,
9:                                      SYSTEM_IVIDEO_MJPEG);
```

Sl. 3.6. Programski kod za postavljanja parametara linkova *UseCase*-a za pohranu snimanog videozapisa kamerom ADAS ALPHA razvojne ploče na razvojno računalo

Implementacijom *UseCase*-a potrebno je kreirani *UseCase* kao i ostale datoteke VisionSDK naredbama u CMD izgraditi za implementaciju na ADAS ALPHA razvojnu ploču. Unutar sučelja CMD na putanji direktorija imena *visionSDK*, naredbama *gmake* generirane su dvije datoteke naziva *AppImage* i *MLO*, potrebne za pokretanje sustava na razvojnoj ploči. Postavljanjem dobivenih datoteka na *microSD* karticu, umetanjem kartice u pripadajući utor ADAS ALPHA razvojne ploče za FFN SoC, povezivanjem razvojnog sustava i razvojne ploče UART kabelom priključkom na FFN SoC te spajanjem ekrana putem HDMI kabela na HDMI izlaz FFN SoC-a, omogućeno je pokretanje kreiranog *UseCase*-a. Kako je sa slike 3.1. vidljivo da FFN SoC ne pruža mrežni modul, kreirani *UseCase* te ostale datoteke VisionSDK potrebno je izgraditi i za FUS SoC te dobivene datoteke *AppImage* i *MLO* postaviti na drugu *microSD* karticu te ju umetnuti u *microSD* utor FUS SoC-a.

Povezivanjem ADAS ALPHA razvojne ploče te razvojnog računala i pokretanjem komunikacijskog terminala TeraTerm, na razvojnom računalu, s postavljanjem brzine komunikacije (engl. *baud rate*) na vrijednost jednakoj brzini komunikacije razvojne ploče, omogućeno je izvođenje implementiranog *UseCase*-a. Prilikom pokretanja, inicijaliziraju se potrebni dijelovi ADAS ALPHA razvojne ploče za rad *UseCase*-a, nakon čega se prikazuje snimana okolina pomoću kamere, na ekranu spojenog na razvojnu ploču.

Pokretanjem kreiranog *UseCase*-a, potrebno je pozicionirati kalibracijski panel na način da se s različitim snimljenim okvirima, kalibracijskim panelom obuhvati cijelo područje snimanja na jednakoj udaljenosti od senzora kamere. Primjer kalibracijskog panela, kao i primjer jednog od potrebnih okvira za skup kalibracijski slika, prikazani su slikom 3.7.



Sl. 3.7. Prikaz jednog okvira iz skupa kalibracijskih slika s kalibracijskim panelom

Prikupljanjem potrebnog skupa slika kalibracijskog panela, sljedeći korak procedure kalibracije kamere podrazumijeva kreiranje *Python* skripte za proračun kalibracijskih parametara. Slikom 3.8. prikazan je ključan dio kreirane skripte za proračun kalibracijskih parametara. U drugoj liniji, u polje slika imena *images* učitava se skup kalibracijskih slika s putanje definirane globalnom varijablom *IMAGE_DIRECTORY*. Prolaskom svake slike u polju slika, mijenja se format boje u prikaz slike u razinama sive boje linijom pet, detektiraju se točke dodira dvaju crnih i dvaju bijelih kvadratića s panela linijom šest te ukoliko su točke detektirane dodaju se skupu točaka objekata kalibracijskog panela linijom osam, a filtrirane ključne točke sa zadanim kriterijem dodaju se skupu točaka slike, linijama (9-11). Proračun kalibracijskih parametara odrađen je pomoću funkcije *calibrateCamera* u liniji 12, definirane i deklarirane u biblioteci *OpenCV*-a. Linijom 14 proračunati parametri spremaju se u datoteku s ekstenzijom „*.npz*“, pri čemu je pohrana parametara omogućena pomoću funkcija biblioteke *NumPy*.

Linija ***Kod***

```
1:      # Proračun kalibracijskih parametara
        images = glob.glob(IMAGE_DIRECTORY)
2:
        for fname in images:
3:
            img = cv2.imread(fname)
4:
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
5:
            ret, corners = cv2.findChessboardCorners(gray, (xOs, yOs), None)
6:
            if ret == True:
7:
                objpoints.append(objp)
8:
                corners2 = cv2.cornerSubPix(gray, corners,
9:
                                            (11, 11), (-1, -1), criteria)
10:
                imgpoints.append(corners2)
11:
        ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints,
12:
                                    imgpoints, gray.shape[::-1], None, None)
13:
        np.savez(NPZ_FILE_OUT, mtx=mtx, dist=dist, rvecs=rvecs, tvecs=tvecs)
14:
```

Sl. 3.8. Programski kod za algoritam proračuna kalibracijskih parametara.

Rezultat kalibracije tri CMOS kamere, tri su datoteke „*.npz“* ekstenzije s proračunatim kalibracijskim parametrima te tri ispisa ukupne pogreške ispravljene projekcije točaka objekta, odnosno kutova spajanja dvaju crnih i dvaju bijelih kvadrata kalibracijskog panela, pomoću proračunatih kalibracijskih parametara u odnosu na točke objekta ne-ispravljene slike. Algoritam proračuna ukupne pogreške prikazan je slikom 3.9. te implementiran pomoću naredbi biblioteke *OpenCV, projectPoints* za ispravljanje projekcije točaka objekata te funkcije *norm* s parametrom

NORM_L2, što označava proračun greške svake ispravljene projekcije točaka objekta s istim neispravljenim točkama objekta na temelju metode korijena sume kvadrata razlike euklidske udaljenosti njihovih pozicija, prikazano formulom 3.1. *src1* te *src2* predstavljaju točke objekata ispravljene i ne-ispravljene slike.

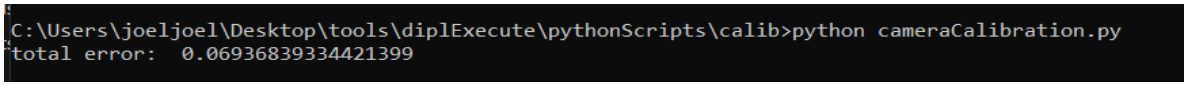
Linija ***Kod***

```
1:      #Proračun ukupne pogreške projekcije ispravljenih elemenata slike
2:      totalError = 0
3:      for i in range(len(objpoints)):
4:          imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i],
5:                                            tvecs[i], mtx, dist)
6:          error = cv2.norm(imgpoints[i],imgpoints2, cv2.NORM_L2) /
7:                           len(imgpoints2)
8:          totalError += error
```

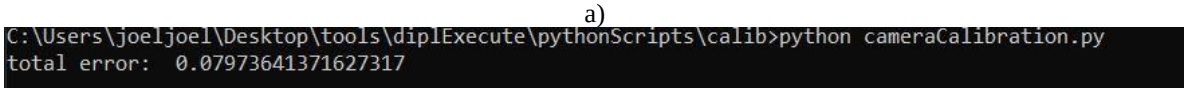
Sl. 3.9. Programski kod za algoritam proračuna ukupne pogreške projekcije ispravljenih elemenata slike

$$||src1 - src2||_{L2} = \sqrt{\sum_{I=1}^{n} (src1(I) - src(I))^2} \tag{3.1.}$$

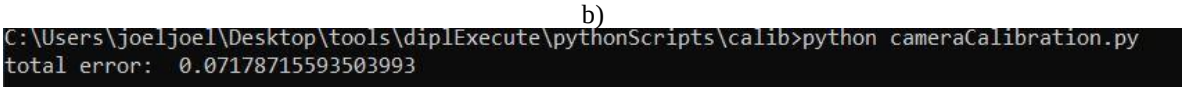
Slikom 3.10. prikazan je dobiveni proračun ukupne pogreške kamera korištenih u ovom radu. Proračun je dobiven u decimalnom zapisu, a pomnožen sa 100 označava postotak ukupne pogreške, čija vrijednost ne smije biti veća od 20%. Ukoliko postotak ukupne pogreške iznosi više od 20%, proračunate kalibracijske parametre treba odbaciti te provesti kalibraciju s novim skupom slika.



a)



b)



c)

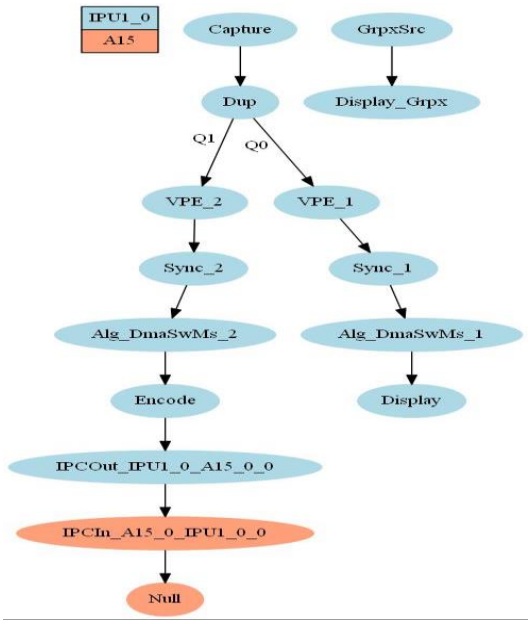
Sl. 3.10. Prikaz proračunate ukupne pogreške ispravljene projekcije korištenih kamera (a) proračun ukupne pogreške lijeve kamere (b) proračun ukupne pogreške srednje kamere (c) proračun ukupne pogreške desne kamere

Proračunati kalibracijski parametri pohranjeni u datotekama s „*.npz*“ ekstenzijom, obuhvaćaju polje distorzijskih koeficijenata te parametre korištene kamere zapisane matricom. Formulom 3.2. prikazan je izgled zapisa i naziva korištenih varijabli prilikom proračuna kalibracijskih parametara. Matricom kamere obuhvaćeni su intrinzični parametri: žarišna duljina varijablama f_x i f_y te koordinate središta slike c_x i c_y [22].

$$Distorzijski\ koeficijenti = (k_1, k_2, p_1, p_2, k_3), matrica\ kamere = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.2.)$$

3.3.2. Snimanje sinkroniziranih videozapisa s područjem preklapanja pomoću kamera ADAS ALPHA razvojne ploče

S ciljem spajanja više okvira različitih videozapisa s područjem preklapanja, snimljenih pomoću kamera ADAS, implementiran je *UseCase* koji omogućava pohranu triju sinkroniziranih videozapisa putem mrežnog modula, snimljenih kamerama sustava ovoga rada. Sinkronizacija snimanja videozapisa pojedinih kamera potrebna je kako bi algoritam spajanja slika u svakoj iteraciji izvođenja prihvaćao scenu snimljenu trima kamerama u istim trenutcima, odnosno skup okvira jednakih vremenskih oznaka. U odnosu na prethodni *UseCase* objašnjen u prethodnome podpoglavlju, unutar ovoga *UseCase*-a implementirat će se rad triju kamera ADAS ALPHA razvojne ploče te dodatno koristiti *Alg_DmaSwMs* algoritamski link. *Alg_DmaSwMs* algoritamski link može prihvatiti do šest kanala, čije sadržaje prikazuje u obliku spojene mozaik slike, čime se podaci ulaznih kanala pohranjuju u jedan red s jednim kanalom. Provedbom navedenih koraka za generiranje te implementaciju *UseCase*-ova objašnjenih u prethodnom podpoglavlju, na slici 3.11. prikazan je rezultat generiranja *UseCase*-a, odnosno tok podataka i veza *UseCase*-a. Također, slikom 3.12. prikazan je poziv funkcija potrebnih za inicijalizaciju i pokretanje više kamera ADAS ALPHA razvojne ploče te postavljanje parametara potrebnih za sinkronizaciju snimanja videozapisa kamerama sustava, kao i parametara *Alg_DmaSwMs* algoritamskog linka.



Sl. 3.11. Prikaz toka podataka i veza *UseCase*-a za pohranu snimanog sinkroniziranog videozapisa kamerama ADAS ALPHA razvojne ploče na razvojno računalo

Linija ***Kod***

```
1:      ChainsCommon_Amv_MultiCam_StartFFNCaptureDevice(
2:                                                  pObj->chainsCfg->captureSrc,
3:                                                  pObj->numLvdsCh,
4:                                                  pObj->sensorId);
5:      ChainsCommon_Amv_MultiCam_SetFFNCapturePrms(&pUcObj->CapturePrm,
6:                                                    CAPTURE_SENSOR_WIDTH,
7:                                                    CAPTURE_SENSOR_HEIGHT,
8:                                                    portId,
9:                                                    pObj->numLvdsCh);
10:     chains_ffn_cam_stitch_streams_SetAlgDmaSwMsPrm(
11:                                                  &pUcObj-> Alg_DmaSwMs_1Prm,
12:                                                   pObj->numLvdsCh,
13:                                                   CAPTURE_SENSOR_WIDTH,
14:                                                   CAPTURE_SENSOR_HEIGHT);
15:     chains_ffn_cam_stitch_streams_SetSyncPrm(&pUcObj->Sync_1Prm,
16:                                                  pObj->numLvdsCh);
```

Sl. 3.12. Programski kod za poziv funkcija postavljanja parametara kamera te linkova *UseCase*-a

Linijama (2-10) inicijalizirane su tri kamere sustava s rezolucijom snimanja 1280 x 720 te omogućene za rad postavljanjem *sensorID* na I2C adrese prvih triju priključaka FFN SoC-a ADAS ALPHA razvojne ploče. Nadalje, linijom 11 prikazan je poziv funkcije za postavljanje parametara algoritamskog linka desne grane toka podataka izlaza *DUP* linka, prikazano slikom 3.11.Ulazni parametar *numLvdsCh* predstavlja ukupni broj kanala na ulazu u algoritamski link, što u ovome radu iznosi tri te se funkciji predaje i rezolucija konačne mozaik slike. Kako je na desnoj grani izlaza *DUP* linka potrebno prikazati snimanu okolinu za potrebe pregleda i kontrole snimanja, za rezoluciju mozaik slike postavlja se vrijednost 1280 x 720, dok se za lijevu granu izlaza *DUP* linka vrijednost rezolucije uvećava tri puta u pogledu širine, čime se sinkroniziranim prikazom u cijelosti obuhvaćaju primljeni podatci svake kamere, bez promjene početne rezolucije. Nadalje, linijom 16 postavljeni su sinkronizacijski parametri, pri čemu *Sync* link regulira slanje podataka kanala na svome izlazu. Implementacijom *UseCase*-a i pokretanjem koracima opisanim u prethodnom podpoglavlju, kao rezultat ostvaren je pohranjeni sinkronizirani videozapis rezolucije 3840 x 720, uz 25 FPS. Primjer okvira pohranjenog videozapisa izvođenjem *UseCase*-a prikazan je slikom 3.13.

Sl. 3.13. Prikaz okvira sinkroniziranog mozaik videozapisa snimljenog trima kamerama

3.3.1. Algoritam ispravljanja izobličenja na snimljenim sinkroniziranim videozapisima

Snimanjem mozaika sinkroniziranih videozapisa s područjem preklapanja, ostvaren je potreban skup videozapisa kao ulaznih datoteka algoritma spajanja slika. Kako je jednim mozaik videozapisom obuhvaćen prikaz snimki s triju kamera, potrebno je implementirati *Python* skriptu s ciljem razdvajanja videozapisa i pohranjivanja u tri zasebna videozapisa jednoznačnog imena. U elektroničkom prilogu P.3.1. ovoga rada dane su datoteke s implementiranim programskim kod, čijim se izvođenjem učitava mozaik videozapis te se svakom iteracijom primljenog okvira mozaik videozapisa, okvir odvaja u tri zasebna prikaza rezolucije 1280 x 720 i pohranjuje u jednoznačno imenovane videozapise. Tri sinkronizirana videozapisa jednakih rezolucija s 25 FPS, također je potrebno urediti algoritmom ispravljanja izobličenja. Proračunatim kalibracijskim parametrima u dijelu 3.1.1., potrebno je ispraviti tangencijalno i radijalno izobličenje svakog okvira sinkroniziranih videozapisa s pripadajući proračunatim kalibracijskim parametrima, ovisno o korištenoj kameri za snimanje pojedinog videozapisa. Implementacija *Python* skripte za ispravljanje izobličenja svakog okvira započinje učitavanjem ulaznog sinkroniziranog videozapisa s pripadajućim proračunatim kalibracijskim parametrima korištene kamere, prikazano slikom 3.14.

Linija Kod

```
1:      video_stream_in = cv2.VideoCapture(FILENAME_IN)
2:      npz_calib_file = np.load(INPUT_CAM_CALIB_FILE)
3:      distCoeff = npz_calib_file['dist']
4:      intrinsic_matrix = npz_calib_file['mtx']
5:      npz_calib_file.close()
```

Sl. 3.14. Programski kod za učitavanje ulaznog videozapisa i pripadajućih kalibracijskih parametara korištene kamere

Nadalje, potrebno je kreirati novi videozapis formata *DIVX* za potrebe pohrane ispravljenih okvira ulaznog videozapisa. Slikom 3.15. prikazan je programski kod za učitavanje ulaznog videozapisa ovoga algoritma te je proračunata nova optimalna matrica senzora kamere pomoću kalibracijskih parametara. Proračunata matrica, zajedno s kalibracijskim parametrima, prosljeđuje se funkciji *initUndistortRectifyMap* za generiranje mape koordinata novih elemenata slike. U prikazanoj formuli 3.3., za svaki element slike s koordinatama (u, v) kao položajem elementa neispravljene slike, provedbom proračuna, u matricama map_x i map_y s indeksima matrica (u, v), pohranjene su koordinate ispravljenog elementa slike [23].

$$
\begin{aligned}
x &= \frac{u - {c'}_x}{{f'}_x} \\
y &= \frac{v - {c'}_y}{{f'}_y} \\
x' &= \frac{X}{W} \\
y' &= \frac{Y}{W} \\
x'' &= x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2\left(r^2 + 2{x'}^2\right) \\
y'' &= y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1\left(r^2 + 2{y'}^2\right) + 2p_2 x' y' \\
map_x(u, v) &= x'' f_x + c_x \\
map_y(u, v) &= y'' f_y + c_y
\end{aligned}
\tag{3.3.}
$$

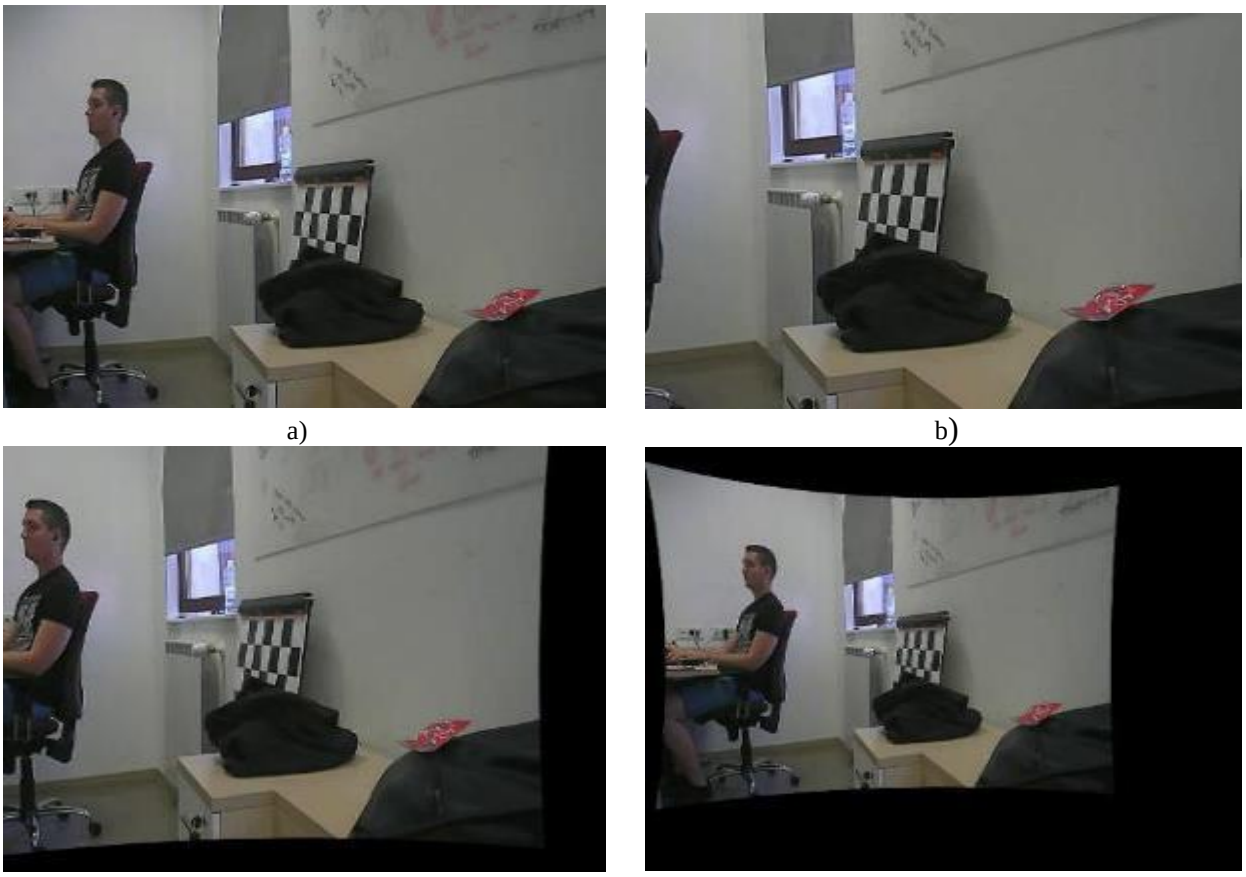
Linija ***Kod***

```
1:      //Ispravljanje izobličenja svakog okvira i pohrana u novi videozapis
2:      fourcc = cv2.VideoWriter_fourcc(*'DIVX')
3:      video_stream_out =cv2.VideoWriter(FILENAME_OUT, fourcc, FPS, size, 1)
4:      newMat, ROI = cv2.getOptimalNewCameraMatrix(intrinsic_matrix,
5:                                                  distCoeff, size, alpha = CROP,
6:                                                  centerPrincipalPoint=1)
7:      mapx, mapy = cv2.initUndistortRectifyMap(intrinsic_matrix, distCoeff,
8:                                                 None, newMat, size,
9:                                                 m1type=cv2.CV_32FC1)
10:     current_frame = 0
11:     while current_frame < total_frames:
12:             success, image = video_stream_in.read()
13:             dst = cv2.undistort(image, intrinsic_matrix, distCoeff,
14:                                 None, newMat)
15:             video_stream_out.write(dst)
16:             current_frame = video_stream_in.get(cv2.CAP_PROP_POS_FRAMES)
17:     video_stream_in.release()
18:     video_stream_out.release()
```

Sl. 3.15. Programski kod za algoritam ispravljanja izobličenja

Nadalje, prolaskom svakog okvira ulaznog videozapisa linijama (10 – 16) pomoću funkcije *undistort* ispravlja se izobličenje uzrokovano tangencijalnim i radijalnim distorzijama. Funkcija *undistort* bilinearnom interpolacijom položaja elemenata slike pomoću funkcija *initUndistortRectifyMap* te *remap*, postiže prikaz točnije aproksimacije stvarne snimljene okoline. Ovisno o vrijednosti *alpha* varijable predane funkciji *getOptimalNewCameraMatrix*, rezolucija ispravljane slike se mijenja. Raspon vrijednosti *alpha* koeficijenta kao decimalne vrijednosti [0, 1] označava očuvanje svih elemenata neispravljenog okvira u okviru s ispravljenim izobličenjem. Vrijednost *alpha* koeficijenta 0 označava da se svi elementi okvira prikazuju točnom aproksimacijom te se uzima valjani dio slike, a rezolucija ispravljene slike uvećava na rezoluciju

primljene slike. S vrijednosti 1, ne provodi se uzimanje valjanog dijela slike te su očuvani svi elementi neispravljene slike, ali interpolacija položaja elemenata slike uzrokuje pojavu crnih površina vidljivih na ispravljenom okviru. Primjer dobivenih ispravljenih okvira s različitim vrijednostima *alpha* koeficijenta prikazan je slikom 3.16.



a) b)

c) d)

Sl. 3.16. Prikaz rezultata algoritma ispravljanja izobličenja: (a) okvir bez ispravljenog izobličenja (b) okvir s ispravljenim izobličenjem s *alpha* vrijednošću 0, (c) okvir s ispravljenim izobličenjem s *alpha* vrijednošću 0,5 (d) okvir s ispravljenim izobličenjem s *alpha* vrijednošću 1

3.3.2. Algoritmi detekcije i stvaranja deskriptora ključnih točaka

Završetkom obrade sinkroniziranih videozapisa s područjem preklapanja, implementirana je *Python* skripta za spajanje svih skupova sinkroniziranih okvira s područjem preklapanja, paralelnim iteracijama prolaska po sinkroniziranim okvirima triju ulaznih datoteka videozapisa. Učitavanjem ulaznih videozapisa, na prvome skupu primljenih okvira provedena je detekcija ključnih točaka. Na slici 3.17. prikazana je inicijalizacija SIFT, SURF te ORB detektora korištenih za detekciju i stvaranje deskriptora ključnih točaka na primljenome skupu sinkroniziranih okvira. Također, prikazan je i poziv funkcije za detekciju ključnih točaka i stvaranje deskriptora na predanom joj okviru, primjenom SIFT detektora. Funkcija *detectAndCompute* biblioteke *OpenCV*, implementirana je za sva tri korištena detektora te se poziva na identičan način. Iako je kodom

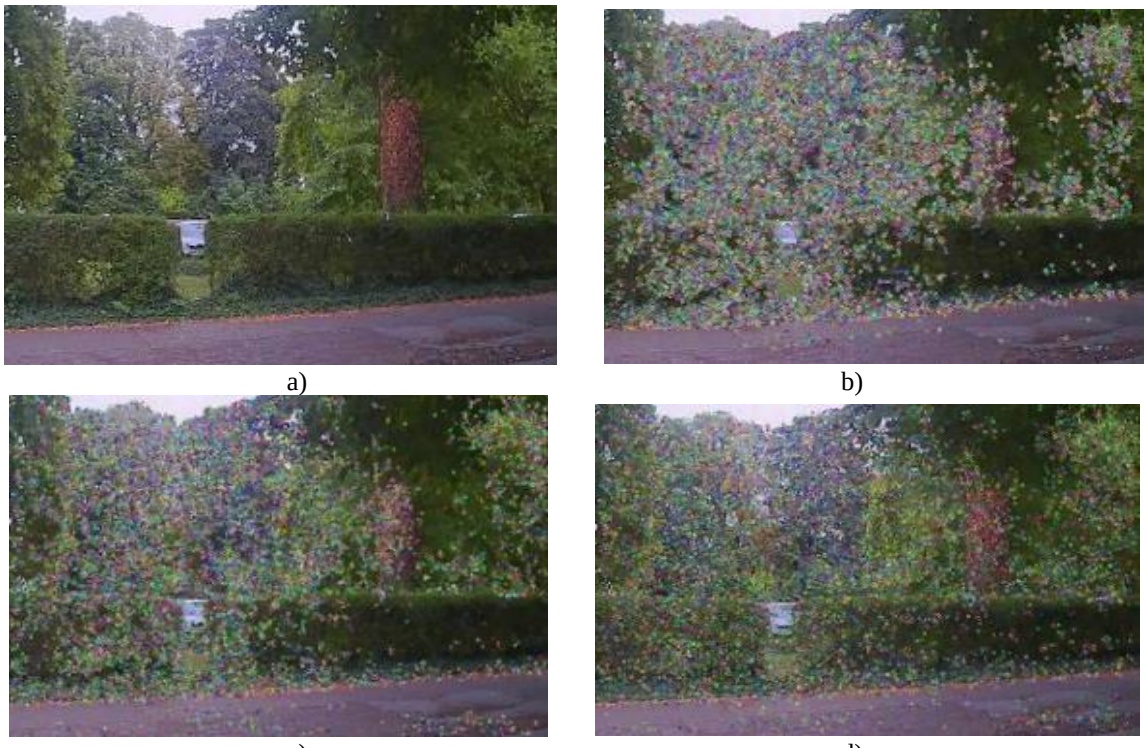
prikazana implementacija SURF detektora, izvođenje algoritma detekcije i stvaranja deskriptora SURF metodom, ostvareno je pomoću druge *Python* skripte i bibliotekom *OpenCV* prijašnjih inačica, kako SURF detektor nije podržan u *OpenCV* korištenoj inačici za potrebe ovoga rada s oznakom 4.4.0.

Linija ***Kod***

```
1:      descriptorSIFT = cv2.xfeatures2d.SIFT_create()
2:      descriptorSURF = cv2.xfeatures2d.SURF_create()
3:      cv2.ORB_create(nfeatures = 100000, scoreType = cv2.ORB_FAST_SCORE)
4:      (keyPoints, features) = descriptorSIFT.detectAndCompute(frame, None)
```

Sl. 3.17. Programski kod za implementaciju korištenih detektora

Slikom 3.18. prikazane su detektirane ključne točke pomoću ORB, SIFT i SURF detektora na predanome okviru.



a) b) c) d)

Sl. 3.18. Prikaz detektiranih ključnih točaka korištenih detektora: (a) okvir za detekciju ključnih točaka (b) okvir s prikazanim ključnim točkama detektiranim pomoću ORB-a (c) okvir s prikazanim ključnim točkama detektiranim pomoću SIFT-a (d)) okvir s prikazanim ključnim točkama detektiranih pomoću SURF-a

Prikazom detektiranih ključnih točaka vidljivo je kako se ORB detektorom na zadanom okviru ne mogu pronaći ključne točke na rubovima okvira, ali nedostatak ključnih točaka u tome području ne utječe na konačni rezultat spajanja okvira te proračun matrice homografije. Nadalje, na slici 3.17. vidljiva je razlika pri implementaciji ORB detektora u odnosu na SURF i SIFT detektore.

Varijablom *nfeatures* određuje se traženi broj ključnih točaka, što se također može postaviti i unutar poziva funkcije inicijalizacije SIFT te SURF detektora, ali unutar detektora ORB je neophodan. Nadalje, varijablom *scoreType* određuje se metoda za odabir najboljih ključnih točaka pri čemu se može postaviti metoda naziva *Harris Score* te *Fast Score*. *Harris Score* označava korištenje *Harris* algoritma za rangiranje ključnih točaka, dok *Fast Score* označava alternativnu metodu za brže izvođenje s detektiranim ključnim točkama podložnijim djelovanju šuma u odnosu na *Harris Score* metodu. Odnosno, *Fast Score* metodom veća je mogućnost pogreške uparivanja odgovarajućih parova ključnih točaka, nego što je moguće korištenjem *Harris Score* metode.

3.3.3. Algoritam spajanja skupa sinkroniziranih okvira s ispravljenim izobličenjem u jedinstveni spojeni okvir pomoću detektiranih ključnih točaka na okvirima za spajanje

Detekcijom ključnih točaka i stvaranjem deskriptora na skupu prvih okvira ulaznih sinkroniziranih videozapisa, omogućeno je traženje odgovarajućih ključnih točaka na različitim parovima okvira s područjem preklapanja. Kako je algoritmom potrebno spojiti tri okvira, prvobitno spojitiće se okvir desnog i srednjeg videozapisa snimane okoline, a zatim će se spojiti okvir lijevog videozapisa s već spojenim okvirima. Slikom 3.19. prikazana je implementacija objekta kojim će se omogućiti pronalaženje jednakih ključnih točaka na različitim okvirima.

Linija ***Kod***

```
1:    matcher = cv2.DescriptorMatcher_create("BruteForce")
2:    rawMatches = matcher.knnMatch( featuresLeft,featuresRight, 2)
3:    matches = []
4:    for m in rawMatches:
5:          if len(m) == 2 and m[0].distance < m[1].distance * ratio:
6:               matches.append((m[0].trainIdx, m[0].queryIdx))
```

Sl. 3.19. Programski kod za algoritam spajanja parova ključnih točaka

Objektu klase *DescriptorMatcher* naziva *matcher* prilikom inicijalizacije u prvoj liniji koda, predan je parametar naziva *BruteForce*, kojim se određuje metoda pronalaženja jednakih ključnih točaka na različitim okvirima. Korištena metoda *BruteForce* usporediti će sve parove ključnih točaka različitih okvira te takvim pristupom pronaći najbolje parove. Druga metoda koja se može koristiti kao alternativna metoda za traženje odgovarajućih ključnih točaka definirana je nazivom *FlannBasedMatcher*. FLANN (engl. *Fast Library for Approximate Nearest Neighbors*) metoda, iako kraćeg vremena izvođenja u odnosu na *BruteForce* metodu, ograničena je u pogledu preciznosti pronalaženja najboljih parova ključnih točaka različitih okvira te pronalazi približno

najbolje parove ključnih točaka. Inicijalizacijom metode traženja, drugom linijom izvršava se samo pretraživanje parova ključnih točaka na predanim deskriptorima funkcijom *knnMatch*. Funkcija *knnMatch* koristeći algoritam *k-NN* (engl. *k-Nearest Neighbors*) vraća *K* najboljih parova ključnih točaka predanih deskriptora, pronađenih *BruteForce* metodom. Prilikom poziva funkcije *knnMatch*, trećim parametrom zadaje se koeficijent *K*, čime se definira broj najboljih parova ključnih točaka. Primjenom predložene metode autora [4] Lowe-a za odabir najboljih parova s linijama (4 – 6), za svaku detektiranu ključnu točku jednog okvira, pohranjene su dvije odgovarajuće „najbliže" ključne točke drugoga okvira, određene minimalnom euklidskom udaljenošću u odnosu na zadanu ključnu točku prvog okvira. Varijablom *ratio* određuje se omjer euklidske udaljenosti uparenih odgovarajućih ključnih točaka, čime se regulira maksimalna euklidska udaljenost odgovarajućih parova ključnih točaka prilikom odabira najboljih. Ukoliko je euklidska udaljenost najbliže ključne točke manja od euklidske udaljenosti druge najbliže odgovarajuće ključne točke pomnožene s vrijednošću *ratio*, u listu *matches* dodaju se indeksi *query* te *train* deskriptora, odnosno indeks deskriptora ključne točke lijevog okvira te indeks deskriptora najbliže odgovarajuće ključne točke desnog okvira. Prikaz spojenih parova ključnih točaka različitih okvira s područjem preklapanja prikazan je slikom 3.20.

a) b)

c)

Sl. 3.20. Prikaz spojenih parova ključnih točaka na skupu okvira za spajanje: (a) lijevi okvir za spajanje (b) desni okvir za spajanje (c) mozaik prikaz lijevog i desnog okvira sa spojenim parovima odgovarajućih ključnih točaka

Povezivanjem odgovarajućih ključnih točaka na području preklapanja okvira za spajanje, potrebno je proračunati matricu homografije funkcijom *findHomograpy,* kojom se objedinjuju inicijalni intrinzični i ekstrinzični parametri predanih okvira te pronalazi odgovarajuća matrica transformacije perspektive. Proračuna matrice homografije prikazan je slikom 3.21.

Linija ***Kod***

```
1:      if len(matches) > 4:
2:        pointsRight = np.float32([keyPointsRight[i] for (i, _) in matches])
3:        pointsLeft = np.float32([keyPointsLeft[i] for (_, i) in matches])
4:       (homography, status) = cv2.findHomography(pointsRight, pointsLeft,
5:                                                 cv2.RANSAC, reprojThresh)
```

Sl. 3.21. Proračun matrice homografije

Matrica homografije može se izračunati ukoliko postoje četiri para spojenih ključnih točaka, dok s više parova spojenih ključnih točaka ostvaruje bolju matricu transformacije perspektive okvira za spajanje. Treći parametar funkcije *findHomography* predstavlja metodu kojom se proračunava matrica homografije. Funkcijom je omogućen proračun matrice homografije *RANSAC* te *Least – Median robust* metodom. Navedene metode svakim setom s više od četiri ključne točke te algoritmom najmanjeg kvadrata (engl. *least – square)* procjenjuju matrice homografije te proračunavaju najbolju. *Least – Median* metodom, matrica homografije proračunava se *median re-projection error* algoritmom na svakoj procijenjenoj matrici, pri čemu se za proračun odabire ona s najmanjom greškom projekcije transformiranih ključnih točaka. *RANSAC* metodom, matrica homografije proračunava se pomoću procijenjenih matrica te brojem *inlier*-a. *Inlier*-i podrazumijevaju ključne točke koje pripadaju zadanom modelu, dok *outlier*-i predstavljaju spojene točke koje ne pripadaju modelu te se odbacuju prilikom proračuna matrice homografije. Prosljeđivanjem parametra *reprojTrech* funkciji *findHomography,* kao praga pogreške projekcije transformiranih ključnih točaka, ključne točke čija je greška transformacijske projekcije manja od zadanog praga smatraju se *inlier*-ima te se uzimaju za proračun matrice homografije. Proračunatom matricom homografije, potrebno je primijeniti matricu homografije na okvir za spajanje te tako dobiven okvir spojiti na drugi okvir za spajanje. Pozivom funkcije *warpPerspective,* što je prikazano slikom 3.22., primjenjuje se izračunata matrica homografije na desnom okviru te se postavlja rezolucija konačno spojenog okvira na ukupnu širinu okvira i visinu srednjeg okvira, odnosno okvira na kojeg će se drugi okvir spojiti. Rezultat funkcije *waprPerspective* te rezultat spajanja desnog i srednjeg okvira prikazani su slikom 3.23.

Linija ***Kod***

```
1:      panFrame = cv2.warpPerspective(frameRight, homographyMat,
2:                                (frameMiddle.shape[1] + frameRight.shape[1],
3:                                 frameMiddle.shape[0]))
4:      panFrame[0:frameMiddle.shape[0],0 frameMiddle.shape[1]]=frameMiddle
```

Sl. 3.22. Programski kod za primjenu matrice homografije na okvir za spajanje

a)

b)

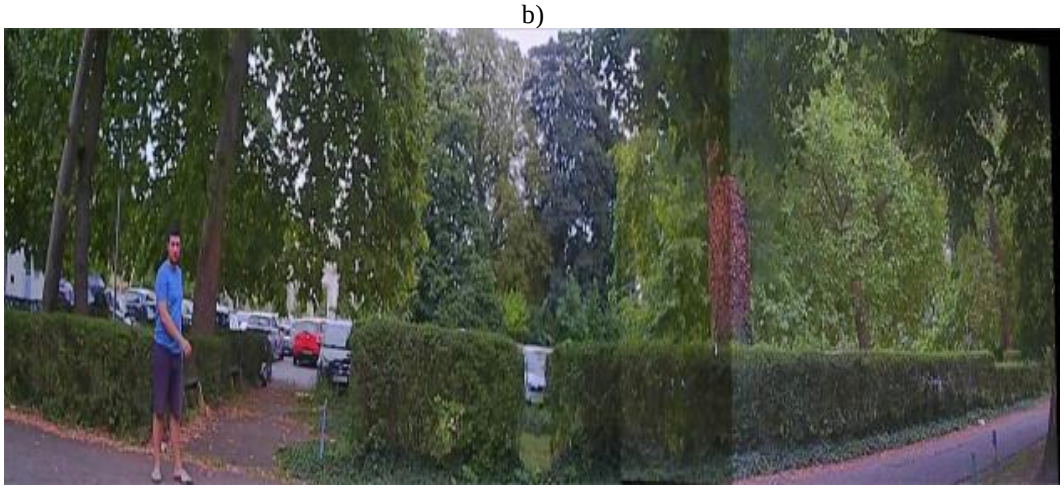
c)

Sl. 3.23. Prikaz rezultat algoritma spajanja okvira: (a) srednji okvir za spajanje (b) desni transformirani okvir za spajanje (c) jedinstveno spojeni okvir

Postupak spajanja desnog i srednjeg okvira, na identičan način treba provesti i za spajanje lijevog okvira s prethodno spojenim desnim i srednjim okvirom, čime se ostvario rezultat spajanja triju sinkroniziranih okvira ulaznih videozapisa, prikazano slikom 3.24.

a)

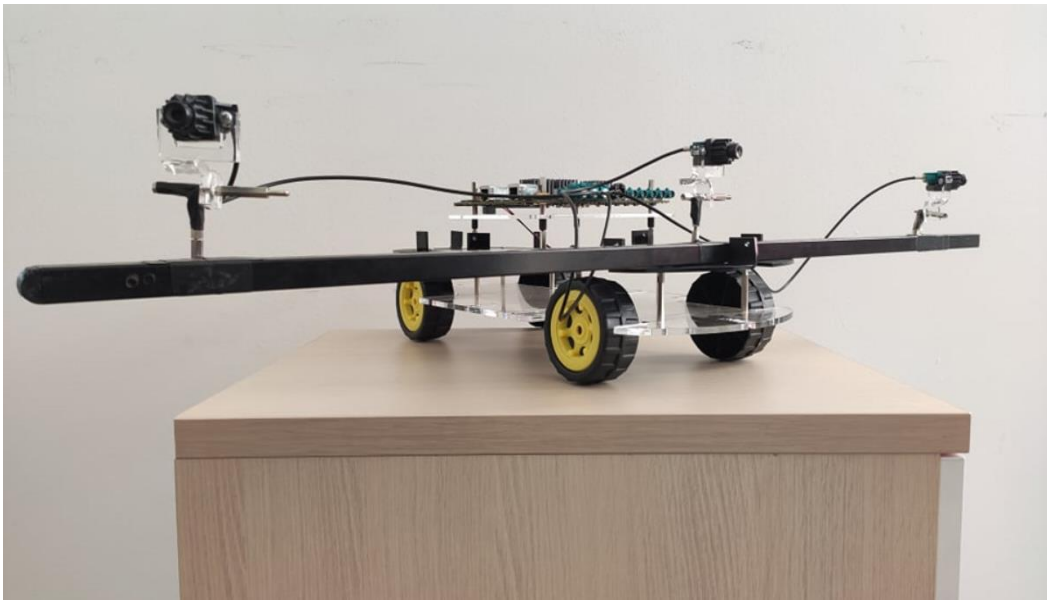
b)



c)

Sl. 3.24. Prikaz rezultat algoritma spajanja okvira: (a) lijevi okvir za spajanje (b) transformirani prethodno spojeni srednji i desni okvir kao desni okvir za spajanje (c) jedinstveno spojeni okvir

Algoritmom spajanja triju okvira rezolucije 1280 x 720, ostvario se jedinstveni spojeni okvir rezolucije 3000 x 600 elemenata slike. Rezolucija spojenih okvira prilagoditi će se u četvrtom poglavlju, prilagođavanjem ulaznog skupa testnih sinkroniziranih videozapisa snimljenih kamerama razvojnog sustava, na način da se obuhvati veće područje snimane okoline te dodatno smanji područje preklapanja snimanih okvira većim kutom zakretanja bočnih kamera u odnosu na srednju kameru, kao što je prikazano slikom korištene konstrukcije nosača kamera 3.25.



Sl. 3.25. Korištena konstrukcija nosača kamera

Razlog modifikacije konstrukcije i međusobnog položaja kamera vidljiv je i na slici 3.24. gdje u jedinstvenom spojenom okviru, okvir srednje kamere prikazuje znatno manju površinu okoline nego što je snimljeno pripadajućom kamerom razvojnog sustava.

3.3.4. Algoritam korekcije osvjetljenja na jedinstvenom spojenom okviru

Jedinstvenim spojenim okvirom snimljenim s trima kamerama razvojnog sustava omogućen je prikaz veće okoline vozila. Kako je jedna od namjena algoritma spajanja slika u području *automotiva* prikaz okoline vozaču, jedinstvenu spojenu sliku također treba naknadno doraditi s ciljem ugodnijeg prikaza slike. Kako bi se ostvario kvalitetniji prikaz jedinstveno spojenog okvira, prilikom spajanja okvira implementirat će se i algoritam korekcije osvjetljenja. Primjenom metode iz [11] i prikazom implementiranog algoritma slikom 3.26., ostvareni su spojeni okviri pri čemu je algoritmom korekcije osvjetljenja ispravio intenzitet osvjetljenja desnog okvira prilikom spajanja na srednji te lijevog okvira prilikom spajanja na spojeni okvir desnog i srednjeg okvira. Odabir srednjeg okvira kao referentnog proizlazi iz već spomenutog problema korekcije osvjetljenja korištene metode, u kojemu se okviri visoke razlike intenziteta prilikom spajanja čine zamućenim. Također, srednji okvir odabire se kao referentni kako bi se izbjeglo ponovno izvođenje algoritma korekcije osvjetljenja na jednome okviru, s obzirom da će jedan okvir dva puta biti dio ulaznih okvira za algoritam spajanja okvira te će se tako smanjiti mogućnost proračuna visoke razlike intenziteta i pojava zamućenja na spojenim okvirima.

Linija ***Kod***

```
1:      leftFrame_HSV = cv2.cvtColor(FrameLeft, cv2.COLOR_BGR2HSV)
2:      rightFrame_HSV = cv2.cvtColor(FrameRight, cv2.COLOR_BGR2HSV)
3:      valueLeft = 0
4:      valueRight = 0
5:      for match in good:
6:           p1 = keyPointsLeft[match.queryIdx].pt
7:           p2 = keyPointsRight [match.trainIdx].pt
8:           pixelHSV_Left = leftFrame_HSV[int(p1[1]),int(p1[0])]
9:           HL,SL,VL = pixelHSV_Left
10:          pixelHSV_Right = rightFrame_HSV[int(p2[1]),int(p2[0])]
11:          HR,SR,VR = pixelHSV_Right
12:          valueLeft += VL
13:          valueRight += VD
14:          luminanceDiffHSV = valueLeft - valueRight
15:     luminanceDiffAvgHSVpixel = luminanceDiffHSV / len(good)
```

Sl. 3.26. Programski kod za algoritam proračuna razlike intenziteta

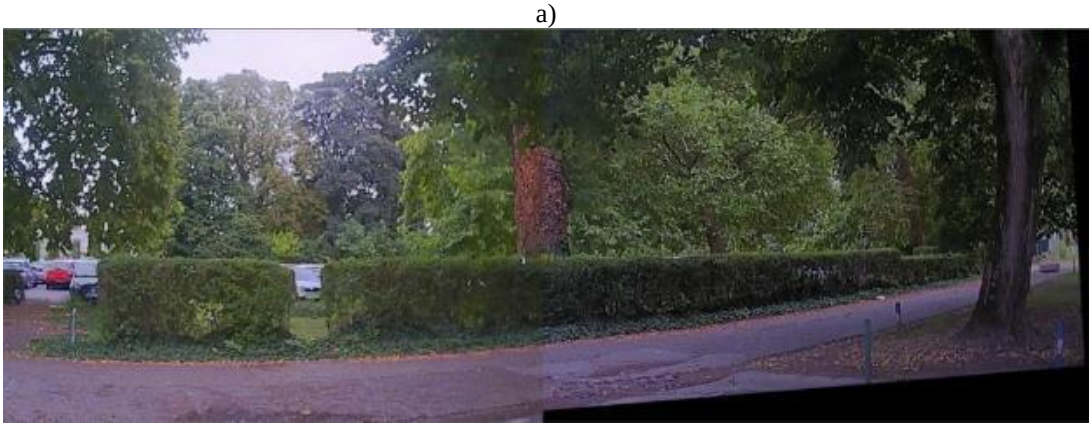
Proračunom prosječne razlike komponente V u HSV formatu na parovima spojenih ključnih točaka okvira predanih algoritmu, proračuna razlike intenziteta osvjetljenja te primjenom proračunate razlike na desni okvir, prikazano slikom 3.27., ostvaren je jedinstveni spojeni okvir s ispravljenim intenzitetom osvjetljenja na desnome okviru prikazanom slikom 3.28. Također metoda iz [11] primijenjena je i u YUV formatu na komponentu Y, pri čemu je ostvareni rezultat također prikazan slikom 3.28.

Linija ***Kod***

```
1:      frameRight_HSV = cv2.cvtColor(frameRight, cv2.COLOR_BGR2HSV)
2:      H_channel, S_channel, V_channel = cv2.split(frameRight_HSV)
3:      if LUMA < 0:
4:         luma = (-1)*int(LUMA)
5:         V_channel[V_channel <= luma] = 0
6:         V_channel[V_channel > luma] -= luma
7:      else :
8:         luma = int(LUMA)
9:         V_channel[V_channel >= (255 - luma)] = 255
10:        V_channel[V_channel < (255 - luma)] += luma
11:     colorCorrected_HSV = cv2.merge((H_channel, S_channel, V_channel))
12:     frameRight_HSV = cv2.cvtColor(colorCorrected_HSV, cv2.COLOR_HSV2BGR)
```

Sl. 3.27. Programski kod za algoritam ispravljanja intenziteta osvjetljenja na desnome okviru

a)



b)

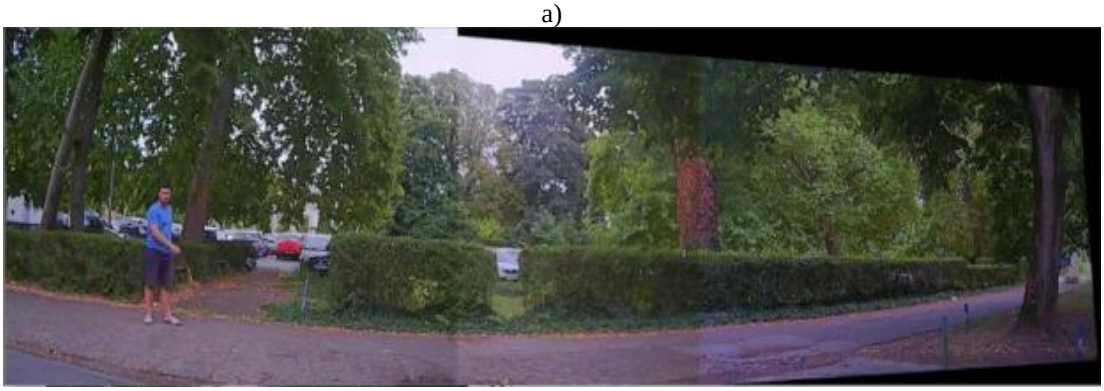
c)

Sl. 3.28. Prikaz rezultata algoritma korekcije osvjetljenja na desnome okviru: (a) jedinstveno spojeni okvir bez ispravljenog osvjetljenja (b) jedinstveno spojeni okvir s ispravljenim osvjetljenjem desnog okvira primjenom algoritma korekcije osvjetljenja na HSV format boje (c) jedinstveno spojeni okvir s ispravljenim osvjetljenjem desnog okvira primjenom algoritma korekcije osvjetljenja na YUV format boje

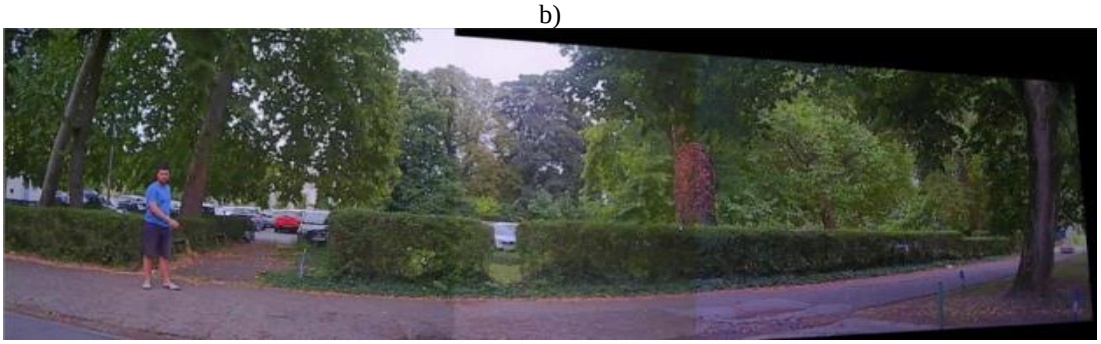
Na prikazu dobivenih rezultata vidljivo je poboljšanje vizualne kvalitete spojenog okvira, kako u HSV tako i YUV420SP formatu boje. Nadalje, algoritam korekcije osvjetljenja primijenjen je i na lijevom okviru za spajanje te spajanjem s već spojenim okvirima dobiveni rezultat prikazan je slikom 3.29. Dobivenim rezultatima prilikom razvoja rješenja također je shvaćena potreba za boljim skupom sinkroniziranih okvira, kako bi se ostvarilo bolje ujednačavanje osvjetljenja na prijelazu spojenih okvira na jedinstvenom spojenom okviru, smanjenjem područja preklapanja prilikom snimanja okoline kamerama razvojnog sustava, jer velikim područjem preklapanja okviri

se spajaju na udaljenijem dijelu okvira u području preklapanja, pri čemu se ne ostvaruje dobar prijelaz osvjetljenja.

a)



b)



c)

Sl. 3.29. Prikaz rezultata algoritma korekcije osvjetljenja na lijevome okviru: (a) jedinstveno spojeni okvir bez ispravljenog osvjetljenja (b) jedinstveno spojeni okvir s ispravljenim osvjetljenjem lijevog okvira primjenom algoritma korekcije osvjetljenja na HSV format boje (c) jedinstveno spojeni okvir s ispravljenim osvjetljenjem lijevog okvira primjenom algoritma korekcije osvjetljenja na YUV format boje

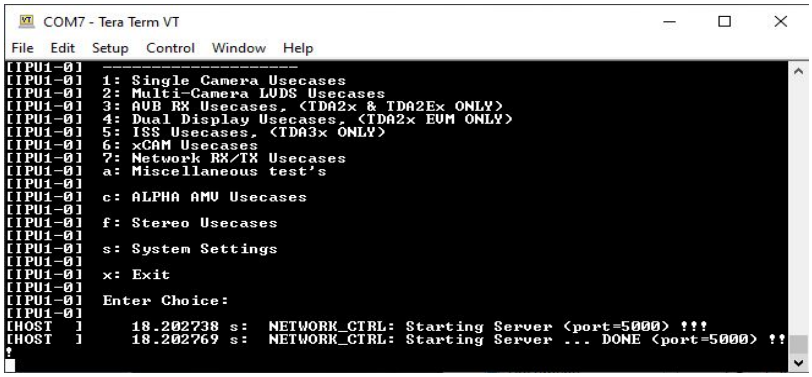
3.3.5. Način pokretanja programskog rješenja

Kako bi ostvarili rezultat spajanja okvira sinkroniziranih videozapisa s ispravljanjem različitog intenziteta osvjetljenja na okvirima snimljenih pomoću različitih kamera razvojnog sustava, potrebno je provesti sljedeće korake:

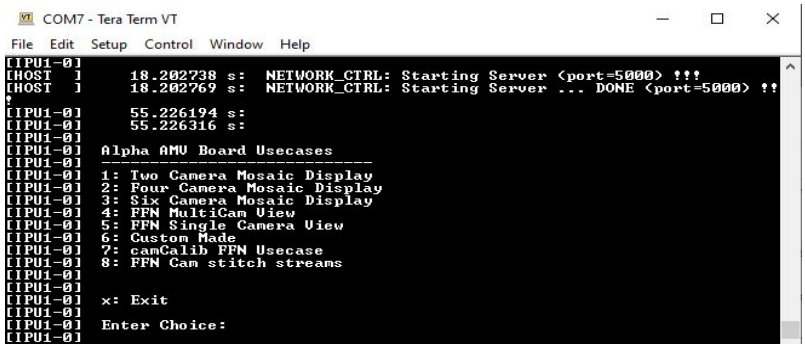
- Osposobiti ADAS ALPHA razvojnu ploču za rad;
- Povezati ADAS ALPHA razvojnu ploču s razvojnim računalom;

- Pokrenuti implementirane *UseCase*-ove;
- Pokrenuti *Python* skripte implementiranih metoda pojedinih funkcionalnosti

Prvi korak podrazumijeva spajanje triju uskokutnih kamera na FFN SoC ADAS ALPHA razvojne ploče. Zatim treba umetnuti *microSD* kartice u pripadajuće utore FFN i FUS SoC-ova. Nadalje, ADAS ALPHA ploči potrebno je omogućiti napajanje s izlaznim naponom od 12V te povezati ekran putem HDMI kabela na odgovarajući FFN SoC radi prikaza rezultata *UseCase*-a. Drugi se korak odnosi na povezivanje ADAS ALPHA razvojne ploče FFN SoC-a s razvojnim računalom pomoću UART te Ethernet kabela. Nadalje, potrebno je na razvojnome računalu otvoriti komunikacijsko sučelje *TeraTerm* te uspostaviti komunikaciju preko UART-a postavljanjem brzine komunikacije razvojnog računala na brzinu komunikacije ADAS ALPHA razvojne ploče (115200 *baud*). Uspostavom komunikacije potrebno je pokrenuti implementirane *UseCase*-ove, kako bi ostvarili skup kalibracijskih videozapisa te skup sinkroniziranih videozapisa s područjem preklapanja susjednih kamera. Slikom 3.30. prikazan je početni izbornik za pokretanje implementiranih *UseCase*-ova pri čemu je potrebno pritiskom tipke znaka „c“ razvojnoga računala odabrati izbornik „*ALPHA AMV Usecases*“. Nadalje, unutar novog izbornika prikazanog slikom 3.31. potrebno je pokrenuti *UseCase* naziva „*camCalib FFN Usecase*“ za snimanje i pohranu kalibracijskog skupa videozapisa kamere spojene na prvi ulaz FFN SoC-a



Sl. 3.30. Prikaz početnog izbornika za pokretanje implementiranih *UseCase*-ova na ADAS ALPHA razvojnoj ploči



Sl. 3.31. Prikaz drugog izbornika za pokretanje implementiranih *UseCase*-ova na ADAS ALPHA razvojnoj ploči

pritiskom tipke „7“. Navedeni *UseCase* potrebno je pokrenuti za svaku korištenu kameru pri čemu je potrebno korištenu kameru spojiti na prvi ulaz kamera FFN SoC-a. Također potrebno je pokrenuti na sličan način i drugi implementirani *UseCase* pod nazivom „*FFN Cam stitch streams*“ za pohranu mozaik sinkroniziranih videozapisa snimljenih s trima kamerama razvojnog sustava pritiskom tipke „8“ unutar drugog izbornika, pri čemu su kamere spojene na prva tri ulaza kamera FFN SoC-a.

Ostvarenim skupovima videozapisa potrebno je pokrenuti *Python* skripte kojima su omogućene sljedeće funkcionalnosti:

1. Proračun kalibracijskih i distorzijski parametara;
2. Odvajanje sinkroniziranih videozapisa iz jednog mozaik videozapisa;
3. Ispravljanje izobličenja na okvirima sinkroniziranih videozapisa;
4. Spajanje okvira sinkroniziranih videozapisa te korekcija osvjetljenja okvira.

Kako bi se izvršio proračun kalibracijskih i distorzijskih parametara, potrebno je iz skupa kalibracijskih videozapisa spremiti okvire s odgovarajućim položajem kalibracijskog panela te ih pohraniti u direktorij s implementiranim *Python* skriptama. Također, u isti direktorij potrebno je i pohraniti mozaik sinkroniziranih videozapisa. Otvaranjem sučelja CMD potrebno je pozicionirati se u navedeni direktorij te pokrenuti *Python* skripte redoslijedom kojim su već navedene naredbom „*python*“ s nazivom skripte potrebne za pokretanje. Prvim izvršavanjem *Python* skripte ostvareni su kalibracijski i distorzijski parametri pohranjeni u isti direktorij s „*.npz*“ ekstenzijom. Potrebno je tri puta pokrenuti skriptu pri čemu svaki put treba učitati drugi skup kalibracijskih okvira različitih kamera. Drugim korakom ostvaren je i pohranjen skup od tri sinkronizirana videozapisa s područjem preklapanja odvajanjem triju okvira iz učitanog mozaik videozapisa. Trećim korakom potrebno je učitati svaki sinkronizirani videozapis s pripadajućim kalibracijskim i distorzijskim parametrima, čime će se ostvariti skup ispravljenih sinkroniziranih videozapisa. Zadnji korak izvođenja programskog rješenja podrazumijeva učitavanje ostvarenog skupa ispravljenih sinkroniziranih videozapisa, čime će se ostvariti videozapis spojenih sinkroniziranih okvira s ispravljenim različitim intenzitetom osvjetljenja okvira prilikom spajanja.

4. TESTIRANJE PERFOMANSI PREDLOŽENOG RJEŠENJA ZA SPAJANJE OKVIRA DOBIVENIH S VIŠE KAMERA U JEDINSTVENI OKVIR I KOREKCIJU OSVJETLJENJA DOBIVENOG OKVIRA

Unutar ovoga poglavlja prikazat će se rezultati spajanja sinkroniziranih okvira ulaznih videozapisa s područjem preklapanja u pogledu vizualne kvalitete spojenih okvira korištenjem SIFT, SURF te ORB detektora. Nadalje prikazati će se i rezultati ispravljanja različitog intenziteta osvjetljenja sinkroniziranih okvira s područjem preklapanja u formatu boje YUV te HSV u odnosu na spojeni okvir bez ispravljanja različitog intenziteta osvjetljenja. Također tabličnim prikazom bit će izneseno vrijeme izvođenja pojedinih algoritama predloženog rješenja te promjena rezolucije ulaznih okvira i konačna rezolucija spojenih okvira. Vrijeme izvođenja pojedinog algoritma izmjereno je koristeći razvojno računalo s procesorom Intel i7-6700 s 3.40GHz te pokretanjem implementiranih *Python* skripti priloženih u elektroničkom prilogu P.3.1. ovoga rada.

4.1. Ulazni skup videozapisa

Ulazni skup videozapisa za potrebe testiranja rada predloženog rješenja za spajanje okvira videozapisa snimljenih pomoću različitih kamera ADAS ALPHA razvojne ploče ostvareno je snimanjem okoline pomoću tri CMOS kamere, međusobno razmaknute 40 centimetara jedna od druge te međusobno zakrenute pod kutom od 30°, a prikazano slikom 3.25. Kako je horizontalni kut snimanja svake korištene kamere približno 60°, zakretanjem vanjskih kamera za 30° u odnosu na srednju, ostvarilo se područje preklapanja snimane okoline susjednih kamera razvojnog sustava od 30°. Prilikom snimanja videozapisa, kamere razvojnog sustava pričvršćene na nosaču, smještene su u sjeni zgrade tijekom sunčanog dana. Primjer skupa sinkroniziranih okvira ulaznih videozapisa prikazan je slikom 4.1. Rezolucija ulaznih okvira iznosi 1280 x 720. Skup svih videozapisa korištenih u testiranju rada predloženog rješenja nalazi se u elektroničkom prilogu P.4.1. ovoga rada danom na DVD-u priloženom uz rad.

a) b) c)

Sl. 4.1. Primjer sinkroniziranih ulaznih okvira snimljenih kamerama razvojnog sustava (a) ulazni okvir snimljen lijevom kamerom razvojnog sustava (b) ulazni okvir snimljen srednjom kamerom razvojnog sustava (c) ulazni okvir snimljen desnom kamerom razvojnog sustava

4.2. Promjena rezolucije prilikom spajanja okvira

Slikom 4.2. prikazan je rezultat ispravljanja izobličenja snimljenih okvira uzrokovanih tangencijalnim i radijalnim distorzijama prilikom snimanja ulaznih videozapisa.

a)

b)

c)

Sl. 4.2. Primjer ispravljenih izobličenja ulaznih sinkroniziranih okvira: (a) ulazni sinkronizirani lijevi okvir s ispravljenim izobličenjem (b) ulazni sinkronizirani srednji okvir s ispravljenim izobličenjem (a) ulazni sinkronizirani desni okvir s ispravljenim izobličenjem

Iako je rezolucija ispravljenih okvira jednaka rezoluciji ulaznih okvira za predloženo rješenje algoritma ispravljanja izobličenja te iznosi 1280 x 720 elemenata slike, usporedbom odgovarajućih slika sa 4.1. i sa 4.2., vidljivo je kako svaki ispravljeni okvir gubi dio elemenata ulazne slike. Gubitak elemenata slike uzrokovan je postavljanjem *alpha* koeficijenta na vrijednost nula, čime se na ispravljenim okvirima zadržavaju samo točno aproksimirani elementi okvira, a valjani dio slike uvećava na zadanu rezoluciju. Iako gubitak elemenata okvira smanjuje površinu obuhvaćenu kamerama sustava, gubitkom elemenata okvira ostvarena je točnija aproksimacija valjanih elemenata okvira, odnosno postignuto je kvalitetnije ispravljanje okvira u pogledu ispravljanja tangencijalne i radijalne distorzije.

Nadalje, prilikom spajanja okvira pomoću detektora ključnih točaka SIFT, SURF te ORB ostvareni su jedinstveni spojeni okviri, pri čemu su rezolucije različite veličine. Rezolucije ostvarene korištenjem pojedinog detektora su:

- SIFT detektor: 3780 x 720 elemenata slike;
- SURF detektor: 3200 x 720 elemenata slike;
- ORB detektor: 3200 x 720 elemenata slike.

Razlika u rezoluciji spojenih sinkroniziranih okvira s područjem preklapanja uzrokovana je različitim proračunima matrica homografije. Kako se matrica homografije računa spajanjem jednakih ključnih točaka na okvirima za spajanje, jasna je razlika rezolucije transformiranih okvira za spajanje prilikom izvođenja predloženog rješenja, jer svaki detektor različitim metodama traži ključne točke te ne pronalaze jednake ključne točke u odnosu na druga dva korištena detektora. Rezultat spojenih sinkroniziranih okvira s područjem preklapanja prikazati će se i objasniti u sljedećem podpoglavlju.

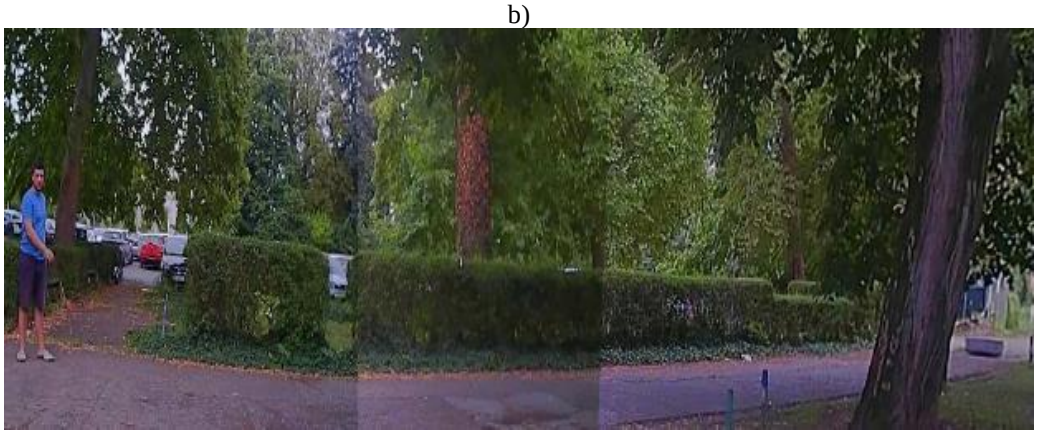
4.3. Rezultati spajanja okvira korištenjem različitih detektora

Implementacijom različitih detektora ključnih točaka SIFT, SURF te ORB, ostvareni su jedinstveni spojeni okviri nastali spajanjem sinkroniziranih okvira s područjem preklapanja ulaznih videozapisa. Slikom 4.3. prikazani su rezultati spajanja navedenih okvira korištenjem različitih detektora. Usporedbom prijelaza između okvira, odnosno šavova spajanja okvira, vidljivo je kako ORB detektor ostvaruje lošiju vizualnu kvalitetu, što je uzrokovano nedostatkom detektiranih ključnih točaka na donjem dijelu područja preklapanja lijevog i spojenog srednjeg i desnog okvira. Nedostatak ključnih točaka na spomenutom donjem dijelu područja preklapanja uzrokuje vizualno lošiji prijelaz na donjem dijelu šava, dok ostatak šava, kao i šav između srednjeg

i desnog okvira ostvaruje zadovoljavajući prijelaz. Nadalje, rezultati ostvareni korištenjem SIFT i SURF detektora ostvaruju oku ugodan prijelaz na šavovima spojenih okvira bez jasno vidljivih razlika.

a)

b)



c)

Sl. 4.3. Primjer spojenih okvira korištenjem različitih detektora: (a) jedinstveni spojeni okvir zasnovan na SIFT detektoru (b) jedinstveni spojeni okvir zasnovan na SURF detektoru (c) jedinstveni spojeni okvir zasnovan na ORB detektoru

Nadalje, na slici 4.3. vidljiva je i razlika u obuhvaćenoj okolini snimljenoj kamerama ADAS ALPHA razvojne ploče, pri čemu je razlika obuhvaćene okoline uzrokovana različitim proračunatim matricama homografije, kao što je objašnjeno u prethodnom podpoglavlju. Također

važno je i napomenuti da vizualno zamućeniji prikaz na srednjem okviru nije uzrokovan primjenom matrice homografije na navedeni okvir, već oštećenjem korištene kamere prilikom snimanja, što je i vidljivo na ulaznom okviru srednje kamere prikazano slikom 4.1.

Također, nedostatak predloženog rješenja vidljiv je i pregledom kreiranog videozapisa sa spojenim okvirima koji sadrži pomične objekte kroz snimljenu okolinu. Prilikom spajanja okvira koji prikazuju navedeni objekt, pri čemu je položaj objekta na području preklapanja susjednih okvira, ne primjenjuje se kvalitetno spajanje što je prikazano slikom 4.4. Netočno spajanje okvira uzrokovano je detekcijom ključnih točaka na prvome skupu sinkroniziranih okvira, čime se novo prisutni objekti bliži kamerama razvojnog sustava spajaju s nedovoljno kvalitetnim prijelazom. Navedeni nedostatak moguće je dobrim dijelom ukloniti detekcijom ključnih točaka svakog skupa sinkroniziranih okvira za spajanje, nauštrb vremena izvođenja samoga rješenja ili implementacijom adaptivne matrice homografije, pri čemu se detekcija ključnih točaka i traženje odgovarajućih parova ključnih točaka provodi na svakom *n*-tom skupu okviru. Adaptivnom matricom homografije veća težina stavlja se na zadnje proračunate matrice homografije, pri čemu starije proračunate matrice homografije omogućuju da se detekcija ključnih točaka ulaznih okvira za spajanje ne provodi na svakom ulaznome skupu sinkroniziranih okvira za spajanje.

Sl. 4.4. Prikaz rezultata spajanja okvira s dinamičnim kretanjem objekata u snimljenoj okolini

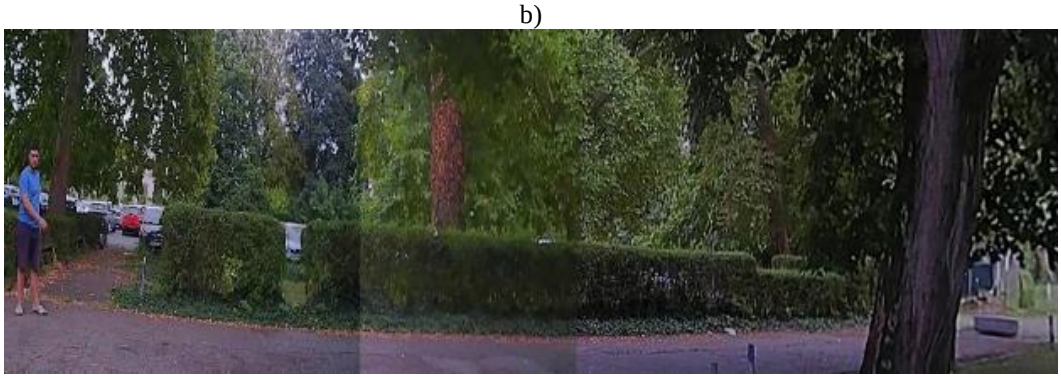
4.4. Rezultati korekcije osvjetljenja okvira za spajanje

Korištenjem metode iz [11] za ispravljanje različitog intenziteta osvjetljenja okvira za spajanje, implementiran je algoritam korekcije osvjetljenja za primjenu na HSV i YUV format boja. Slikom 4.5. prikazan je rezultat korekcije osvjetljenja na okvirima za spajanje u navedenim formatima, u odnosu na spojeni jedinstveni okvir bez implementiranog algoritma korekcije osvjetljenja okvira.

Ispravljanje različitog osvjetljenja na lijevome i desnome okviru prilikom spajanja ostvarilo je oku ugodniju sliku, pri čemu se u oba formata boje intenzitet osvjetljenja na okvirima neznatno ali ipak dovoljno promijenio. Uzrok male promjene intenziteta na okvirima za spajanje ostvaren je proračunom razlike osvjetljenja spojenih parova ključnih točaka susjednih okvira na području preklapanja, pri čemu je međusobni intenzitet osvjetljenja okvira za spajanje male razlike. Kako područje preklapanja susjednih okvira obuhvaća dosta veliku okolinu prilikom snimanja okoline kamerama razvojnog sustava, razlikom intenziteta osvjetljenja referentnog okvira i okvira za spajanje u području preklapanja uspješno se izjednačuje intenzitet osvjetljenja okvira za spajanje, postiže se jedinstveno osvjetljenje cjelokupne spojene slike ali i dalje ne rješava vidljive šavove spajanja okvira.

a)

b)



c)

Sl. 4.5. Prikaz rezultata ispravljena različitog intenziteta osvjetljenja okvira: (a) jedinstveni spojeni okvir bez ispravljenog različitog osvjetljenja okvira za spajanje (b) jedinstveni spojeni okvir s ispravljenim različitim intenzitetom osvjetljenja okvira za spajanje u YUV formatu boje (c)) jedinstveni spojeni okvir s ispravljenim različitim intenzitetom osvjetljenja okvira za spajanje u HSV formatu boje

4.5. Subjektivna ocjena kvalitete spojenih okvira s ispravljenim osvjetljenjem

Kako se rezultati predloženog rješenja korekcije osvjetljenja jedinstvenog spojenog okvira ne mogu evaluirati određenom metrikom, provedena je anketa za subjektivnu ocjenu rezultata korekcije osvjetljenja spojenih okvira. Za potrebe ankete, iz kreiranog videozapisa nastalog izvođenjem predloženog rješenja, izdvojeno je 50 okvira bez ispravljenog osvjetljenja koji su služili kao referente slike za usporedbu, istih 50 okvira s ispravljenim osvjetljenjem u formatu boje HSV te istih 50 okvira s ispravljenim osvjetljenjem u formatu boje YUV. Ukupno 100 slika za evaluaciju, 50 referentnih slika za usporedbu kao i rezultati ankete prikazani *Excel* tablicom nalaze se u elektroničkim prilozima P.4.2. te P.4.3. ovoga rada danim na DVD-u priloženom uz rad.. U anketi je sudjelovalo 10 ispitanika, pri čemu su ispitanici ocijenili vizualnu kvalitetu ispravljenog osvjetljenja svakog okvira u YUV i HSV formatu, usporedbom s tim okvirom bez korekcije osvjetljenja, ocjenom u rasponu 0 - 10. Tablicom 4.1. prikazane su prosječne ocjene korekcije osvjetljenja primjenom u YUV i HSV formatu boje svakog ispitanika te ukupne prosječne ocijene svih sudionika za pojedini format boje.

Tablica 4.1. Rezultati ankete subjektivne ocjene kvalitete spojenih okvira s ispravljenim osvjetljenjem

ISPITANICI:	***Prosječna ocjena kvalitete ispravljenog osvjetljenja primjenom na YUV format boje***	***Prosječna ocjena kvalitete ispravljenog osvjetljenja primjenom na HSV format boje***
ISPITANIK 1	8.06	7.66
ISPITANIK 2	7.94	7.50
ISPITANIK 3	8.26	7.82
ISPITANIK 4	8.16	7.56
ISPITANIK 5	7.86	7.66
ISPITANIK 6	8.18	7.56
ISPITANIK 7	8.26	7.48
ISPITANIK 8	8.28	7.46
ISPITANIK 9	8.38	7.56
ISPITANIK 10	8.34	7.24
UKUPNA PROSJEČNA OCJENA	**8.17**	**7.55**

Razlika u subjektivnim prosječnim ocjenama vizualnog prikaza korekcije osvjetljenja između korištenih formata boja, uzrokovana je proračunom razlike intenziteta osvjetljenja parova ključnih točaka na komponenti osvjetljenja pojedinog formata boje. Iz tog razloga jasna je razlika u konačnim prikazima ispravljenih osvjetljenja dobivenih jedinstvenih okvira te razlike u subjektivnoj ocjeni pojedinog ispitanika. Iz rezultata ankete vidljivo je kako se kvalitetniji prikaz

ostvario primjenom korekcije osvjetljenja na YUV formatu u odnosu na primjenu korekcije osvjetljenja na HSV format boje. Poboljšanje vizualne kvalitete programskog rješenja korekcije osvjetljenja moguće je postići smanjenjem područja preklapanja te većim brojem parova odgovarajućih ključnih točaka na okvirima za spajanje. Smanjenjem područja preklapanja postigao bi se točniji proračun razlike intenziteta osvjetljenja okvira za spajanje u području preklapanja, jer smanjenjem područja preklapanja manje su oscilacije vrijednosti krajnjeg proračuna, kako se udaljeniji parovi ključnih točaka ne bi uzimali u obzir prilikom proračuna. Nadalje, poboljšanje je moguće ostvariti i primjenom već spomenute adaptivne matrice homografije, čime bi se osvjetljenje uzimalo na stvarnim lokacija odgovarajućih parova ključnih točaka umjesto na lokacijama parova odgovarajućih ključnih točaka proračunatih na prvome skupu ulaznih okvira za spajanje.

4.6. Vrijeme izvođenja algoritama predloženog rješenja

Kako bi se utvrdila moguća uporaba detektora ključnih točaka za primjenu u algoritmima spajanja okvira s područjem preklapanja dobivenih kamerama vozila, za upotrebu u ADAS algoritmima ili za upotrebu unutar centra za obradu podataka, izmjerena su vremena izvođenja pojedinih algoritama potrebnih za ostvarivanje spajanja okvira za rad u stvarnome vremenu. Tablicama 4.2., 4.3. te 4.4. prikazana su vremena izvođenja algoritama ovisno u korištenom detektoru ključnih točaka, kako algoritam proračuna matrice homografije te algoritam proračuna razlike intenziteta osvjetljenja ovise o pronađenim ključnim točkama pojedinog detektora.

Usporedbom vremena detekcije ključnih točaka pojedinog algoritma, vidljivo je kako bi se za vrijeme od jedne do dvije sekunde uspjelo detektirati potrebne ključne točke svakim korištenim detektorom, što za potrebe algoritma spajanja više okvira snimljenih različitim kamerama vozila nije dovoljno za rad u stvarnome vremenu. Nadalje, usporedbom vremena izvođenja algoritma povezivanja jednakih ključnih točaka, jasno je kako veća količina detektiranih ključnih točaka na predanim okvirima utječe na potrebno vrijeme izvođenja navedenog algoritma. Što veći broj detektiranih ključnih točaka okvira za spajanje, iako ostvaruje kvalitetniji proračun matrice homografije, a s time i kvalitetniju transformaciju okvira za spajanje, uzrokuje potrebu za dužim vremenom izvođenja algoritma povezivanja jednakih ključnih točaka na području preklapanja, jer se algoritmom provjeravaju svi šturo povezani parovi ključnih točaka na cjelokupno predanim okvirima te naknadno uzimaju samo oni parovi koji odgovaraju zadanim parametrima i pragovima. Rezultatom vremena izvođenja algoritma povezivanja ključnih točaka u tablici 4.4. prilikom korištenja detektora ORB, pri čemu je postavljen traženi broj ključnih točaka na sto tisuća, jasno

je vidljivo kako veliki broj detektiranih ključnih točaka svakog okvira za spajanje može uzrokovati duže vrijeme izvođenja u odnosu na vrijeme izvođenja s puno manjim brojem detektiranih ključnih točaka pronađenih korištenjem SIFT te SURF detektora ključnih točaka.

Tablica 4.2. Vrijeme izvođenja algoritama na prva tri sinkronizirana okvira korištenjem SIFT detektora.

Detektor ključnih točaka	*Lijevi okvir*	*Spojeni srednji i desni okvir*	*Srednji okvir*	*Desni okvir*
SIFT	0.31915 sec	0.61233 sec	0.30518 sec	0.45482 sec
• *Algoritam povezivanja jednakih ključnih točaka*	7.3244		5.5041 sec	
• *Algoritam proračuna razlike osvjetljenja spojenih ključnih točaka*	0.001701 sec		0.001792 sec	
• *Algoritam korekcije osvjetljenja okvira za spajanje*	0.00698 sec	X	X	0.01296 sec
• *Proračun matrice homografije*	0.0009984		0.0013626	

Tablica 4.3. Vrijeme izvođenja algoritama na prva tri sinkronizirana okvira korištenjem SURF detektora.

Detektor ključnih točaka	*Lijevi okvir*	*Spojeni srednji i desni okvir*	*Srednji okvir*	*Desni okvir*
SURF	0.35804 sec	0.60139 sec	0.33409 sec	0.55967 sec
• *Algoritam povezivanja jednakih ključnih točaka*	1.1469 sec		0.70312 sec	
• *Algoritam proračuna razlike osvjetljenja spojenih ključnih točaka*	0.001422 sec		0.001381 sec	
• *Algoritam korekcije osvjetljenja okvira za spajanje*	0.00698 sec	X	X	0.01296 sec
• *Proračun matrice homografije*	0.0008913 sec		0.0009995	

Tablica 4.4. Vrijeme izvođenja algoritama na prva tri sinkronizirana okvira korištenjem ORB detektora.

Detektor ključnih točaka	*Lijevi okvir*	*Spojeni srednji i desni okvir*	*Srednji okvir*	*Desni okvir*
ORB	0.09498 sec	0.16300 sec	0.08987 sec	0.27237 sec
• *Algoritam povezivanja jednakih ključnih točaka*	21.09836 sec		15.486567	
• *Algoritam proračuna razlike osvjetljenja spojenih ključnih točaka*	0.000998 sec		0.000963 sec	
• *Algoritam korekcije osvjetljenja okvira za spajanje*	0.00698 sec	X	X	0.01296 sec
• *Proračun matrice homografije*	0.0007022 sec		0.0009973 sec	

Nadalje, ostvarenim povezanim parovima odgovarajućih ključnih točaka na području preklapanja okvira za spajanje, izmjereno je vrijeme potrebno za proračun razlike intenziteta osvjetljenja okvira za spajanje. Iako su vremena izvođenja izrazito kratka te dovoljna i za ADAS algoritme, temelj algoritma proračuna razlike intenziteta osvjetljenja okvira za spajanje i dalje su detektirane ključne točke s pripadajućim deskriptorima, čime se upotreba ovoga pristupa i dalje smatra neisplativom. Vrijeme izvođenja navedenog algoritma razlikuje se kako između različitih detektora tako i unutar istoga prilikom proračuna razlike intenziteta osvjetljenja između lijevog i srednjeg te srednjeg i desnog okvira. Razlika u vremenu jasna je jer se svakim detektorom ključnih točaka ne pronalazi jednak broj parova istih ključnih točaka na okvirima za spajanje u području preklapanja te se ne pronalazi ni jednak broj spomenutih parova korištenjem istog detektora na različitim parovima okvira za spajanje. Zbog istog razloga jasna je i razlika potrebnog vremena za proračun matrice homografije, kako korištenjem istog detektora na različitim parovima okvira za spajanje, tako i korištenjem različitih detektora. Također, na vrijeme proračuna matrice homografije utječe i broj pronađenih parova odgovarajućih ključnih točaka u području preklapanja okvira, pri čemu veći broj zahtjeva duže vrijeme izvođenja, a sukladno tome manji broj parova jednakih ključnih točaka kraće vrijeme izvođenja.

Usporedbom vremena izvođenja algoritma ispravljanja različitog intenziteta osvjetljenja na okvirima za spajanje, odnosno na lijevome i desnome okviru, vidljivo je kako su vremena izvođenja jednaka. Razlog jednakosti vremena izvođenja vidljiv je implementacijom algoritma ispravljanja intenziteta osvjetljenja okvira za spajanje, pri čemu se osvjetljenje okvira za spajanje ispravlja jednako za sva tri detektora te ovise samo o ulaznim okvirima koji su jednaki za sva tri detektora te jednom proračunatom vrijednošću razlike intenziteta.

4.7. Osvrt na predloženo rješenje i dobivene rezultate

Kako je predloženim rješenjem ostvaren skup videozapisa s jedinstveno spojenim okvirima s ispravljenim različitim intenzitetom okvira za spajanje iz ulaznog skupa sinkroniziranih videozapisa, korištenjem različitih detektora, ovim podpoglavljem naglasit će se prednosti i nedostatci predloženog rješenja. Iako oku ugodan prikaz spojenih okvira omogućava vozaču lakši pregled veće okoline iza vozila u odnosu na klasični „pogled preko ramena“, evaluacija rezultata vizualnih prikaza temelji se na subjektivnoj procjeni sudionika ankete za procjenu kvalitete rezultata. Nedostatak predloženog rješenja vidljiv je u jasnim šavovima na spojevima susjednih okvira, što bi se moglo riješiti implementacijom prvo metode korekcije različitog osvjetljenja okvira za spajanje metodom iz [10], a zatim naknadnom korekcijom osvjetljenja predloženom

metodom ovoga rada, odnosno metodom rada iz [11]. Nadalje, nedostatak predloženog rješenja diplomskog rada vidljiv je i u vremenu izvođenja pojedinih algoritama. Vrijeme izvođenja detekcije ključnih točaka na okvirima za spajanje pa tako i ostalih algoritama koji ovise o detektiranim ključnim točkama, mogu se ubrzati određivanjem područja interesa na područje preklapanja, čime bi se ograničio broj detektiranih ključnih točaka svakog okvira za spajanje na manji broj detektiranih ključnih točaka potrebnih za proračun matrica homografije te razlike intenziteta osvjetljenja okvira za spajanje, odnosno izostavio bi se veći dio okvira za detektiranje ključnih točaka od kojega svakako nema koristi. Iako se može postići brže vrijeme izvođenja pojedinih algoritama, rezultati testiranja ostvareni su pomoću procesora Intel i7-6700 s 3.40GHz koji je puno veće obradive moći od procesora ADAS ALPHA ploče čiji takt obrade na određenim procesorima može doseći maksimum do 1150 MHz. Povodom toga, jasno je kako implementacija detektora s deskriptorima korištenih metoda u ovome radu nije isplativa za ADAS algoritme, ali omogućuju jednostavnu i dovoljno brzu obradu podataka za implementaciju u centrima za obradu podataka prometa, pri čemu bi se trebao riješiti problem komunikacije vozila te okolne infrastrukture za rad u stvarnome vremenu. Kako je ovim radom predstavljen problem implementacije postojećih detektora ključnih točaka, jasan je pristup razvoja algoritma spajanja više okvira dobivenih kamerama vozila predloženih radom [1], čiji se pristup nije mogao replicirati te poboljšati zbog ograničenja korištene inačice VisionSDK te starije inačice TDA SoC-a u odnosu na inačice korištene u [1].

5. ZAKLJUČAK

Problem ovog diplomskog rada s ciljem spajanja više okvira s područjem preklapanja dobivenih različitim kamerama vozila te ispravljanja različitog intenziteta osvjetljenja na snimljenim okvirima u jedinstveni spojeni okvir, zasniva se na implementaciji detektora ključnih točaka SIFT, SURF te ORB. Detektiranjem ključnih točaka pronađeni su jednaki elementi susjednih okvira za spajanje s područjem preklapanja potrebni za proračun matrica homografije. Matricom homografije omogućeno je jednostavno spajanje sinkroniziranih okvira ulaznih videozapisa. Nadalje, problem različitog osvjetljenja riješen je proračunom prosječne razlike intenziteta spojenih parova jednakih ključnih točaka u području preklapanja okvira za spajanje. Prosječna razlika intenziteta parova ključnih točaka primijenjena je na Y kanal YUV formata boje te V kanal formata boje HSV. Implementacijom *UseCase*-a s algoritmima pohrane videozapisa, pomoću ADAS ALPHA razvojne ploče te triju CMOS kamera ostvareni su potrebni ulazni videozapisi implementiranih *Python* skripti za proračun kalibracijskih parametara korištenih kamera te mozaik videozapis potreban kao ulazni videozapis predloženog rješenja za spajanje više okvira s područjem preklapanja.

Izvođenjem implementiranih *Python* skripti te prikazanim rezultatima, ostvareni su i vidljivi jedinstveni spojeni okviri. Jedinstvenim spojenim okvirom prikazuje se veća snimljena okolina nego što se može obuhvatiti snimanjem pomoću jedne kamere vozila. Nadalje, ispravljen je različit intenzitet osvjetljenja okvira za spajanje, čime se ostvario oku ugodan prikaz spojenih okvira s jedinstvenim osvjetljenjem. Iako predloženim rješenjem nije ostvareno uklanjanje vidljivih šavova spajanja okvira, implementacijom *improved gradual fusion* metode postigli bi se kvalitetniji jedinstveni spojeni okviri bez jasno vidljivih šavova. Nadalje, nedostatak predloženog rješenja ovoga rada također je vidljiv u pogledu potrebnog vremena izvođenja pojedinih algoritama, čije se vrijeme izvođenja dodatno treba skratiti određivanjem područja interesa. Određivanjem područja interesa izostavlja se obrada većeg dijela ulaznih okvira koji su svakako nepotrebni prilikom proračuna matrice homografije te razlike intenziteta osvjetljenja okvira za spajanje. Primjena ovako implementiranog rješenja moguća je za centre za obradu podataka, koji bi s vozilima komunicirali putem *Vehicle to Infrastructure* komunikacije, a ne za primjenu u ADAS algoritmima koji se izvode u samom vozilu, jer zahtijevaju puno veću računalnu moć nego što ADAS može pružiti.

LITERATURA

[1] J. Pan, V. Appia, J. Villarreal, L. Weaver, i D.-K. Kwon, „Rear-Stitched View Panorama: A Low-Power Embedded Implementation for Smart Rear-View Mirrors on Vehicles“, u *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, srp. 2017, str. 1184–1193, doi: 10.1109/CVPRW.2017.157.

[2] E. R. Fossum i D. B. Hondongwa, „A Review of the Pinned Photodiode for CCD and CMOS Image Sensors“, *IEEE J. Electron Devices Soc.*, sv. 2, izd. 3, str. 33–43, svi. 2014, doi: 10.1109/JEDS.2014.2306412.

[3] S. Wang, S. Song, i X. Shi, „An improved adaptive correction method for camera distortion“, u *Proceedings of the 33rd Chinese Control Conference*, srp. 2014, str. 4821–4825, doi: 10.1109/ChiCC.2014.6895756.

[4] D. G. Lowe, „Distinctive Image Features from Scale-Invariant Keypoints“, *Int. J. Comput. Vis.*, sv. 60, izd. 2, str. 91–110, stu. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[5] H. Bay, T. Tuytelaars, i L. Van Gool, „SURF: Speeded Up Robust Features“, u *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 2006, str. 404–417, doi: 10.1007/11744023_32.

[6] E. Rublee, V. Rabaud, K. Konolige, i G. Bradski, „ORB: An efficient alternative to SIFT or SURF“, u *2011 International Conference on Computer Vision*, stu. 2011, str. 2564–2571, doi: 10.1109/ICCV.2011.6126544.

[7] E. Rosten i T. Drummond, „Machine Learning for High-Speed Corner Detection“, u *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 2006, str. 430–443, doi: 10.1007/11744023_34.

[8] M. Calonder, V. Lepetit, C. Strecha, i P. Fua, „BRIEF: Binary Robust Independent Elementary Features“, u *Computer Vision – ECCV 2010*, Berlin, Heidelberg, 2010, str. 778–792, doi: 10.1007/978-3-642-15561-1_56.

[9] S.-H. Yeh i S.-H. Lai, „Real-time video stitching“, u *2017 IEEE International Conference on Image Processing (ICIP)*, ruj. 2017, str. 1482–1486, doi: 10.1109/ICIP.2017.8296528.

[10] C. Xiu i Y. Ma, „Image Stitching Based on Improved Gradual Fusion Algorithm“, u *2019 Chinese Control And Decision Conference (CCDC)*, lip. 2019, str. 2933–2937, doi: 10.1109/CCDC.2019.8832814.

[11] A. A. Mohammed, F. Ming, i R. Zhengwei, „Color Balance for Panoramic Images“, *Mod. Appl. Sci.*, sv. 9, izd. 13, Art. izd. 13, stu. 2015, doi: 10.5539/mas.v9n13p140.

[12] A. Lindgren i F. Chen, „State of the Art Analysis: An Overview of Advanced Driver Assistance Systems (ADAS) and Possible Human Factors Issues“, *undefined*, 2006. /paper/State-of-the-Art-Analysis%3A-An-Overview-of-Advanced-Lindgren-Chen/6f2d1b9bbd563f8ff2c3c23691df49d249877088 (pristupljeno ruj. 03, 2020).

[13] „(PDF) Review of advanced driver assistance systems (ADAS)“, *ResearchGate*. https://www.researchgate.net/publication/321364551_Review_of_advanced_driver_assistance_systems_ADAS (pristupljeno ruj. 03, 2020).

[14] „RT-RK - Automotive“. https://www.rt-rk.com/services/automotive (pristupljeno ruj. 14, 2020).

[15] „CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) | TI.com“. https://www.ti.com/tool/CCSTUDIO?utm_source=google&utm_medium=cpc&utm_campaign=epd-null-null-GPN_EN_EVM-cpc-evm-google-wwe&utm_content=CCSTUDIO&ds_k=CCSTUDIO&DCM=yes&gclid=CjwKCAjw4rf6BRAvEiwAn2Q76lC6afXF33qh9eNr5RgTGBdYs8_hLIuI9mhVAe89C1EzHn6EtbquHRoCyM4QAvD_BwE&gclsrc=aw.ds (pristupljeno ruj. 03, 2020).

[16] T. pip developers, *pip: The PyPA recommended tool for installing Python packages.* .

[17] „OpenCV“. https://opencv.org/ (pristupljeno ruj. 03, 2020).

[18] „NumPy“. https://numpy.org/about/ (pristupljeno ruj. 03, 2020).

[19] A. Rosebrock, *imutils: A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.* .
[20] „Pillow — Pillow (PIL Fork) 3.0.0 documentation“. https://pillow.readthedocs.io/en/3.0.x/index.html (pristupljeno ruj. 03, 2020).
[21] „android - Rotate YUV420Sp image by 90 degrees counter clockwise“, *Stack Overflow*. https://stackoverflow.com/questions/31859708/rotate-yuv420sp-image-by-90-degrees-counter-clockwise (pristupljeno ruj. 10, 2020).
[22] „Camera Calibration — OpenCV-Python Tutorials 1 documentation“. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html (pristupljeno ruj. 08, 2020).
[23] „Geometric Image Transformations — OpenCV 2.4.13.7 documentation“. https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html?highlight=remap (pristupljeno ruj. 08, 2020).

SAŽETAK

Ovim radom razvijeno je programsko rješenje za spajanje više okvira s područjem preklapanja snimljenih kamerama vozila, kako bi se postigao prikaz šire okoline vozila u odnosu na onu prikazanu jednom kamerom vozila. Predloženo rješenje namijenjeno je za spajanje okvira snimljenih kamerama vozila unutar bočnih zrcala vozila, te jedne kamere sa stražnje strane vozila. Rješenje je postignuto snimanjem odgovarajućeg ulaznog skupa videozapisa, pri čemu je odrađeno ispravljanje izobličenja ulaznih okvira, spajanje sinkroniziranih okvira te korekcija osvjetljenja okvira za spajanje. Predloženo rješenje ostvaruje rezultat u pogledu jedinstvenih spojenih okvira testiranjem na razvojnome računalu pomoću implementiranih *Python* skripti potrebnih funkcionalnosti, a rezultati su prikazani te evaluirani subjektivnom procjenom sudionika ankete od strane 10 neutralnih gledatelja. Iz rezultata provedene ankete može se zaključiti da je primjena programskog rješenja za korekciju osvjetljenja na jedinstvenom spojenom okviru ostvarila kvalitetniji prikaz rezultata primjenom na YUV format boje, u odnosu na primjenu na HSV formatu boje.

Ključne riječi: detekcija ključnih točaka, korekcija osvjetljenja, spajanje slika, OpenCV, ADAS

STITCHING OF IMAGES CAPTURED BY MULTIPLE CAMERAS INTO A SINGLE IMAGE AND PHOTOMETRIC CORRECTION FOR THE USAGE IN ADAS ALGORITHMS

ABSTRACT

This paper developed a software solution for stitching multiple frames with the overlap area recorded by the vehicle cameras, in order to achieve a view of the wider vehicle environment in relation to that shown by a single vehicle camera. The proposed solution is intended for stitching frames captured by vehicle cameras inside the side mirrors of the vehicle, and one camera at the rear of the vehicle. The solution was achieved by recording the appropriate video input set, correcting the distortion of the input frames, stitching the synchronized frames, and correcting the brightness of the stitched frames. The proposed solution achieves a result in terms of unique stitched frames by testing on a development computer using implemented Python scripts of the required functionality and the results are presented and evaluated by subjective evaluation of survey participants by 10 neutral viewers. By conducting the survey, the application of a software solution for luminance correction on a single stitched frame, achieved a better presentation of the results by applying to the YUV color format compared to the application to the HSV color format.

Keywords: keypoint detection, brightness correction, image stitching, OpenCV, ADAS

ŽIVOTOPIS

Josip Kundid, univ. bacc. ing. comp. rođen je 9. srpnja 1996. godine u Osijeku, Republika Hrvatska. Završenom osnovnom školom „Frana Krste Frankopana Osijek“, srednjom školom „III. Prirodoslovno matematička gimnazija Osijek“ te preddiplomskim studijem „Računarstvo“ na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija, upisuje diplomski studij „Automobilsko računarstvo i komunikacije“ na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek, Sveučilišta J. J. Strossmayera u Osijeku. Tijekom pisanja diplomskoga rada, Josip Kundid stipendist je Instituta Računalnih tehnologija i računalnih komunikacija (Institut RT-RK Osijek).

_____

Potpis autora

PRILOZI

P.3.1. – Mapa u kojoj se nalaze sve potrebne Python skripte za uspješno pokretanje programskog rješenja

P.4.1. – Mapa s ulaznim videozapisima korištenih za ostvarivanje rezultata

P.4.2. – Mapa sa skupom slika korištenih za provedbu evaluacijske ankete

P.4.3. – Mapa s rezultatima evaluacijske ankete programskog rješenja korekcije osvjetljenja