Prikaz stanja prostora oko automobila iz različitih pogleda pomoću sustava kamera na automobilu

Buljeta, Daniel

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

Sveučilišni studij

PRIKAZ STANJA PROSTORA OKO AUTOMOBILA IZ RAZLIČITIH POGLEDA POMOĆU SUSTAVA KAMERA NA AUTOMOBILU

Diplomski rad

Daniel Buljeta

Osijek, 2018.

SADRŽAJ

1. UVOD

Automobilska industrija je znatno napredovala u posljednjem desetljeću. Svakim danom razvijaju se vozila koja su sve samostalnija prilikom vožnje, pouzdanija, sigurnija, manje zagađuju okoliš i troše manje sirovina. Vozila koja imaju ugrađenu umjetnu inteligenciju nazivaju se autonomnim vozilima i sve su zastupljenija na tržištu. Umjetna inteligencija omogućuje vozilima da neke funkcije u vožnji obavljaju samostalno, bez interakcije vozača. Informacije potrebne za rad umjetne inteligencije samostalnih vozila prilikom vožnje prikupljaju se najčešće preko kamera postavljenih na automobilu. Pomoću informacija dobivenih preko kamera automobil odlučuje kako reagirati u određenom trenutku.

U radu je potrebno osmisliti računalni algoritam za generiranje pogleda prostora oko automobila. Nakon analize postojećih algoritama, potrebno je razviti svoj vlastiti algoritam u nekom od viših programskih jezika, zatim ga prebaciti u C programski jezik. Kada je algoritam prebačen u C programski jezik, potrebno ga je pokrenuti na ADAS (engl. *Advanced Driver Assistance System*) ploči. ADAS ploča predstavlja hardverski dio koji služi za pokretanje stvarno-vremenskih aplikacija, na kojoj bi se izvršavao algoritam. U ovom diplomskom radu stvoren je i opisan algoritam koji omogućuje prikaz stanja prostora oko automobila iz različitih pogleda (engl. *surround view*). Sustav u koji se algoritam može implementirati radi pomoću četiri kamere postavljene na različitim položajima na automobilu i obrađene informacije prikazuje na zaslonu unutar automobila. Pogled odozgo prikazuje prostor 360° oko automobila. Pogledi s različitih kamera su stopljeni u jedan obavljanjem različitih geometrijskih operacija nad trenutnim okvirima svake kamere.

U drugom poglavlju diplomskog rada opisani su osnovni pojmovi strojnog vida (engl. *machine vision*) i objašnjeni neki od postojećih algoritama za generiranje pogleda prostora oko automobila. Treće poglavlje opisuje razvoj vlastitog algoritma za kalibraciju kamere i algoritma za generiranje pogleda prostora oko automobila, dok su u četvrtom poglavlju opisana testiranja tog algoritma. Peto poglavlje donosi zaključke rada.

2. GENERIRANJE POGLEDA PROSTORA OKO AUTOMOBILA

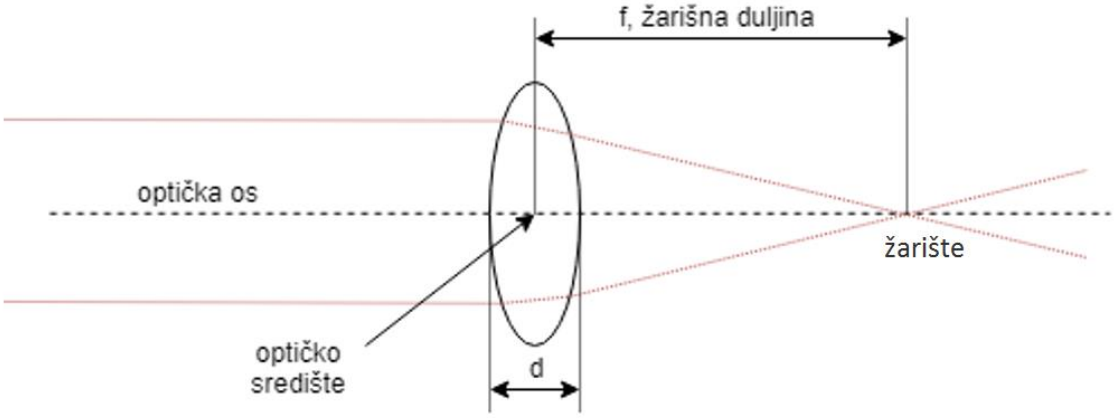
Kako bi se lakše razumjeli postojeći algoritmi generiranja prostora oko automobila, ukratko su objašnjeni neki od pojmova projiciranja scene i geometrijskih operacija nad slikom.

2.1. Projiciranje 3D scene na 2D sliku

3D (engl. *three-dimensional*) scena predstavlja trodimenzionalni prostor, koji sadrži: visinu, širinu, dubinu, dok 2D (engl. *two-dimensional*) scena predstavlja sliku, izraženu s dvije dimenzije visinom i širinom.

Umjetna inteligencija automobila djeluje na osnovu informacija dobivenih sa senzora postavljenih na automobilu. Najčešći senzor je kamera, koja predstavlja strojni vid. Kako bi se razumio strojni vid, bitno je znati kako je točka iz 3D svijeta projicirana (engl. *mapped*) na točnu poziciju na slici (2D svijet). Parametri koji opisuju položaj i orijentaciju kamere u odnosu na stvarni svijet predstavljaju ekstrinzične parametre kamere, dok su intrinzični parametri kamere konstanta kamere i ostali parametri koji utječu na projiciranje točke iz 3D svijeta na sliku. Prije nego se definiraju ti parametri potrebno je poznavati pojmove u optici (optička os, žarišna duljina, žarište):

- Optička os je zamišljena linija koja prolazi kroz centar zakrivljene leće te se podudara s osi simetrije leće. Na slici ispod vidljiva je skica leće s naznačenim osnovnim pojmovima.



Sl.2.1. *Žarišna duljina, optička os, žarište leće*

- Žarišna duljina predstavlja udaljenost između središta leće i žarišta, ovisi o obliku i o tvari od koje je leća napravljena. Postoje sabirne i rastresene leće, gdje je žarišna duljina sabirnih pozitivna, a rastresenih negativna.
- Žarište ili fokus, u geometrijskoj optici je točka kroz koju prolaze sve zrake svjetlosti, koje padaju na neki optički sustav paralelno s optičkom osi tog sustava. Kod sabirnih leća te

zrake se sijeku u žarištu dok kod rastresenih se sijeku samo njihovi produžeci u smjeru suprotnom od smjera širenja.

Postoji nekoliko koordinatnih sustava koji su bitni za razumijevanje projiciranja podataka iz 3D svijeta na 2D sliku:

- Koordinatni sustav objekta (KSO): koordinatni sustav objekta kojeg projiciramo pomoću kamere (3D scena).
- Koordinatni sustav kamere (KSK): sve je opisano relativno u odnosu na ishodište ovog koordinatnog sustava. Projekcija između ova dva koordinatna sustava (KSO u KSK) se postiže translacijom i rotacijom podataka (3D scena). Definira se položaj koordinatnog sustava kamere u odnosu na koordinatni sustav objekta. Transformacija iz 3D u 3D scenu, ne dolazi do gubitaka dimenzija.
- Slikovni koordinatni sustav (SKS): koordinatni sustav 3D svijeta se projicira na slikovni koordinatni sustav (2D). Transformacija 3D u 2D svijet, dolazi do gubitka jedne dimenzije.
- Koordinatni sustav senzora kamere (KSSK): ovaj koordinatni sustav opisuje koji element slike odgovara s kojim elementom 3D svijeta. Ovo je dvodimenzionalni koordinatni sustav.

Transformacija 3D scene u 2D (Sl.2.2.), odnosno projiciranje 3D podataka u 2D podatke definirano je izrazom (2-1) [1].

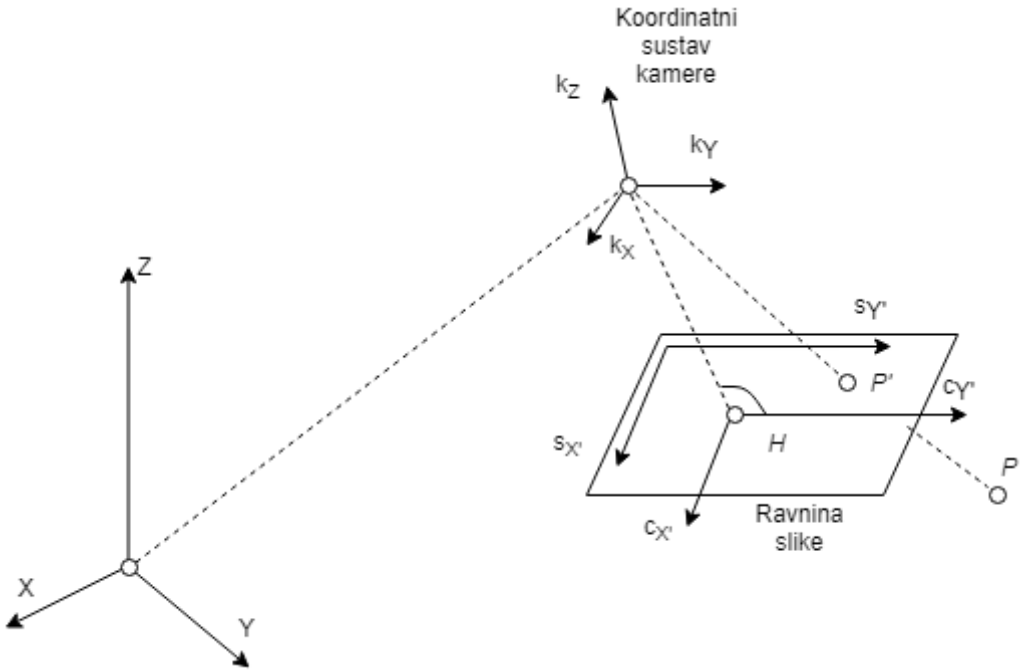
$$\begin{bmatrix} s_X \\ s_Y \\ 1 \end{bmatrix} = s_{Hc} c_{Hk} k_{Ho} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2-1}$$

gdje je:

- matrica $\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$ predstavlja koordinate u KSO (x, y, z) koordinate u 3D svijetu, 1 znači da se radi o homogenom koordinatnom sustavu
- k_{Ho} definira transformaciju iz KSO u KSK (opisuje gdje se kamera nalazi u 3D svijetu)
- c_{Hk} definira transformaciju, odnosno projiciranje točke iz KSK na ravninu slike
- s_{Hc} projiciranje točke s ravnine slike na ravninu senzora (2D)

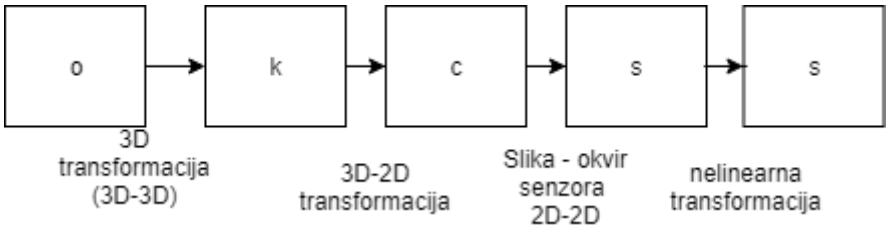
Na Sl.2.2. skicirana je transformacija točke iz 3D koordinatnog sustava u 2D koordinatni sustav. U početku je jednostavna transformacija KSO u KSK (položaj kamere), KSK i SKS su paralelni te se pod pravim kutom projicira točka iz KSK u SKS, što definira glavnu točku (engl. *principal point*) pri čemu udaljenost između ishodišta ta dva koordinatna sustava predstavlja konstantu

kamere. Zatim se definira transformacija iz SKS u KSSK i primjetno da je KSSK translatiran u odnosu na SKS, ta translacija je definirana parametrima kamere.



Sl.2.2. *Projekcija točke objekta na senzor kamere [1]*

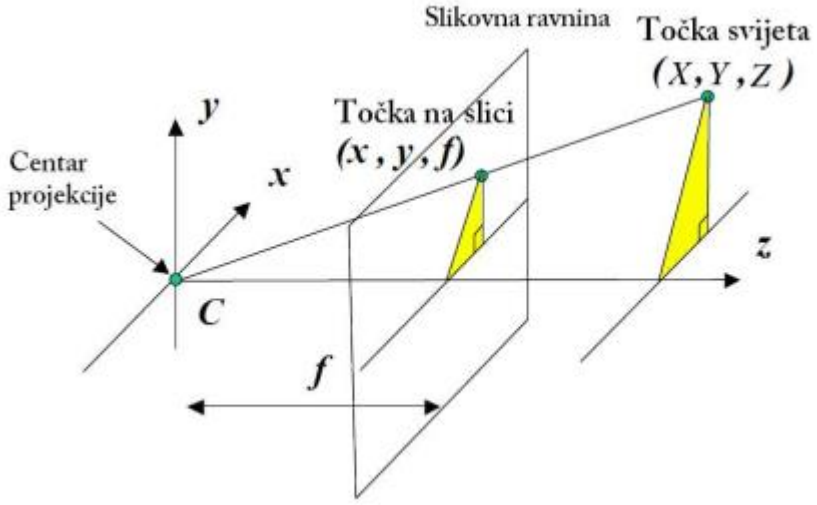
Slijed transformacija potrebnih za projiciranje točke 3D svijeta u točku 2D svijeta vidljiv je na slici ispod (Sl.2.3). Ukoliko kamera posjeduje idealnu leću (bez distorzije itd.) tada proces potrebnih transformacija ne sadrži nelinearnu transformaciju. Nelinearnu transformaciju prilikom projiciranja točke posjeduju širokokutne kamere kod kojih dolazi do nelinearnog projiciranja elementa slike, čime se stvara distorzija slike. Ekstrinzični parametri (matrica rotacije u kojoj je definirana rotacija po sve tri osi i vektor translacije) određeni su 3D transformacijom (o i k kvadratićem) i definiraju položaj kamere u stvarnom svijetu. Intrinzični parametri su svojstveni za svaku kameru i definiraju projiciranje scene ispred kamere na konačnu sliku. Kalibracijom kamere prikupljaju se intrinzični i ekstrinzični parametri.



Sl.2.3. *Transformacije prilikom projekcije točke 3D svijeta u 2D sliku[1]*

2.1.1. Kalibracija kamere

Kalibracija kamere predstavlja proces utvrđivanja parametara kamere i čini poveznicu između točke u svijetu (X, Y, Z) s koordinatama elementa slike u koji je projicirana ta točka 3D svijeta. Suvremene kamere se uspoređuju s osnovnim modelom kamere, tzv. *pinhole* kamerom. *Pinhole* kamera je jednostavna kamera bez leća, s malim otvorom konusnog oblika kroz koji prolazi svjetlost. Što je veći otvor, slika je manje oštra i duljina fokusa je definirana kutom konusa, a žarišna duljina određuje uvećanje slike[2]. Suvremene kamere sadrže unutar sebe leću, te time ne odgovaraju standardnoj *pinhole* kameri, odnosno odstupaju od tog osnovnog modela kamere. Kalibracijom kamere dobivaju se određeni parametri koji matematičkim operacijama nad slikom uklanjaju odstupanja nastale slike od slike koja bi se dobila *pinhole* modelom kamere. Parametri koji se izračunavaju kalibracijom kamere dijele se na intrinzične i ekstrinzične parametre. Intrinzični (unutarnji) parametri sadrže podatke koji sadrže informaciji o projekciji 3D prostora u 2D prostor, dok ekstrinzični podaci određuju položaj kamere u odnosu na stvarni svijet. Intrinzičnih parametara je 5 i dani su matricom intrinzičnih parametara (**I** matrica). Na Sl.2.4. vidljiv je položaj točke u svijetu i njena projekcija na slikovnu ravninu [3].



Sl.2.4. *Projekcija točke svijeta na slikovnu ravninu u ovisnosti o žarišnoj duljini[3]*

Jednadžbe projekcije točke određenih koordinata dani su izrazima (2-2, 2-3)[3]:

$$x = f\frac{X}{Z} \tag{2-2}$$

$$y = f\frac{Y}{Z} \tag{2-3}$$

gdje je:

- *x, y* - koordinate točke u slikovnoj ravnini
- *X, Y, Z* - koordinate točke u KSO
- f - žarišna duljina

Zbog nesavršenosti proizvodnje leće i njenog nesavršenog ugrađivanja u kućište kamera, prilikom kalibracije, potrebno je odrediti i centar slike, jer on ne mora biti središnji element senzora i koordinate glavne točke. Centar slike ima svoje koordinate (x_o, y_o). Da bi se rezultati jednadžbi proračuna intrinzičnih parametara slagali s dobivenim rezultatima prilikom programske kalibracije unose se i korekcijski faktori koji predstavljaju odnos između dimenzije elementa slike kamere koji se kalibrira i mjerne jedinice korištene u sceni. Matrica intrinzičnih parametara **I** dana je izrazom (2-4)[3]:

$$I = \begin{bmatrix} a_x & s & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2-4}$$

gdje je:

- a_x, a_y - žarišne duljine ($a_x = \mathrm{f}k_x, a_y = \mathrm{f}k_y$), definiraju uvećanje slike
- x_o, y_o - koordinate centra slike (definiraju translaciju između glavne točke i nultog elementa na senzoru)
- *s* - faktor iskrivljenosti senzora, u današnje vrijeme senzori se skoro pa savršeno izrađuju te se može zanemariti

Ekstrinzični parametri daju međusobni položaj između koordinatnog sustava kamere, i koordinatnog sustava scene u svijetu, gdje je njihov položaj određen matricom rotacije **R** i vektorom translacije $\vec{t}$ (pomaka) u sve tri dimenzije (x, y, z) (2-5) [3]:

$$\begin{bmatrix} R & \vec{t} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{2-5}$$

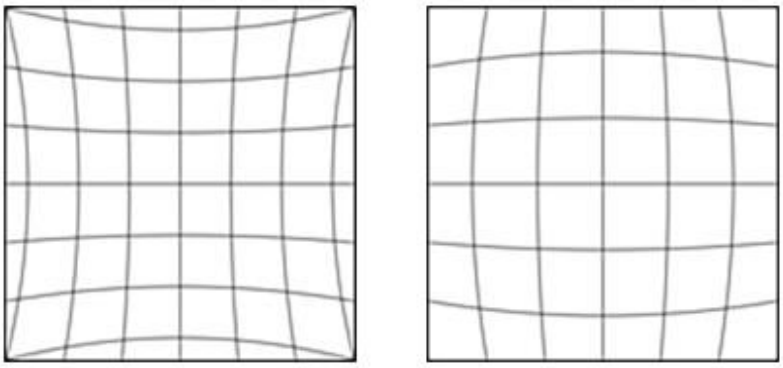
Postoje različiti načini kalibracije kamere kao što su:

- Kalibracija kalibracijskim kavezom - koristi se trodimenzionalna fizička struktura. Sastavlja se od dvije kalibracijske ravnine postavljene pod kutom od 90°.

- Kalibracija štapom - ravni štap na sebi sadrži kalibracijske točke koje su kolinearne u prostoru.
- Kalibracija kalibracijskom ravninom - ima dvije dimenzije (x, y), treća dimenzija (z) je nula. Prilikom kalibracije kalibracijskom ravninom uzima se čvrsta ravnina koja na sebi sadrži klasične geometrijske oblike kao što su: kvadrati, ucrtane kružnice. Podloga kalibracijske ravnine treba biti bijela/crna, odnosno elementi na kalibracijskoj ravnini crno/bijeli, jer računalo prilikom detekcije kalibracijskih točaka uvijek koristi neku vrstu binarne klasifikacije.

2.1.2. Distorzija slike

Distorzija u optici leća predstavlja odstupanje od idealnog modela projekcije (Sl.2.5.). Kod idealnog modela projekcije ravne linije u sceni ostaju ravne linije na projekcijskoj ravnini. Najizraženija dva tipa distorzije leća kod kamera su tangencijala i radijalna. Razlog nastanka distorzije je jeftiniji proizvodni proces sfernih leća, zbog kojih nastaje radijalna distorzija, neporavnata leća sa senzorom slike dovodi do tangencijalne distorzije. Radijalna distorzija se dijeli na bačvastu (engl. *barrel*) i jastučastu (engl. *pincushion*). Bačvasta distorzija čini sliku većom u centru i smanjuje ju prema rubovima, dok je kod jastučaste distorzije obrnut slučaj.



(a) (b)

Sl.2.5. *Dva tipa radijalne distorzije (a) jastučasta, (b) bačvasta distorzija [4]*

Radijalna distorzija je jednaka nuli u centru leće te se prema rubovima njen iznos povećava. Najbolja aproksimacija radijalne distorzije dobivena je u slučaju korištenja prva dva, odnosno tri elementa Taylorovog reda. Tri elementa se koriste ukoliko leća ima puno distorzije kao što je primjer *fish-eye* kamere koje su korištene u izradi ovog diplomskog rada. Ispravljanje distorzije određenog elementa slike odnosno njegova nova lokacija na slici može se prikazati izrazima (2-6, 2-7, 2-8)[4].

$$r^2 = x^2 + y^2 \tag{2-6}$$

$$\mathrm{x}_{ispravljeno} = x + x(\mathrm{k}_1 r^2 + \mathrm{k}_2 r^4 + \mathrm{k}_3 r^6) \tag{2-7}$$

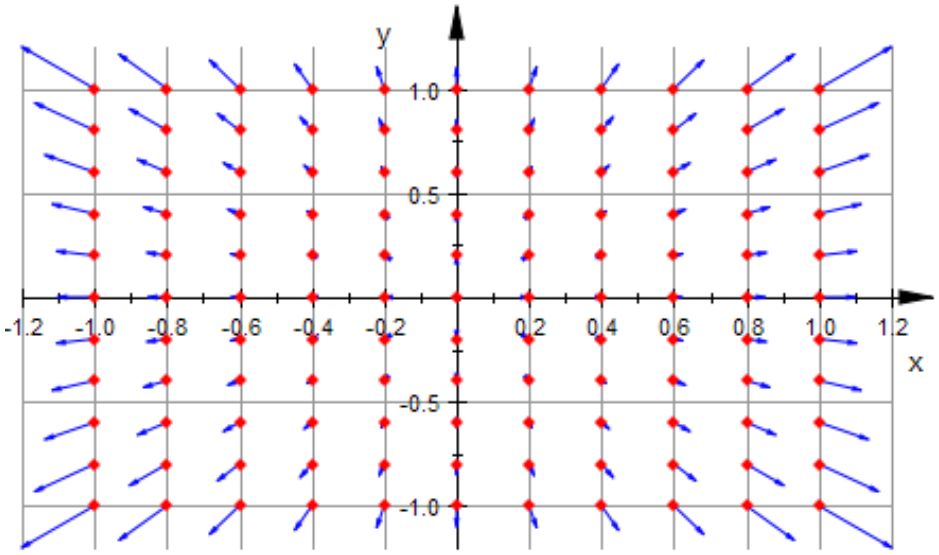
$$\mathrm{y}_{ispravljeno} = y + y(\mathrm{k}_1 r^2 + \mathrm{k}_2 r^4 + \mathrm{k}_3 r^6) \tag{2-8}$$

gdje je:

- r – udaljenost elementa slike određenih koordinata od centra slike
- x, y – koordinate elementa slike
- $x_{ispravljeno}$, $y_{ispravljeno}$ – koordinate ispravljenog elemenata slike

U izrazu (2-7 i 2-8) prikazan je izračun položaja svakog ispravljenog elementa slike pomoću udaljenosti elementa slike od centra slike i koeficijenata radijalne distorzije *k1*, *k2*, *k3* dobivenih geometrijskom kalibracijom kamere.

Na Sl.2.6. prikazana je distorzija elementa slike. Crvene točke na slici predstavljaju poželjni položaj elementa slike, dok plave strjelice definiraju njihov položaj prilikom slikanja kamerom koja nije idealna, odnosno kamerom koja stvara distorziju elementa slike. Što je dulja plava strjelica to je distorzija elementa slike veća. Vrh strjelice definira konačni položaj elementa slike na slici.

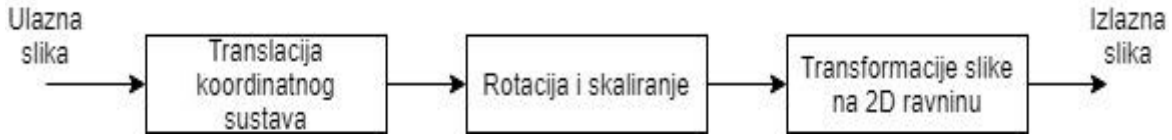


Sl.2.6. *Distorzija elementa slike u ovisnosti o udaljenosti od centra slike [5]*

Tangencijalna distorzija je pojava do koje se dolazi zbog neparalelnog smještanja leće i senzora u kućištu kamere ali ona nije toliko izražena na kvalitetnije izrađenim kamerama.

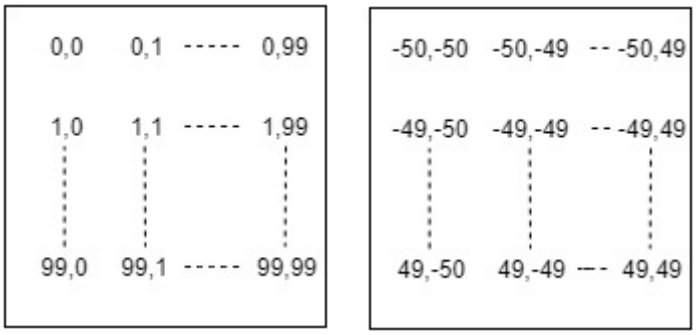
2.1.3. Prebacivanje slike u ptičju perspektivu

Transformacija slike u ptičju perspektivu je tehnika koja stvara pogled odozgo. Ovaj proces bi se mogao podijeliti u osnovna tri koraka: translacija koordinatnog sustava, rotacija slike, transformacija slike na 2D ravninu (Sl.2.7.) [6].



Sl.2.7. *Operacije nad slikom za dobivanje pogleda odozgo [6]*

Ukoliko je ulazna slika razlučivosti 100x100 elemenata slike, položaj svakog elementa slike je prikazan Sl.2.8., prije i nakon translacije koordinatnog sustava ulazne slike. Kako bi se mogla izvesti rotacija potrebno je translatirati koordinatni sustav tako da središte slike bude u koordinati (0, 0) (Sl.2.8. (b)) gdje je početak slike (-50, -50), a kraj u točci (49, 49). Ukoliko (*x, y*) predstavljaju koordinate slike onda translatirane koordinate (*X, Y*) su dane izrazima (2-9, 2-10).



(a) (b)

Sl.2.8. *Položaj elemenata slike (a) prije translacije (b) nakon translacije koordinatnog sustava ulazne slike [6]*

$$X = x - (W/2) \tag{2-9}$$

$$Y = y - (H/2) \tag{2-10}$$

gdje su:

- *X, Y* – koordinate elementa slike nakon translacije koordinatnog sustava
- *x, y* – koordinate elemenata slike prije translacije koordinatnog sustava
- *W* – širina slike u broju elemenata slike
- *H* – visina slike u broju elemenata slike

Rotacija se izvodi množenjem matrica, matrice rotacije **R** i translatiranih koordinata, čiji produkt je definiran izrazom (2-11) [6].

$$\begin{bmatrix} p \\ q \\ e \\ 1 \end{bmatrix} = R \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} \tag{2-11}$$

gdje su :

- p, q, e – koordinate elementa slike rotirane slike
- X_1, Y_1, Z_1 – koordinate ulazne slike.

Ukoliko je kamera montirana pod kutom od 45° u odnosu na površinu, rotiranjem slike za 45° dobiva se pogled odozgo. Nakon rotiranja potrebno je izmijeniti dimenzije slike, širinu i visinu odnosno izvršiti potrebno skaliranje. Kako bi se dobio pogled odozgo (2D ravnina), potrebno je projicirati sliku u 2D prostor. Jasnije objašnjenje može se naći u [6].

Ovisno iz kojeg ugla i s kolike udaljenosti je neka scena zabilježena (ovisno o perspektivi), svaki element slike je povezan s drugačijom informacijom scene. Prilikom procesiranja slike, potrebno je uzeti u obzir efekt perspektive i odrediti važnost pojedinog elementa slike, prema informacijama koje sadrži. Tom pojavom se bavi geometrijska transformacija (engl. *Inverse Perspective Mapping*) inverzna transformacija perspektive. Ova transformacija omogućava da se ukloni efekt perspektive s određene slike i čitava slika transformira u 2D sliku, gdje su sadržani samo potrebni elementi čitave slike. Prilikom izrade ovog algoritma najbitnije informacije su informacije koje se nalaze neposredno u blizini vozila. Iz tog razloga je potrebno dobiti pogled odozgo koji će sadržavati te informacije.

2.2. Postojeći algoritmi za generiranje pogleda prostora oko automobila

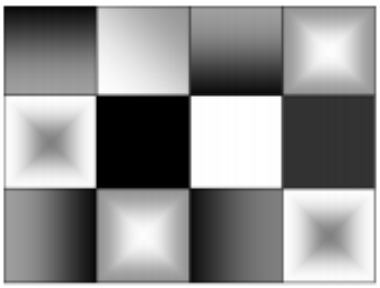
Neki od postojećih algoritama za generiranje pogleda prostora oko automobila objašnjeni su u nastavku i razlikuju se po načinu kalibracije kamere, korištenim tehnikama spajanja slike i izjednačavanja boje na mjestu spojeva. Njihovom analizom utvrđen je pristup koji će se koristiti pri izradi vlastitog algoritma za generiranje pogleda prostora oko automobila.

2.2.1. Kalibracija kamere kalibracijskim uzorkom

Prema [7] sustav za generiranje pogleda prostora oko automobila obično se sastoji od četiri do šest *fish-eye* kamera postavljenih na automobilu: jedna na prednjem braniku, druga na zadnjem braniku i po jedna na bočnim retrovizorima automobila. Najprodavaniji sustav umjetne inteligencije je sustav za pogled nad autom iz ptičje perspektive. Ovaj sustav najviše nedostataka ima na mjestu spojeva slika s različitih kamera (granicama susjednih slika koje se preklapaju) i mjestima nejednakosti boja zbog različitih ulaznih scena svake kamere te koristi algoritam za dobivanje složenog pogleda odozgo koji se sastoji od tri ključna dijela:

- geometrijsko poravnanje,
- fotometrijsko poravnanje,
- sinteza složenog pogleda.

Geometrijsko poravnanje, tzv. kalibracija, predstavlja osnovnu komponentu sustava pogled odozgo. Ovaj korak obuhvaća ispravljanje distorzije LDC (engl. *Lens Distortion Correction*) i transformaciju perspektive. Nakon LDC istovremeno se primjenjuju 4 matrice perspektive, po jedna za svaku kameru, kako bi se četiri ulazna okvira nad kojima je izvršen LDC transformirala u ptičju perspektivu. Ovaj algoritam je zasnovan na kalibracijskom dijagramu. Sadržaj kalibracijskog dijagrama je dizajniran kako bi pojednostavnio i povećao preciznost i pouzdanost algoritma (Sl.2.9.).

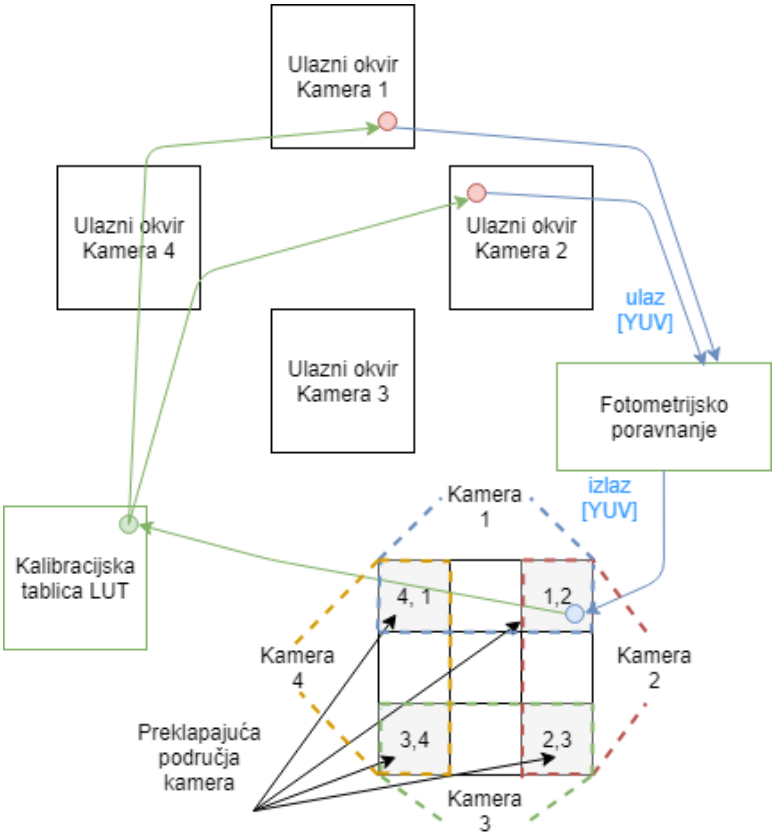


Sl.2.9. *Primjer kalibracijskog dijagrama [7]*

Prilikom kalibracije, četiri kalibracijska dijagrama su postavljena oko vozila. Svaki dijagram bi trebao biti postavljen u zajedničko vidno područje dviju odgovarajućih kamera. Nakon toga, svaka kamera bilježi okvir istovremeno. Za odgovarajuće parove kamera koristi se Harrisov detektor kuta (engl. *Harris corner detector*) nad slikom i preklapajućim područjima te se dobivaju određeni podaci, tzv. BRIEF (engl. *Binary Robust Independent Elementary Features*) opisivači. Zatim se minimizira razlika između povezanih značajki određivanjem optimalne matrice perspektive. U konačnici se kreira LUT (engl. *Look up table*) tablica, u kojoj se kodiraju informacije LDC-a i transformacije perspektive. LUT tablica se koristi u sintezi složenog pogleda, kako bi se kreirao konačni povezani pogled odozgo.

Zbog AE (engl. *Auto Exposure*), AWB (engl. *Auto White Balance*), boja i osvjetljenje istog objekta uhvaćenog različitim kamerama mogu biti jako različiti. Kao rezultat, spojena slika može imati primjetne fotometrijske razlike između dvaju odgovarajućih pogleda. Cilj fotometrijskog poravnanja je ujednačiti osvjetljenje i boje, kako bi složeni pogled izgledao kao da je dobiven jednom kamerom iz ptičje perspektive. Kako bi se to postiglo, dizajnirana je korekcijska funkcija globalne boje i osvjetljenja, stvarajući manja međusobna odstupanja između svakog pogleda te u konačnici manja odstupanja u preklopljenim područjima.

U ovom dijelu algoritma, korištenjem sekvenci s ulazne četiri *fish-eye* kamere kreira se složena slika, odnosno spojena slika. U preklapajućim područjima gdje su potrebni podaci s dviju odgovarajućih kamera, svaki izlazni element slike je povezan s lokacijama elemenata slike s dviju ulaznih slika. U preklapajućim područjima moguće je stopiti odgovarajuće podatke s dviju odgovarajućih slika, ili binarnom klasifikacijom odrediti koja slika od dvije koje se preklapaju će biti korištena za dobivanje podataka u tom području. Većina neslaganja u boji i osvjetljenju riješeni su u prethodnom koraku (fotometrijsko poravnanje), dodatno uklanjanje neslaganja u ovom dijelu algoritma izvršava se korištenjem AB (engl. *Alpha Blending*). AB uklanja preostale vidljive spojeve slika. AB veličine su također pohranjene u odgovarajućim LUT tablicama. Izlazni elementi slike su generirani prema linearnoj kombinaciji odgovarajućih veličina ulaznih elementa slike i AB veličina. Na Sl.2.10. prikazano je način dobivanja izlaznog elementa slike, ili od dva ulazna elementa slike, ukoliko je izlazni element slike u preklapajućem području, ili od jednog ukoliko nije, što se raspoznaje korištenjem LUT tablice, dobivene geometrijskom kalibracijom. Ovaj algoritam koristi YUV sustav boja koji omogućuje određenu kompresiju slike (poduzorkovanje) jer je ovaj sustav boja pogodniji za digitalnu obradu i uzima u obzir veću osjetljivost ljudskog oka na luminaciju, gdje slika ima jednu luminantnu (Y) i dvije krominantne (U, V) komponente.



Sl.2.10. *Sinteza složenog pogleda[7]*

2.2.2. Ispravljanje distorzije slike bez korištenja kalibracijskog uzorka

U literaturi [8] opisane su tehnike: ispravljanje distorzije, te geometrijsko poravnanje zasnovano na homografiji (engl. *homography*), zajedno s pronalaženjem zajedničkih značajki preklapajućih dviju slika, kako bi se generirao precizan pogled odozgo slika s više kamera.

Širokokutne leće su općenito sferične, često područja vidljivosti blizu $180°$. Kako bi se obuhvatilo veće područje, rubni elementi slike moraju biti izobličeni. Nastala distorzija slike se ispravlja prije geometrijskog poravnanja slika generiranih s više kamera. Većina tehnika za ispravljanje distorzije zasnovane su na parametrima kamere, koji se procjenjuju na osnovu nekoliko slika određenog kalibracijskog uzorka. Postojeće rješenje ispravljanja distorzije predstavljeno ovdje ne koristi kalibracijski uzorak i kreira gotovo idealan model leće, optimizirajući dimenzije slike te pruža kontrolu jačine korekcije distorzije i razinu povećanja slike (engl. *zoom*). Temeljeno na širokokutnoj projekciji, korekcijski radijus R_c se može prikazati funkcijom visine (H) i širine (W) ispravljene slike (ispravljena distorzija) i korekcijskog iznosa (s) prema izrazu (2-12) [8].

$$R_c = f(H^2, W^2, s^{-1}) \tag{2-12}$$

Nakon centriranja ($X = \left(x - \frac{W}{2}\right), Y = (y - W/2)$) izvorišta slike (engl. *image origin*), udaljenost (D) elementa slike od izvorišta slike je $D = f(X^2, Y^2)$. Omjer r udaljenosti od izvorišta slike i korekcijskog radijusa slike računa se prema izrazu (2-13) [8].

$$r = D/R_c \tag{2-13}$$

Ukoliko se koordinate elementa izobličene i ispravljene slike prikažu kao (x', y') i (x, y), moguće je napraviti i reverzno projiciranje prema izrazu (2-14) [8].

$$x' = \frac{W}{2} + (X \times \theta \times z); y' = \frac{H}{2} + (Y \times \theta \times z) \tag{2-14}$$

Gdje z u izrazu predstavlja iznos povećanja (engl. *amount of zoom*) u odredišnoj ispravljenoj slici, a sferični kutni parametar θ se računa prema izrazu (2-15) [8].

$$\theta = \begin{cases} 1, & r = 0 \\ \dfrac{tan^{-1}(r)}{r}, & r \neq 0 \end{cases} \tag{2-15}$$

Ukoliko je vrijednost (x', y') unutar granica izobličene slike, samo tada se vrijednost elementa slike projicira u (x, y) ispravljene slike.

Ispravljena slika se zatim prebacuje u ptičju perspektivu korištenjem inverzne transformacije perspektive. Transformacijski model je fleksibilan što se tiče rotacije, translacije, parametara uvećanja slike, čime se dobiva dobra kontrola projekcije slike. Bikubična interpolacija je korištena kako bi se uklonilo zamućenje dobiveno primjenom inverzne transformacije perspektive nad slikom.

Četiri slike su registrirane i poravnate kako bi se kreirao složeni pogled odozgo. Za registraciju slike koristi se 8 setova koordinata dviju slika (stražnja i prednja kamera). Te koordinate pripadaju preklapajućim točkama između prednje i desne kamere, prednje i lijeve kamere, stražnje i desne kamere, stražnje i lijeve kamere. Ove točke je jako teško ručno odrediti, iz tog razloga se koristi algoritam koji koristi detekciju kuta poznatog uzorka. Sljedeći koraci su potrebni za izdvajanje kutova interesa prednje i stražnje kamere [8]:

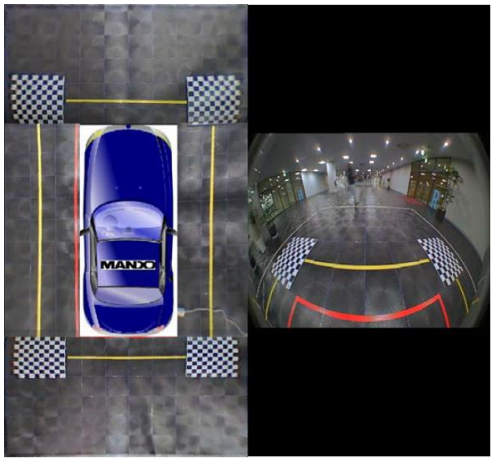
- Korak 1: Odrediti sve moguće kutove (x_i, y_i) unutar područja interesa (engl. *Region Of Interest*) koristeći Harrisov detektor kuta.
- Korak 2: Ručno označiti klikom mišem najbliže točke (x, y) kutovima(x_i, y_i) unutar područja interesa .
- Korak 3: Izračunati udaljenost (δ) između označenih točaka i svih kutova unutar područja interesa prema $\theta = \sqrt{(x - x_i)^2 + (y - y_i)^2}$
- Korak 4: Izdvajaju se četiri seta koordinata s prednje odnosno zadnje kamere, kasnije definiranih kao izvorišne točke, s najmanjim udaljenostima (δ).

Nakon što su slike svake kamere prošle kroz prethodne geometrijske operacije, potrebno je te slike spojiti u pogled odozgo. Kako bi se generirao pogled odozgo, prvo se kreira prazna slika na koju se nakon toga postavljaju dijelovi ostalih slika, odnosno slika pojedine kamere. Sve slike imaju iste dimenzije, tako da je njihov odnos na čitavoj spojenoj slici pogleda odozgo jednolik. Dvije slike istih planarnih površina su povezane prema homografiji. Tip homografije korišten u ovom rješenju se opisuje matricom veličine 3×3, što znači da ta matrica ima 9 elemenata koji utječu na projekciju izvorišnih točaka na odredišne.

Za $N \geq 4$, gdje je N broj veza između izvorišnih točaka $\{\{x_i, y_i\}\}_{i=0}^{N-1}$ i odgovarajućih odredišnih točaka $\{\{x'_i, y'_i\}\}_{i=0}^{N-1}$ se opisuje izrazom (2-16):

$$x'_i = \frac{h_{00} \cdot x_i + h_{01} \cdot y_i + h_{02}}{h_{20} \cdot x_i + h_{21} \cdot y_i + 1},\ y'_i = \frac{h_{10} \cdot x_i + h_{11} \cdot y_i + h_{12}}{h_{20} \cdot x_i + h_{21} \cdot y_i + 1} \qquad (2\text{-}16)$$

Ako matricu homografije definiramo kao red-stupac matricu $\boldsymbol{H} = [h_{00} \dots h_{21}]^T$, moguće je korištenjem izraza (2-16), sve izvorišne točke projicirati u njihove odredišne točke. Kamere korištene u ovom algoritmu izložene su različitim osvijetljenima, tako da je potrebno uraditi korekciju boja, koja se postiže primjenom tehnike temperaturnog balansiranja boja (engl. *colour temperature balancing techique*) nad četiri geometrijski poravnate slike. Nakon toga primjenjuje se poravnavanje spojeva slika (engl. *image blending*). Konačni pogled prostora oko automobila opisanog algoritma prikazan je na Sl.2.11(a), dok je na Sl.2.11(b) prikazan pogled s desne *fish-eye* kamere.



(a) (b)

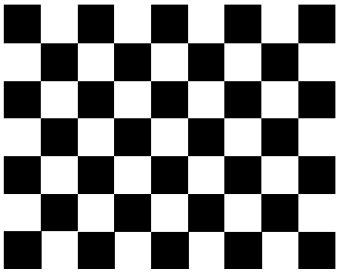
Sl.2.11. *Postojeći algoritam za generiranje prostora oko automobila (a) prikaz prostora oko automobila postojećeg algoritma, (b) prikaz slike s desne kamere korištene u ovom algoritmu [8]*

2.2.3. Bouget kalibracijski alat

Prema [9] algoritam za parkiranje automobila uključuje tri modula: kalibracija *fish-eye* kamere, transformacija perspektive, poravnanje i povezivanje slika (engl. *stitching*).

Kako bi se postigla veća preglednost prilikom parkiranja vozila, na njega se postavljaju četiri *fish-eye* kamere gdje se između dviju odgovarajućih kamera formira određeno područje, jer dolazi do preklapanja područja koja spadaju u područje pokrivanja obiju kamera. Slike snimljene *fish-eye* kamerom sadrže veliku distorziju elemenata slike. Kako bi se koristile projekcijske informacije takvih slika, potrebno ih je ispraviti. Ljudskom oku udaljeniji objekti izgledaju manji od bližih objekata, što nazivamo perspektivom. Kalibracijski parametri kamere su potrebni kako bi se dobile ispravljene slike. Postupak kalibracije izravno utječe na rezultate povezivanja slika. U literaturi [9] korišten je Bouget *fish-eye* kalibracijski alat, jer daje ponajbolje rezultate preciznosti

i robusnosti u smislu DAS (engl. *Driver Assistance Systems*). Nakon što su zabilježene kalibracijske slike (njih 50) s kalibracijskim uzorkom (Sl.2.12.), moguće je ispravljati izobličene slike pomoću Bouget kalibracijskog alata. Detaljniji opis korištenja spomenutog alata te njegovih funkcija može se vidjeti u literaturi [9].

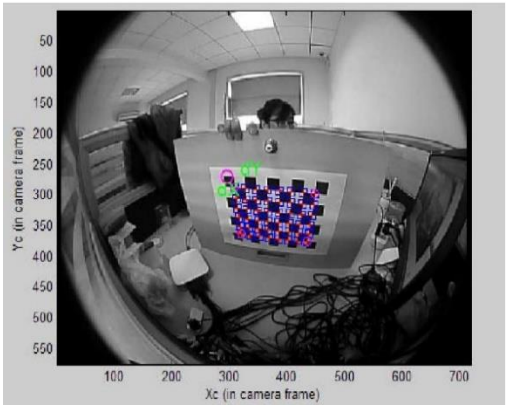


Sl.2.12. *Kalibracijski uzorak korišten za kalibraciju slika Bouget kalibracijskim alatom [9]*

Kalibracijske slike trebaju biti zabilježene s različitim položajima i usmjerenja kalibracijskog uzorka te svaki dio kalibracijskog uzorka treba biti unutar slike i što je moguće veći na slici. Kalibracijski proces kamere sastoji se od sljedećih koraka:

1. Učitavanje kalibracijskih slika
2. Izdvajanje kutova na kalibracijskim slikama (Sl.2.13.)
3. Računanje kalibracijskih parametara

   Intrinzični parametri su inicijalizirani korištenjem spojnih točaka kvadratića na kalibracijskom uzorku.



Sl.2.13. *Izdvojeni kutovi (spojne točke) na kalibracijskoj slici korištenjem Bouget kalibracijskog alata [9]*

Kalibracijski parametri u MATLab-u nakon inicijalizacije su kako slijedi[9]:

- Žarišna duljina: *fc*=[128,01 128,01457]
- Glavna točka: *cc* = [359,5000 287,5000]
- Iskrivljenost (engl. *Skew*): *alpha_c* = [0,0000], kut elementa slike = 90°
- Distorzija: *kc* = [0,000 0,000 0,000 0,000 0,000]

Kalibracijski parametri nakon optimizacije u MATLab-u, korištenjem kalibracijskih slika su kako slijedi[9]:

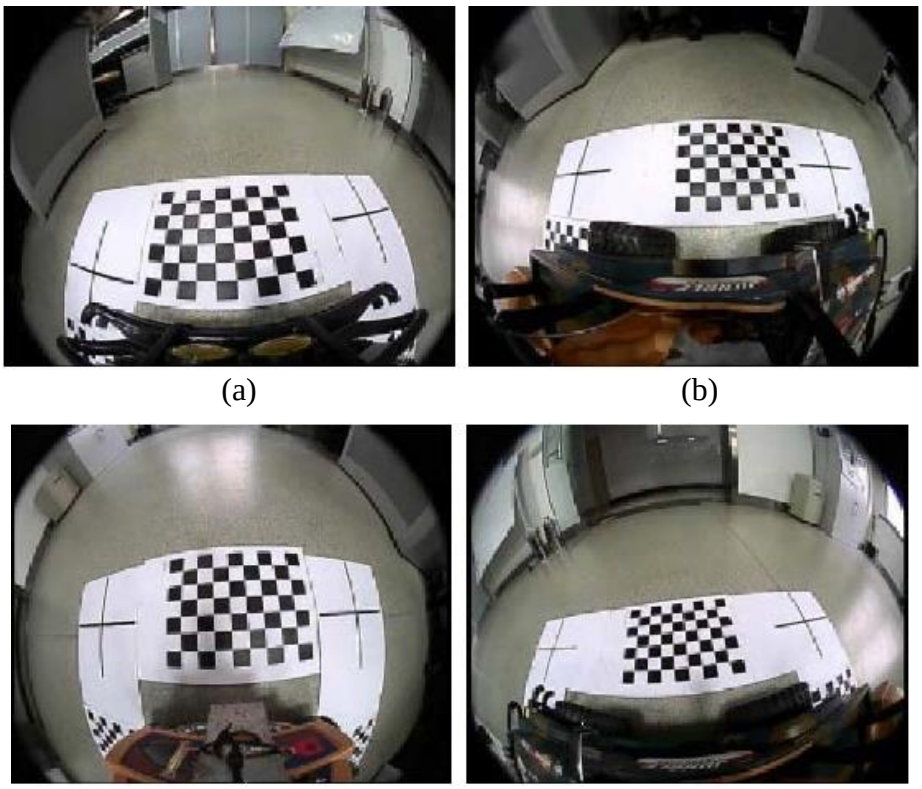
- Žarišna duljina: *fc*=[250,84720 258,61783] +/- [5,94402 6,18146]
- Glavna točka: *cc* = [362,74657 297,43090] +/- [2,15964 2,41706]
- Iskrivljenost (engl. *Skew*): *alpha_c* = [0,00000] +/- [0,00000]=> , kut elementa slike = 90°
- Distorzija: *kc* = [-0,38452 0,29703 -0,00361 0,00143 0,00000] +/- [0,02789 0,11145 0,00238 0,00208 0,00000]
- Greška elementa slike: *err* = [0,078704 0,11970]

4. Ispravljanje distorzije slike

   Pomoću *OpenCV* (engl. *Open Source Computer Vision Library*) moguće je ispraviti distorziju pojedine slike.

5. Popunjavanje tablice veze između slika s distorzijom i ispravljenih slika

   Kalibracijske slike oko automobila prikazane su Sl.2.14., na kojoj vidljivo platno s kalibracijskom pločom.

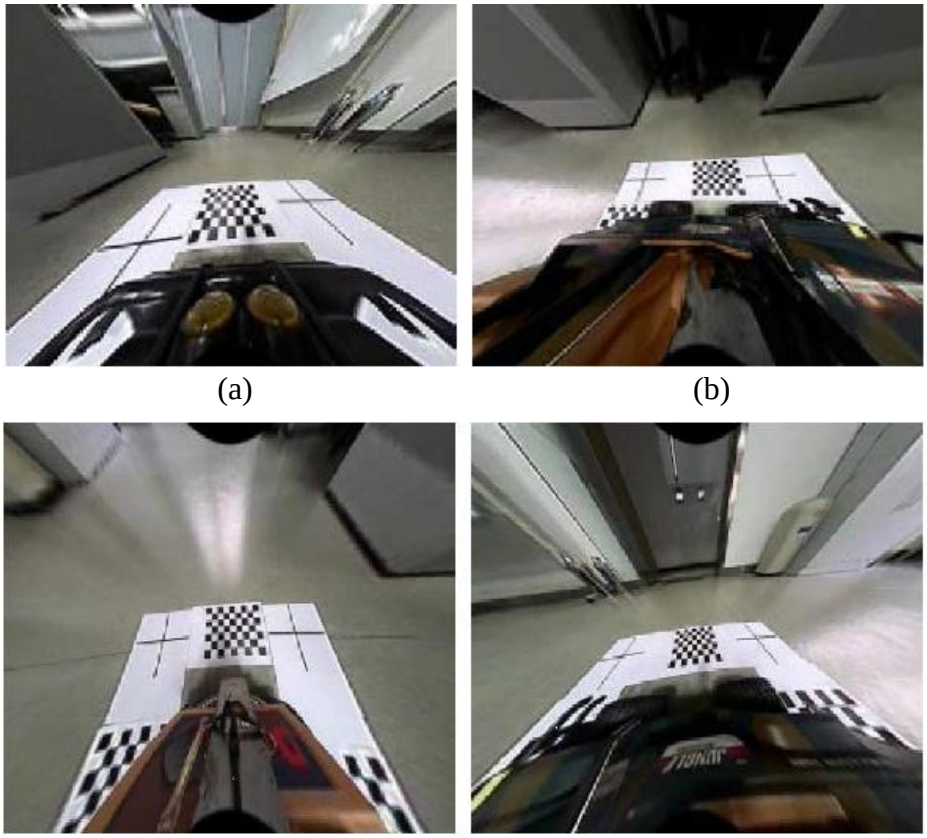


(a) (b)

(c) (d)

Sl.2.14. *Kalibracijske slike oko automobila sa kamera (a) prednja, (b) lijeva, (c) stražnja, (d) desna [9]*

Korištenjem parametara kamere i spomenutog kalibracijskog alata, veza između slika s distorzijom i ispravljenih slika se računa i sprema u tablicu. Ispravljene slike prikazane su na Sl.2.15.



(a) (b)

(c) (d)

Sl.2.15. *Ispravljena distorzija sa slika predstavljenih na Sl.2.14. (a) prednja, (b) lijeva, (c) stražnja, (d) desna [9]*

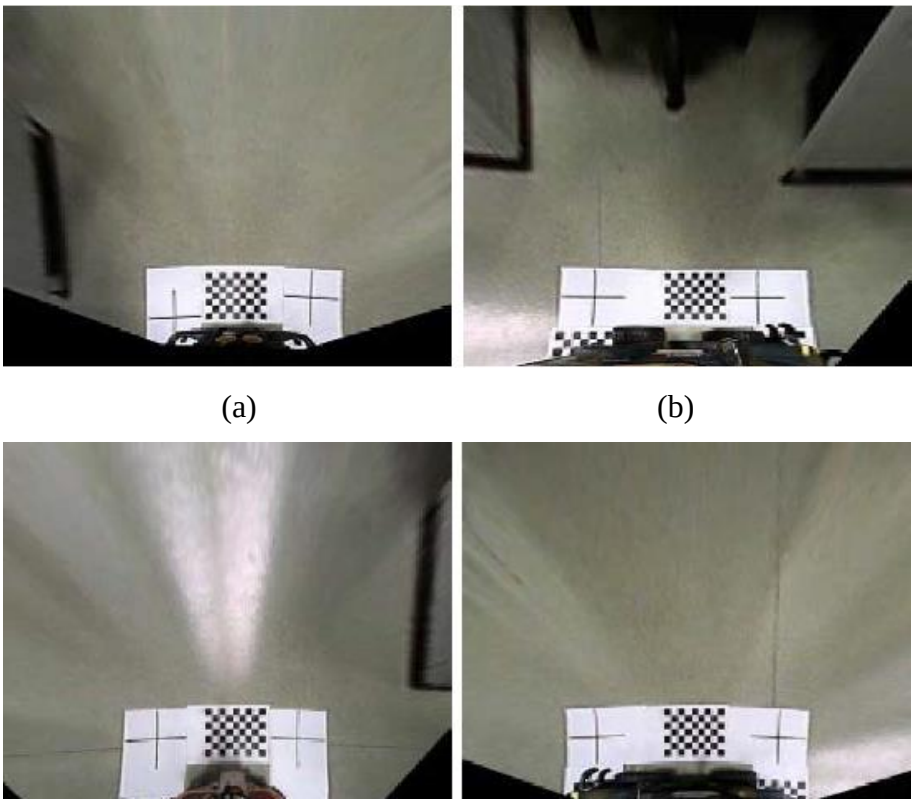
Kako bi se dobile slike pogleda odozgo potrebno je napraviti transformaciju perspektive ispravljenih slika. Transformacija perspektive, predstavlja matematički model preslikavanja slike na novu ravninu, odnosno rotiranje ravnine oko osi pod određenim kutom, čime održava geometriju nepromijenjenu u novoj ravnini. Geometrijska transformacija slike ne mijenja sadržaj slike ali deformira elemente slike prilikom prebacivanja u odredišnu sliku. Kako bi se izbjegla distorzije, projiciranje se radi obrnutim redoslijedom, od odredišta (*dst*) prema izvorištu (*src*), odnosno za svaki element odredišne slike $dst(x, y)$, funkcija računa odgovarajuće koordinate izvorišne slike, i kopira vrijednost elementa slike prema izrazu (2-17).

$$dst(x, y) = src(fx(x, y), fy(x, y)) \tag{2-17}$$

Proces transformacije perspektive baziran na *OpenCV* može se opisati funkcijama:

- *cvFindChessboardCorners* – koristi se za određivanje kutova kalibracijske ploče na ispravljenim slikama.

- *cvGetPerspectiveTransform* – koristi se za računanje matrice transformacije perspektive iz četiri para odgovarajućih točaka kuta.
- *cvWarpPerspective* – koristi se za primjenu transformacije perspektive nad slikom. Rezultati transformacije perspektive nad ispravljenim slikama vidljivi su na Sl.2.16.



(a) (b)

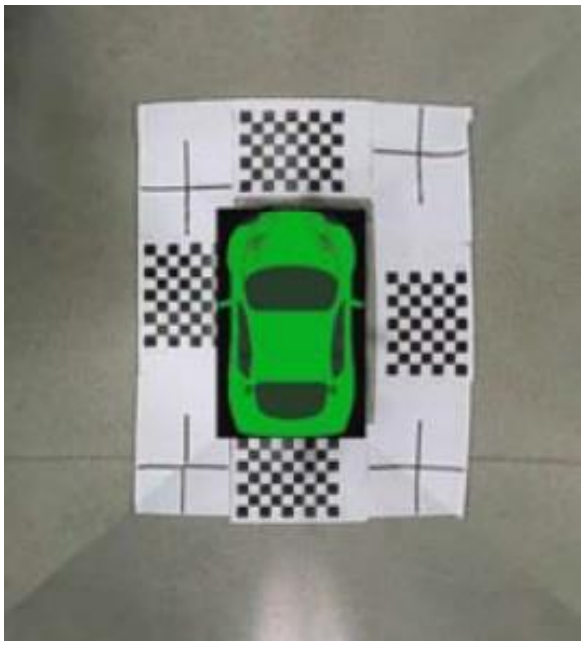
(c) (d)

Sl.2.16. *Rezultati transformacije perspektive nad slikama prikazanih na Sl.2.15. (a) prednja, (b) lijeva, (c) stražnja, (d) desna [9]*

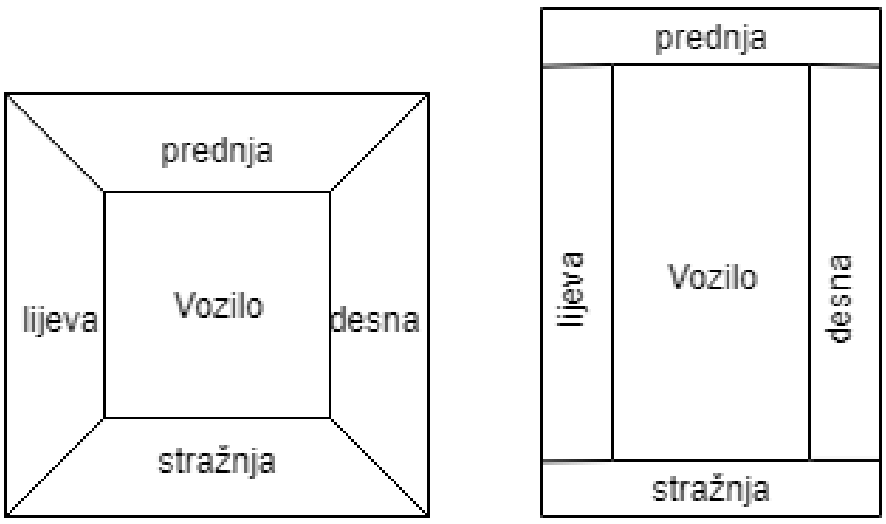
Načini poravnanja i povezivanja slika dijele se prema području, značajkama te prema četiri osnovna koraka registracije slike: detekcija značajki, povezivanje značajki, dizajn projekcijske funkcije i transformacija slike. Prema [9] kako bi se pronašao najbolji način povezivanja slika između dviju odgovarajućih kamera, postavljeni su „križevi“ između dvije preklapajuće kamere. Moguće je i ručno složiti četiri slike pogleda odozgo kako je napravljeno u ovom algoritmu. Rezultati dobiveni ovim algoritmom prikazani su na Sl.2.17.

Prema postojećim algoritmima za generiranje prostora oko automobila za razvoj vlastitih algoritama definirani su pristupi za: kalibraciju kamere kalibracijskim uzorkom, prebacivanje slike u ptičju perspektivu inverznom transformacijom perspektive. Pri međusobnom spajanju slika za dobivanje konačnog pogleda odozgo pristupa se na dva načina: prema zajedničkim značajkama dviju slika i ručno. Ovisno o oblicima slika definiranih pri određivanju njihove pozicije na

konačnoj slici i prevladavajućeg elementa slike na konačnoj slici u preklapajućem području dviju kamera definirana su dva obliak: trapezni oblik slike i pravokutni oblik slike prikazani na Sl.2.18. Standardna vozila se najviše razlikuju u dimenziji duljine zbog čega je više podataka potrebno s bočnih kamera nego s prednje i stražnje kamere, zbog toga što bočne kamere daju podatke o okolini dulje dimenzije automobila te se pristup na Sl.2.18(b) smatra boljim i takav je korišten u izradi algoritma.



Sl.2.17. *Pogled odozgo generiran korištenjem tehnika iz literature [9] te u konačnici ručno složenih pogleda sa Sl.2.16.*



(a) (b)

Sl.2.18. *Pogled oko automobila: (a) trapezni oblik, (b) pravokutni oblik slika kamera*

3. RAZVOJ VLASTITOG PROGRMASKOG RJEŠENJA ZA GENERIRANJE POGLEDA PROSTORA OKO AUTOMOBILA

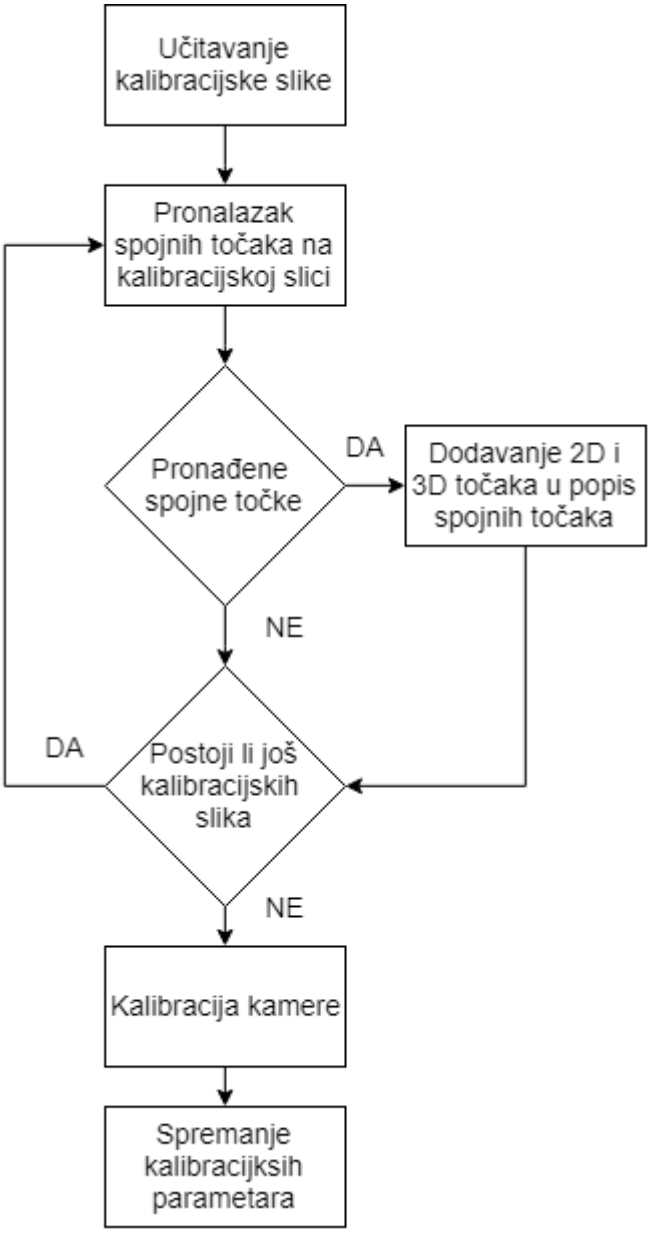
U sklopu zadatka diplomskog rada bilo je predviđeno razviti vlastito rješenje za generiranje pogleda prostora oko automobila i implementirati ga u nekom od viših programskih jezika, zatim ga spustiti u C programski jezik. Kako se prostor oko automobila dobiva snimanjem slika *fish*-eye kamerama potrebno je ispraviti distorziju dobivenih slika. Na samom početku rješenje je se pokušalo implementirati u MATLAB-u. Rezultati ispravljanja distorzije slika fotografiranih pomoću mobitela (Huawei P9 Lite 2016.) i android aplikacije „Fisheye Lens Pro“ (koju je moguće preuzeti besplatno s Google Store-a) koja programski stvara distorziju na slici bili su prilično dobri, dok rezultati ispravljanja distorzije *fish-eye* slika snimljenih pravom *fish-eye* kamerom nisu bili zadovoljavajući zbog dostupne verzije MATLAB-a. Bolji rezultati kalibracije kamere postignuti su u Python programskom jeziku. Kako ispravljanje distorzije slike izravno utječe na sve ostale geometrijske operacije nad slikom, za nastavak razvijanja diplomskog rada odabran je Python programski jezik. Programsko rješenje za generiranje pogleda prostora oko automobila sastoji se od dva algoritma: algoritma za kalibraciju kamere, algoritma za generiranje pogleda prostora oko automobila. Na Sl.3.1. prikazan je dijagram toka vlastitog algoritma za kalibraciju kamere. Prije pokretanja algoritma kalibracije kamere potrebno je prikupiti kalibracijske slike. Cilj ovog algoritma je dobiti kalibracijske matrice kamere koje služe za ispravljane distorzije slike.

Na Sl.3.2. prikazan je dijagram toka vlastitog algoritma za generiranje pogleda prostora oko automobila. Prije samog pokretanja algoritma potrebno je prikupiti kalibracijske parametre koji služe za ispravljanje distorzije slike i odrediti ostale parametre objašnjene u sljedećim potpoglavljima.

3.1. Rješenje za generiranje pogleda prostora oko automobila u Python programskom jeziku

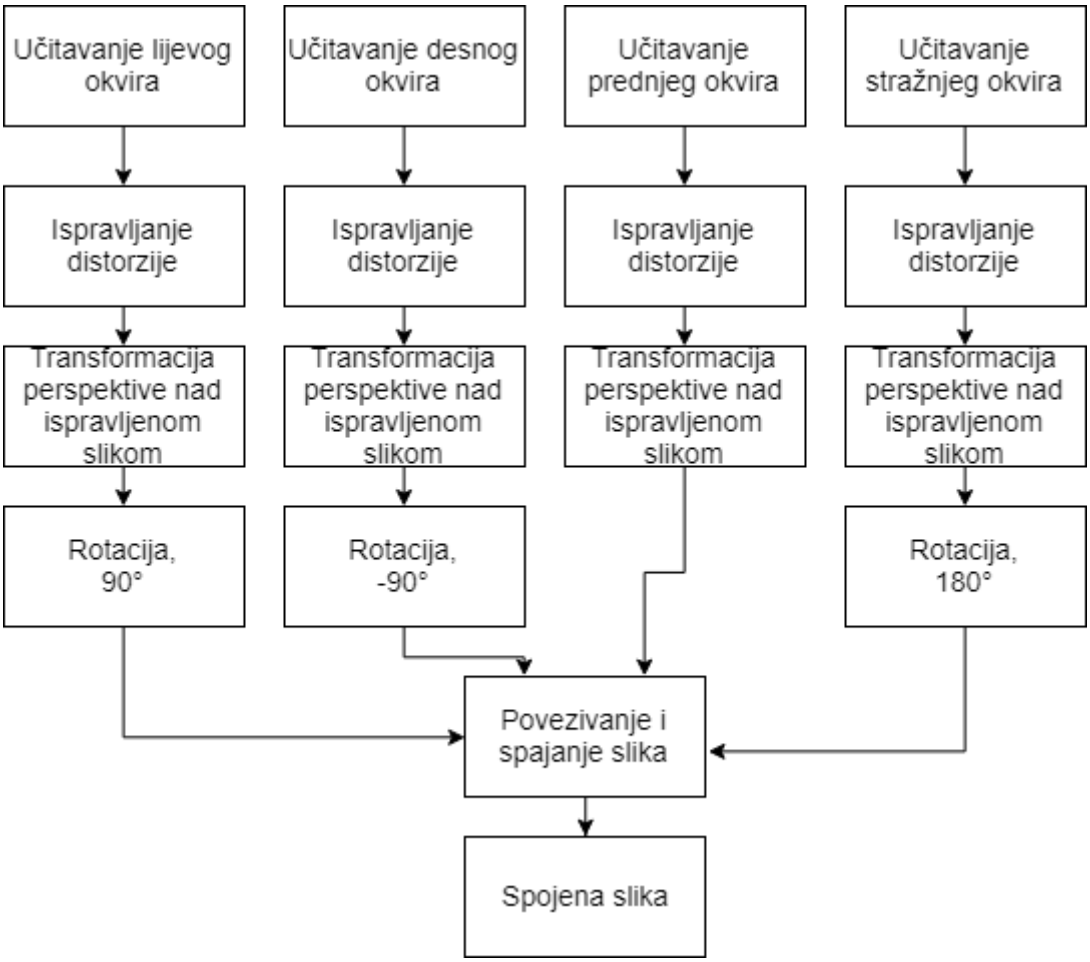
Python je programski jezik opće namjene, što znači da se može koristiti za razvoj bilo kakvih programskih rješenja korištenjem odgovarajućih alata i biblioteka. Općenito, Python je odličan za *backend* web razvoj, analizu podataka, umjetnu inteligenciju, bilo kakve znanstvene proračune te postoje različiti izvori koji pomažu u razvoju i razumijevanju alata. Postoji nekoliko verzija Python programskog jezika. Sintaksa Python programskog jezika je općenito lako čitljiva, ne koristi zagrade kao ostali programski jezici, nego koristi uvlačenje kao metodu razlikovanja

programskih blokova. Ključne riječi i kontrola toka u Pythonu definirane su riječima iz engleskog jezika kao što su*: if, for, while, try, class, def, assert, import*.



Sl.3.1. *Dijagram toka vlastitog algoritma za kalibraciju kamere*

Za razvoj programskog rješenja ovog diplomskog rada unutar Python programskog jezika korištena je *OpenCV* (engl. *Open Source Computer Vision Libary*) biblioteka. Ova biblioteka je slobodna za akademsku, komercijalnu upotrebu, sadrži više od 2500 optimiziranih algoritama, sučelja za C++, Python i Java te podržava operacijske sustave : Windos, Linux, Mac OS, iOS i Android. Razvijena je s posebnim naglaskom na stvarno-vremenske aplikacije. Biblioteka je razvijana u optimiziranom C te C++ i može se izvoditi na više jezgri. Uz *OpenCL*, može iskoristiti hardversko ubrzanje.



Sl.3.2. *Dijagram toka vlastitog algoritma za generiranje pogleda prostora oko automobila*

Generirani pogled prostora oko automobila je realiziran pomoću dvije Python skripte koje predstavljaju zasebne algoritme, gdje jedna skripta služi za kalibraciju kamere, dok druga skripta služi za generiranje pogleda prostora oko automobila.

3.1.1. Funkcije i biblioteke korištene za realizaciju algoritama u Python-u

Prva skripta *undistortKoef.py* [10] koja se nalazi u P.3.1. služi za dobivanje parametara kamere, odnosno njenu kalibraciju. Navedenu skriptu je dovoljno pokrenuti jednom za određeni tip kamere. Nakon što su prikupljeni parametri određene kamere, ovu skriptu više nije potrebno pokretati, odnosno raditi kalibraciju iste kamere. Ta skripta koristi biblioteke:

- *OpenCV* – olakšava rad sa slikama i sadrži velik broj integriranih algoritama.
- *NumPy* – Python biblioteka za znanstveno računanje, N dimenzijska polja i višedimenzijski kontejner podataka.
- *Glob* – Python modul za pronalazak svih naziva putanja koji odgovaraju odgovarajućem uzorku.

Navedena skripta koristi sljedeće funkcije [11]:

- Iz biblioteke *NumPy*, koristi funkciju *zeros(shape, dtype=float, order='C')* koja stvara polje (matricu) određenog tipa (*dtype*) i veličine (*shape*), popunjava nulama po redovima odnosno stupcima ovisno o vrsti slaganja (*order*), koja su definirana s *C* po redovima, *F* po stupcima.
- Iz biblioteke *NumPy* koristi funkciju *mgrid* koja nad određenim elementima matrice postavlja određene vrijednosti, *T* nad tom funkcijom stvara transponiranu matricu.
- Iz biblioteke *Glob* koristi *Glob* funkciju, koja stvara skup putanja određenog uzorka.
- Iz biblioteke *OpenCV* koristi funkciju *imread* koja prima putanju kao parametar i služi za učitavanje slike s te putanje.
- Iz biblioteke *OpenCV* koristi funkciju *cvtColor* koja prima dva parametra, prvi parametar je slika, drugi parametar u koji format treba pretvoriti primljenu sliku.
- Iz biblioteke *OpenCV* koristi funkciju *findChessboardCorners (image, patternSize, flags=CV_CALIB_CB_ADAPTIVE_THRESH)*. Parametar *image* je 8-bitna slika, *patternSize* broj redaka i stupaca kalibracijske ploče, *flags* zastavice koje mogu biti postavljena na nulu ili kombinacija nekih od konstanti iz *OpenCV*. Funkcija vraća polje detektiranih kutova na ploči.
- Iz biblioteke *OpenCV* koristi funkciju *drawChessboardCorners (image, patternSize, corners, patternWasFound)* koja služi za crtanje zakrivljenosti ploče nad kalibracijskom slikom. Parametar *image* je odredišna slika mora biti 8-bitna slika, *patternSize* broj redaka i stupaca kalibracijske ploče, *patternWasFound* inicira je li kalibracijska ploča nađena ili ne.
- Iz biblioteke *OpenCV* koristi funkciju *fisheye.calibrate(objectPoints, imagePoints, image_size, K, D[, rvecs[, tvecs[, flags[, criteria]]]])* koja provodi kalibraciju kamere. Parametri: *objectPoints* vektor vektora točaka kalibracijskog uzorka u koordinatnom sustavu, *imagePoints* vektor vektora projekcija točaka kalibracijskog uzorka, *image_size* – veličina slike koja je korištena za inicijalizaciju intrinzične matrice kamere, ***K*** izlazna matrica kamere, *D* izlazni vektor koeficijenata distorzije, *rvecs* rotacijski vektori, *tvecs* translacijski vektori, *flags* različite zastavice koje mogu biti postavljena na nulu ili kombinacija nekih od konstanti iz *OpenCV*, *criteria* uvjet za prekidanje.

Druga Python skripta *stitchFourImages.py* koja se nalazi u P.3.2. služi za uklanjanje distorzije sa slike, njeno prebacivanje u ptičju perspektivu te slaganje pogleda odozgo. Ova skripta

se vrti dok god ima ulaznih okvira i obrađuje, za što koristi prethodno objašnjenje biblioteke te dvije navedene u nastavku:

- *Imutilis* – Python biblioteka za jednostavne operacije nad slikama kao što su: translacija, rotiranje te promjena veličine slike.
- *MatplotLib* – Python biblioteka koja služi za prikazivanje slika. U ovom diplomskom radu je korištena zato što iscrtava sliku kao dijagram te pomoću *x* i *y* osi lakše se orijentirati koji element slike je na kojoj poziciji.

Navedena skripta koristi sljedeće funkcije [11]:

- Iz biblioteke *NumPy* koristi funkciju *float32(points)* koja služi za kreiranje početnih i krajnjih točaka paralelnih pravaca, koji se koriste prilikom prebacivanja slike u ptičju perspektivu. Parametar *points* u ovom slučaju predstavlja točke koje imaju *x* i *y* koordinate te funkciju *array(points, dtype)*, služi za upisivanje točaka kvadrata nad kojim se izvršava druga transformacija perspektive. Parametar *points* predstavlja skup koordinata točaka, dok *dtype* predstavlja tip točaka. Iz ove biblioteke koristi funkciju i *diff(a, n=1, axis=-1)* koja služi za izračun diskretne razlike po određenoj osi. Parametri: *a* ulazno polje elemenata, *n* broj koliko puta se računa diferencija nad brojevima, *axis* os po kojoj se uzima diferencija.
- Iz biblioteke *OpenCV* koristi funkciju *getPerspectiveTransform(src, dst)* koja računa perspektivnu matricu [12]. Parametar *src* su koordinate kvadrata na ulaznoj slici, dok su *dst* odgovarajuće koordinate u odredišnoj slici, prema odgovarajućim kombinacijama točaka. Izlazni parametar ove funkcije je matrica perspektive. Iz ove biblioteke koristi funkciju *warpPerspective(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])* koja primjenjuje transformaciju perspektive nad slikom. Parametri su: *src* ulazna slika, *dst* izlazna slika (mora biti istog tipa kao ulazna slika *src*), ***M*** matrica transformacije, *dsize* veličina izlazne slike, *flags* zastavice koje iniciraju koja će interpolacijska metoda biti primijenjena prilikom transformacije perspektive slike, *borderMode* metoda ekstrapolacije, *borderValue* vrijednost korištena u slučaju *borderMode(BORDER_CONSTANT)*.
- Iz biblioteke *OpenCV* koristi funkciju *fisheye.initUndistortRectifyMap(K, D, R, P, size, m1type[, map1[, map2]])* koja koristi za izračun karti distorzije i korekcije slike. Ulazni parametri su: ***K*** matrica kamere, *D* vektor distorzije, *R* korekcijska transformacija, ***P*** nova matrica kamere/matrica projekcije, *size* veličina ispravljene slike, *m1type* tip prve

izlazne karte. Izlazni parametar ove funkcije su dvije karte distorzije i korekcije (*map1, map2*). Iz ove biblioteke koristi funkciju „*remap(src, map1, map2, interpolation[, dst[, borderMode[, borderValue]]])*“ koja primjenjuje geometrijsku operaciju ispravljanja distorzije slike. Parametri su: *src* ulazna slika, *dst* izlazna slika (iste veličine kao i ulazna slika), *map1* prva karta koja sadrži samo *x* koordinate ili *x*,*y* parove ovisno o tipu, *map2* druga karta ili samo *y* koordinate ukoliko *map1* sadrži i *x*, *y* koordinate, *interpolation* metoda interpolacije, *borderMode* metoda ekstrapolacije, *borderValue* vrijednost korištena u slučaju *borderMode(BORDER_CONSTANT).*

- Iz biblioteke *Imutilis* koristi funkciju *rotate(image, angle)* koja služi za rotiranje slike. Parametri su: *image* ulazna slika, koju se rotira, *angle* kut rotacije slike.
- Iz biblioteke *OpenCV* koristi funkciju *treshold(src, tresh, maxval, type[, dst])* koja se primjenjuje nad slikom, najčešće kako bi se izdvojio određeni kanal boje. Parametri su: *src* ulazna višekanalna slika (polje) koja mora biti 8/32-bitna, *dst* izlazno polje istog tipa kao i ulazna slika te isti broj kanala, *tresh* granična vrijednost, *maxval* maksimalna vrijednost koja se koristi s različitim tipovima graničnih vrijednosti, *type* tip. Još jedna funkcija iz ove biblioteke koja se primjenjuje nad slikom, je funkcija *split (image*) koja služi za izdvajanje kanalnih boja (RGB), odnosno komponenti slike u slučaju ukoliko je potrebno raditi zasebno s komponentama. Ulazni parametar funkcije je ulazna slika, povratne vrijednosti su komponente boje. Funkcija koja se koristi za spajanje zasebnih komponenata kako bi se dobila slika sa svim kanalnim komponentama boje naziva se „*merge(params)*“, gdje su ulazni parametri zasebne komponente boje.

3.1.2. Implementacija algoritama u Python programskom jeziku

U ovom dijelu je objašnjena upotreba prethodno objašnjenih Python skripti, odnosno potrebne promjene u kodu pri pokretanju zato što se kalibracijske slike ne nalaze u istom direktoriju, odnosno dimenzije kalibracijskog uzorka ne moraju biti iste kao u ovom primjeru.

Skripta *undistortKoef.py*:

U kodu ispod opisan je dio uključivanja potrebnih biblioteka za kalibraciju kamere, te provjera instalirane verzije *OpenCV* biblioteke, čija verzija mora biti ili 3.0.0 ili novija da bi sadržavala modul za rad s *fish-eye* kamerama.

```
import cv2
assert cv2.__version__[0] == '3', 'The fisheye module requires opencv version
>= 3.0.0'
import numpy as np
```

Definirane su dimenzije kalibracijskog uzorka (ploče) na kalibracijskim slikama.

```
CHECKERBOARD = (6,9)
```

Linija koda ispod definira putanju do kalibracijskih slika. Neke od kalibracijskih slika korištenih u ovom diplomskom radu prikazane su na Sl.3.3.. Ovu putanju je potrebno promijeniti ovisno o tome gdje se nalaze kalibracijske slike. Dimenzije kalibracijskog uzorka i putanja do datoteke u kojoj se nalaze kalibracijske slike su jedine promjene koje je potrebno napraviti u skripti za kalibraciju kamere. Bitna stavka kalibracijskih slika jest da obuhvaćaju razne kutove i usmjerenja kalibracijske ploče kako bi program mogao bolje odrediti distorziju odnosno parametre kamere. U P.3.1. dani su komentari koda.

```
images=glob.glob('C:/Users/dane/Desktop/Diplomski/python/gPictures/*.jpg')
```

Sl.3.3. *Neke od kalibracijskih slika korištenih za kalibraciju kamere u ovom diplomskom radu*

Ukoliko su instalirane potrebne biblioteke za rad skripte, skripta se pokreće tako što se pokrene *Command Prompt* i postavi u direktorij gdje se nalazi skripta, gdje se ovisno o operacijskom sustavu upisuju se komande:

- Windows: samo naziv skripte u ovom slučaju *undistortKoef.py* obavezno s ekstenzijom „.py“, zatim na tipkovnici pritisak „Enter“.
- Verzija Linux-a: *python 3.6.3 undistortKoef.py*, zatim na tipkovnici pritisak „Enter“, odnosno ovisno o instaliranoj verziji i nazivu skripte.

Nakon što je skripta za kalibraciju kamere pokrenuta, prolazi kroz svaku kalibracijsku sliku te traži spojne točke, ukoliko su spojne točke pronađene prikazuje prihvaćenu kalibracijsku slika s naznačenim kutovima kalibracijske ploče, kao što je vidljivo na Sl.3.4. Kalibracijska ploča predstavlja 2D ravninu, koordinate označenih točaka na spojevima kvadratića su poznate zbog poznate veličine ploče. Jedna točka kalibracijske ploče uzima se za ishodište i udaljenosti svake druge točke su poznate. Tako program računa način projekcije točaka iz stvarnog svijeta

(kalibracijske ploče) na senzor kamere i parametre kamere. Izlazne vrijednosti ove skripte su izračunati parametri kamere, odnosno vrijednosti matrica **K** i **D**, koji su potrebni za rad skripte *stitchFourImages.py* kako bi se uklanjala distorzija ulaznih slika. Razlozi zbog kojih parametri mogu odstupati, odnosno program nedovoljno dobro procijeniti, a samim tim izobličenu sliku nedovoljno dobro ispraviti su:

- nevaljane kalibracijske slike
- Neodgovarajući položaj kalibracijskog uzorka na čitavoj slici. Dio kalibracijskog uzorka nije vidljiv na čitavoj slici.
- Nedovoljan broj kalibracijskih slika.
- Kalibracijske slike ne pokrivaju sva područja distorzije koju stvara kamera.
- Nedovoljno dobra razlučivost slike te nejasnost spojnih točaka kvadratića na kalibracijskom uzorku.

(a) (b)

Sl.3.4. *Naznačene spojne točke na kalibracijskim slikama prilikom kalibracije kamere (a) kalibracijska slika 1, (b) kalibracijska slika 2*

Skripta *stitchFourImages.py*:

U kodu ispod opisan je skup biblioteka potrebnih za rad skripte za ispravljane distorzije, prebacivanje u ptičju perspektivu, stapanje četiri slike u jednu. Funkcija za rotaciju slike je korištena iz biblioteke *Imutilis*, dok je *matplotlib* biblioteka korištena za iscrtavanje slike s konkretnim koordinatama pojedinog elementa, takvim ispisom se određuje pozicija slike, odnosno gdje bi se koja slika s pojedine kamere trebala nalaziti i koordinate kvadrata za prebacivanje slike u ptičju perspektivu.

```
import cv2
import numpy as np
import imutils
import matplotlib.pyplot as plt
```

U kodu ispod prikazano je definiranje dimenzije ulazne slike i parametara za transformaciju perspektive. Nakon što je sa slike uklonjena distorzija potrebno je odrediti varijable *src, dst* koje definiraju dvije paralelne linije koje na ispravljenoj slici nisu paralelne, dok kad se slika prebaci u ptičju perspektivu trebaju biti paralelne. Varijabla *pts* predstavlja koordinate točaka četverokuta za kvalitetnije prebacivanje slike u ptičju perspektivu. Ove vrijednosti su u skripti popunjene za određeni položaj kamere na automobilu. Ukoliko se koriste drugačije kamere i drugačiji položaj kamere u odnosu na površinu koju snima potrebno je odrediti vrijednosti tih varijabli.

```
#Size of input images:IMAGE_H/height, IMAGE_W/Width
IMAGE_H = 720
IMAGE_W = 1280

#defines which lines on undistorted images need to be parallel,
# so the src variable has start points, and the dst variable has the end
points of those lines
src = np.float32([[0, IMAGE_H], [1217, IMAGE_H], [0, 0], [IMAGE_W, 0]])
dst = np.float32([[431, IMAGE_H], [801, IMAGE_H], [0, 0], [IMAGE_W, 0]])

#->getTopViewSecondTime
pts = np.array([(464,0), (864,0), (1111,260), (216,293)], dtype = "float32")
```

Dobiveni kalibracijski parametri K i D iz skripte *undistortKoef.py* nad kalibracijskim slikama korištene *fish-eye* kamere prikazani su u kodu ispod. Važni za ispravljanje distorzije slike.

```
#->Undistortion parameters
DIM=(1280, 720) #picture dimensions
# Camera and Distortion coefficients matrix, got from undistortKoef.py
script
K=np.array([[335.4360116970886,0.0,638.3853408401494],
[0.0,335.6314521829435, 403.2844174394132], [0.0, 0.0, 1.0]])
D=np.array([[-0.010624391932770951],[0.0692322261552681],
[-0.018577927478565195], [0.0006725877509963431]])
```

Kod ispod definira putanje do slika s četiri kamere.

```
#Input streams
leftStream = r'C:/Users/rtrk/Desktop/python/getTop -
Copy/paintRefPictures1/leftDistorted' #putanja1
left=[]
topStream = r'C:/Users/rtrk/Desktop/python/getTop -
Copy/paintRefPictures1/frontDistorted'# putanja2
top=[]
rightStream = r'C:/Users/rtrk/Desktop/python/getTop -
Copy/paintRefPictures1/rightDistorted' # putanja3
right=[]
bottomStream = r'C:/Users/rtrk/Desktop/python/getTop -
Copy/paintRefPictures1/backDistorted' # putanja4
bottom=[]
```

Unutar vlastite funkcije *merge_images* vidljive u P.3.2. za svaku sliku su definirane *x* i *y* koordinate, odnosno njihov položaj na čitavoj spojenoj slici (*x_offset*, *y_offset*). Kod ispod prikazuje koordinate lijeve slike na čitavoj spojenoj slici. Te koordinate se određuju ručno tako što se iscrta čitava spojena slika preko funkcija iz *matplotlib* biblioteke *plt.imshow* i *plt.show*. Na tako iscrtanoj slici treba odrediti za koliko je elementa slike potrebno sliku kamere translatirati da bi se poklapala s ostalim slikama kako treba. Unutar *for* petlje se prolazi kroz čitavu sliku te se postavljaju elementi slike dobivene transformacijom perspektive na konačnu sliku. Ovaj dio je fiksiran za određeni položaj kamera (fiksne koordinate), jer se položaj ne mijenja i nije potrebno stalno osvježavanje. Nadogradnja ovog algoritma bila bi traženje zajedničkih značajki na slikama, ukoliko bi kamere na automobilu imale određene oscilacije i bile nedovoljno dobro fiksirane, podatke bi trebalo stalno osvježavati te bi se položaj pojedine kamere na čitavoj slici generirao određivanjem zajedničkih značajki preklapajućeg područja dviju odgovarajućih kamera.

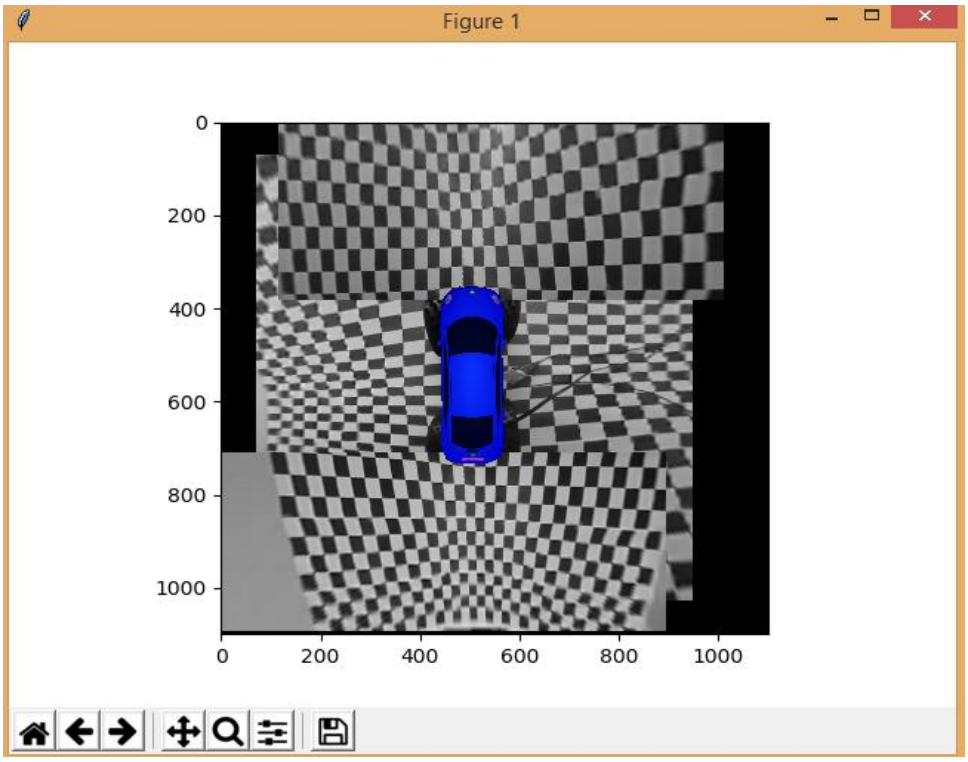
```
x_offset=70
y_offset=70
y1, y2 = y_offset, y_offset + left.shape[0]
x1, x2 = x_offset, x_offset + left.shape[1]
alpha_s = left[:, :, 3] / 255.0
alpha_l = 1.0 - alpha_s
for c in range(0, 3):
      Stitched[y1:y2, x1:x2, c] = (alpha_s * left[:, :, c] + alpha_l *
Stitched[y1:y2, x1:x2, c])
```

Ako su instalirane potrebne biblioteke za rad skripte i upisane eventualne izmjene opisane iznad, skripta se pokreće tako što se otvori *Command Prompt* i postavi u direktorij gdje se nalazi skripta i ovisno o operacijskom sustavu upisuju se komande:

- Windows: samo naziv skripte u ovom slučaju *stitchFourImages.py*, obavezno s ekstenzijom „.py“, zatim na tipkovnici pritisak „Enter“.
- Verzija Linux-a: *python 3.6.3 stitchFourImages.py*, zatim na tipkovnici pritisak „Enter“, odnosno ovisno o instaliranoj verziji i nazivu skripte.

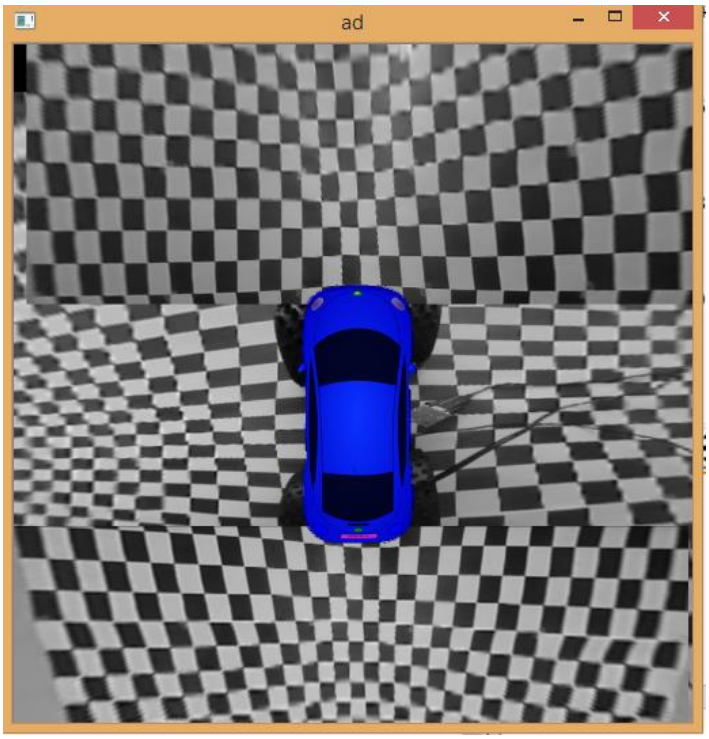
Na Sl.3.5. i Sl.3.6. prikazane su spojene slike dobivene unutar Python programskog jezika. Sl.3.5. prikazuje spojenu sliku s vidljivim koordinatama pojedinog elementa slike, čime je olakšano definiranje položaja pojedine slike na čitavoj spojenoj slici. Jednostavnim promjenama iznos varijabli *offset* definiramo položaj slike.

Slika 3.6. sadrži automobil i najvažniji prostor oko automobila izrezan sa Sl.3.5.



Sl.3.5. *Prikaz spojene slike pomoću funkcija iz matplotlib biblioteke (Python rješenje)*

.



Sl.3.6. *Prikaz spojene slike izrezane sa Sl.3.5. (Python algoritam za generiranje pogleda prostora oko automobila)*

3.2. Rješenje za generiranje pogleda prostora oko automobila u C++ programskom jeziku

Godine 1979. Stroustrup, danski računalni znanstvenik tvrtke Bell Labs-a, započeo je razvoj C++ jezika. C++ je objektno orijentirani programski jezik. Razvijen je kao proširenje programskom jeziku C i smatra se jezikom srednje razine, jer enkapsulira programske značajke i visokih i niskih programskih jezika. Glavna značajka ovog programskog jezika je velik broj prije definiranih klasa, kao što su tipovi podataka, koji se mogu instancirati više puta te omogućava korisniku stvaranje vlastitih klasa. Unutar klasa mogu se implementirati različite funkcije, kako bi se omogućila određena funkcionalnost klase. C++ sadrži binarne operatore, aritmetičke, operacije nad bitovima, operatore usporedbe i logičke operatore i omogućava preopterećenje (engl. *overloading*) nekih operatora.

Generirani pogled prostora oko automobila je realiziran pomoću jedne C++ datoteke koja ispravlja distorziju slike, zatim ju prebacuje u ptičju perspektivu te međusobno spaja slike koje su prebačene u ptičju perspektivu u jednu konačnu sliku kako bi se generirao pogled prostora oko automobila.

3.2.1. Funkcije i biblioteke korištene za realizaciju algoritma u C++

Unutar C++ programskog jezika algoritam za generiranje pogleda prostora oko automobila je realiziran C++ datotekom *stitchImages.cpp* koja se nalazi u P.3.3. i podataka dobivenih iz Python skripte *undistortKoef.py* koji predstavljaju kalibracijske parametre kamere koji su ujedno zajednički za oba programska jezika, odnosno nije potrebno realizirati kalibraciju kamere u C++ programskom jeziku, već je dovoljno preuzeti kalibracijske parametre dobivene pokretanjem kalibracijskog algoritma u Python programskom jeziku.

Rješenje u C++ programskom jeziku koristi samo *OpenCV* biblioteku i funkcije iz te biblioteke[11]:

- Funkcija *cv::imread(const String & filename, int flags = IMREAD_COLOR)* služi za učitavanje konkretne slike određene putanje i naziva. Ulazni parametri funkcije su *filename* naziv ulazne datoteke (slike), *flags* zastavice koje definiraju tip učitane slika.
- Funkcija *estimateNewCameraMatrixForUndistortRectify(InputArrayK, InputArray D, const Size & image_size, InputArray R, OutputArray P, double balance = 0.0, new_size = const Size & Size(), double fov_scale = 1.0)* računa novu matricu kamere koja služi za uklanjanje distorzije slike. Ulazni parametri su: ***K*** matrica kamere, ulazni vektor distorzije

*D, image_*size veličina slike, ***P*** nova izračunata matrica (izlazna vrijednost ove funkcije), *balance* postavlja novu vrijednost fokusa između minimalne i maksimalne vrijednost fokusa (vrijednosti u intervalu [0,1]), *new_size* nova veličina, *fov_scale* faktor za novu žarišnu duljinu.

- Funkcija *remap(InputArray src, OutputArray dst, InputArray map1, InputArray map2, int interpolation, int borderMode=BORDER_CONSTANT, Scalar&borderValue=Scalar())* primjenjuje geometrijsku transformaciju nad slikom. Ulazni parametri su jednaki kao i kod Python funkcije ovog naziva te su u prethodnom poglavlju i objašnjeni.
- Funkcija *fisheye::initUndistortRectifyMap(InputArray K,InputArray D, InputArray R,InputArray P,const cv::Size & size, int m1type, OutputArray map1,OutputArray map2)* služi za izračunavanje karti za uklanjanje distorzije slike. Ulazni i izlazni parametri su jednaki kao i kod Python funkcije ovog naziva te su u prethodnom poglavlju i objašnjeni.
- Funkcija *create(Size size, int type)* služi za alokaciju memorije. Ulazni parametri funkcije su: *size* veličina nove matrice: *Size(cols, rows), type* tip podataka nove matrice.
- Funkcija *rotate (InputArray src, OutputArray dst, int rotateCode)* rotira sliku na tri moguća načina, 90° i 180° u smjeru kazaljke na satu i za 90° obrnuto od smjera kazaljke na satu. Ulazni parametri su: *src* ulazna slika, *dst* odredišna slika, *rotateCode* je kod rotacije odnosno makro koji definira za koliko stupnjeva i u kojem smjeru će slika biti zarotirana.
- Funkcija *fourPointTransform* je vlastita funkcija napravljena za potrebe transformacije slike u pogled odozgo. Koristi *OpenCV* funkcije: *getPerspectiveTransform* i warpPerspective, koje imaju ista svojstva kao i Python funkcije tih naziva te su tamo i objašnjene.
- Funkcija *mergeImagesFourPoints* je vlastita funkcija napravljena za spajanje slika, jednostavno pristupa elementima slike iteracijom kroz *Mat* objekte (slike) sadržane u *OpenCV* biblioteci. Iteriranjem kroz sliku pristupa se točno određenim elementima slike, koji se postavljaju na točno određenu lokaciju na spojenoj slici.

3.2.2. Implementacija algoritma u C++

U ovom dijelu je objašnjena upotreba prethodno objašnjene C++ datoteke i potrebne izmjene koda. Programsko rješenje u C++ koristi funkcije samo iz *OpenCV* biblioteke. U kodu ispod definirane su putanje do ulaznih slika nad kojima se izvršavaju geometrijske operacije. Ako su ulazne slike u drugim direktorijima potrebno je izmijeniti ovaj dio koda.

```
// path to images
  Mat left = imread("paintLinedPictures/leftPaint.bmp", IMREAD_COLOR);
  Mat leftUnd;
  Mat right = imread("paintLinedPictures/rightPaint.bmp", IMREAD_COLOR);
  Mat rightUnd;
  Mat front = imread("paintLinedPictures/frontPaint.bmp", IMREAD_COLOR);
  Mat frontUnd;
  Mat back = imread("paintLinedPictures/backPaint.bmp", IMREAD_COLOR);
  Mat backUnd;
  Mat car = imread ("car1.png", IMREAD_COLOR);
```

U kodu ispod su podaci dobiveni kalibracijom kamere u Python programskom jeziku i isti se mogu koristiti kao ulazni podaci za ispravljanje distorzije slike u C++. Varijable su drugačije definirane u C++ programskom jeziku nego u Pythonu, odnosno tipovi se ne definiraju dinamički dodjelom vrijednosti nego svaki tip je strogo definiran. Ako je drugačiji tip kamera potrebno je ponovno napraviti kalibraciju kamere te promijeniti podatke unutar varijabli *K* i *D*.

```
//Camera params used from python, same procedure for camera calibration in
C++
  Matx33d   K   (335.4360116970886,   0.0,   638.3853408401494   ,   0.0,
335.6314521829435, 403.2844174394132,  0, 0, 1);
  Vec4d D (-0.010624391932770951,0.0692322261552681, -0.018577927478565195,
0.0006725877509963431);
```

Unutar vlastite funkcije *fourPointTransform* nalaze se koordinate 4 točke (*src[0]*, *src[1]*…) o kojima ovisi koje će se područje slike prebaciti u ptičju perspektivu te veličinu četverokuta definiranog s te 4 točke na slici. Veličina četverokuta (1000, 250) određuje raširenost između elemenata u četverokutu, definirana je u funkciji *warpPerspective* i vidljiva u kodu ispod.

```
  //Points
  src[0] = Point2f(420, 380);
  src[1] = Point2f(870, 370);
  src[2] = Point2f(1260, 500);
  src[3] = Point2f(20, 515);
  // resolution in warpPerspective transformation defines the ratio betwen
elements, this resolution depends of number of charts.
  warpPerspective(draw, (*img), M, Size (1000, 250),INTER LINEAR,0);
```

Unutar funkcije *mergeImagesFourPoints* definirano je spajanje slike u povezani pogled, tako da se određuje lokacija elementa ulazne slike (kojoj je ispravljena distorzija i perspektiva) na točnu *x*, *y* lokaciju na odredišnoj slici. Te koordinate su definirane za svaku od četiri kamere. U kodu iznad definirana je transformacija slike dobivene s kamere postavljene na desnoj strani automobila. Transformacijom perspektive je definirana razlučivost slike i vidljivo je da se iterira kroz čitavu sliku, no lokacija elementa slike na odredišnoj slici je korigirana određenim pomakom. Ovisno o željenim lokacijama pojedinog pogleda na spojenoj slici potrebno je ispraviti lokacije elementa

slike.

```
//right image stitching on stitched image->all pixels
  for (j = 0; j<1000;j++)
  for(i = 0; i<250;i++)
  {
    // getting a specified pixel from stitched and from right image
    (*stitched).at<cv::Vec3b>(j,i+405) = (right).at<cv::Vec3b>(j,i);
    //x offset on big (stitched) image for pasting right image pixels
  }
```

Pokretanje C++ rješenja:

Skripta se pokreće tako što se otvori *Command Prompt* i postavi u direktorij gdje se nalazi skripta i upisuje komanda kako bi se generirala izgrađena (engl. *builded*) datoteka, kojom su ustvari vidljivi rezultati:

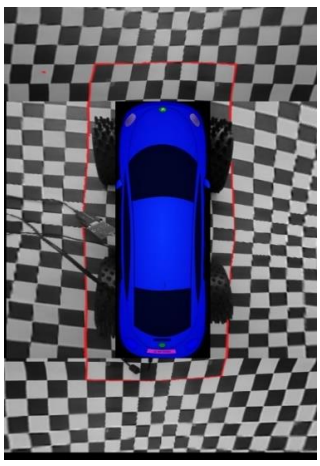
- „g++ -std=c++11 stitchImages.cpp `pkg-config --libs --cflags opencv` -o test“

Nakon što se izgradi C++ datoteka s potrebnim modifikacijama u kodu, potrebno je samo pokrenuti izvršnu datoteku dobivenu gradnjom *.cpp* datoteke.

Izvršna datoteka se pokreće tako što se ponovno unutar *Command Prompt* postavi u direktorij gdje se nalazi izgrađena datoteka i jednostavnim upisom naziva izgrađene datoteke i pritiskom tipke „Enter“ na tipkovnici, u ovom slučaju pokretanje bi se izvodilo na način:

- Unutar *Command Prompt* prozora upisati: *test* jer je to naziv izvršne datoteke koji se definira prilikom gradnje *.cpp* datoteke, nakon toga pritisnuti tipku „Enter“ na tipkovnici.

Na Sl.3.7. vidljiva je spojena slika, odnosno kompletan pogled odozgo s dodanim pomoćnim linijama na referentnim slikama kako bi preklapanje slika bilo preciznije.



Sl.3.7. *Prikaz spojene slike (C++ algoritam za generiranje pogleda prostora oko automobila)*

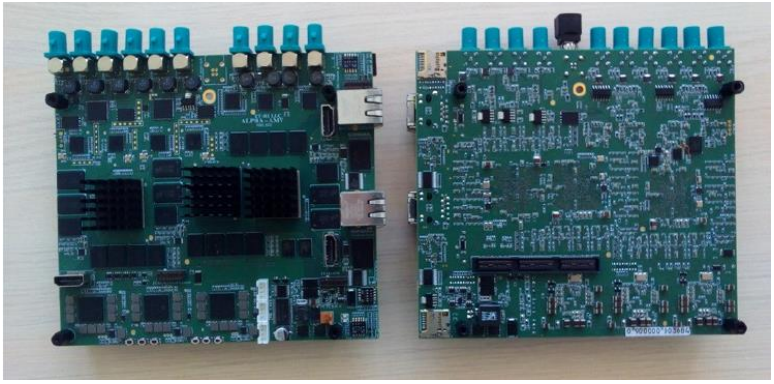
3.3. Implementacija rješenja na ADAS razvojnu platformu

ADAS su sustavi razvijeni kako bi poboljšali sigurnost vozača, nudeći tehnologije koje naznačuju vozaču potencijalne opasnosti i probleme. ADAS razvojna platforma oslanja se na više ulaza kao što su: kamere, laserski daljinometar, radari, međusobna komunikacija različitih sustava u vozilu. Za potrebe ovog rada korištena je ALPHA ploča koja je prikazana na Sl.3.8. i koristi TI (engl. *Texas Instruments*) čipove i ostale neophodne komponente za rad ADAS aplikacija te podržava[13]:

- osnovne i napredne upozoravajuće sustave
- stalne kontrolne sustave
- polu-autonomne operacije

te funkcionalnosti kao što su:

- pogled unazad
- upozorenja u prometu
- prepoznavanje znakova
- povezani pogled odozgo oko automobila
- monitori unutar vozila
- pomaganje vozaču u prometu i parkiranju
- samostalna vožnja autoputom prepoznajući linije na traci
- samoprilagodljiva brzina vožnje i razmaka između vozila



Sl.3.8. *ALPHA ploča korištena u ovom diplomskom radu [13]*

Na Sl.3.8. je prikazana ALPHA ploča, a neke od njenih značajki su:

- Na ploči se nalaze tri TDA2sxx čipa koji rade na istoj frekvenciji (600 MHz), iste su organizacije memorije te istog tipa. Podijeljeni su prema funkcijama koje izvršavaju:

#1 SC (engl. *Surround Camera*) čip - za pokretanje algoritma povezanog pogleda odozgo oko automobila.

#2 FFN (engl. *Front view camera near angle stereoscopic view, Froint view camera wide angle, Night vision camera*) čip – za pokretanje algoritama za prepoznavanje znakova, procjenu brzine i sl.

#3 FUS (engl. *Fusion*) čip - za obradu informacije senzora priključenih na prva dva čipa.

- Načini rada sustava - otkrivanje grešaka (engl. *Debug*), SD (engl. *Secure Digital*) kartica, *flash*.
- Priključci - dva mrežna priključka, HDMI (engl. *High-Definition Multimedia Interface*) izlaz za svaki čip, 6 priključaka za kameru (za SC čip), 4 priključka za kameru (za FFN čip), JTAG priključak za svaki čip.

3.4.1. Problemi prilikom implementacije rješenja na ALPHA ploču

Algoritam povezanog pogleda prostora oko automobila pokreće se na SC čipu ALPHA ploče, pomoću SD kartice i korištenjem četiri *fish-eye* 24B kamere koje se priključuju na četiri od šest dostupnih priključaka kamere na tom čipu.

24B kamera je kamera napravljena na RDACM24B platformi pogodna za ADAS aplikacije. Opskrbljuje videom elektroničku upravljačku jedinicu (ECU, engl. *Electronnic control unit*) koji procesira podatke s kamere u informacije koje određuju radnju vozila. Pomoću koaksijalnog kabla kamera se povezuje s ECU, koji služi za prijenos informacija i napajanje kamere. Leća i elektronički dijelovi kamere su smješteni unutar kućišta kamere koje je napravljeno da osigura rad kamere u raznim vremenskim uvjetima.

Početna ideja zadatka diplomskog rada bila je realizirati algoritam za generiranje pogleda prostora oko automobila u nekom od viših programskih jezika te pretvoriti u C programski jezik, a nakon toga implementirati na ADAS (ALPHA) ploči. Kako bi se obuhvatilo svih 360° oko automobila koriste se *fish-eye* kamere, zbog čega je potrebno prvo kalibrirati kameru. Zbog toga programsko rješenje u Pythonu sadrži i algoritam za kalibraciju kamere i algoritam za generiranje pogleda prostora oko automobila. Algoritam za kalibraciju nije potrebno implementirati na ploči jer se samo jednom koristi, dok se algoritam za generiranje pogleda prostora oko automobila razvijen u Python programskom jeziku se pokušao postupno pretvoriti u niže programske jezike, a u konačnici u C programski jezik. Zbog toga je taj algoritam iz Pythona prebačen u C++

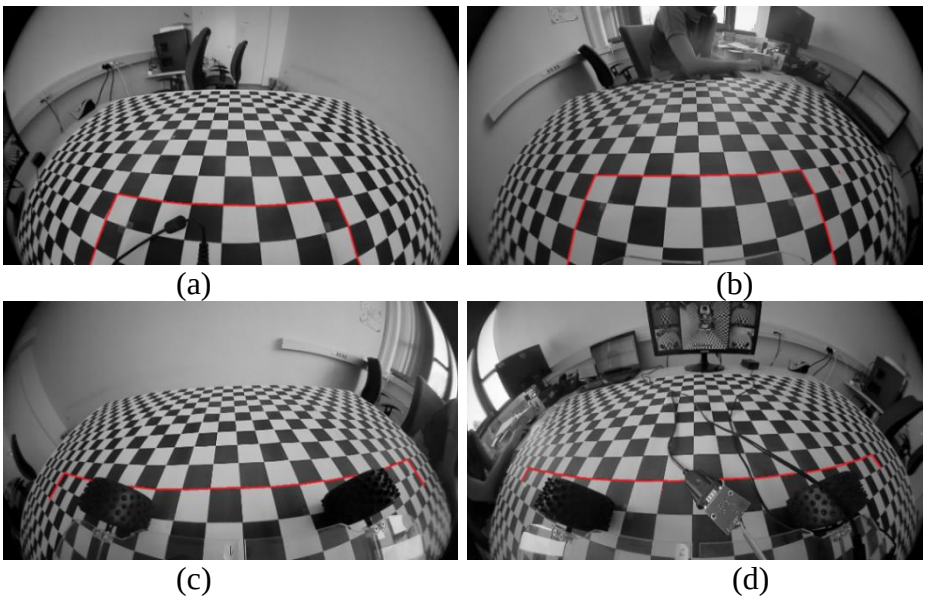
programski jezik iz kojeg se zatim pokušao prebaciti u C programski jezik. Algoritam izrađen u C++ programskom jeziku koristi *OpenCV* biblioteku te je potrebno sve korištene funkcije iz C++ programskog jezika pretvoriti u C programski jezik. Prilikom prebacivanja C++ funkcija u C funkcije nije bilo moguće prebaciti funkciju za ispravljanje distorzije slike u C programski jezik jer koristi procesorske funkcije kojima se ne može pristupiti [14]. Iz tog razloga C programsko rješenje nije prikazano u diplomskom radu. *OpenCV* koji se nalazi u programskoj podršci (VISION_SDK) ALPHA ploče uspješno je pokrenut na ploči, čime je iskorišteno previše resursa, što je uzrokovalo sporo izvršavanje algoritma na ploči. Nakon toga osposobljen je postojeći slučaj korištenja (engl. *use-case*) povezanog pogleda prostora oko automobila koji se također nalazi u spomenutoj programskoj podršci te je napravljen novi slučaj korištenja po uzoru na postojeći za koji su namijenjeni podaci dobiveni u vlastitom algoritmu. Te podatke nije moguće iskoristiti zato što postojeći algoritmi za dobivanje povezanog pogleda koriste unaprijed popunjene LUT tablice te im nije moguće mijenjati broj ulaza i izlaza, odnosno svaki algoritam ima unaprijed definirane ulaze, izlaze i sam broj ulaza i izlaza. Slučaj korištenja definira način pokretanja ploče, obuhvaćene algoritme, kamere koje se koriste, njihovo pokretanje i izlazne podatke tog slučaja korištenja.

Nakon dugog vremenskog perioda pokušavanja implementacije algoritma za generiranje pogleda prostora oko automobila, algoritam nije uspješno implementiran na ALPHA ploču zbog prije navedenih poteškoća na koje se nailazilo prilikom pokušavanja implementacije. Te poteškoće se mogu generalizirati kao: ograničenja memorije i nedovoljno dobre programske podrške ploče u kojoj nisu implementirani svi potrebni moduli za algoritme opisane u ovom radu.

4. TESTIRANJE ALGORITMA ZA GENERIRANJE POGLEDA PROSTORA OKO AUTOMOBILA

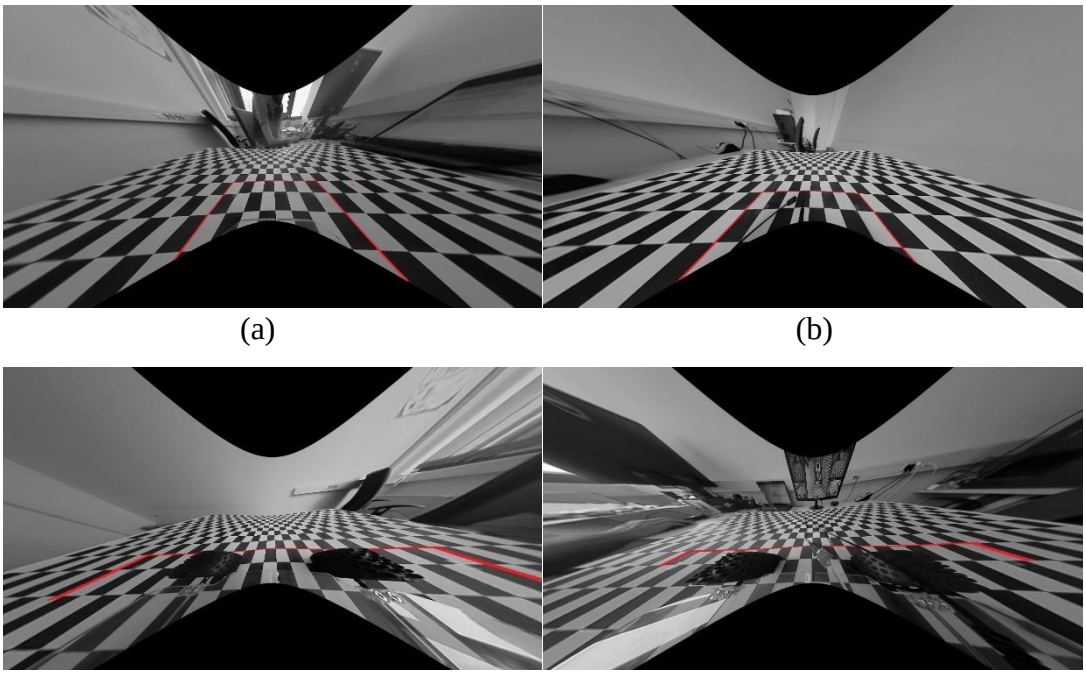
Algoritam za generiranje pogleda prostora oko automobila je kompletno razvijen u Python i C++ programskim jezicima. Python i C++ koriste iste funkcije iz *OpenCV* biblioteke za realizaciju algoritma. Samim tim rezultati rješenja se ne bi trebala razlikovati. Zbog toga je korišten drugačiji pristup prebacivanja slike u ptičju perspektivu (transformacija perspektive) u C++ i Python algoritmu, tako da su u C++ algoritmu postignuti bolji rezultati.

Za potrebe kalibracije kamere korišteno je nekoliko slika koje obuhvaćaju različite položaje kalibracijske ploče, a samim tim i distorzije elementa na različitim lokacijama slike. Slike su procesirane pomoću ALPHA ploče i *fish-eye* kamere. Kako bi se spremile slike s kamere potrebne za kalibraciju, napravljen je novi slučaj korištenja u programskoj podršci ploče, koji video određenog trajanja šalje na računalo putem mreže i šalje na HDMI izlaz SC čipa te se na računalu video klipovi spremaju. Problem koji se događao prilikom slanja slika je taj što je slika treperila na tada dostupnoj verziji ALPHA ploče R.002, tako da se za svaki položaj kalibracijske ploče morao spremiti čitav video, a zatim izvući određeni okvir videa na kojemu nema pogrešaka stvorenih treperenjem kamere. Pripremljene slike za kalibraciju kamere su vidljive u P.4.1. Za potrebe testiranja generiranja pogleda prostora oko automobila, originalne slike s testnog automobila su izmijenjene tako što su dodane pomoćne linije crvene boje oko vozila, kako bi se lakše odredilo preklapanje pojedine slike na cjelokupnom pogledu. Na platnu oko vozila su vidljivi svjetliji rubovi na spoju susjednih četverokuta, prema čemu orijentiralo pri određivanju položaja pomoćnih linija (Sl.4.1.).



Sl.4.1. *Testne slike s pomoćnim linijama: (a) prednja, (b) stražnja, (c) lijeva, (d) desna kamera*

Rezultati ispravljanja distorzije slike stvorene prilikom korištenja *fish-eye* kamere prikazani su na Sl.4.2. i identični za oba programska jezika. Prilikom ispravljanja distorzije *fish-eye* slike vidljivo je kako su ravne linije postale ravne, no primjećuju se crne površine na vrhu i dnu svake slike i određeno produljenje nekih objekata na slikama. Crne površine se stvaraju jer se prilikom ispravljanja distorzije elementi slike ustvari „guraju“ na drugu lokaciju, samim time neki dijelovi slike se produljuju neki smanjuju. Primjetno je kako su kotači automobila u ravnini što bi i trebalo tako biti nakon ispravljanja slike.

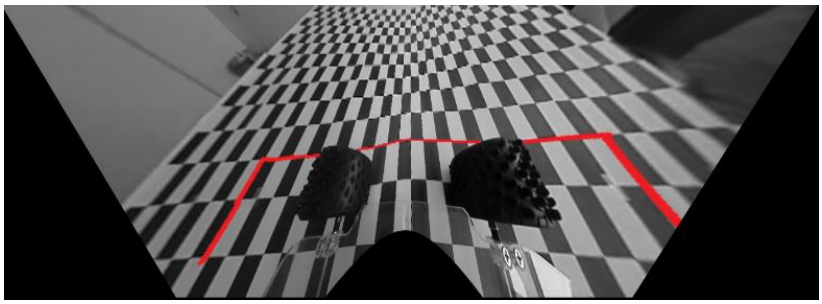


(a) (b)

(c) (d)

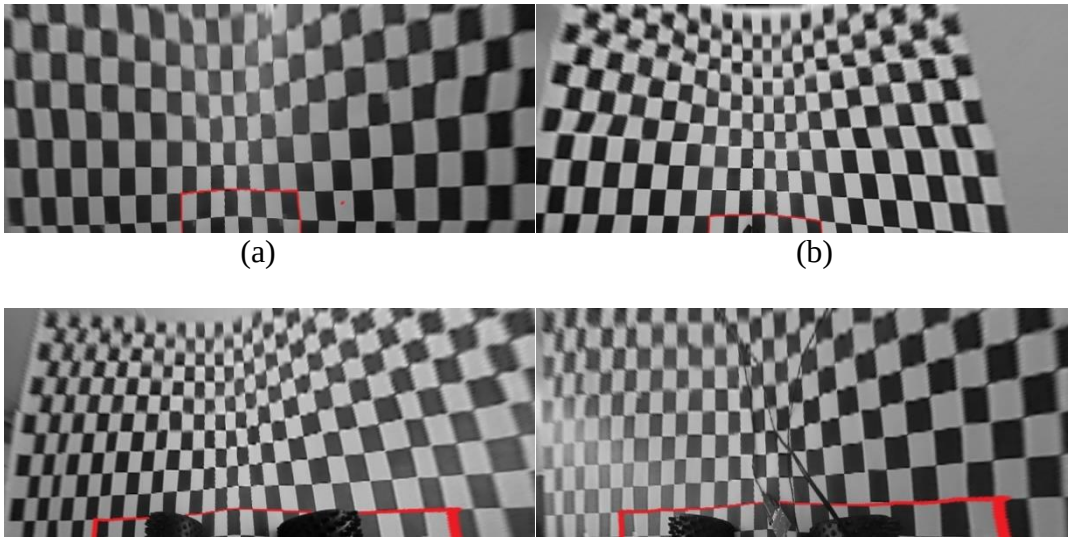
Sl.4.2. *Ispravljena distorzija slika prikazanih na Sl.4.1. (a) prednja, (b) stražnja, (c) lijeva, (d) desna*

Transformacija perspektive u Python programskog jeziku je napravljena tako što se nad ispravljenom slikom primjene dvije transformacije perspektive. Rezultat prve transformacije perspektive prikazan je na Sl.4.3.



Sl.4.3. *Rezultat prve transformacije perspektive (lijeva kamera) u Python programskom jeziku*

Na Sl.4.3. je vidljivo kako su kvadratići produljeni i deformirani. Zatim se nad rezultatom prve transformacije perspektive primjenjuje još jedna transformacija perspektive, kako bi se dobio pravilniji oblik kvadratića. Na Sl.4.4. prikazan je rezultat druge transformacije perspektive.

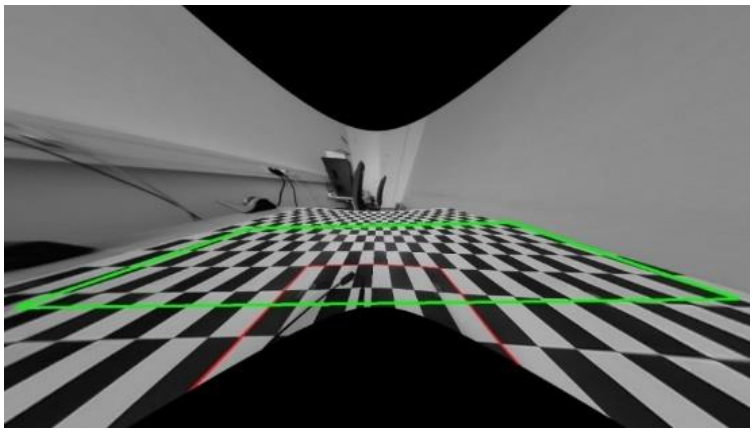


Sl.4.4. *Rezultat druge transformacije perspektive u Python programskom jeziku nad slikama prikazanim na Sl.4.2. (a) prednja, (b) stražnja, (c) lijeva, (d) desna*

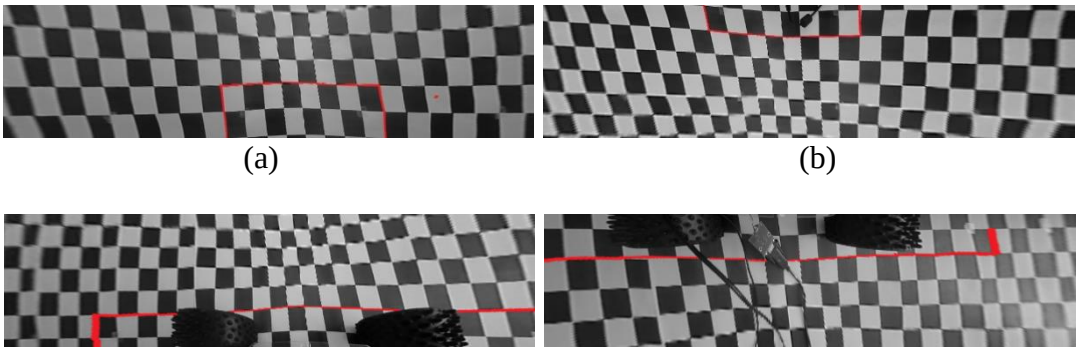
Na Sl.4.4. su vidljiva izobličenja slike nastala primjenom još jedne transformacije perspektive. Slika je savijena na sredini. Time se zaključuje da se treba pokušati dobiti što bolji rezultat što manjim brojem primjena određene funkcije. Osim distorzije koja se stvori većim brojem primjene određene funkcije, dolazi i do povećanja potrošnje računalnih resursa, što produljuje vrijeme izvođenja programa.

U C++ programskom rješenju transformacija perspektive slike je napravljena jednom primjenom funkcije. Prvo je definiran četverokut (Sl.4.5.) nad ispravljenom slikom čije dimenzije određuju raširenost elemenata unutar njega. Brojanjem kvadratića u horizontalnom i vertikalnom smjeru moguće je odrediti dimenzije (širinu i visinu) četverokuta. Primjenom funkcije za transformaciju perspektive u C++ dobiju se rezultati prikazani na Sl.4.6.

Konačno spojena slika je stvorena tako što su ručno određene koordinate položaja pojedinog pogleda transformiranih perspektiva sa Sl.4.6. (C++ rješenja) i Sl.4.4. (Python rješenja). Ručno su određene koordinate pogleda jer su kamere na vozilu fiksirane i ti podaci se ne mijenjaju. Preklapanje pogleda je određeno pomoću pomoćnih crvenih linija dodanih na osnovne testne slike. Definiranje položaja pojedinog pogleda definira se međusobnim preklapanjem pomoćnih crvenih linija koje formiraju četverokut oko vozila na kojemu su postavljene kamere.



Sl.4.5. *Definirani četverokut određenih dimenzija za transformaciju perspektive u C++ programskom jeziku*



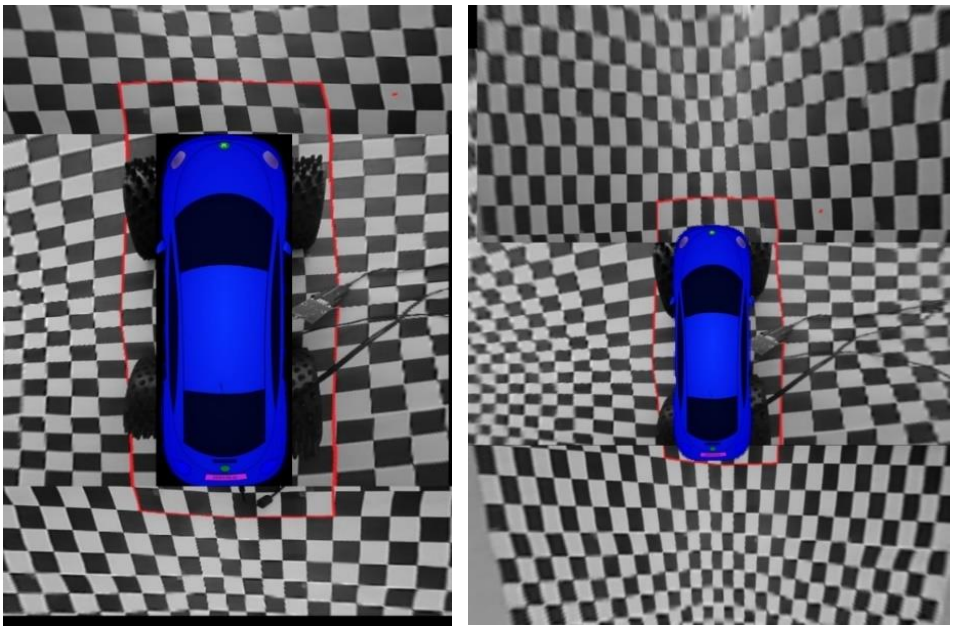
(a) (b)

(c) (d)

Sl.4.6. *Rezultat transformacije perspektive u C++ nad ispravljenim slikama prikazanim na Sl.4.2. (a) prednja, (b) stražnja, (c) lijeva, (d) desna*

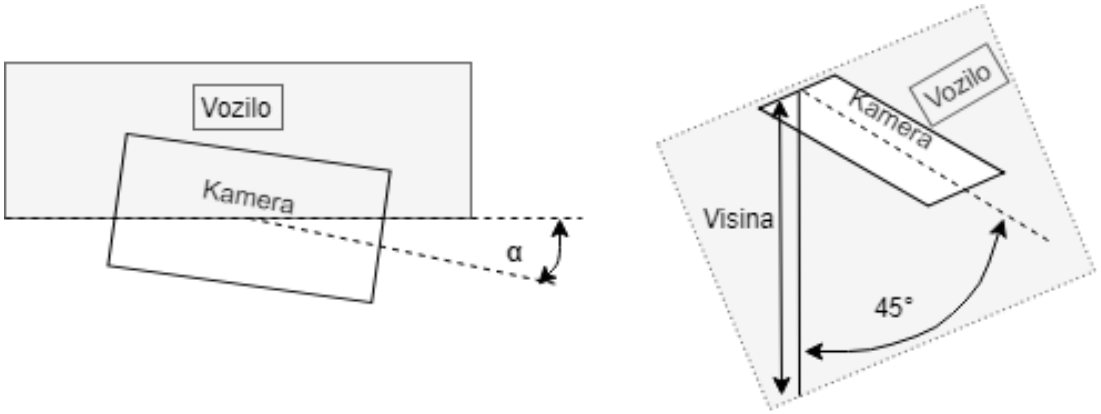
Krajnji rezultat povezanog pogleda prikazan je na Sl.4.7(a), gdje C++ je rezultat uvećan u usporedbi s Python rezultatom, no primjetna je manja distorzija i bolje oblikovanje kvadratića na podlozi oko vozila nego u Python rezultatu algoritma Sl.4.7(b).

Za potrebe testiranja uzete su slike zabilježene pomoću *fish-eye* kamera koje nisu pravilno postavljene na testnom vozilu, odnosno nisu u istom položaju kao kamere postavljene na testnom vozilu za prvi testni slučaj (Sl.4.1.). Pretpostavlja se da kamere nisu pravilno postavljene zbog toga što kut (α) između vozila i kamere prikazan na Sl.4.8(a) nije jednak nuli i/ili nije pod kutom od 45° (Sl.4.8(b)) u odnosu na vozilo. Sve kamere na automobilu moraju biti postavljene na jednakoj visini.



(a) (b)

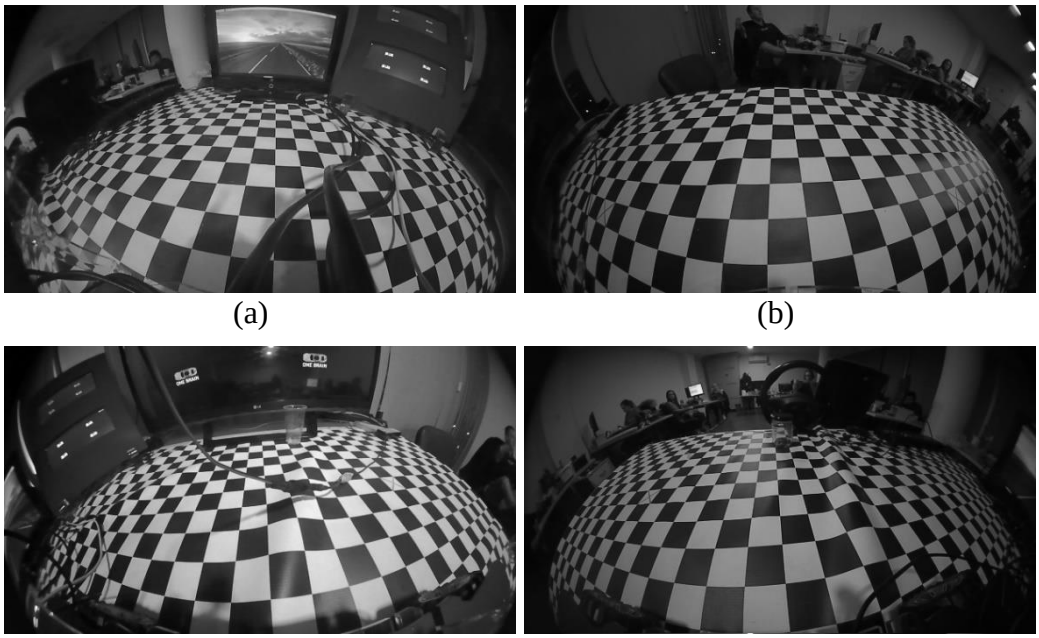
Sl.4.7. *Konačni rezultat povezanog pogleda odozgo prostora oko automobila: (a) C++, (b) Python*



(a) (b)

Sl.4.8. *Položaj kamere na testnom automobilu (a) tlocrt, (b) bokocrt*

Na Sl.4.9. prikazane su testne slike koje su korištene za prikazivanje rezultata algoritam za generiranje pogleda prostora oko automobila (C++), prilikom drugačijeg postavljanja kamera na automobil. Usporedbom testnih slika prikazanih na Sl.4.9. i Sl.4.1. primjećuje se drugačija veličina vidnog područja kamera i drugačija usmjerenost kamera zbog druge pozicije montiranja na automobilu. Za testne slike prikazane na Sl.4.9. kao i za slike prikazane na Sl.4.1. korištene su kamere istog tipa, zbog čega je uklanjanje distorzije tih slika uspješno korištenjem prije dobivenih kalibracijskih parametara.

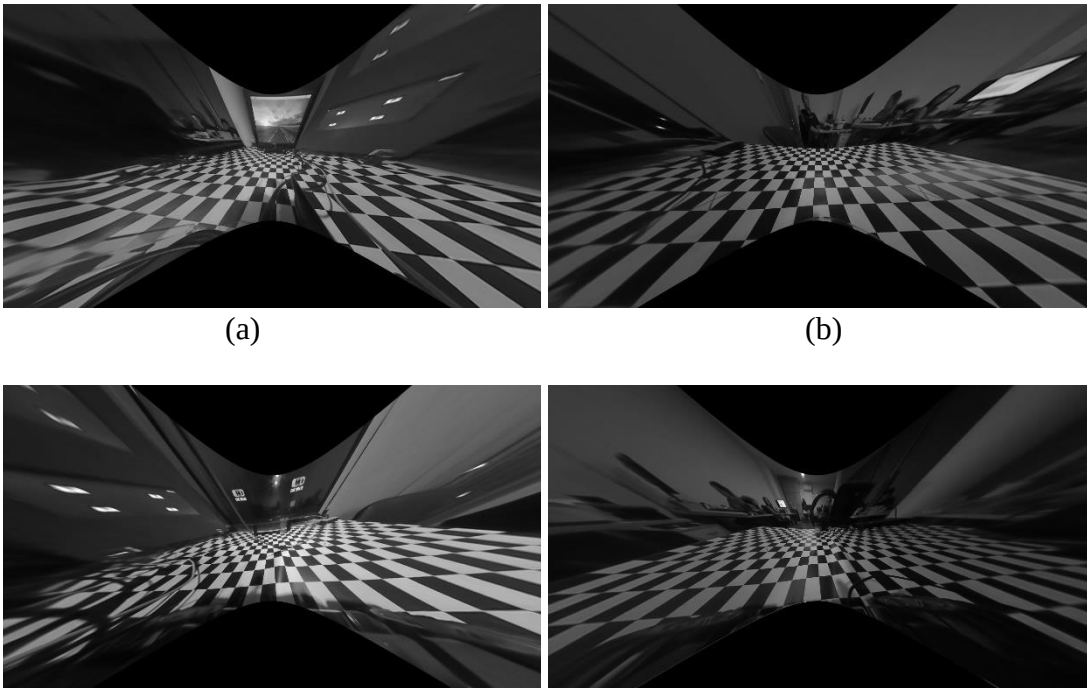


(a) (b)

(c) (d)

Sl.4.9. *Testne slike s drugim položajem kamera na automobilu: (a) prednja, (b) stražnja, (c) lijeva, (d) desna*

Na Sl.4.10. je vidljivo kako su rezultati ispravljanja distorzije prilično dobri, no već je primjetno kako je površina na kojoj je automobil, zakrivljena odnosno da kamere na automobilu nisu postavljane pod istim kutovima kao na testnim slikama prikazanima na Sl.4.1.

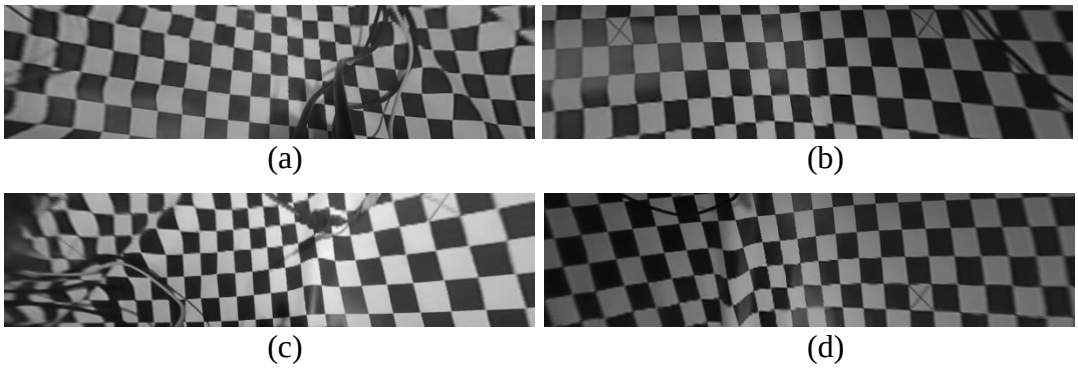


(a) (b)

(c) (d)

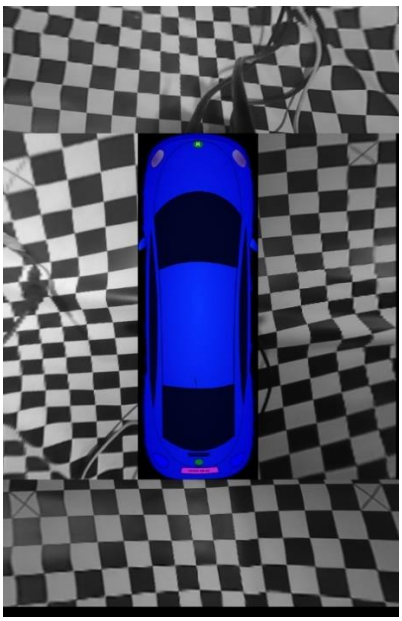
Sl.4.10. *Ispravljena distorzija slika prikazanih na Sl.4.9. (a) prednja, (b) stražnja, (c) lijeva, (d) desna*

Primjenom transformacije perspektive nad ispravljenim slikama dobivaju se loši rezultati zato što je transformacija perspektive napravljena s podacima definiranim za slike prikazane na Sl.4.2. Samim tim zaključuje se kako se zbog promjene položaja i usmjerenja kamera na automobilu moraju ponovno definirati podaci za transformaciju perspektive. Na Sl.4.11. prikazane su slike nakon što je nad njima primijenjena transformacija perspektive s prije definiranim podacima transformacije.



Sl.4.11. *Transformacija perspektive nad ispravljenim slikama prikazanim na Sl.4.10. (a) prednja, (b) stražnja, (c) lijeva, (d) desna kamera*

Nakon transformacije perspektive dolazi do povezivanja slika prikazanih na Sl.4.11. prema definiranim položajima za slike prikazane na Sl.4.6. koje su zabilježene na automobilu s drugačijim pozicijama i usmjerenjima kamera. Iz tog razloga i rezultati povezivanja slika je lošiji na Sl.4.12. nego na Sl.4.7(a).



Sl.4.12. *Konačni rezultat povezanog pogleda odozgo prostora oko automobila (definirani parametri nisu za postavljeni položaj kamera)*

5. ZAKLJUČAK

Umjetna inteligencija automobila sve se više razvija kako bi se vozačima olakšala vožnja te sam vozač i putnici automobila bili sigurniji. U ovom diplomskom radu opisani su algoritmi kalibracije kamere i generiranja pogleda prostora oko automobila koji bi se mogli implementirati u sustav koji pomaže vozaču pri parkiranju automobila, odnosno pri vožnji u neposrednoj blizini predmeta i objekata koje je teško vidjeti iz automobila. Taj sustav se naziva prikaz prostora oko automobila (pomoću četiri *fish-eye* kamere). *Fish-eye* kamere su specifične po tome što imaju šire vidno područje 180°. Toliko vidno područje se postiže stvaranjem distorzije na slici koju je prije svega potrebno ispraviti. Za potrebe izrade spomenutih algoritama prvotno je korišten Python programski jezik korištenjem funkcija iz biblioteke *OpenCV*, zatim C++ također korištenjem funkcija biblioteke *OpenCV*.

Kako bi se ispravila distorzija slike, potrebno je prikupiti intrinzične parametre kamere koji se dobivaju postupkom kalibracije kamere. Za potrebe kalibracijskih slika potrebnih za postupak kalibracije kamere snimljeno je nekoliko videa pomoću ALPHA ploče, kamere 24B (*fish-eye*) i slučaja korištenja ploče implementiranog u njenoj programskoj podršci u kojemu je definiran način snimanja i spremanja videa s ploče na računalo putem mreže. Iz kalibracijskih videa preuzeto je nekoliko okvira koji su najpogodniji za kalibracijske slike. Dobiveni parametri kamere korišteni su u Python i C++ programskom rješenju algoritma. Konačno rješenje povezanog pogleda sastoji se od nekoliko faza nakon kalibracije kamera. Prva faza je ispravljanje distorzije svake kamere, zatim prebacivanje okvira svake kamere u ptičju perspektivu, rotacija potrebnih okvira te njihovo spajanje i pozicioniranje na konačnoj slici. Prilikom izrade algoritma veliki problem je stvarao proces ispravljanja distorzije slike zbog nevaljanih kalibracijskih slika. Prilikom povezivanja slika različitih pogleda, fiksirani su podaci pozicije slike svake kamere zato što se kamere nalaze na fiksnom položaju na automobilu i nepotrebno je stalno računanje preklapajuće matrice za generiranje prikaza prostora oko automobila. U algoritmu su podaci koje je moguće fiksirati (ispravljanje distorzije slike, transformacija perspektive, položaj slike svake kamere na konačnoj slici) fiksirani zato što svaka dodatna operacija usporava algoritam. Zbog dimenzija automobila više podataka nose bočne kamere nego prednja i stražnja kamera, zbog toga što bočne kamere daju podatke o okolini dulje dimenzije automobila. U vlastitom algoritmu generiranja povezanog pogleda odozgo više podataka uzima se s bočnih kamera.

Rezultati algoritma za kalibraciju kamere izravno ovise o kvaliteti kalibracijskih slika, što utječe na ispravljanje distorzije slike i predstavlja važan dio za daljnje geometrijske operacije nad

slikom. Algoritam za generiranje pogleda prostora oko automobila jako dobro ispravlja sliku dobivenu spomenutom *fish-eye* kamerom, što znači da su kalibracijski parametri dobiveni iz algoritma za kalibraciju kamere dobri. Dio algoritma za generiranje pogleda prostora oko automobila koji realizira prebacivanje slika u ptičju perspektivu daje dobre rezultate za definirane parametre transformacije i određenu poziciju kamere na automobilu, dok za iste parametre i drugu poziciju kamere ne dobiva se dobra transformacija perspektive, što znači da se za drugu poziciju kamere moraju definirati drugi parametri transformacije. Zadnji dio spomenutog algoritma predstavlja spajanje slika transformirane perspektive u povezani pogled, gdje se slike svake kamere postavljaju na određenu lokaciju na konačnoj slici, ovisno o vidnom području kamera. U ovom dijelu ručno su definirane lokacije slika kamera. Nedostatak algoritma za generiranje pogleda prostora oko automobila je što ne povezuje automatski slike prema njihovim zajedničkim značajkama.

Podaci dobiveni algoritmom za kalibraciju poznate kamere su konačni i koriste se za ispravljanje distorzije slike u algoritmu za generiranje pogleda prostora oko automobila. Algoritam za generiranje pogleda prostora oko automobila nije uspješno realiziran na ALPHA ploči zbog njenih ograničenja memorije i nedovoljno dobre programske podrške u kojoj nisu implementirani svi potrebni moduli za algoritme opisane u ovom radu.

LITERATURA

[1] Photogrammetry & Robotics, Bonn, 2015, http://www.ipb.uni-bonn.de/ [3.9.2018.]

[2] D. Balihar, PINHOLE.CZ, http://www.pinhole.cz/en/index.php [28.8.2018.]

[3] M. Kosanović, Diplomski rad „Metode kalibracije kamera“, Zagreb, 2010.

[4] M. Holl, N. Heran, V. Lepetit, Augmented Reality Oculus Rift, Inst. for Computer Graphics and Vision Graz University of Technology, Austria, 5.2018.

[5] The MathWorks, Developing an Algorithm to Undistort Pixel Locations of an Image, https://jp.mathworks.com/products/demos/symbolictlbx/Pixel_location/Camera_Lens_Undistortion.html [3.9.2018.]

[6] M. Venkatesh, P. Vijayakumar, Transformation Technique, International Journal of Scientific & Engineering Research, Vol. 3, Issue 5, 5.2012, https://www.ijser.org/researchpaper/A-Simple-Birds-Eye-View-Transformation-Technique.pdf [28.7.2018.]

[7] B. Zhang, V. Appia, I. Pekkucuksen, A. U. Batur, P. Shastry, S. Liu, S. Sivasankaran, K. Chitnis, A surround view camera solution for embedded systems, Conference on Computer Vision and Pattern Recognition Workshops, IEEE, 2014., str. 676-681

[8] S.M. Santhanam, V. Balisavira, S.H. Roh, Lens Distortion Correction and Geometrical Alignment for Around View Monitoring System, ISCE, IEEE, 2014., str. 1-2

[9] X. Yan, Y. Li, L. Wang, The Research of Surround View Parking Assist System, International Conference on Computer and Communications, IEEE, 3rd, 2017. , str.2145-2148

[10] K. Jiang, Calibrate fisheye lens using OpenCV—part 2, Medium, 2017, https://medium.com/@kennethjiang/calibrate-fisheye-lens-using-OpenCV-part-2-13990f1b157f [28.8.2018.]

[11] OpenCV documentation, https://docs.opencv.org/ [3.9.2018.]

[12] N. Falaleev, Bird's Eye View Transformation, OpenCV, https://nikolasent.github.io/OpenCV/2017/05/07/Bird's-Eye-View-Transformation.html [30.8.2018.]

[13] RT-RK, Automotive Machine Vision ALPHA reference board on Texas Instruments SoCs, http://www.rt-rk.com/download/rt-rk_ALPHA_ADAS_board.pdf [3.9.2018.]

[14] OpenCV team ,CPU optimizations build options, https://github.com/OpenCV/OpenCV/wiki/CPU-optimizations-build-options [3.9.2018.]

SAŽETAK

Diplomski rad opisuje algoritam za kalibraciju kamere i algoritam za generiranje pogleda prostora oko automobila koji se mogu iskoristiti u sustavu za parkiranje automobila. Algoritam za kalibraciju kamere na osnovu kalibracijskih slika izračunava parametre kamere koji služe za ispravljanje distorzije slike te su ovom radu opisane važne stavke prilikom kalibracije kamere kalibracijskim uzorkom. U algoritmu za generiranje pogleda prostora oko automobila definirane su potrebne geometrijske operacije nad slikom kako bi se generirao cjelokupni povezani pogled koje se mogu podijeliti na: ispravljanje distorzije, transformacija perspektive i spajanje slika. Algoritam uspješno radi ukoliko su na automobil postavljene kamere s poznatim kalibracijskim parametrima i za unaprijed predviđeni položaj kamera na automobilu.

Ključne riječi: distorzija slike, fish-eye, transformacija perspektive, povezivanje slika, kalibracija kamere, povezani pogled odozgo, auto-motiv.

OVERVIEW OF THE SPACE ARROUND THE CAR FROM DIFFERENT VIEWS USING THE CAMERA SYSTEM

ABSTRACT

This paper describes a camera calibration algorithm and an algorithm for generating the view around the car that can be used in a car park system. The camera calibration algorithm based on calibration images calculates camera parameters used to eliminate image distortion, and this paper describes important items used in camera calibration with calibration pattern. In the algorithm for generating the view around the car, the geometric operations of the image are defined in order to generate the entire related view that can be divided into: distortion correction, perspective transformation, and image stitching. The algorithm works successfully if the placed cameras are with known calibration parameters and for the predefined position of the cameras on the car.

Keywords: image distortion, fish-eye, perspective transformation, image stitching, camera calibration, surround view, automotive.

ŽIVOTOPIS

Daniel Buljeta je rođen u Ozimici, Žepče 20. prosinca 1994.godine. Tijekom osnovnoškolskog obrazovanja sudjeluje na natjecanju iz matematike, te u Žepču, 2009. godine završava osnovnu školu „OŠ Žepče“. Iste godine upisuje srednju školu za zanimanje „Tehničar za mehatroniku“ u Katoličkom školskom centru „Don Bosco“ u Žepču. Tijekom srednjoškolskog obrazovanja sudjeluje na natjecanjima iz mehanike u Osijeku, te iz osnova elektrotehnike u Zagrebu. Srednju školu završava 2013. godine te iste godine upisuje preddiplomski studij elektrotehnike na Elektrotehničkom fakultetu u Osijeku. U rujnu 2016. godine završava preddiplomski studij i upisuje diplomski studij komunikacija i informatike na istom fakultetu.

U Osijeku, rujan 2018.

Daniel Buljeta

PRILOZI

Prilog P.3.1. – Skripta za prikupljanje podataka kamere potrebnih za ispravljanje distorzije slike, nalazi se u digitalnom formatu na CD-u priloženom u mapi *PythonSV* pod nazivom *undistortKoef.py*. Komentari koda dani su u prilogu na priloženom CD-u.

Prilog P.3.2. – Skripta za generiranje povezanog pogleda, nalazi se u digitalnom formatu na CD-u priloženom u mapi *PythonSV* pod nazivom *stitchFourImages.py*. Komentari koda dani su u prilogu na priloženom CD-u.

Prilog P.3.3. – Programski kod za generiranje povezanog pogleda, nalazi se u digitalnom formatu na CD-u priloženom u mapi *SV_C++* pod nazivom *stitchImages.cpp*. Komentari koda dani su u prilogu na priloženom CD-u.

Prilog P.4.1. – Slike potrebne za kalibraciju 24B kamere, nalaze se u digitalnom formatu na CD-u priloženom u mapi *cal_Images*.