

Monokularna i stereo vizualna odometrija za autonomna vozila

Kandidat: Faris Hajdarpašić

Mentor: doc.dr. Dinko Osmanković

Završni rad na
I ciklusu studija



Elektrotehnički fakultet Sarajevo
Odsjek za automatiku i elektroniku
Univerzitet u Sarajevu
Bosna i Hercegovina
Ak. godina 2020/2021.

Sažetak

Ovaj rad se bavi problemom vizualne odometrije. Zadatak vizualne odometrije jeste određivanje pozicije i orijentacije agenta koji se kreće, koristeći neki sistem kamere. Korišten je monokularni i stereo sistem kamere. Prezentovani i objašnjeni su glavni aspekti vizualne odometrije, od načina projekcije scene na slikovnu ravan kamere, do detekcije značajki, njihove deskripcije i metoda podudaranja, kao i princip stereo vizije. Dalje, objašnjeni su principi estimacije matrice relativne transformacije i to korištenjem 2D-2D i 2D-3D korespondencije značajki. Prikazani su rezultati dobijeni na osnovu *KITTI* podataka te podataka dobijenih korištenjem *CARLA* simulatora autonomne voznje.

Abstract

This paper deals with visual odometry problem. The task of visual odometry is estimation of position and orientation of moving agent, by using some camera system. Monocular and stereo camera system are used. Main aspects of visual odometry are presented and explained, from scene projection to the image plane, to feature detection, its description and matching methods, as well as the principle of stereo vision. Furthermore, principles of relative transformation matrix estimation are explained using 2D-2D and 2D-3D feature correspondence. The results obtained on the basis of KITTI dataset and data obtained using the CARLA autonomous driving simulator are presented.

Univerzitet u Sarajevu

Naziv fakulteta/akademije: Elektrotehnički fakultet

Naziv odsjeka i/ili katedre: Odsjek za automatiku i elektroniku

Predmet: Završni rad

Izjava o autentičnosti radova

Seminarski rad, završni (diplomski odnosno magistarski) rad za I i II ciklus studija i integrirani studijski program I i II ciklusa studija, magistarski znanstveni rad i doktorska disertacija¹.

Ime i prezime: Faris Hajdarpašić

Naslov rada: Monokularna i stereo vizualna odometrija za autonomna vozila

Vrsta rada: Završni rad za I(prvi) ciklus studija

Broj stranica: 84

Potvrđujem:

- Da sam pročitao/la dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- Da sam svjestan/na univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- Da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- Da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- Da sam jasno naznačio/la prisustvo citiranog ili parafraziranog materijala i da sam se referirao/la na sve izvore;
- Da sam dosljedno naveo/la korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- Da sam odgovarajuće naznačio/la svaku pomoć koju sam dobio/la pored pomoći mentora/ice i akademskih tutora/ica.

Mjesto, datum: _____

Potpis kandidata: _____

Potpis mentora: _____

¹U radu su korišteni sljedeći dokumenti: Izjava autora koju koristi Elektrotehnički fakultet u Sarajevu; Izjava o autentičnosti završnog rada Centra za interdisciplinarne studije – master studij „Evropske studije”, Izjava o plagijarizmu koju koristi Fakultet političkih nauka u Sarajevu.

Sadržaj

Sažetak	iii
Abstract	iv
Popis slika	x
Popis tabela	xi
1. Uvod	1
1.1. Model perspektivne kamere	2
2. Opis problema	4
2.1. Vizuelna odometrija	4
2.1.1. Formulacija problema vizuelne odometrije	5
2.2. Značajke slike	8
2.3. Detektori značajki	10
2.3.1. Detektori uglova	10
2.3.2. Detektori grumenova	11
2.4. Deskriptor značajki	20
2.4.1. SIFT deskriptor	21
2.4.2. SURF deskriptor	22
2.4.3. BRIEF deskriptor	24
2.5. Podudaranje značajki	25
2.5.1. BF Matcher	26
2.5.2. FLANN Matcher	27
2.6. Stereo vizija	27

2.6.1. Pojednostavljeni slučaj	28
2.6.2. Generalni slučaj	29
2.6.3. Triangulacija	30
2.6.4. Problem korespondencije tačaka	32
2.7. Zaključak	37
3. Opis rješenja	38
3.1. Estimacija kretanja	38
3.2. 2D-2D estimacija kretanja	40
3.3. 3D-2D estimacija kretanja	43
3.4. CARLA simulator	45
3.4.1. Klijent	46
3.4.2. Svijet	46
3.4.3. Mapa	47
3.4.4. Biblioteka nacrtta	49
3.4.5. Akteri	49
3.4.6. Skripte u CARLA Simulatoru	50
3.5. Zaključak	51
4. Rezultati	52
4.1. Trajektorije	52
4.2. Analiza performansi	65
4.2.1. 2D-2D estimacija kretanja	65
4.2.2. 3D-2D estimacija kretanja	66
4.3. Analiza kvaliteta	67
4.3.1. 2D-2D estimacija kretanja	68
4.3.2. 3D-2D estimacija kretanja	69
4.4. Zaključak	69
5. Zaključak	71

Popis slika

1.1.	Model perspektivne kamere	2
1.2.	Proces prelaska iz 3D koordinata svijeta u 2D koordinate slikovne ravni	3
2.1.	Transformacije koordinatnih sistema	6
2.2.	Koraci vizuelne odometrije	7
2.3.	DoG kao aproksimacija LoG-a	12
2.4.	Piramida slika različite rezolucije	13
2.5.	Konvolucija slike sa Gaussovim funkcijama različitih širina	13
2.6.	Prostorna skala	14
2.7.	Konvolucija originala sa LoG filterom različitih širina preko DoG-a	15
2.8.	Odredjivanje lokalnog maksimuma preko susjednih piksela različitih skala	15
2.9.	SIFT značajke različitih veličina	16
2.10.	Gradjeni u svakom pikselu SIFT značajke	17
2.11.	Histogram uglova gradjenata uzorka SIFT značajke	17
2.12.	Box filteri kao aproksimacija <i>LoG</i> filtera širine $\sigma = 1.2$ korišteni u SURF algoritmu	18
2.13.	Kreiranje piramide smanjivanjem rezolucije slike (lijevo) i povećavanjem dimenzije filtera (desno)	18
2.14.	Deskriptor	21
2.15.	SIFT deskriptor	21
2.16.	Histogram	22
2.17.	Klizni prozor na osnovu kojeg se određuje orijentacija značajke	23
2.18.	Haarov filter za x i y osu	23

2.19. Vektor od 4 elementa formiran izračunatim vrijednostima za svaki kvadrat uzorka	24
2.20. Različiti pristupi za odabir piksela koji će biti poređeni za BRIEF detektor	25
2.21. 3D tačka i njeni 2D korespondenti na paru slika stereo kamere	28
2.22. Stereo sistem - Pojednostavljeni slučaj	28
2.23. Stereo vizija - Generalni slučaj	29
2.24. Uticaj smetnji na rekonstrukciju 3D tačke	30
2.25. Geometrijska interpretacija određivanja 3D tačke	31
2.26. Korespondentne značajke stereo kamere	32
2.27. Epipolarna geometrija	33
2.28. Epipolarna geometrija - epipolarne linije stereo kamere	33
2.29. Rektificirane slike stereo sistema	34
2.30. Neporavnate slike stereo sistema	35
2.31. Poravnate slike stereo sistema	35
2.32. Rekonstrukcija 3D tačaka na osnovu stereo kamere	36
3.1. 2D-2D estimacija kretanja	39
3.2. 3D-3D estimacija kretanja	40
3.3. 3D-2D estimacija kretanja	40
3.4. Mape nekih gradova CARLA simulatora	48
3.5. Grad u CARLA simulatoru	48
3.6. Grad u CARLA simulatoru sa dodanim akterima	50
3.7. Manuelno upravljanje vozilom	50
4.1. Rezultati za KITTI set podataka koristeći SIFT i 2D-2D korespondenciju tačaka	54
4.2. Rezultati za CARLA set podataka koristeći SIFT i 2D-2D korespondenciju tačaka	56
4.3. Rezultati za KITTI set podataka koristeći ORB i 2D-2D korespondenciju tačaka	58

4.4. Rezultati za CARLA set podataka koristeći ORB i 2D-2D korespon-	
denciju tačaka	60
4.5. Rezultati za KITTI set podataka koristeći SIFT i 3D-2D korespon-	
denciju tačaka	62
4.6. Rezultati za KITTI set podataka koristeći ORB i 3D-2D korespon-	
denciju tačaka	64

Popis tabela

4.1.	Ukupno vrijeme izvršavanja za 2D-2D estimaciju na osnovu KITTI seta podataka	65
4.2.	Ukupno vrijeme izvršavanja za 2D-2D estimaciju na osnovu CARLA seta podataka	66
4.3.	Ukupno vrijeme izvršavanja za 3D-2D estimaciju na osnovu KITTI seta podataka	66
4.4.	Vrijeme izvršavanja jedne iteracije zajedničkih etapa oba načina esti- macije kretanja	67
4.5.	Vrijeme izvršavanja jedne iteracije etapa za 2D-2D estimaciju kretanja	67
4.6.	Vrijeme izvršavanja jedne iteracije etapa za 3D-2D estimaciju kretanja	67
4.7.	MSE po svim koordinatama za 2D-2D estimaciju na osnovu KITTI seta podataka	68
4.8.	MSE po svim koordinatama za 2D-2D estimaciju na osnovu CARLA seta podataka	68
4.9.	MSE po svim koordinatama za 3D-2D estimaciju na osnovu KITTI seta podataka	69

Poglavlje 1.

Uvod

Kompjuterska vizija je polje nauke ciji je cilj razvoj tehnika koje će pomoci računarima da "vide" i razumiju sadržaj digitalnih slika, kao što su fotografije i video snimci. To podrazumijeva ekstrakciju značajnih informacija koje nose fotografije i video snimci. Problem kompjuterske vizije izgleda jednostavan, jer je taj problem trivijalno rjesiv za ljudska bica. Međutim, kompjuterska vizija je bazirana na ne tako jednostavnim algoritmima. Ona podrazumijeva metode za prikupljanje, precesiranje, analizu i razumijevanje digitalnih slika od strane računara.

Zbog razvoja umjetne inteligencije, kompjuterska vizija je dosta uznapredovala. Brzi razvoj kompjuterske vizije, njena upotreba i njeno daljnje razvijanje su posljedica velike kolicine podataka koje se danas generisu. Da bi se ti podaci obradili potrebni su brzi i efikasni algoritmi. Također, modernizacija u raznim sferama nauke i života, dovela je do toga da su potrebne moćne i sofisticirane aplikacije, pa je kompjuterska vizija nasla svoj udio u tome, i uveliko potpomaze razvoj tehnologije.

Mnoge su primjene kompjuterske vizije. Autonomnost raznih vozila, objekata i robova podrazumijeva njihovo percipiranje svijeta i na osnovu toga, donose se daljnje odluke. Percipiranje svijeta se zasniva na kompjuterskoj viziji. Tako da bi postigli autonomnost, potrebno je integrirati kompjutersku viziju u takve aplikacije. Također, razvoj umjetne inteligencije i autonomnosti uveliko proistice iz razvoja vojne industrije. Pa se tako u modernom ratovanju koriste dronovi i bespilotne letjelice,

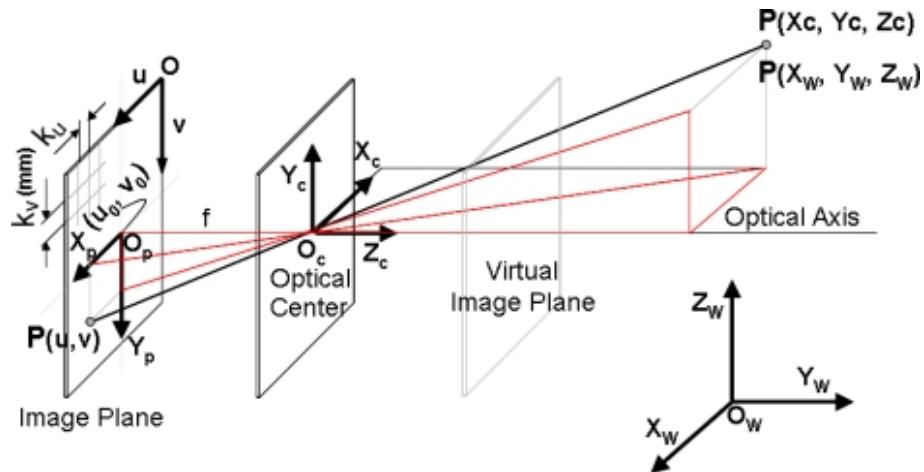
koje se pokazuju veoma efektivne, i vrse najzahtjevnije vojne operacije. Postoje još mnoge druge primjene kompjuterske vizije. Neke od njih su: autonomna vozila, prepoznavanje lica, AR, primjena u medicinske svrhe, svemirske operacije, itd.

Ovaj rad se bavi problemom vizuelne odometrije. Vizuelna odometrija omogućava određivanje trajektorije agenta, na osnovu sistema kamere. U ovom uvodnom poglavlju će biti objasnjen princip projekcije scene na slikovnu ravan kamere, što je veoma važan princip za daljnje razumijevanje.

1.1. Model perspektivne kamere

Proces projekcije 3D tačke koordinata svijeta (eng. *world frame*) (u metrima) u 2D tačku slike (eng. *image plane*) (u pikselima) može se objasniti na principu rada perspektivne kamere. Perspektivna kamera se može modelirati modelom rupe (eng. *pinhole model*). Stvarna kamera, umjesto rupe, ima leću, pa dolazi do izobličenja slike prilikom projekcije.

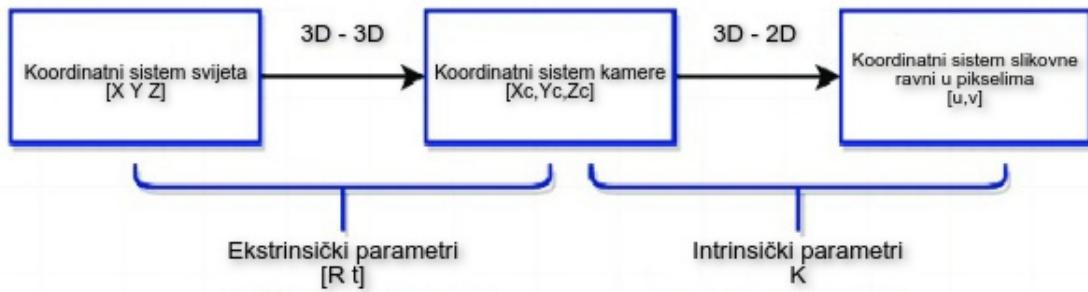
Kod modela rupe, slika se formira intersekcijom svjetlosnih zraka sa objekta, koje prolaze kroz centar projekcije, sa slikovnom ravni.



Slika 1.1. Model perspektivne kamere

Pretpostavimo da imamo neku 3D tačku u koordinatnom sistemu svijeta \mathbf{P}_w sa koordinatama $[x_w \ y_w \ z_w]$. Preslikavanje 3D tačke \mathbf{P}_w u 2D tačku cije su koordinate izražene u pikselima se vrši sljedećom procedurom:

1. Prebacivanje 3D tačke iz koordinatnog sistema svijeta (\mathbf{P}_w) u koordinatni sistem kamere (\mathbf{P}_c)
2. Projekcija 3D tačke (\mathbf{P}_c) u 2D tačku na slikevnu ravan sa koordinatama (x, y)
3. Konvertovanje koordinata (x, y) u koordinate izražene u pikselima (u, v)



Slika 1.2. Proces prelaska iz 3D koordinata svijeta u 2D koordinate slikevne ravni

Transformacija 3D tačke koordinatnog sistema svijeta u 3D tačku koordinatnog sistema kamere je definisana matricom **ekstrinsičkih** parametara.

Projekcija 3D tačke u koordinatnom sistemu kamere u 2D tačku slikevne ravni je definisana matricom **intrinsičkih** parametara. Ona sadrži podatke o fokalnoj dužini kamere i koordinatama centra slikevne ravni.

Matrica intrinsickih parametara \mathbf{K} i matrica ekstrinsickih parametara $[\mathbf{R} \ \mathbf{t}]$, zajedno formiraju matricu projekcije P .

$$\mathbf{P} = \mathbf{K} [\mathbf{R} \ \mathbf{t}] = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{21} & r_{22} & r_{23} & t_{21} \\ r_{31} & r_{32} & r_{33} & t_{31} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \quad (1.1.)$$

Matrica projekcije \mathbf{P} projicira 3D tačku iz svjetskih koordinata u 2D tačku (koordinate izraženih u pikselima) slikevne ravni.

Poglavlje 2.

Opis problema

U prethodnom poglavlju smo se upoznali sa baznim stvarima kompjuterske vizije. U ovom poglavlju cemo se prvo upoznati sa time sta je vizuelna odometrija, kao i sa nekim koracima koje je potrebno ispuniti pri analizi ovog problema. Bit ce objasnjeni algoritmi za detekciju i deskripciju znacajki. Takodjer, spomenut cemo dva nacina na koja se ove znacajke uporedjuju i na osnovu toga formiraju korepsodentni parovi znacajki. Bit ce spomenuta i estimacija kretanja, na osnovu koje se vrsti lokalizacija agenta. Na kraju poglavlja, dotaci cemo se stereo vizije, i objasniti neke principe vezane za stereovizijski sistem sa dvije kamere.

2.1. Vizuelna odometrija

Vizuelna odometrija (eng. *visual odometry*) je proces odredjivanja kretanja agenta, koristenjem samo ulaznih podataka jedne ili vise kamera nakacenih na njega. Pojam VO je prvi put pomenut 2004. godine od strane Nister-a u njegovom naučnom radu. Pojam je odabran zbog njegove sličnosti sa odometrijom tockova (eng. *wheel odometry*), koja postepeno određuje putanju vozila integriranjem broja skretanja njegovih tockova tokom vremena. Vizuelna odometrija radi na sličnom principu. Ona određuje putanju vozila na osnovu promjena na slikama koje daju kamere, a koje su uzrokovane promjenom kretanja vozila. Da bi VO radila korektno, potrebno je da scena koju snima kamera ima dovoljno osvjetljenja i teksture. Takodjer, konsekutivni frejmovi koje snima kamera trebaju imati dovoljno preklapanja tj. trebaju

imati minimalno dovoljno istih informacija scene koju slikaju.

Prednost VO u odnosu na WO jeste da VO ne ovisi od terena po kojem se kreće vozilo, jer npr. neravan teren neće uticati na kameru, dok na tockove hoće. Tako da je VO precizniji nacin odredjivanja trajektorije u odnosu na WO. VO u sprezi sa drugim navigacionim sistemima (GPS,IMU,...) može dati odlicne rezultate.

2.1.1. Formulacija problema vizuelne odometrije

Agent se kreće kroz okruzenje i uzima slike pomocu rigidno nakacenog sistema kamere na njega, u diskretnim vremenskim trenucima. Postoje dva tipa vizuelne odometrije:

- Monokularna VO - koristi se jedna kamera
- Stereo VO - koriste se dvije kamere

U monokularnom sistemu, set slika u diskretnim vremenima je ozначен sa $I_{0:n} = I_0, \dots, I_n$. U stereo sistemu, postoje dvije kamere (lijeva i desna) pa se set slika za obje kamere u diskretnim vremenima označava sa $I_{L,0:n} = I_{L,0}, \dots, I_{L,n}$ za lijevu kameru i $I_{R,0:n} = I_{R,0}, \dots, I_{R,n}$ za desnu kameru.

Radi jednostavnosti, uzima se da je koordinatni sistem kamere odgovara koordinatnom sistemu vozila. Treba napomenuti da se za stereo VO, transformacije koordinata i odredjivanje kretanja vozila, po pravilu, određuje u odnosu na koordinatni sistem lijeve kamere.

Za dvije pozicije kamere u diskretnim vremenskim trenucima $k - 1$ i k , transformacija koordinata jedne pozicije kamere u odnosu na drugu je opisana matricom transformacije $T_{k,k-1} \in \mathbb{R}^{4x4}$:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \quad (2.1.)$$

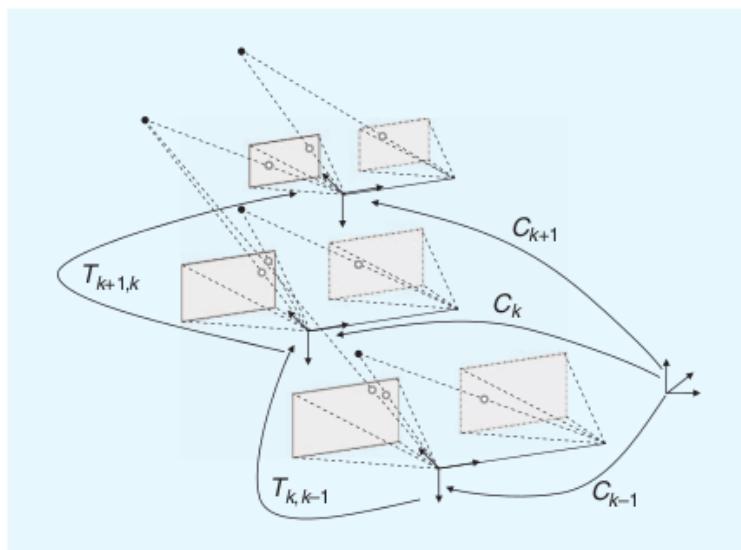
gdje je:

- $R_{k,k-1} \in \mathbb{R}^{3x3}$ - matrica rotacije

- $t_{k,k-1} \in \mathbb{R}^{3x1}$ - vektor translacije

Radi jednostavnosti možemo koristiti i notaciju T_k umjesto $T_{k,k-1}$.

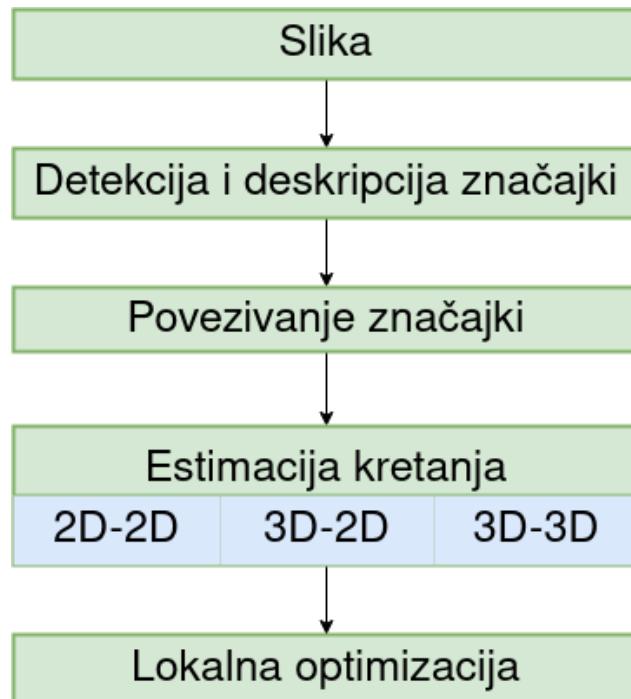
Set $T_{w:n} = T_{0,w}, T_{1,0}, \dots, T_{n,n-1}$ sadrzi sve transformacije izmedju uzastopnih frej-mova. $T_{0,w}$ predstavlja transformaciju koordinata izmedju koordinatnog sistema svijeta i prvog frejma (indeksacija pocinje od nule). Vecinom se uzima da se koordinatni sistem svijeta i koordinatni sistem prvog frejma poklapaju, pa je ova matrica matrica identiteta dimenzija 4x4 (jer nema rotacije - matrica identiteta, i nema rotacije - nul-vektor). Sada kada imamo ove uzastopne transformacije, možemo formirati set transformacija koje se odnose na transformacije kamere u svakom diskretnom trenutku k u odnosu na koordinatni sistem svijeta. Pa to označavamo sa $C_{w:n} = C_{0,w}, C_{1,w}, \dots, C_{n,w}$. Odatle, možemo izvuci matricu rotacije i translacije, te znati kako je vozilo orijentisano i koliko je udaljeno u diskretnom vremenskom trenutku k u odnosu na koordinatni sistem svijeta. Trenutna poza C_m se racuna matricnim mnozenjem svih transformacija T_k ($k=0,\dots,m$). Posto svaka trenutna poza C_k je samo transformacija vise od prethodne, C_k se može racunati i kao $C_k = C_{k-1} \cdot T_k$.



Slika 2.1. Transformacije koordinatnih sistema

Glavni zadatak u vizuelnoj odometriji jeste odrediti uzastopne transformacije (tj. relativne transformacije) T_k iz slika I_k i I_{k-1} , te kontinuirano spajati ove transformacije, da bi iz njih izdvojili pozicije vozila u odnosu na mjesto pocetka njegovog kretanja. Znaci vizuelna odometrija konstruise putanju, pozu po pozu. Također moguce je vrsiti iterativno usavršavanje putanje na zadnjih m frejmova, tako da se dobije bolja aproksimacija. Ovo iterativno usavršavanje radi na principu minimiziranja sume kvadratne greske reprojekcije rekonstruisanih 3D tacaka u zadnjih m frejmova (eng. *window bundle adjustment*). 3D tačke se dobijaju postupkom triangulacije 3D tacaka na osnovu 2D tacaka slikovne ravnini.

Koraci koje je potrebno ispostovati za VO su prikazani na slici 2.2..



Slika 2.2. Koraci vizuelne odometrije

Za svaku sliku (ili par slika u slučaju stereo kamere) prva dva koraka se sastoje od detekcije i deskripcije znacajki i povezivanja znacajki sa onima iz prethodnog frejma. Znacajke na dva razlicita frejma, koje predstavljaju reprojekciju istih 3D tacaka se nazivaju korespondentne znacajke (eng. *image/features correspondences*). Znacajke 2 razlicita frejma se mogu povezivati (eng. *feature matching*) ili pratiti (eng. *feature tracking*). Povezivanje znacajki jeste traženje istih znacajki na dva razlicita frejma

koristeci neku metriku sličnosti, kao što je vec objasnjeno ranije, dok pracenje značajki podrazumijeva detektovanje znacajki na jednom frejmu i njihovo pracenje na frejmovima sto slijede pomoću neke tehnike lokalnog pretrazivanja (npr. korelacija).

Naredni korak jeste estimacija kretanja. Ona podrazumijeva da se odredi relativna transformacija $T_{k,k-1}$. U zavisnosti od toga kako su znacajke predstavljene (u 2D ili 3D prostoru) postoje razliciti pristupi rjesavanju ovog problema. Naknadno ce biti detaljnije govora o ovim pristupima. Na osnovu relativnih transformacija, racunaju se poze kamere (i agenta) u odnosu na koordinatni sistem svijeta, tako sto se pretvodna poza pomnozi sa trenutnom relativnom transformacijom.

Na kraju, iterativno usavršavanje (*bundle adjustment*) može se izvršiti na posljednjih m frejmova da dobijemo precizniju estimaciju trajektorije agenta.

U nastavku cemo se detaljnije upoznati sa detekcijom i deskripcijom znacajki (eng. *feature detection and description*), kao i sa njihovim povezivanjem (eng. *feature matching*), te ce biti prezentovani nacini estimacije kretanja agenta (eng. *motion estimation*) i principima na kojima su oni bazirani.

2.2. Značajke slike

Detekcija znacajki slike (eng. *Feature detection*) i njihovo povezivanje na razlicitim slikama (eng. *Feature matching*) je jedan od znacajnijih dijelova kompjuterske vizije, te ima koristi u raznim aplikacijama.

Znacajke su specificki uzorci na slikama. Znacajka se sastoji od ključne tačke (eng. *keypoint*) i deskriptora. Ključna tačka je piksel na slici. Oko ključne tačke se uzima uzorak piksela. Taj uzorak opisujemo pomoću deskriptora. Dvije najznačajnije vrste znacajki su:

- Ugao (eng. *corner*)
- Grumen (eng. *blob*)

Postoje dva glavna nacina za nalazenje znacajki i njihovih korespondenata. Prvi nacin jeste da nadjemo znacajke jedne slike i onda ih pratimo na ostalim slikama koristeci tehniku lokalnog pretrazivanja (korelacija ili najmanji kvadrati). Drugi nacin jeste da neovisno odredimo znacajke dvije ili vise slika i potom ih povezemo.

Detekciju ključnih tacaka i njihovo povezivanje ćemo podijeliti na tri faze:

- Detekcija znacajki - na svakoj slici trazimo lokacije koje se mogu lako povezati sa istim na drugoj slici. Te lokacije su ustvari pikseli koji se nazivalju ključne tačke (eng. *keypoints*)
- Deskripcija znacajki - uzorak sastavljen od piksela oko detektovane ključne tačke se zapisuje kao tzv. deskriptor koji se poslije koristi za poredjenje sa drugim deskriptorima
- Povezivanje ili pracenje znacajki - traženje korespondenata znacajki dvaju slika.
Za ovaj dio su nam potrebni pomenuti deskriptori

Detektori i deskriptori moraju biti invarijantni na geometrijske i fotometrijske promjene. Fotometrijske promjene su promjene u osvjetljenju. Geometrijske promjene su:

- Rotacija
- Skaliranje
- Promjena perspektive

Dva glavna zahtjeva koja se moraju ispostovati da bi uspjeli naci korespondente znacajki jesu:

- Ponovljivost - osobina detektora znacajki. Detektor znacajki, ako detektuje znacajku na jednoj slici, bi trebao biti u mogucnosti detektovati istu znacajku na drugoj slici iste scene
- Pouzdanost i karakteristicnost - osobina deskriptora znacajki. Pouzdan deskriptor znacajke jasno identificira jednu znacajku od druge.

Pored invarijantnosti, ponovljivosti i pouzdanosti, neke dodatne osobine su:

- Tacnost lokalizacije (pri promjeni pozicije i skale)
- Efikasnost i brzina izvršenja
- Robusnost

Detektori uglova su brzi ali manje pouzdani za ponovnu detekciju, dok za detektore grumenova vrijedi obratno. Dodatno, uglovi su bolje lokalizirani od grumenova, ali pri promjeni skale su losije lokalizirani. To znaci da se uglovi manje redetektuju od gurmenova pri promjeni skale.

Svaki detektor znacajki na sliku primjenjuje odredjenu funkciju (tzv. *feature-response function*) na osnovu koje ce odrediti poslije lokalne maksimume/minimume (*non-maxima suppression*). Ti maksimumi/minimumi koji predstavljaju znacajke slike.

U koraku deskripcije znacajke, uzorak piksela oko detektovane ključne tačke se pretvara u kompaktnu formu koja je poslije pogodna za uporedjivanje sa drugim deskriptorima.

Za poredjenje takvih deskriptora koriste se neke metrike slicnosti kao npr SSD i NCC. Za binarne deskriptore moze se koristiti tzv. Hamming-ova udaljenost.

2.3. Detektori značajki

Detektori znacajki se mogu podijeliti na one koji detektuju uglove i oni koji detektuju grumenove. U nastavku ce biti navedeni jedni i drugi, te detaljnije objasnjeni.

2.3.1. Detektori uglova

Najkoristeniji detektori uglova su:

- Harris detektor
- Shi-Tomashi detektor
- FAST detektor

Ovi detektori uglova se baziraju na istom principu. Svaki od njih primjenjuje određenu *feature response* funkciju. Nakon toga se formira tzv. mapa jacine ugla. Pikseli takve mape imaju vrijednosti primjenjene funkcije na sliku. Nakon toga se eventualno postavlja određeni prag T , pa pikseli koji imaju manju ili vecu vrijednost od zadanog praga, poprimaju vrijednost 0 ili 1. Sada na ovakvu novu mapu, potrazimo lokalne maksimume, i ti lokalni maksimumi predstavljaju uglove slike. Sumarno, svi ovi detektori baziraju na primjeni *feature response* funkcije, postavljanju praga T i na kraju traženje lokalnih maksimuma (*non-maximum suppression*).

2.3.2. Detektori grumenova

Grumenovi (eng. *blob*) su uzorci slike koji su upečatljivi po intenzitetu, boji i teksturi i kao takvi se ističu medju svojim susjedima. Grumenovi kao znacajke su opisani sa 4 parametra:

- Pozicija (eng. *Location*)
- Velicina (eng. *Scale*)
- Orientacija (eng. *Orientation*)
- Deskriptor (eng. *Descriptor*)

Najkoristeniji detektori grumenova su:

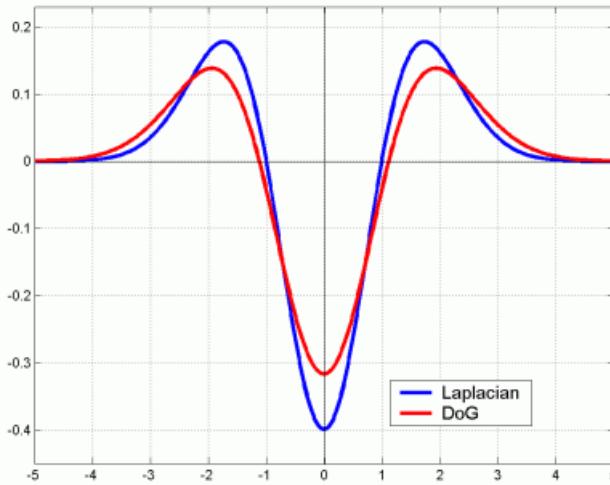
- SIFT detektor
- SURF detektor
- ORB detektor (Oriented FAST detektor)

SIFT detektor

SIFT detektor se zasniva na upotrebi *DoG*-a kao *feature-response* funkcije. To je detektor grumenova i invarijantan je na skalu i rotaciju. Za opis grumena potrebno je da poznajemo lokaliziranu ključnu tačku (eng. *keypoint localization*), velicinu i orijentaciju grumena.

Kao što smo ranije objasnili **DoG** filter se koristi kao dobra aproksimacija **LoG** filtera. Ova aproksimacija se vrši iz razloga što je **DoG** brzi za izvršavanje.

$$\text{LoG}(x, y, \sigma) \approx \text{DoG}(x, y, \sigma) = G(x, y, k\sigma) - G_\sigma(x, y, \sigma) \quad (2.2.)$$



Slika 2.3. DoG kao aproksimacija LoG-a

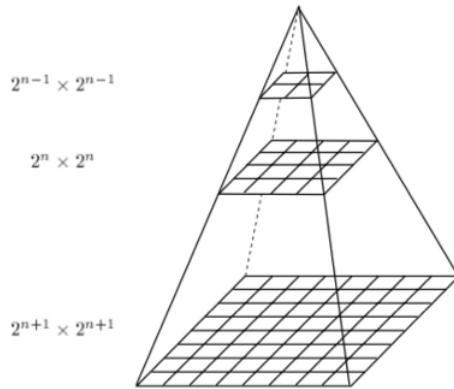
Pojam lokacije znacajke je vec od ranije poznat. To je lokacija piksela kojeg smatramo ključnom tačkom. Medjutim za grumenove se uvodi i pojma velicina. To je sastavni dio grumena. Velicina se odnosi na okolinu obuhvacenu krugom odredjenog radiusa oko ključne tačke/piksela. Ta okolina je ustvari ono sto se smatra grumenom. Grumeni su istaknute znacajke slike - te kao takve, mogu biti manjih i vecih dimenzija (tj. pomenutih velicina). SIFT detektor je na elegantan nacin rjesio ovo pitanje velicine grumena i nacina odredjivanja te velicine. Algoritam SIFT detektora ćemo podijeliti u nekoliko dijelova.

Osnovni koncept SIFT detektora jeste konvolucija slike sa LoG filterima za razlicite skale (tj. razlicite vrijednosti parametra σ). Posto koristimo aproksimaciju LoG filtera preko DoG-a, nećemo direktno vrsiti konvoluciju nego se to radi na sljedeci nacin.

1. Konstrukcija prostorne skale

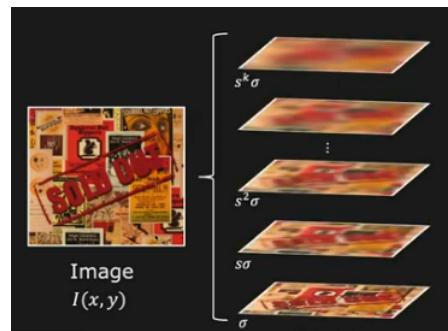
Prvi dio jeste konstrukcija prostorne skale. Prostorna skala je kolekcija slika

razlicitih rezolucija dobijena iz jedne slike. Uzmemo originalnu sliku i smanjujemo joj rezoluciju. Na taj nacin dobijamo niz slika razlicite rezolucije koje formiraju piramidu kao na slici 2.4..



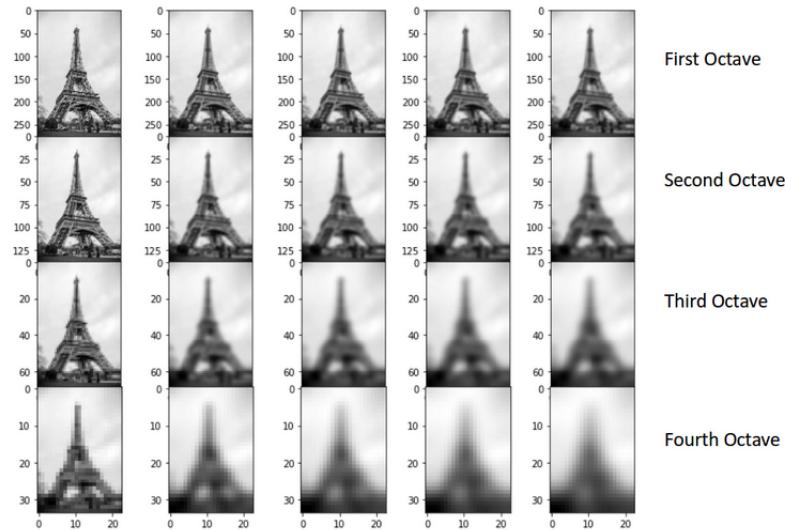
Slika 2.4. Piramida slika različite rezolucije

Svaku od tih slika je potrebno onda konvoluirati sa Gaussovim funkcijama razlicitih sirina. Na taj nacin dobijamo set slika.. Primjer konvolucije originalne slike sa Gaussovim funkcijama razlicite sirine je dat na slici 2.5..



Slika 2.5. Konvolucija slike sa Gaussovim funkcijama različitih širina

Za svaku od slika razlicitih rezolucija uradimo isto, i taj set konvoluiranih slika se naziva oktava. Najidealnije je da postoje 4 ili 5 oktava, i za svaku oktavu 5 konvoluiranih fotografija. Na ovaj nacin je formirana prostorna skala.



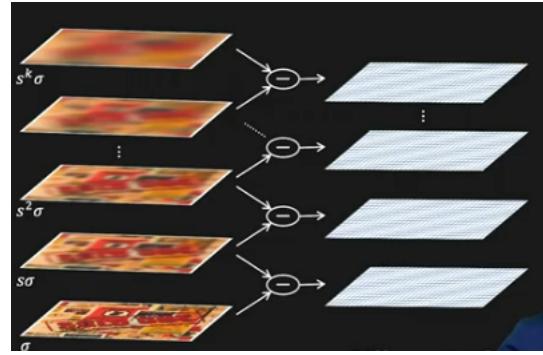
Slika 2.6. Prostorna skala

Razlog zasto se vrsi smanjivanje rezolucije jeste radi brzine izvršavanja algoritma. Kada ne bi smo smanjivali rezoluciju morali bismo koristiti Gaussove kernele sve vecih i vecih dimenzija (tj. sve vecih i vecih vrijednosti σ) i imali bi stalno isti broj piksela, i ove dvije stvari bi usporile algoritam. Gladjenje slike Gaussovim kernelom vecih dimenzija i gladjenje slike manje rezolucije sa kernelom manjih dimenzija daje isti rezultat, a ovo nam ubrzava algoritam, s obzirom da i slika i kernel su manjih dimenzija.

2. Slike konvoluirane sa LoG filterima razlicitih sirina

Vidimo da je DoG ustvari razlika dvije Gaussove funkcije razlicitih sirina. Pa posto mi vec imamo slike koje su filtrirane Gaussovim funkcijama razlicitih sirina - njihovim oduzimanjem dobijamo kao rezultat originalne slike konvoluirane sa DoG filterima razlicitih sirina.

Vidimo da smo na kraju dobili slike konvoluirane sa LoG filterima ali preko aproksimacije pomocu DoG filtera. Ova aproksimacija se koristi radi efikasnijeg izvršavanja.

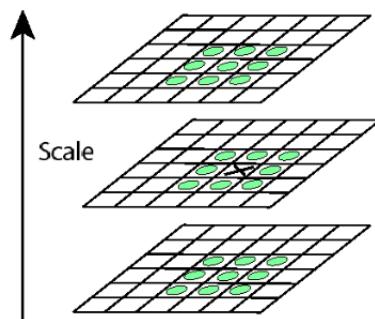


Slika 2.7. Konvolucija originala sa LoG filterom različitih širina preko DoG-a

3. Lokalizacija ključnih tacaka

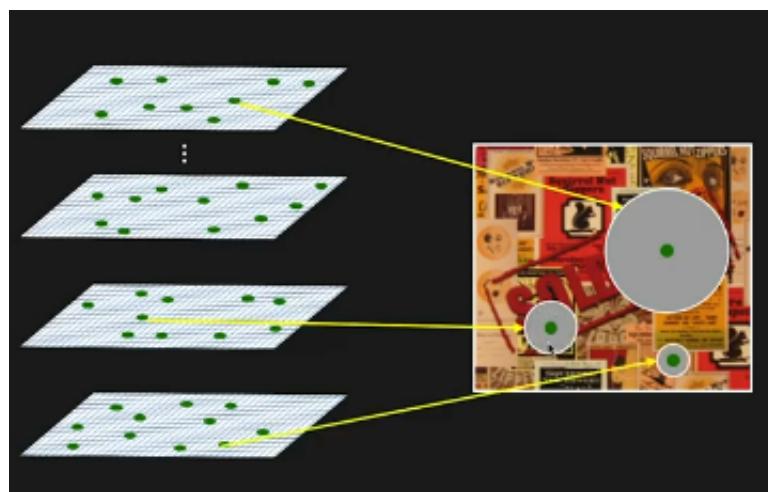
Za lokalizaciju ključnih tacaka potrebno je naci sve potencijalne ključne tačke te nakon toga izbaciti one koje to nisu.

U ovom nizu slika je sada potrebno odrediti lokalne maksimume koji predstavljaju potencijalne ključne tačke. Lokalni maksimumi se sada određuju i u domenu prostora i u domenu skale. To znači, za provjeru svakog piksela da li je potencijalni lokalni maksimum, uzmememo njegovu okolinu i takodje uzmememo okolinu više i nize skale. Svaki piksel se provjerava sa svojih 26 susjeda. 8 susjeda ima na trenutnoj slici, 9 susjeda na slici veće skale i 9 susjeda na slici manje skale. Ako je taj piksel lokalni maksimum medju svojih 26 susjeda tada se radi o SIFT znacajki.



Slika 2.8. Odredjivanje lokalnog maksimuma preko susjednih piksela različitih skala

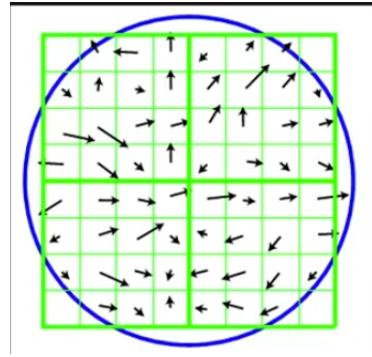
Sada imamo dosta pronadjenih ključnih tacaka, te je potrebno odrediti koje to zaista jesu, a koje nisu. Treba izbaciti one koji imaju nizak kontrast i one koje su rubovi. Za izbacivanje značajki koje se nalaze na rubovima koristi se matrica Hessijana. Također na slici 2.9. vidimo da svaka SIFT značajka ima svoju poziciju i veličinu. Velicina značajke je određena sirinom (σ) LoG filtera tj. velicina ovisi od skale na kojoj je ključna tačka značajke bila lokalni maksimum. Znaci SIFT detektor pored pozicije značajke određuje i njenu veličinu.



Slika 2.9. SIFT značajke različitih veličina

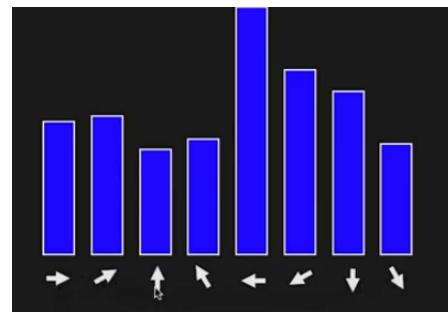
4. Dodjela orijentacije ključnim tačkama

Sada je potrebno odrediti orijentaciju SIFT značajke. Zasad znamo lokaciju SIFT značajke i njenu veličinu. Za svaki piksel SIFT značajke (broj piksela ovisi od velicine SIFT značajke) racuna se gradijent. Kako velicina SIFT značajke ovisi od vrijednosti σ , sirina uzorka susjednih piksela za tu značajku koji se uzimaju u obzir prilikom racunanja orijentacije, je velicine prozora Gausso-vog filtera kada bi sirina Gaussove funkcije bila $1.5 \cdot \sigma$.



Slika 2.10. Gradijenți u svakom pikselu SIFT značajke

Oni gradijenți koji imaju malu magnitudu se mogu iskljuciti iz razmatranja koji slijedi. Formira se histogram smjera gradijenta piksela SIFT znacajki koji se najlakse vizualizira tzv. kosaricama (eng. *bins*). Histogram se dijeli na 36 ili 8 kosarica. Ako se koristi 36 kosarica tada te kosarice predstavljaju podjelu kruga od 360 stepeni na 36 jednakih opsega uglova, i u svaku kosaricu se ubacuje ugao gradijenta svakog piksela SIFT znacajke.



Slika 2.11. Histogram uglova gradijenata uzorka SIFT značajke

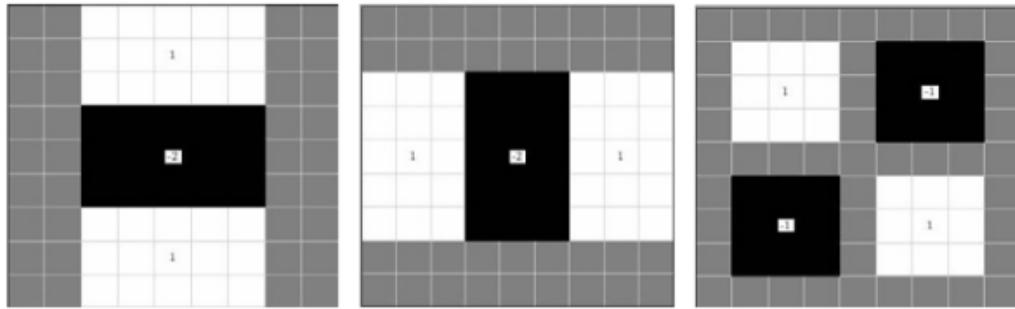
U ovisnosti od toga koja kosarica ima najvecu vrijednost ta orijentacija predstavlja orijentaciju SIFT znacajke. Sada znamo lokaciju ključne tačke, velicinu i orijentaciju SIFT znacajke.

SURF detektor

SURF detektor je predstavljen 2006. godine. SURF detektor je uveliko realiziran prema SIFT – u, i donosi niz matematičkih pojednostavljenja i aproksimacija čime smanjuje potrebno vrijeme proračuna značajki. On je dosta brzi algoritam od SIFT-a.

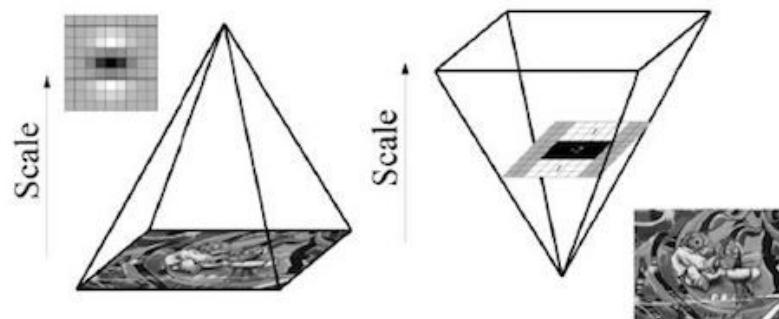
1. Prostorna skala

SIFT je kao drugi izvod Gaussove funkcije koristio *DoG* operator, te je tako postigao odlicne rezultate. Međutim SURF koristi *box* filter kao aproksimaciju drugog izvoda Gaussove funkcije tj. kao aproksimaciju *LoG* filtera. Na slici 2.12. su prikazani *box* filteri dimenzija 9x9..



Slika 2.12. Box filteri kao aproksimacija *LoG* filtera širine $\sigma = 1.2$ korišteni u SURF algoritmu

Prostorna skala obično se implementira kao piramida slike. Slike se više puta izglađuju Gaussovom funkcijom i zatim se poduzorkuju kako bi se postigao viši nivo piramide te se onda opet izglađuju. Zbog koristenja *box* filtera SURF algoritam ne mora vrsiti poduzorkovanje slike da bi konstruisao ovu piramidu. Konstrukcija piramide kod SURF algoritma se postize povećavanjem dimenzija *box* filtera, jer na taj nacin dolazi do veceg intenziteta gladjenja slike.



Slika 2.13. Kreiranje piramide smanjivanjem rezolucije slike (lijevo) i povećavanjem dimenzije filtera (desno)

2. Detekcija kljucnih tacaka znacajki bazirana na matrici Hessijana

SURF koristi matricu Hessijana kao detektor grumenova radi dobrih performansi i vremena izvršavanja. Hessijanova matrica u tački $p(x,y)$ i skali σ je:

$$H(p, \sigma) = \begin{bmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{xy}(p, \sigma) & L_{yy}(p, \sigma) \end{bmatrix} \quad (2.3.)$$

gdje je $L_{xx}(p, \sigma)$ konvolucija drugog izvoda Gaussove funkcije sirine σ po x osi i slike u tački p . Analogno vrijedi za ostale elemente matrice Hessijana.

Nakon formiranja prostorne skale i konvolucije sa box filterima, imamo potrebne elemente da formiramo matricu Hessijana za razlicite skale i izračunamo determinante. Determinanta Hessijana se koristi kao funkcija jacine grumenata (eng. *feature-response* funkcija). Formira se mapa jacine grumenova sa vrijednostima ovih determinanti za razlicite skale, i onda se traže lokalni maksimumi da bi se tako detektovali grumenovi tj. znacajke (eng. *non-maxima suppression*). Takodjer, kao i kod SIFT-a, za detekciju kljucnih tacaka potrebno je odrediti lokalni maksimum u odnosu na susjede trenutne, vise i nize skale. Posmatra se okolina 3x3x3 oko piksela, te ako je piksel lokalni maksimum oko te sredine, odabire se kao potencijalna kljucna tačka.

Zaključno, konvolucija sa *box* filterom može biti lako izračunata, te također može se paralelno računati za različite skale, što predstavlja osnovu za brzi rad algoritma. Takodjer, brža detekcija značajki je postignuta ne računanjem orijentacije ali je ostavljena opcija za dodatno računanje po zelji.

ORB detektor

ORB stoji za *Oriented FAST and Rotated BRIEF*. Njegov detektor je baziran na *FAST* detektoru. ORB detektor je *FAST* detektor sa dodatkom brze i precizne komponente orijentacije. *FAST* detektor nema osobinu orijentacije i skale. ORB detektor, na isti nacin kao i SIFT, formira prostornu skalu tj. piramidu. Uzima se originalna slika i smanjuje joj se rezolucija, pa se tako kreira piramida skale.

Kada se kreira piramida skale, tada ORB koristi FAST detektor za detekciju ključnih tacaka na svakoj slici tj. svakom levelu te piramide. Detektovanjem ključnih tacaka na svakoj slici (tj. levelu) piramide ORB locira ključne tačke na drugacijim skalama slike. Kada locira ove ključne tačke, ORB im dodjeljuje orijentaciju. Nacin dodjele orijentacije je preko tzv. *centroida intenziteta*. Centroid intenziteta pretpostavlja da je intenzitet ugla offset od njegovog centra, i taj se vektor koristi za određivanje orijentacije. Na ovaj nacin ORB detektor je i invarijantan na rotaciju.

Definisimo moment uzorka kao:

$$m_{pq} = \sum_{x,y} x^p \cdot y^q \cdot I(x, y) \quad (2.4.)$$

Parametri x i y predstavljaju koordinate svih piksela uzorka koji posmatramo oko ugla. Taj uzorak bi trebao biti kruznog oblika, sa radiusom r i sa centrom u ključnoj tački (tački koja se smatra centrom ugla). Pomocu ovog momenta možemo naci centroid kao:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (2.5.)$$

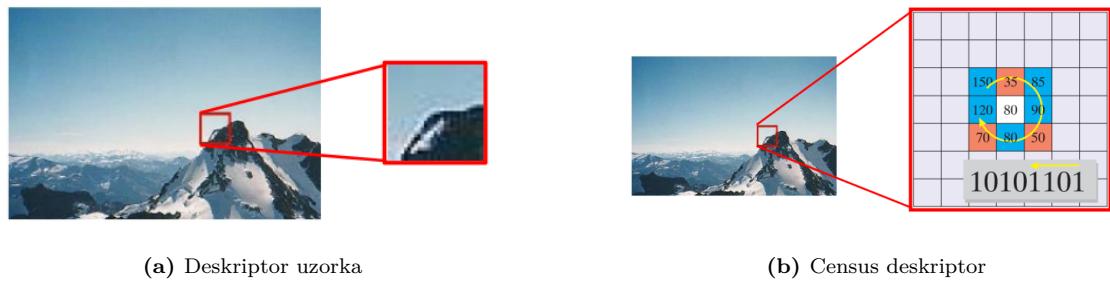
Sada možemo kreirati vektor od centra ugla do ovog centroida \vec{OC} . Orijentacija ugla je :

$$\theta_c = \text{atan2}(m_{01}, m_{10}) \quad (2.6.)$$

2.4. Deskriptor značajki

Sada kada znamo kako se znacajke nalaze, potrebno ih je i opisati da bi mogli nalaziti njihove parove u razlicitim slikama. Deskriptori znacajki opisuju znacajke. Deskriptori se mogu podijeliti na 2 tipa:

- Deskriptor uzorka - imaju cjelobrojne vrijednosti kao elemente (npr. intenziteti piksela tog uzorka)
- Census deskriptor - imaju binarne vrijednosti kao elemente (0 ili 1)



Slika 2.14. Deskriptor

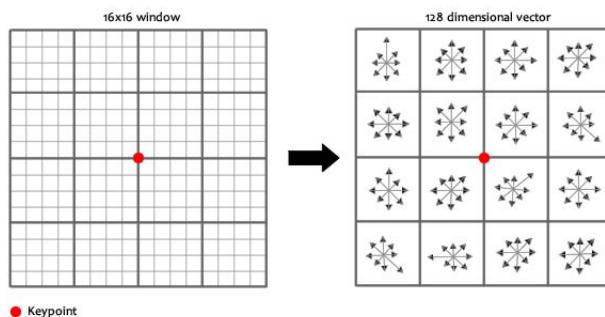
Idealni deskriptor bi trebao biti invarijantan na:

- Geometrijske promjene (rotacija, skala/velicina, perspektiva)
- Fotometrijske promjene (osvjetljenje)

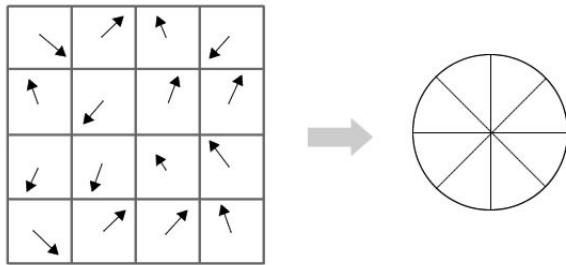
2.4.1. SIFT deskriptor

SIFT deskriptor je vektor od 128 elemenata. Dosada imamo znacajke ciju lokaciju ključne tačke, velicinu i orijentaciju znamo. Sad im je potrebno dodijeliti deskriptor.

Oko ključne tačke znacajke se posmatra uzorak dimenzija 16x16. Svakom pikselu tog uzorka potrebno je dodijeliti gradijent. Nakon sto sada imamo gradijente u svim pikselima uzorka, potrebno je taj uzorak podijeliti na manje uzorke dimenzija 4x4. Sada imamo 16 pod-uzorka unutar glavnog uzorka. Za svaki pod-uzorak formirajmo histogram koji se sastoji od 8 kosočetvornih elementa (zbog 8 smjerova). Ovaj histogram je histogram uglova gradijenta piksela uzorka. Sada svaki pod-uzorak ima svoj histogram od 8 elemenata.



Slika 2.15. SIFT deskriptor



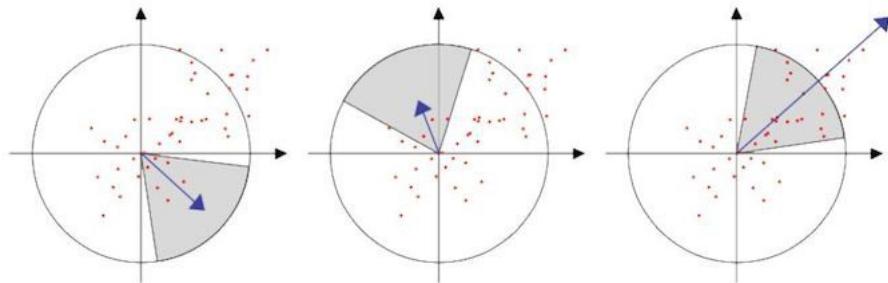
Slika 2.16. Histogram

Posto je ukupno 16 pod-uzoraka, imaćemo 16 histograma. Elemente svih histograma zapisemo kao 1D vektor. 16 histograma po 8 elemenata daje 128 elemenata. Ovo predstavlja SIFT deskriptor znacajke.

2.4.2. SURF deskriptor

SURF deskriptor je invarijantan na rotaciju. To se postize tako što se deskriptoru dodjeli orijentacija. Orijentacija se određuje na osnovu susjeda u kružnom okruženju oko ključne tačke znacajke. Nakon toga posmatramo kockasti uzorak oko ključne tačke koji će biti nas deskriptor. Taj uzorak je orijentisan u skladu sa prethodno odredjenom orijentacijom znacajke.

Da odredimo orijentaciju znacajke, odredimo odziv Haarove funkcije po x i y osi u svakom pikselu uzorka oko ključne tačke znacajke. Uzorak je u ovom slučaju kružni, sa radijusom $6s$ od ključne tačke, gdje je s skala na kojoj je detektovana ključna tačka. Velicina Haarovog filtera je $4s$. Nakon što se izračunaju odzivi Haarove funkcije po x i y osi u pikselima tih susjeda, ti odzivi se predstavljaju kao vektori u koordinatnom sistemu, cije su koordinate upravo ovi odzivi po x i y osama. Orijentacija se sada nalazi tako što se određuje dominantna orijentacija. Dominantna orijentacija određuje se izračunavanjem zbiru svih ovih odziva unutar kliznog prozora sirine $\pi/3$. Horizontalni i vertikalni odzivi unutar prozora se sumiraju. Svi sumirani horizontalni i vertikalni odzivi tada daju novi vektor sa tim koordinatama. Najduži takav vektor se odabire kao orijentacija znacajke.



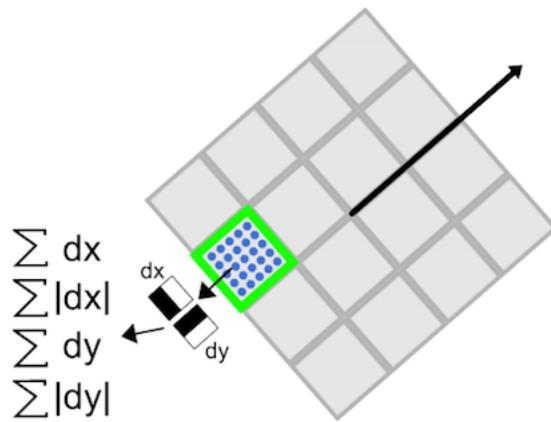
Slika 2.17. Klizni prozor na osnovu kojeg se određuje orijentacija značajke

Sada je potrebno kreirati deskriptor na osnovu ove orijentacije. Prvi korak sastoji se od izgradnje kvadratnog uzorka centriranog oko ključne tačke i orijentiranog prema orijentaciji koju smo već dobili za znacajku. Veličina ovog uzorka je $20s$. Ovaj uzorak se podijeli na 16 kvadrata (tj. napravi se mreza dimenzija 4×4), gdje je svaki kvadrat dimenzija 5×5 . Radi jednostavnosti dx ćemo zvati Haarov filter po x -osi, a dy Haarov filter po y -osi. Ovi filteri su velicine $2s$ (tj. dimenzija $2s \times 2s$).



Slika 2.18. Haarov filter za x i y osu

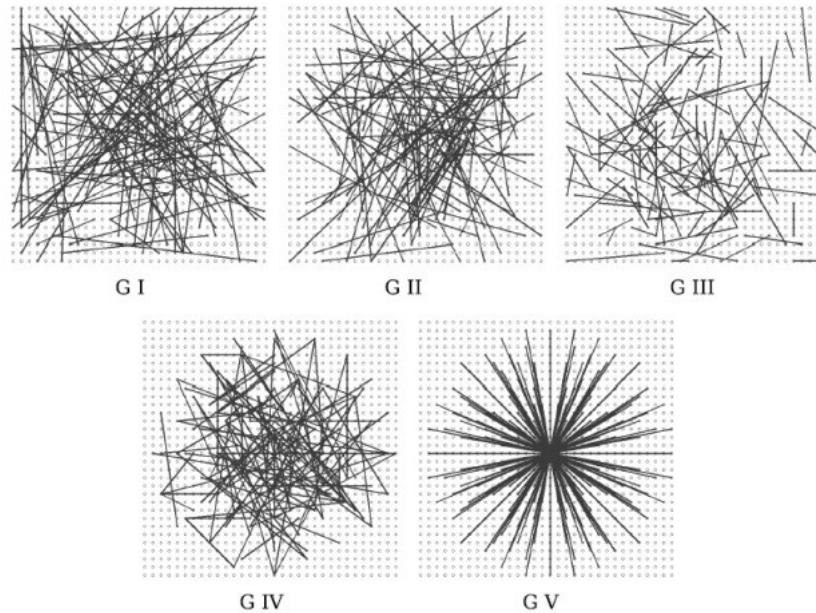
Ovi filteri se sada primjenjuju na svih 16 kvadrata naseg uzorka. Nakon ovog filtriranja, za svaki od ovih 16 kvadrata, racunamo sume $\sum dx$, $\sum |dx|$, $\sum dy$, $\sum |dy|$. Tako da za 16 kvadrata naseg uzorka, imamo po 4 vrijednosti za svaki kvadrat, što rezultira sa 64 dobijene vrijednosti. Upravo ove vrijednosti su elementi vektora tj. SURF deskriptora. Dakle, SURF deskriptor je duzine 64.



Slika 2.19. Vektor od 4 elementa formiran izračunatim vrijednostima za svaki kvadrat uzorka

2.4.3. BRIEF deskriptor

BRIEF deskriptor je binarni deskriptor. BRIEF prvo izglađi sliku sa Gaussovim kernelom, da bi bio otporan na sumove velikih frekvencija. Velicina ovog deskriptora tj. vektora je 128. Posto je ovo binarni deskriptor, njegovi elementi su 0 i 1, te oni opisuju odnose dva nasumično odabrana piksela oko ključne tačke. Tako da, ako imamo 2 piksela p i q koja su susjedna ključnoj tački, ako je piksel p svjetlij od piksela q tada element deskriptora poprima vrijednost 1, inace 0. BRIEF deskriptor predstavlja odnose nasumicno odabranih piksela u uzorku oko ključne tačke, te zbog toga je ovo veoma brz deskriptor. Nacin odabira nasumicnih piksela i njihovog poredjenja se odabire jednom, te se onda isti nacin primjenjuje na sve ostale ključne tačke, tako da se primjeni isti kriteriji za sve.



Slika 2.20. Različiti pristupi za odabir piksela koji će biti poređeni za BRIEF detektor

Originalni BRIEF deskriptor nije invarijantan na rotaciju. Zato ORB deskriptor koristi modifikaciju BRIEF deskriptora i dodaje mu osobinu invarijantnosti na rotaciju, pa se taj deskriptor naziva rBRIEF (Rotated BRIEF). Tokom faze detekcije znacajki koristeci oFAST (oriented FAST) detektor odredjena je i orijentacija znacajke, tako da ova orijentacija poslje koristi i primjenjuje na BRIEF deskriptor i dobija se rBRIEF deskriptor.

2.5. Podudaranje značajki

Podudaranje znacajki (eng. *Feature matching*) je proces određivanja istih ili sličnih znacajki na razlicitim slikama iste scene/objekta. Ako imamo dvije slike iste scene, potrebno je detektovati znacajke na obje slike te ih onda poređiti i traziti iste znacajke. Podudaranje znacajki se vrši pomoću deskriptora znacajki, jer su oni vektori brojeva, pa ih je lako uporedjivati. Dva najčešća *matcher-a* su:

- Brute-Force Matcher
- FLANN Matcher

2.5.1. BF Matcher

Brute-Force matcher je jednostavan. On uzima deskriptor jedne znacajke u prvom skupu i poredi se sa svim ostalim deskriptorima znacajki u drugom skupu koristeći neku metriku udaljenosti. Ova udaljenost predstavlja slicnost izmedju dvije znacajke. Par sa najmanjom vrijednosti ovog poredjenja se vraca kao odabrani par podudarenih znacajki.

Vidimo da po defaultu uzima se znacajka prve slike i poredi sa svim znacajkama druge slike, te se bira najbolje podudaranja. Znaci, podudaranje se vrsi u odnosu na prvu sliku. Svaka znacajka prve slike nadje svoj odgovarajuci par na drugoj slici. Kako svaka znacajka prve slike dobije neku sebi pridruzenu znacajku druge slike, to ne znaci da je taj par ispravan, tj. da su te dvije znacajke zaista iste znacajke na dvije razlicite slike. Zato je potrebno filtrirati dobijene rezultate, tako da se izbacue *outlier-i*. *Outlier-i* su lose podudarene znacajke tj. znacajke koje su podudarene ali ne predstavljaju istu znacajku. Za to postoje 2 nacina.

Prvi nacin da dobijemo sto bolje podudaranje znacajki, tj. da izbacimo *outlier-e* jeste da za svakoj znacajki prve slike pridruzimo znacajku druge slike, i da isti postupak uradimo u obratnom smjeru, pa da svakoj znacajki druge slike pridruzimo znacajku prve slike. Ako je znacajki i pridruzena znacajka j druge slike kada se podudaranje vrsilo za prvu sliku, te ako je znacajki j druge slike pridruzena znacajka i prve slike kada se podudaranje vrsilo u obratnom smjeru tj. za drugu sliku, taj par znacajki (i, j) se smatra tacnim, te je vrlo vjerovatno da su te dvije znacajke zaista iste.

Drugi nacin da dobijemo sto bolje podudaranje znacajki, tj. da izbacimo *outlier-e* jeste da svakoj znacajki prve slike nadjemo k najbližih znacajki druge slike. Ako uzmemo da je $k = 2$, za svaku znacajku prve slike bice nadjene dvije znacajke druge slike cija je metrika udaljenosti najmanja. Ako sa m oznamimo par znacajke prve slike i njemu odgovarajuce znacajke druge slike najmanje udaljenosti, a sa n oznamimo par znacajke prve slike i njemu odgovarajuce znacajke druge slike sa drugom

najmanjom udaljenosti, m se smatra dobim parom znacajki ako je zadovoljen uslov od Lowe-a koji glasi:

$$m.distance < 0.6 \cdot n.distance \quad (2.7.)$$

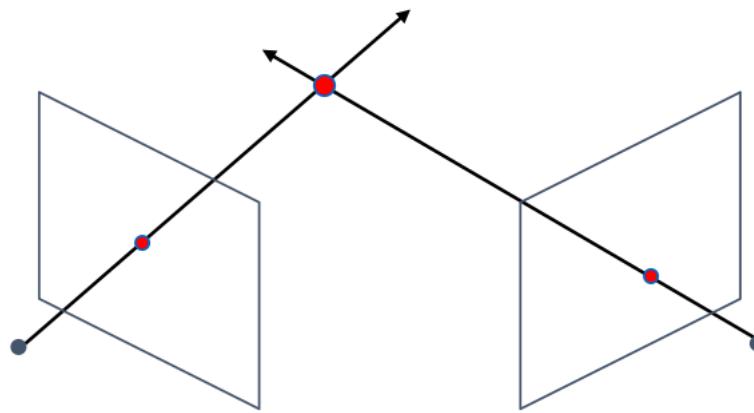
Ovaj uslov nam govori sljedeće. Ako smo nasli dva para, koja su dovoljno blizu jedna drugom, tada ne možemo tacno znati koji par znacajki je pravi, tako da takve parove odbacujemo. Ako su ta dva para dosta udaljena jedna od drugog, onda sa vrlo vjerovatno znamo da je ovaj drugi par outlier, pa zaključujemo da je prvi par ispravan tj. *inlier*.

2.5.2. FLANN Matcher

FLANN (Fast Library for Approximate Nearest Neighbors) sadrzi kolekciju algoritama optimiziranih za nalazenje najblizih susjeda. Dakle, FLANN ce za svaku znacajku prve slike odrediti njegov par na drugoj slici. Medjutim, tacnost da se nasaо odgovarajuci par znacajki je kod BF-a veca nego kod FLANN-a. FLANN je brži od BFMatcher-a, jer koristi algoritme za nalazenje parova znacajki (a ne poredjenje sa svim znacajkama kao BF), ali pronalazi samo približnog najbližeg susjeda, što je dobro podudaranje, ali nije nužno i najbolje (dok BFMatcher nalazi najboljeg). Može se igrati sa parametrima FLANN -a kako bi se povecala njegova brzina ili preciznost.

2.6. Stereo vizija

Stereo vizija se bavi problemom određivanje dubine pomocu stereo kamere. Pod dubinom misli se na z komponentu 3D tačke. Tako da, stereo vizija predstavlja 3D rekonstrukciju tacaka (tj. scene) pomocu stereo kamere. Pretpostavka je da su K, R i t za obje kamere poznati. Problem se bazira na određivanju 2D korespondentnih tacki koje su projekcija iste 3D tacke, te na osnovu toga, treba odrediti dubinu 3D tacke.



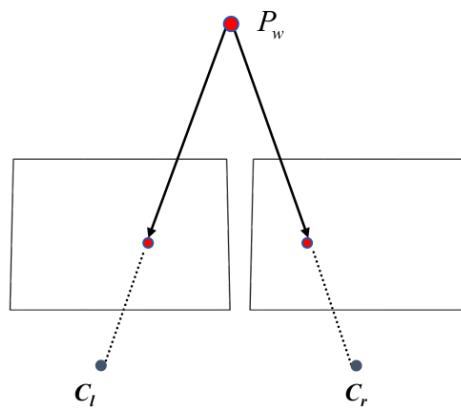
Slika 2.21. 3D tačka i njeni 2D korespondenti na paru slika stereo kamere

Postoje dva slučaja kako kamere mogu biti pozicionirane jedna u odnosu na drugu:

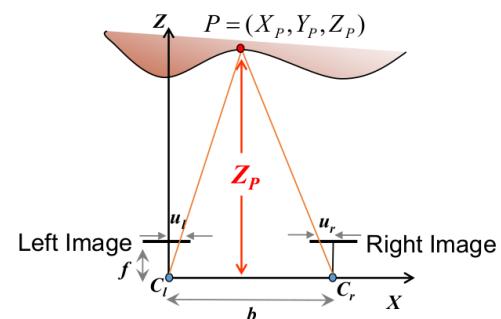
- Pojednostavljeni slučaj
- Generalni slučaj

2.6.1. Pojednostavljeni slučaj

U ovom slučaju kamere su identnične (imaju iste intrinsične parametre) i poravnate su po x-osi, kao na slici 2.22.a.



(a) Poravnate kamere



(b) Odredjivanje dubine

Slika 2.22. Stereo sistem - Pojednostavljeni slučaj

Na osnovu sličnosti trouglova vrijedi sa slike 2.22.b:

$$\frac{f}{Z_p} = \frac{u_L}{X_p} \quad (2.8.)$$

$$\frac{f}{Z_p} = \frac{-u_R}{b - X_p} \quad (2.9.)$$

Pa je:

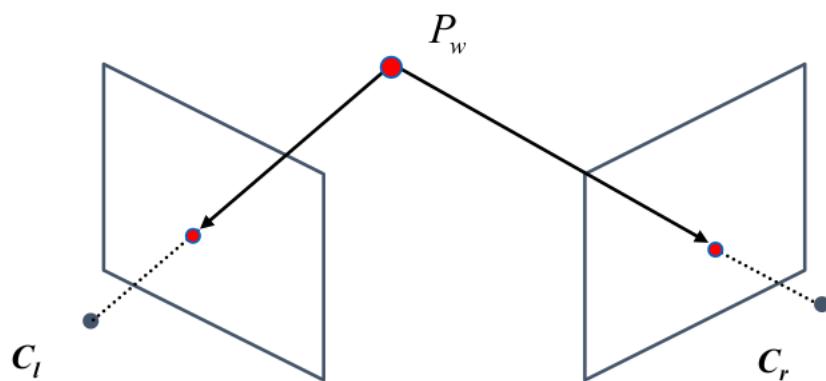
$$Z_p = \frac{bf}{u_L - u_R} = \frac{bf}{d} \quad (2.10.)$$

gdje je b udaljenost (eng. *baseline*) izmedju dvije kamere po x-osi, a d disparitet. Disparitet predstavlja razliku u pikselima po x ili y koordinati (u nasem slučaju po x koordinati) dvije tačke na slici koje predstavljaju reprojekciju iste 3D tačke.

Za sve korespondentne tačke koje predstavljaju neke 3D tačke, potrebno je na ovaj nacin odrediti z koordinatu, te se tako može rekonstruisati 3D scena na osnovu 2D tacaka u ovom pojednostavljenom slučaju.

2.6.2. Generalni slučaj

U stvarnosti ne postoje iste kamere (sa istim intrinsickim parametrima). Takodjer perfektno poravnjanje dvije kamere na horizontalnoj osi je veoma tesko. Potrebno je poznavati intrinsicke parametre obje kamere kao i ekstrinsicke parametre tj. relativnu transformaciju izmedju ove dvije kamere. Lijeva kamera predstavlja koordinatni sistem svijeta, pa tako R i t iz relativne transformacije predstavljaju kako je postavljena desna kamera u odnosu na lijevu.



Slika 2.23. Stereo vizija - Generalni slučaj

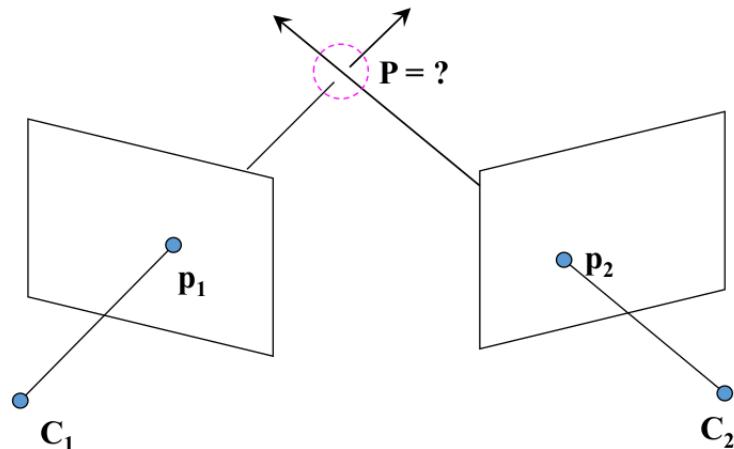
Za pojednostavljeni slučaj vidjeli smo da se 3D tacka rekonstruise pomocu mape dispariteta. To je laksi nacin za odredjivanje 3D tacke, međutim kamere moraju

biti poravnate. Za generalni slučaj može se primjeniti ovaj nacin ali kamere se prvo moraju poravnati. Također još jedan nacin rekonstrukcije za ovaj slučaj je tzv. triangulacija 3D tacke. Oba nacina će biti objasnjena u nastavku.

2.6.3. Triangulacija

Triangulacija je dio stereo vizije koja ima za cilj da odredi 3D koordinate. Pretpostavka je da su korespondentne tačke poznate.

Na slici 2.21. vidimo dvije korespondentne tacke. Presjecanjem zraka koje prolaze kroz njih od centra svake kamere, dobijamo 3D tačku. Ovo presijecanje zraka ce se aproksimirati sistemom jednacina.



Slika 2.24. Uticaj smetnji na rekonstrukciju 3D tačke

Za lijevu kameru vrijedi:

$$\lambda_1 \cdot \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = K_1 \begin{bmatrix} I & 0 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (2.11.)$$

$$\lambda_1 \cdot p_1 = M_1 \cdot P \implies p_1 \times \lambda_1 p_1 = p_1 \times M_1 \cdot P \implies 0 = p_1 \times M_1 \cdot P \quad (2.12.)$$

Za desnu kameru vrijedi:

$$\lambda_2 \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = K_2 \begin{bmatrix} R & t \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (2.13.)$$

$$\lambda_2 \cdot p_2 = M_2 \cdot P \implies p_2 \times \lambda_2 p_2 = p_2 \times M_2 \cdot P \implies 0 = p_2 \times M_2 \cdot P \quad (2.14.)$$

Na osnovu pravila vektorskog proizvoda vrijedi:

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [a_{\times}] \cdot b \quad (2.15.)$$

Sada za lijevu kameru, jednacina (88) postaje:

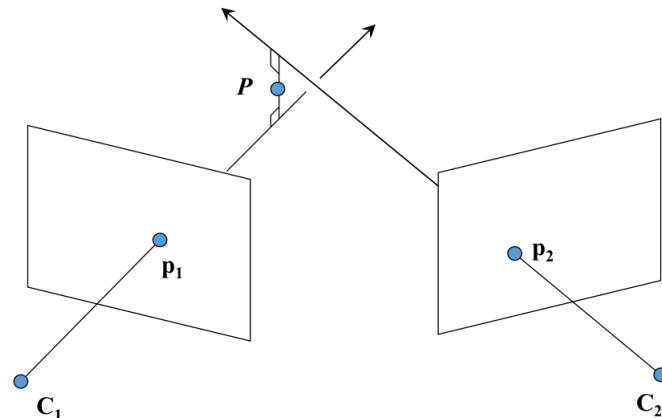
$$[p_{1\times}] \cdot M_1 \cdot P = 0 \quad (2.16.)$$

Za desnu kameru, jednacina (90) postaje:

$$[p_{2\times}] \cdot M_2 \cdot P = 0 \quad (2.17.)$$

Sada imamo ovaj sistem jednacina, cijim rjesavanjem dobijamo 3D tačku P. P se može odrediti koristeci SVD.

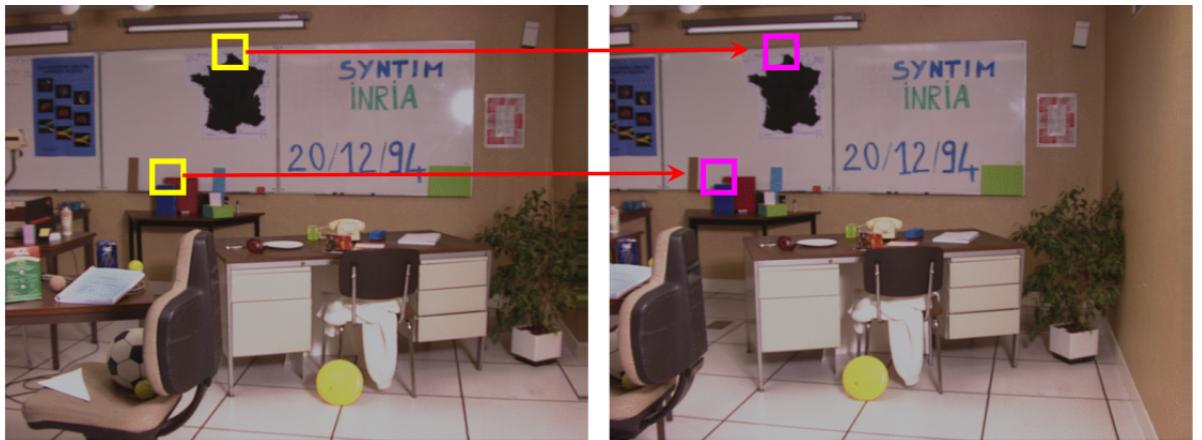
Geometrijska interpretacija ovog nacina, jeste da je tačka P dobijena kao srediste najkraceg segmenta koji povezuje dvije linije, kao na slici 2.25..



Slika 2.25. Geometrijska interpretacija određivanja 3D tačke

2.6.4. Problem korespondencije tačaka

Za datu tačku p_L na lijevoj slici, potrebno je naci njegovog korespondenta p_R na desnoj slici. To možemo uraditi tako što ćemo za svaku tačku na lijevoj slici pretrazivati sve tačke na desnoj slici, te odabratи onu koja joj najviše sliči. Ta sličnost se definise nekom metrikom udaljenosti, koristeci uzorke oko piksela.



Slika 2.26. Korespondentne značajke stereo kamere

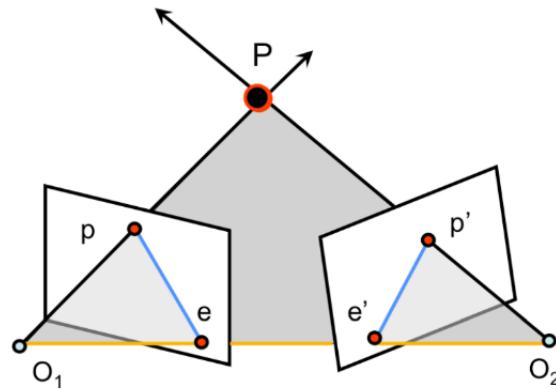
Medjutim, Ovo 2D iscrpno pretraživanje računski je jako skupo. Ima veoma mnogo poredjenja, s obzirom da za svaki piksel lijeve slike provjeravamo piksel desne slike. Da bi se ovaj proces ubrzao, traženje korespondenata se može vršiti u 1D umjesto u 2D. Ovaj postupak traženja korespondenata u 1D se bazira na epipolarnoj geometriji (eng. *epipolar geometry*).

Epipolarna geometrija

Motiv za uvodjenje epipolarne geometrije jeste smanjivanje vremena komputacije prilikom traženja korespondenata tacaka u stereo sistemu.

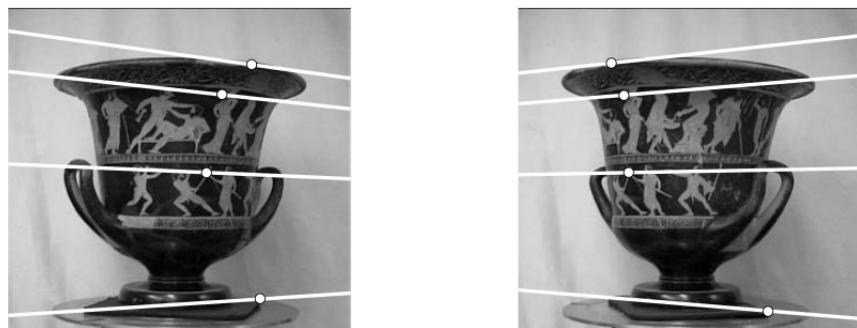
Prepostavimo da znamo tacku 2D tacku lijeve kamere \mathbf{x} i da joj zelimo naci korespondenta na desnoj kameri \mathbf{x}' . Centri kamera O_1 i O_2 , te poznata taka \mathbf{x} formiraju tzv. epipolarnu ravan π . Intersekcijom epipolarne ravni π sa slikovnim ravnima kamera, formiraju se tzv. epipolarne linije. Korespondentne 2D tacke \mathbf{x} i \mathbf{x}' upravo leže na ovim epipolarnim linijama svojih slikovnih ravni. Ovo se naziva epipolarno ogranicenje.

Upravo ovo epipolarno ogranicenje smanjuje pretragu korespondenata sa 2D pretrage na 1D pretragu. Za svaku 2D tacku lijeve kamere, njen korespondent lezi na odgovarajucoj epipolarnoj liniji desne kamere, pa se tako korespondentna tacka trazi samo duz te epipolarne linije.



Slika 2.27. Epipolarna geometrija

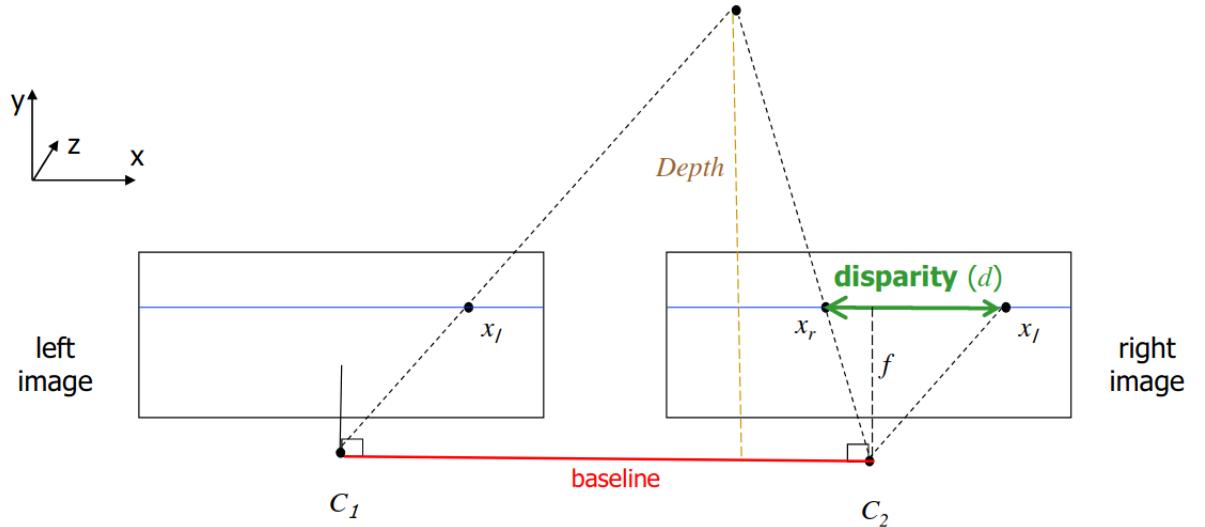
Primjer epipolarne geometrije je dat na slici 2.28..



Slika 2.28. Epipolarna geometrija - epipolarne linije stereo kamere

Stereo rektifikacija

Lijeva i desna kamera nikada nisu perfektno poravnate. Stereo rektifikacija ili stereo poravnanje rjesava ovaj problem, te poravnava ova dvije slikovne ravni. Poravnavanjem slikovnih ravni, jos brze dolazi do odredjivanja korespondenata tacaka na dvije slike.



Slika 2.29. Rektificirane slike stereo sistema

Cilj stereo rektifikacije jeste da lijeva i desna kamera treba da imaju isto R (matricu rotacije u odnosu na koordinatni sistem svijeta) i istu matricu intrinsickih parametara K . Za svaku od ove dvije kamere se izračuna matrica homografije. Ova matrica homografije će biti razlicita za ove dvije kamere, iz razloga što one nisu orijentisane isto u odnosu na koordinatni sistem svijeta, a matrica homografije sadrži i transformaciju koordinatnih sistema ovih kamera u odnosu na koordinatni sistem svijeta.

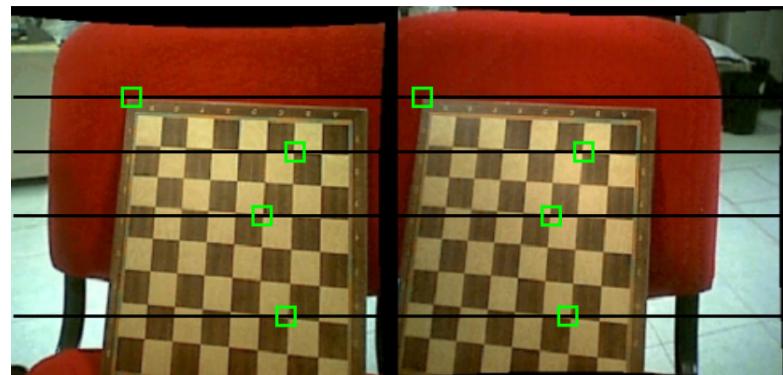
Kada su lijeva i desna slika poravnate, tada epipolarne linije odgovaraju linijama skeniranja (eng. *scanlines*). I tada traženje korespondenata se vrši samo duž tih linija. Uporedjivanje se vrši standardno, tako što se uzme uzorak određene velicine oko tačke, se se kreće epipolarne linije druge slike koja odgovara toj tački, i koristenjem neke metrike sličnosti (npr. SSD), te se trazi tačka koja najbolje odgovara onoj sa prve slike.

Na slici 2.30. su prikazane neporavante slike, na kojima epipolarne linije ne odgovaraju linijama skeniranja.



Slika 2.30. Neporavnate slike stereo sistema

Na slici 2.31. su prikazane poravnate slike, na kojima epipolarne linije odgovaraju linijama skeniranja.



Slika 2.31. Poravnate slike stereo sistema

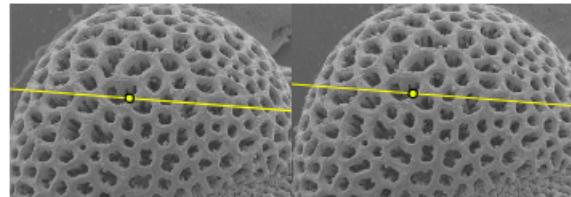
Mapa dispariteta

Nakon rektifikacije, veza izmedju koordinata korespondentnih tacaka je:

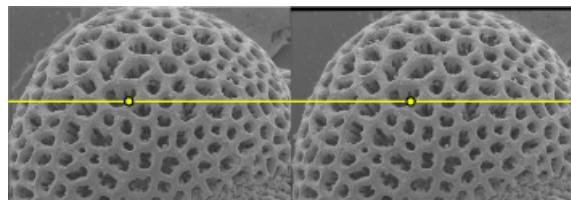
$$(x', y') = (x + d, y) \quad (2.18.)$$

gdje je d disparitet (razlika) izmedju horizontalnih koordinata ove dvije korespondentne tačke ($d = u_L - u_R$). Kao što smo imali u pojednostavljenom slučaju kada je stereo par bio automatski poravnat, tako i sada, nakon ovih nekoliko dodatnih koraka, opet racunamo mapu dispariteta. Na osnovu nje ćemo opet moci odrediti z koordinatu 3D tačke (tj. njenu dubinu) kao:

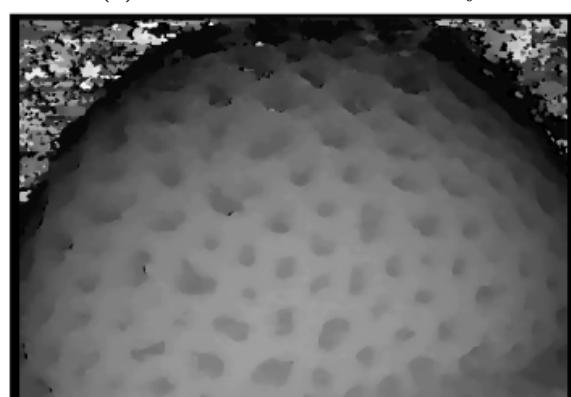
$$z = \frac{f \cdot b}{d} \quad (2.19.)$$



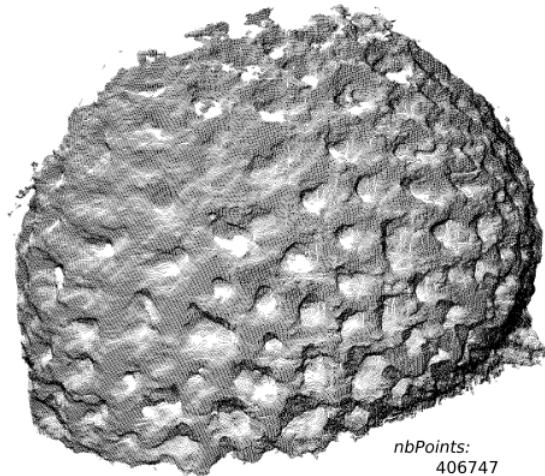
(a) Par stereo slika



(b) Par stereo slika nakon rektifikacije



(c) Mapa dispariteta



(d) Rekonstruisane 3D tačke (3D pointcloud)

Slika 2.32. Rekonstrukcija 3D tačaka na osnovu stereo kamere

2.7. Zaključak

Za rjesavanje problema vizualne odometrije moguce je koristiti monokularnu ili stereo kameru. Bilo koji sistem kamere da se koristi, potrebno je detektovati i opisati znacajke, te ih povezati. SIFT detektor se zasniva na koristenju Gaussovog kernela kako bi se aproksimiralo filtriranje sa LoG kernelom, SURF na koristenju box filtera, a ORB na FAST detektoru uglova, sa dodanom orijentacijom. SURF je brzi od SIFT-a iz razloga sto koristi box kernele umjesto Gaussove kernele. ORB je brzi od SURF-a, jer koristi FAST detektor, koji je sam po sebi brz, jer koristi kruzni uzorak oko ključne tacke, i brzo provjerava da li se radi o korneru. Sto se tice deskriptora, SIFT i SURF deskriptori rade na slicnom principu i to su cjelobrojni deskriptori. SIFT deskriptor ima 128 elemenata, dok SURF deskriptor ima 64 elementa. ORB deksriptor je ustvari BRIEF deskriptor invarijantan na rotaciju. BRIEF je binarni deskriptor. Sto se tice povezivanja znacajki, BF Matcher preispituje sve ključne tacke, dok FLANN sadrži optimizirane algoritme za povezivanje znacajki dvije slike. Tako da BF Matcher je vise vjerovatno tacan, medjutim FLANN je brzi. Tako detektovani algoritmi mogu sadrzavati *outlier*-e (pogresno povezane znacajke), pa se to moze donekle popraviti primjenom *Lowe*-ovog pravila ili dvostrukim povezivanjem.

Stereo vizija predstavlja problem određivanja z komponente 3D tacke koristenjem stereo sistema kamere. Ako se radi o horizontalno poravnatim kamerama stereo sistema dovoljno je kreirati mapu dispariteta i odatle odrediti dubine 3D tacaka. Ako kamere nisu horizontalno poravnate, 3D tacke se mogu odrediti triangulacijom ili mapom dispariteta. Sada, prije konstrukcije mape dispariteta, potrebno je poravnavati kamere stereo sistema (stereo rektifikacija) te nakon toga je moguce naci mapu dispariteta.

Poglavlje 3.

Opis rješenja

Na pocetku prethodnog poglavlja smo opisali glavne korake vizuelne odometrije. Spomenuta je i estimacija kretanja, koja zapravo predstavlja srz vizuelne odometrije. Ovo poglavlje je posveceno metodama estimacije kretanja. Bit ce spomenuta tri pristupa odredjivanja matrice transformacije, od cega su detaljno objasnjena dva pristupa i to 2D-2D i 3D-2D estimacija kretanja, o cemu ce vise rijeci biti u nastavku. Takodjer, bit ce rijeci i o *CARLA* simulatoru autonomne voznje, pomocu kojeg se prikupljaju slike. Posto za svaki pristup rjesavanja problema vizuelne odometrije, koraci detekcije i povezivanja znacajki su neophodni, ovi pristupi se onda razlikuju prema odredjivanju matrice transformacije. Tako da razlicita rjesenja ovise od razlicite estimacije matrice transformacije, tako da je ovo poglavlje posveceno koraku vizuelne odometrije koji odreduje matricu relativne transformacije.

3.1. Estimacija kretanja

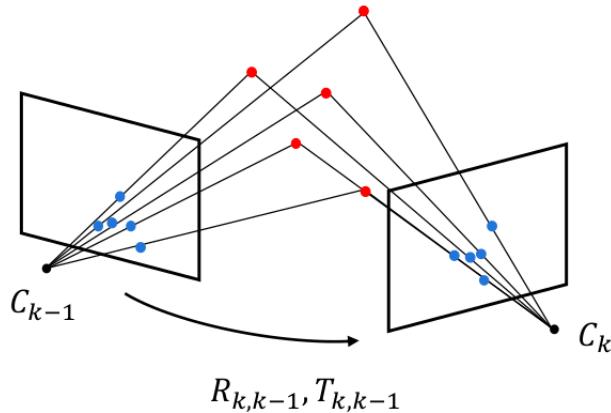
Estimacija kretanja je osnovni korak prilikom realiziranja vizuelne odometrije. U ovom koraku, odreduje se relativno kretanje kamere izmedju dva diskretna vremenska trenutka u kojima je kamera slikala scenu. Kao rezultat toga dobivamo matricu koja sadrzi transformaciju koordinata izmedju dva koordinatna sistema, i to koordinatnog sistema kamere u diskretnom vremenskom trenutku $k-1$ i koordinatnog sistema kamere u diskretnom vremenskom trenutku k . Preciznije, u koraku estimacije kretanja, kretanje kamere između trenutne slike i prethodne slike se racuna.

Spajanjem svih ovih relativnih transformacija , citava trajektorija kamere i agenta se može odrediti. Ovo poglavlje objasnjava kako relativna transformacija $T_{k,k-1}$ izmedju dvije slike I_{k-1} i I_k , uslikane u vremenskim trenucima $k-1$ i k , može biti izračunata od dva seta korespondentnih znacajki f_{k-1} i f_k , dobijenih u diskretnim vremenskim trenucima $k-1$ i k . Postoje tri nacina na koja mozemo odrediti matricu transformacije. Ona se razlikuju po tome kako predstavljamo korespondentne znacajke. Tako da, ovisno od toga da li su korespondentne znacajke predstavljene u 2D ili 3D prostoru, postoje tri razlicita nacina odredjivanja matrice transformacije T_k .

Ta 3 nacina su:

- **2D-2D**

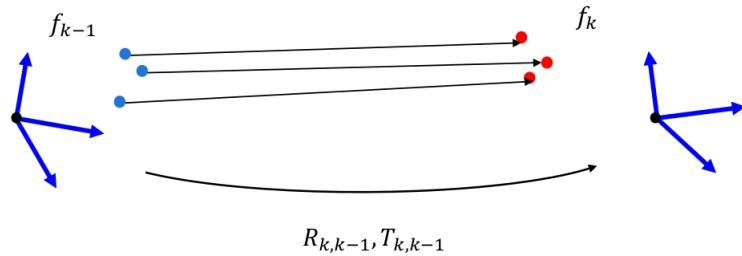
U ovom slučaju oba seta znacajki f_{k-1} i f_k su predstavljeni u 2D koordinatama slike



Slika 3.1. 2D-2D estimacija kretanja

- **3D-3D**

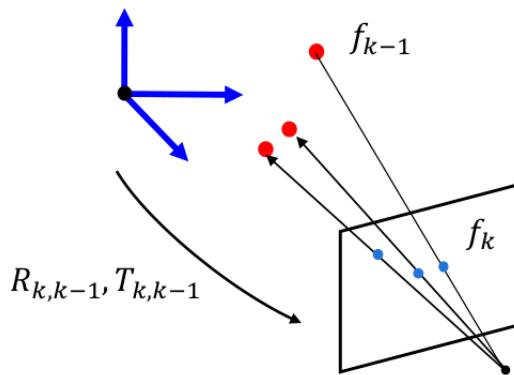
U ovom slučaju oba seta znacajki f_{k-1} i f_k su predstavljeni u 3D. Za ovaj pristup, neophodno je, za oba trenutka $k-1$ i k tj. za oba frejma, izvršiti triangulaciju 3D tacaka preko 2D koordinata znacajki, a to se može uraditi ako koristimo stereo kameru.



Slika 3.2. 3D-3D estimacija kretanja

- **3D-2D**

U ovom slučaju f_{k-1} su predstavljeni u 3D, dok su f_k njihovi korespondentne 2D reprojekcije na frejmu I_k . Za monokularni slučaj, bila bi potrebna dva uzastopna frejma I_{k-2} i I_{k-1} da se ove 3D tačke triangulisu, te da se onda povezu sa svojim korespondentnim 2D projekcijama koje su ustvari 2D znacajke na slici I_k . Vidimo da za ovaj monokularni slučaj, potrebna su nam 3 frejma u ovom slučaju.



Slika 3.3. 3D-2D estimacija kretanja

3.2. 2D-2D estimacija kretanja

Geometrijski odnos izmedju dvije slike I_{k-1} i I_k kalibriranih kamara se može opisati pomocu tzv. *esencijalne matrice* (eng. *essential matrix*). Njenom dekompozicijom dobijamo relativnu transformaciju izmedju dva položaja kamere. Treba napomenuti da je vektor translacije te transformacije jedinicni vektor. Dobijanje esencijalne matrice se zasniva na epipolarnoj geometriji.

Pretpostavimo da se 3D tacka \mathbf{P} projicirala na lijevu i desnu slikovnu ravan, u 2D tacke \mathbf{p}_L i \mathbf{p}_R respektivno. Pretpostavimo da znamo matricu kamere \mathbf{K} . Radi jednostavnosti, mozemo preci u normalizirane koordinate. Normalizirane koordinate ovih 2D tacaka dobijamo kao $\mathbf{x}_L = \mathbf{K}^{-1} \cdot \mathbf{p}_L$ i $\mathbf{x}_R = \mathbf{K}^{-1} \cdot \mathbf{p}_R$. Oznacimo sa \mathbf{t} vektor translacije. Ova tri vektora leze u epipolarnoj ravni π . Definisimo vektor \mathbf{n} okomit na epipolarnu ravan kao:

$$\mathbf{n} = \mathbf{t} \times \mathbf{x}_L \quad (3.1.)$$

Vrijedi da je:

$$\mathbf{x}_L \cdot \mathbf{n} = \mathbf{x}_L \cdot (\mathbf{t} \times \mathbf{x}_L) = 0 \quad (3.2.)$$

Ako ovo zapisemo u matricnom obliku, vrijedi:

$$\mathbf{x}_L^T \cdot [\mathbf{T}_x] \cdot \mathbf{x}_L = 0 \quad (3.3.)$$

Na osnovu relativne transformacije izmedju ova dva frejma, vrijedi:

$$\mathbf{x}_L = \mathbf{R} \cdot \mathbf{x}_R + \mathbf{t} \quad (3.4.)$$

Pa imamo:

$$\mathbf{x}_L^T \cdot ([\mathbf{T}_x] \cdot \mathbf{R} \cdot \mathbf{x}_R + [\mathbf{T}_x] \cdot \mathbf{t}) = 0 \quad (3.5.)$$

Posto vrijedi da je:

$$[\mathbf{T}_x] \cdot \mathbf{t} = \mathbf{t} \times \mathbf{t} = 0 \quad (3.6.)$$

Sada imamo:

$$\mathbf{x}_L^T \cdot [\mathbf{T}_x] \cdot \mathbf{R} \cdot \mathbf{x}_R = \mathbf{x}_L^T \cdot \mathbf{E} \cdot \mathbf{x}_R = 0 \quad (3.7.)$$

Odavde dobijamo esencijalnu matricu \mathbf{E} :

$$\mathbf{E} = [\mathbf{T}_x] \cdot \mathbf{R} \quad (3.8.)$$

Esencijalna matrica sadrzi u sebi relativnu transformaciju izmedju ova dva frejma. Tako da je za nalazenje transformacije izmedju dva frejma potrebno odrediti esencijalnu matricu, te iz nje izvaditi matricu rotacije i vektor translacije.

Kako matrica rotacije i vektor translacije imaju po 3 stepena slobode, esencijalna matrica, zbog toga sto vektor translacije je jedinicni vektor, ima 5 stepeni slobode umjesto 6. Posto esencijalna matrica ima 5 stepeni slobode to indicira da nam je potrebno najmanje 5 korespondentnih tacaka da odredimo \mathbf{E} . Medjutim cesce se koristi pristup od 8 korespondentnih tacaka da se odredi \mathbf{E} , s obzirom da je on jednostavniji, a imamo mnogo korespondenata.

Posmatrajmo dvije korespondentne tačke, sa svojim normaliziranim koordinatama $\mathbf{x}_1 = [u_1, v_1, 1]$ i $\mathbf{x}_2 = [u_2, v_2, 1]$. Prema jednacini (3.7) vrijedi:

$$\begin{pmatrix} u_1, v_1, 1 \end{pmatrix} \begin{pmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = 0 \quad (3.9.)$$

Ako matricu \mathbf{E} zapisemo u formi vektora kao:

$$\mathbf{e} = \begin{bmatrix} e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9 \end{bmatrix}^T \quad (3.10.)$$

tada jednacina (112) postaje:

$$\begin{bmatrix} u_2u_1, u_2v_1, u_2, v_2u_1, v_2v_1, v_2, u_1, v_1, 1 \end{bmatrix} \cdot \mathbf{e} = 0 \quad (3.11.)$$

Analogno zapisemo jednacinu epipolarnog ogranicenja za ostale korespondente tacaka, te na taj nacin formiramo set jednacina koje je potrebno rjesiti. Taj set jednacina se matricno može zapisati kao:

$$\mathbf{M} \cdot \mathbf{E} = 0 \quad (3.12.)$$

gdje je \mathbf{M} matrica poznatih elemenata. Ovaj sistem se može rjesiti primjenom *SVD*-a na matricu \mathbf{M} . Kada odredimo matricu \mathbf{E} , iz nje trebamo odrediti R i t . Treba naponemuti da je vektor trasnslacije jedinicni vektor, tako da ga je potrebno skalirati prije nego sto ovu matricu transformacije T_k pomnozimo sa matricom C_{k-1} . Moguce je izračunati relativnu skalu, na osnovu ove dvije slike, kojom ce biti reskaliran vektor translacije. Jedan nacin jeste triangulacijom 3D tacaka. Potrebna su nam dva para 3D tacaka X_{k-1} i X_k , sto znaci da su nam potrebna dva uzastopna para slika (npr.

I_{k-2} i I_{k-1} , te I_{k-1} i I_k). Relativna skala se na osnovu ovih oblaka 3D tacaka racuna kao:

$$r = \frac{\|X_{k-1,i} - X_{k-1,j}\|}{\|X_{k,i} - X_{k,j}\|} \quad (3.13.)$$

Ovaj odnos r se izračuna za mnogo parova 3D tacaka, te se onda izračuna srednja vrijednost ovih odnosa, i ta srednja vrijednost predstavlja relativnu skalu. Sada reskaliramo vektor t , te formiramo matricu C_k . Umjesto relativne skale, može se racunati i absolutna skala. Na osnovu prethodnog znanja o samoj sceni može se izračunati ova skala (npr. visina kamere, velicina objekta, brzina vozila, bazna linija stereo kamere, ...) ili koristenjem nekog senzora (npr. IMU ili GPS, koji mogu dati podatke npr. o položaju kamere tj. njene koordinate za svaku sliku, brzini vozila za svaku sliku, ...).

Koraci koje je potrebno ispostovati za ovaj pristup rjesavanja problema vizelne odometrije dati su ispod, i njih je potrebno ponavljati:

- Uzimanje frejma I_k
- Detektovanje znacajki na dva uzatopna frejma I_{k-1} i I_k
- Povezivanje znacajki frejmova I_{k-1} i I_k
- Racunanje esencijalne matrice E_k
- Dekompozicija esencijalne matrice E_k na R_k i t_k
- Racunanje skale
- Reskaliranje vektora translacije t_k
- Formiranje matrice transformacije T_k
- Formiranje matrice $C_k = C_{k-1} \cdot T_k$

3.3. 3D-2D estimacija kretanja

Ovaj nacin estimacije matrice transformacije T_k podrazumijeva da imamo set 3D tacaka X_{k-1} i njihove 2D korespondente reprojicirane slikovnoj ravni lijeve kamere

stereo para slika u sljedecem trenutku, p_k . Ovaj nacin može se koristiti i za monokularni i stereo sistem kamere. Kod stereo sistema, potrebna su nam dva uzastopna para slika. Prvi stereo par sluzi za odredjivanje 3D tacaka X_{k-1} . Ove 3D tačke mogu se dobiti triangulacijom ili koristenjem mapa dispariteta. Na drugom stereo paru slika, potrebno je na lijevoj slici detektovati 2D znacajke p_k . Na osnovu ova dva seta tacaka, potrebno je naci korespondente izmedju 3D i 2D tacaka.

Algoritam na kojem se zasniva ovaj metod estimacije je *PnP* (eng. *Perspective from n points*) algoritam. *PnP* odredjuje matricu transformacije kada su poznate 3D tačke i njihove 2D reprojekcije. Minimalno rjesenje zahtjeva tri 3D-2D korespondenta (*P3P*). *P3P* je standardna metoda za estimaciju kretanja u prisustvu outliers-a. Po red *P3P* rjesenja, jednostavnije rjesenje *PnP* problema za $n \geq 6$ korespondenata, je DLT (eng. *direct linear transformation*) algoritam.

Posmatrajmo 3D tačku u homogenim koordinatama $P=[X, Y, Z, 1]^T$ i da se ona projicirala u 2D tačku slikovne ravni kamere $x_1 = (u_1, v_1, 1)$ (izrazenu u normaliziranim homogenim koordinatama). Ovu projekciju možemo zapisati koristeci jednacina projekcije, pa vrijedi:

$$s \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ a_9 & a_{10} & a_{11} & a_{12} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = A \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (3.14.)$$

Predstavimo A kao:

$$A = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ a_9 & a_{10} & a_{11} & a_{12} \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{pmatrix} = \mathbf{K} \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix} \quad (3.15.)$$

Sada imamo dvije jednacine:

$$\mathbf{a}_1^T \mathbf{P} - \mathbf{a}_3^T \mathbf{P} u_1 = 0 \quad (3.16.)$$

$$\mathbf{a}_2^T \mathbf{P} - \mathbf{a}_3^T \mathbf{P} v_1 = 0 \quad (3.17.)$$

Istu proceduru ponovimo za ostalih $n - 1$ tacaka, te dobijemo sistem jednacina koji je potrebno rjesiti.

Matricni zapis je:

$$\mathbf{M} \cdot \mathbf{A} = 0 \quad (3.18.)$$

gdje je \mathbf{M} matrica poznatih elemenata, dok \mathbf{A} matrica koju treba odrediti. Ovaj sistem se može rjesiti primjenom *SVD*-a na matricu \mathbf{M} . Kada odredimo matricu \mathbf{A} , koja je ustvari matrica projekcije, iz nje možemo odrediti transformaciju (\mathbf{R} i \mathbf{t}) i matricu kalibracije \mathbf{K} .

Koraci koje je potrebno ispostovati za ovaj pristup rjesavanja problema vizelne odometrije (koristeci stereo kameru) dati su ispod, i njih je potrebno ponavljati:

- Uzimanje stereo slika I_{k-1} i I_k
- Odrediti mapu dispariteta i mapu dubine za stereo par I_{k-1}
- Detekcija 2D znacajki lijeve kamere iz para slika $I_{L,k-1}$ i $I_{L,k}$
- Povezivanje znacajki frejmova $I_{L,k-1}$ i $I_{L,k}$
- Za svaku tacku frejma $I_{L,k-1}$ odrediti 3D tacku cija je ona reprojekcija
- Formiramo set 3D tacaka frejma I_{k-1} i set 2D tacaka frejma I_k
- Odredjivanje R_k i t_k pomocu *PnP* algoritma
- Formiranje matrice transformacije T_k
- Formiranje matrice $C_k = C_{k-1} \cdot T_k$

3.4. CARLA simulator

CARLA je open-source simulator autonomne voznje. CARLA je zasnovana na Unreal Engine-u za pokretanje simulacije i koristi OpenDRIVE standard za definiranje cesta i urbanih postavki. Kontrola simulacije je omogućena putem API-ja kojim se

rukaje u Pythonu i C++.

CARLA simulator se sastoje od skalabilne arhitekture klijent-server. Server je odgovoran za sve što se odnosi na samu simulaciju: iscrtavanje senzora, računanje fizike, ažuriranje stanja svijeta i njegovih aktera i još mnogo toga.

Klijentska strana sastoje se od klijentskih modula koji kontrolisu logiku aktera na sceni i postavljaju uslove svijeta. To se postiže korištenjem CARLA API-ja (u Pythonu ili C ++).

3.4.1. Klijent

Klijent (eng. client) i svijet dva su osnovna dijela CARLA-e, neophodna za rad simulacije i njenih aktera.

Klijenti su jedan od glavnih elemenata u CARLA arhitekturi. To je modul koji korisnik koristi da traži informacije ili promjene u simulaciji. Povezuju se sa serverom, preuzimaju informacije i mijenjaju naredbe. To se radi putem skripti. Klijent se identificira i povezuje sa svijetom kako bi zatim operirao simulacijom. Klijent radi s IP -om i određenim portom. Sa serverom komunicira putem terminala.

Glavna svrha klijentskog objekta je dobiti ili promijeniti svijet i primijeniti naredbe (eng. commands). Naredbe su prilagodbe nekih od najčešćih metoda CARLA koje se mogu primijeniti u skupinama. Na primjer, *command.SetAutopilot* ekvivalentna je *Vehicle.setAutopilot()*, omogućava autopilot za vozilo. Ovo postaje izuzetno korisno za metode koje se obično primjenjuju na čak stotine elemenata. Postoje i naprednije funkcije koje klijent obezbjedjuje, a ove pobrojane su osnovne.

3.4.2. Svijet

Svijet (eng. world) je objekt koji predstavlja simulaciju. To je glavni vladar simulacije. Klijent bi trebao preuzeti njegovu instancu. Sadrži glavne metode za

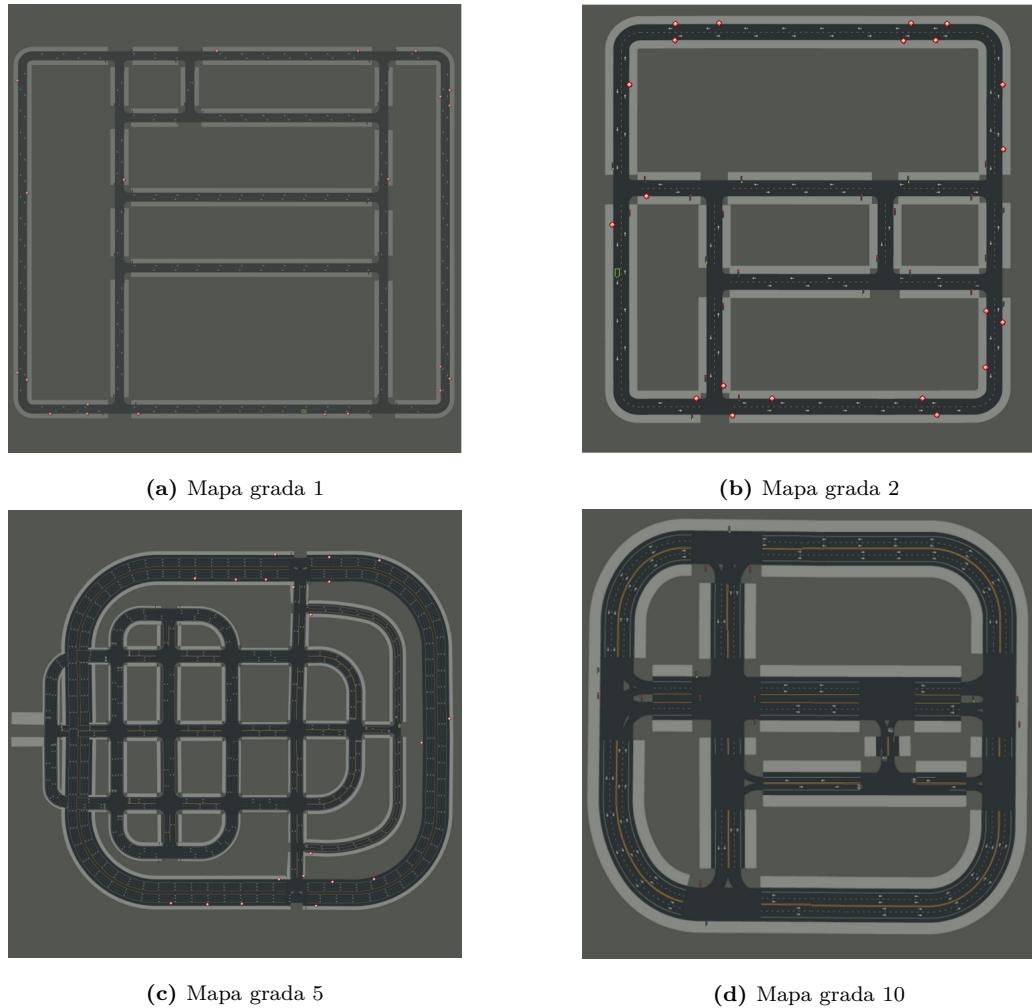
generiranje aktera, promjenu vremena, dobijanje trenutnog stanja svijeta itd. Postoji samo jedan svijet po simulaciji. Bit će uništena i zamijenjena novom kada se karta promijeni. Većini informacija i općim postavkama može se pristupiti iz ove klase. Kroz svijet se može pristupiti sljedecim elementima:

- Akterima (eng. actors)
- Biblioteci nacrta (eng. blueprint library)
- Mapi (eng. map)
- Postavkama svijeta (eng. world settings)
- Snimcima (eng. snapshot)
- Postavkama vremena i osvjetljenja (eng. weather and light manager)

3.4.3. Mapa

Mapa je objekat koji predstavlja simulacijski svijet, uglavnom grad. U CARLA simulatoru je dostupno 8 mapa, a sve koriste OpenDRIVE 1.4 standard za opisivanje puteva.

Putevima, trakama i raskrsnicama upravlja PythonAPI kojem se pristupa sa klijentske strane. Koriste se zajedno sa znacajnim tačkama puta, s ciljem osiguravanja vozila navigacijskom stazom.



Slika 3.4. Mape nekih gradova CARLA simulatora



Slika 3.5. Grad u CARLA simulatoru

3.4.4. Biblioteka nacrt

Ovi nacrti omogućuju korisniku da veoma jednostavno uključi nove aktere u simulaciju. To su već napravljeni modeli s animacijama i nizom atributa. Neki atributi se mogu mijenjati, neki ne.

Nacrti imaju svoj ID, preko kojih je moguce naci taj nacrt u biblioteci, sto nam kasnije omogucava kreiranje tog aktera. Takodjer, moguce je preko uzorka naci grupu nacrta, te odabrati neki po zelji.

3.4.5. Akteri

Akter je sve što igra ulogu u simulaciji. Pod aktere spadaju:

- Vozila (eng. vehicles)
- Senzori (eng. sensors)
- Pjesaci (eng. walkers)
- Gledalac (eng. spectator)
- Saobracajni znakovi i semafori (eng. traffic signs and traffic lights)

Aktere je moguce kreirati preko nacrt. Nacrti su fakticki urnek za aktera. Pored odabranog nacrta, potrebno je jos odabrati mjesto gdje se akter stvoriti. I na taj nacin se efikasno kreira akter.

Nakon sto je kreiran akter, postoje razne metode za upravljanje tim akterom i dobijanje zeljenih podataka o njemu. Kada nam akter vise nije potreban, moguce ga je unistiti.

Treba napomeuti da kada simulacija započne, saobracajni znakovi i semafori automatski se generiraju pomoću informacija u OpenDRIVE datoteci. Oni nisu dio biblioteke nacrt, tj. ne postoje urneci za njih, pa se stoga ne mogu ni stvoriti. Medjutim njima je moguce pristupiti i na taj nacin utjecati na njih.



Slika 3.6. Grad u CARLA simulatoru sa dodanim akterima

3.4.6. Skripte u CARLA Simulatoru

U *PythonAPI/examples* folderu postoje vec kreirane skripte za pokretanje koje rade odredjeni posao. Pomocu njih korisnik se može bolje upoznati sa CARLA okruzenjem, pokretajuci razlicite skripte. Pokretanjem skripte *manualControl.py* može se rucno upravljati vozilom koristeci tastaturu.



Slika 3.7. Manuelno upravljanje vozilom

Ovo je bio kratki uvod u CARLA Simulator. Za sve ostale funkcionalnosti preporučuje se dokumentacija CARLA simulatora, u kojoj imaju navedene sve mogućnosti, kao što su dokumentacija za PythonAPI u kojoj se nalaze sve klase u CARLA simulatoru i njihove metode, opisana biblioteka nacrta, osnovni koncepti koristenja itd.

3.5. Zaključak

Glavni dio vizualne odometrije jeste odrediti matricu relativne transformacije. U ovisnosti od dimenzije u kojoj imamo korespondentne znacajke, postoje tri načina određivanja matrice relativne transformacije. U ovom poglavlju su opisana 2, i to koristenjem 2D-2D i 3D-2D korespondencije znacajki.

Za 2D-2D korespondenciju znacajki, koristena je monokularna kamera. Potrebno je na dva uzastopna frejma detektovati i povezati znacajke, te odrediti esencijalnu matricu. Esencijalna matrica se bazira na epipolarnoj geometriji, te se njenom dekompozicijom dobiva relativna matrica rotacije i vektor translacije. Vektor translacije je jedinicni vektor, te ga je potrebno skalirati računajući relativnu skalu (triangulacija 3D tacaka) ili apsolutnu skalu (poznavanje nekih stvarnih vrijednosti scene).

Za 3D-2D korespondenciju znacajki, koristena je stereo kamera. Potrebno je za dva uzastopna para frejmova, na lijevim kamerama stereo sistema detektovati i povezati znacajke. Nakon toga, za prvi stereo par, formira se mapa disapeariteta i na taj način se rekonstruisu 3D tacke. Posto imamo povezane znacajke u 2D, za rekonstruisane 3D znacajke treba još odrediti njihove 2D parove. Nakon toga se primjenjuje PnP algoritam za određivanje relativne matrice transformacije.

CARLA simulator je simulator autonomne vožnje. Koristen je za dobijanje slika sa sistema kamere. Postavljene su dvije RGB kamere na udaljenosti od 0.54[m] po horizontalnoj y osi. Za kretanje vozila, koristen je autopilot.

Poglavlje 4.

Rezultati

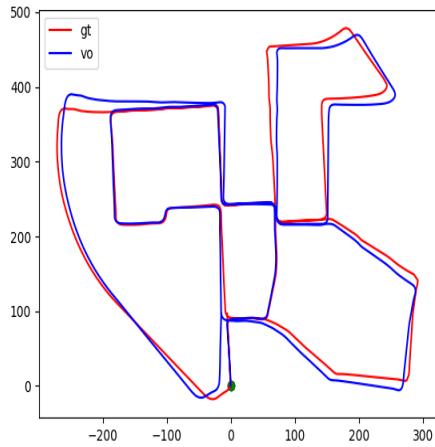
U ovom poglavlju bit će prikazani rezultati tj. trajektorije koje su dobijene na osnovu 2D-2D i 3D-2D estimacije kretanja. Također, izvršena je analiza kvaliteta vizualne odometrije pomoću MSE metrike, kao i analiza performansi s obzirom na korištene algoritme (za svaku bitnu etapu jedne iteracije vizualne odometrije mjeri se vrijeme izvršavanja, kao i ukupno vrijeme). Kao izvori podataka koristen je *KITTI* set podataka, kao i *CARLA* simulator. Pored *CARLA* simulatora odlucio sam se koristiti još i *KITTI* set jer je to veoma popularan set podataka za razne aplikacije kompjuterske vizije.

Korištena je sljedeća konfiguracija:

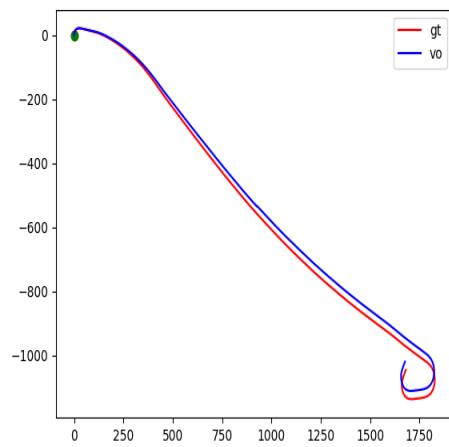
- AMD Ryzen 7 4800H @ 2.9GHz
- RAM 16 GB
- Nvidia GeForce GTX 1650 4GB
- Ubuntu 20.04.5

4.1. Trajektorije

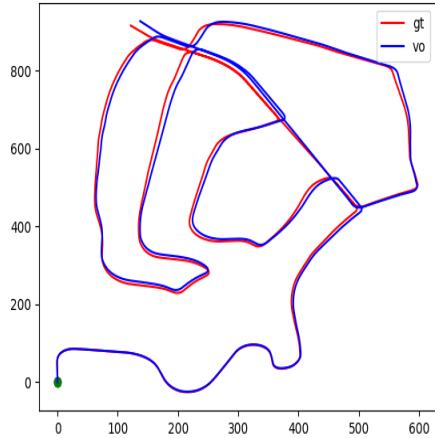
Rezultati dobijeni na osnovu *KITTI* seta podataka te 2D-2D estimacije kretanja i 3D-2D estimacije kretanja su dati na slikama 4.3. i 4.6. respektivno, a na osnovu *CARLA* simulatora i 2D-2D estimacije kretanja na slici 4.4..



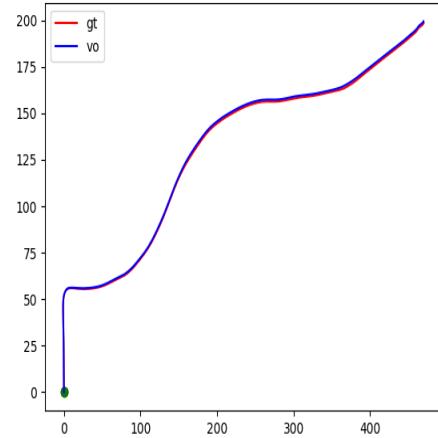
(a) Set 0



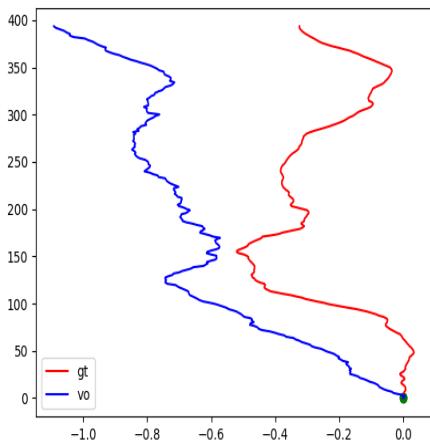
(b) Set 1



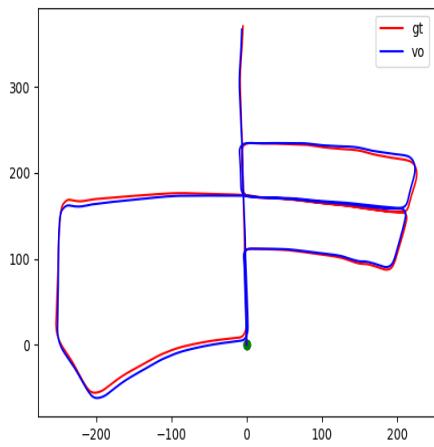
(c) Set 2



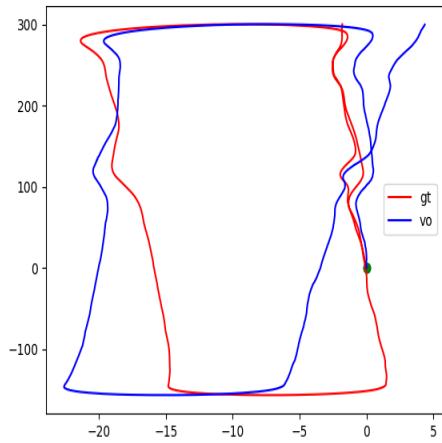
(d) Set 3



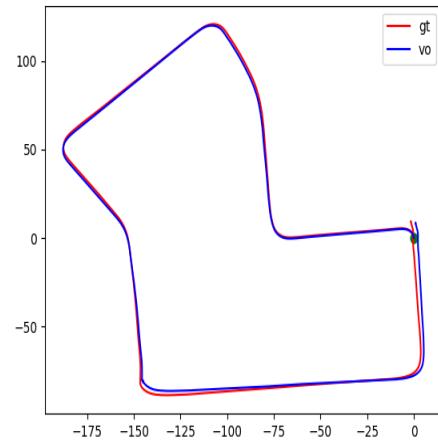
(e) Set 4



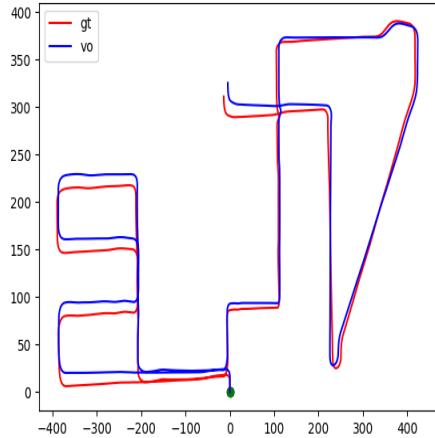
(f) Set 5



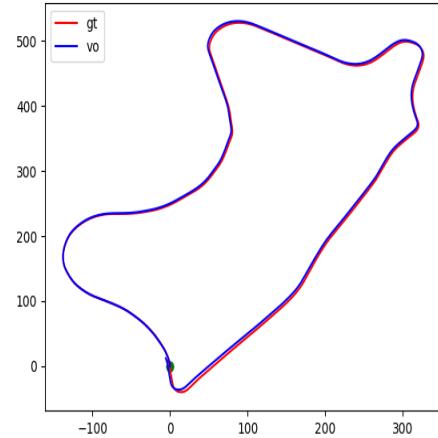
(g) Set 6



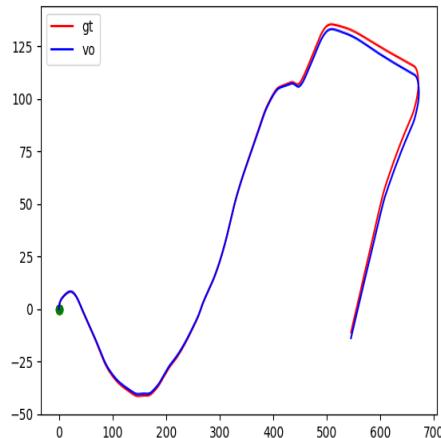
(h) Set 7



(i) Set 8

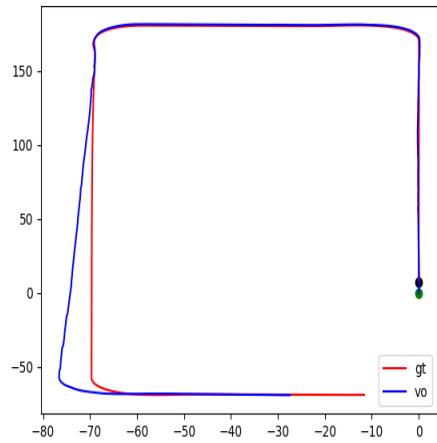


(j) Set 9

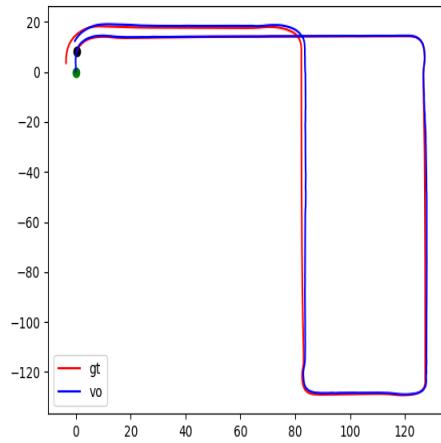


(k) Set 10

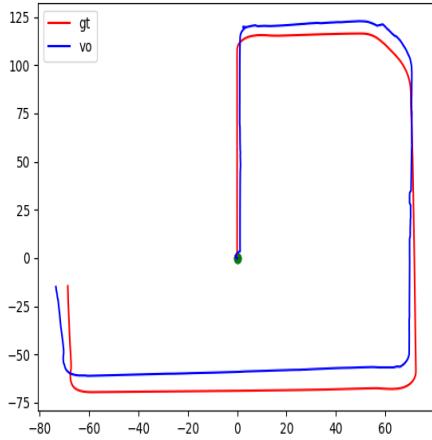
Slika 4.1. Rezultati za KITTI set podataka koristeći SIFT i 2D-2D korespondenciju tačaka



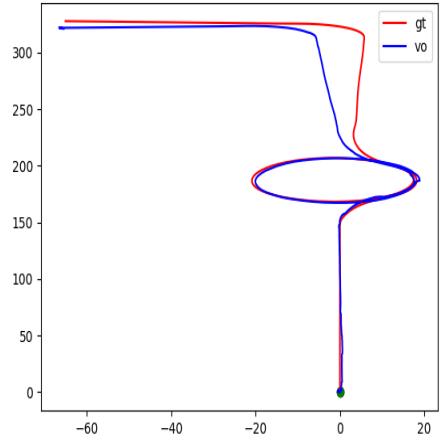
(a) Set 1



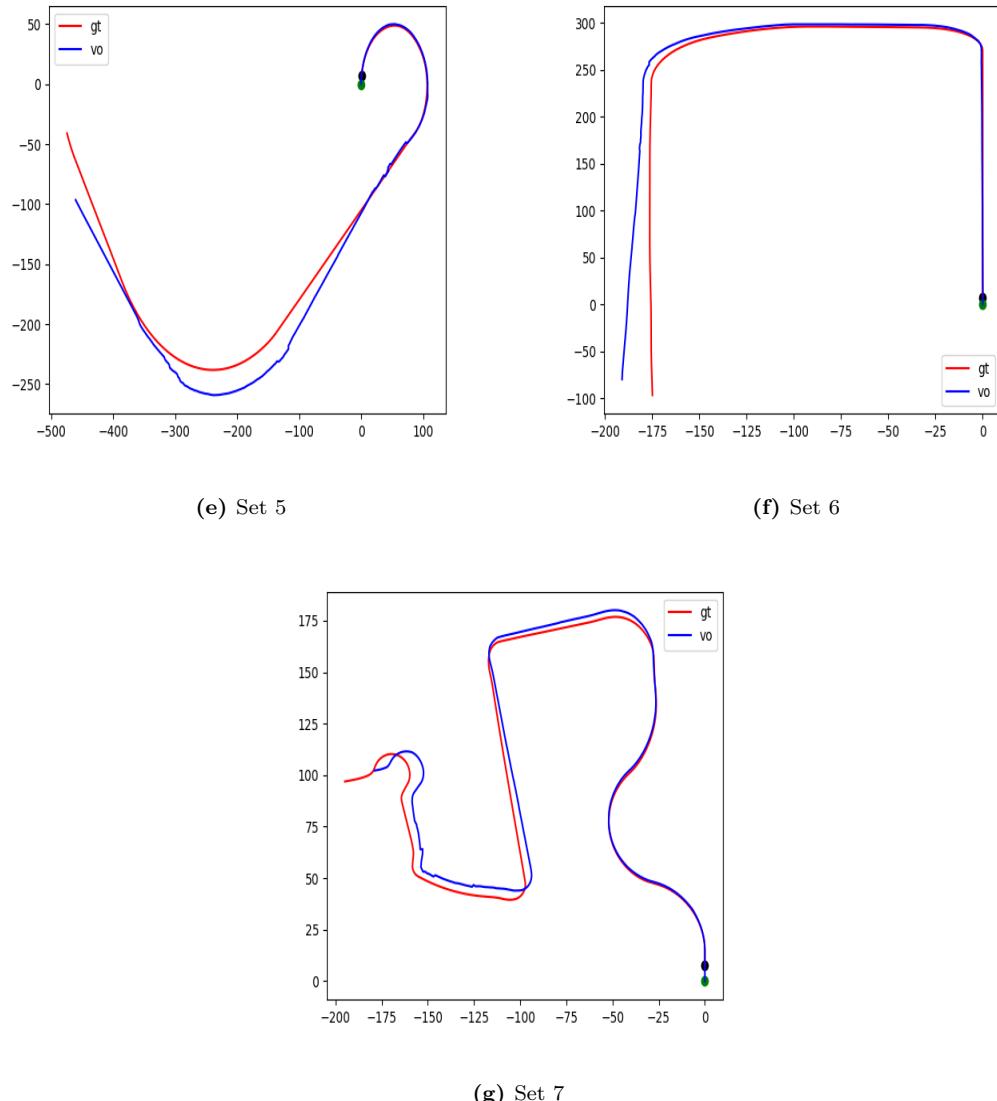
(b) Set 2



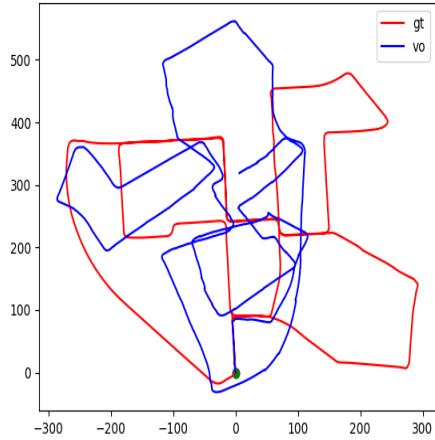
(c) Set 3



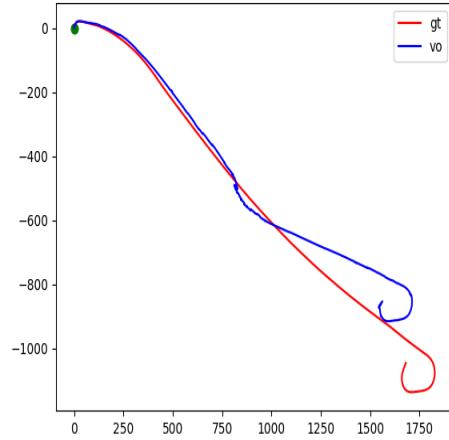
(d) Set 4



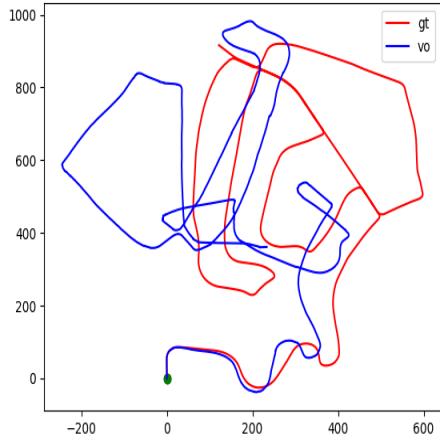
Slika 4.2. Rezultati za CARLA set podataka koristeći SIFT i 2D-2D korespondenciju tačaka



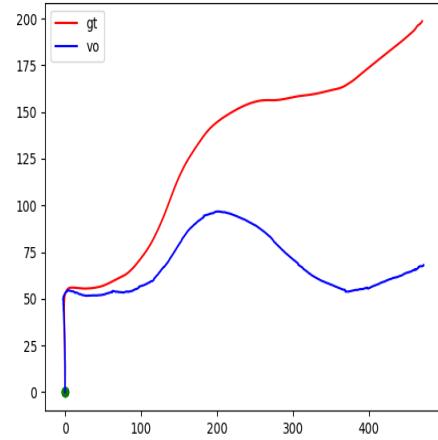
(a) Set 0



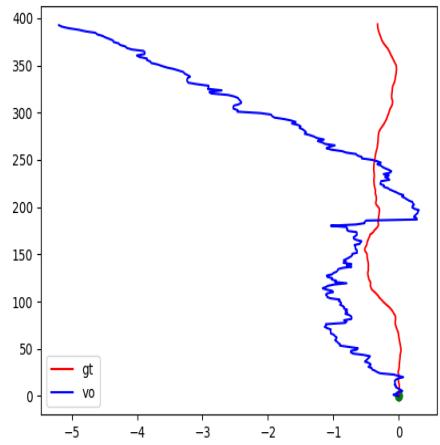
(b) Set 1



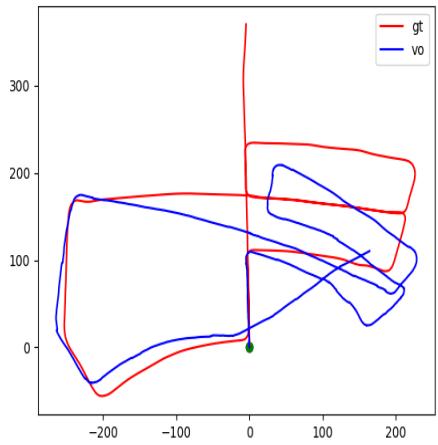
(c) Set 2



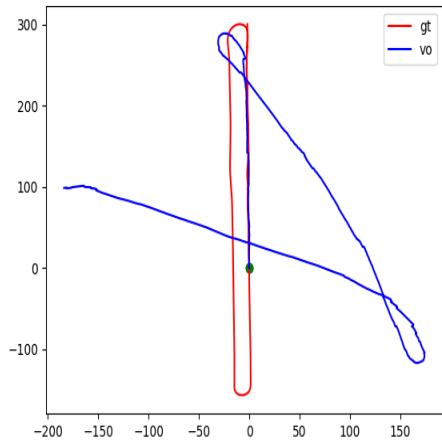
(d) Set 3



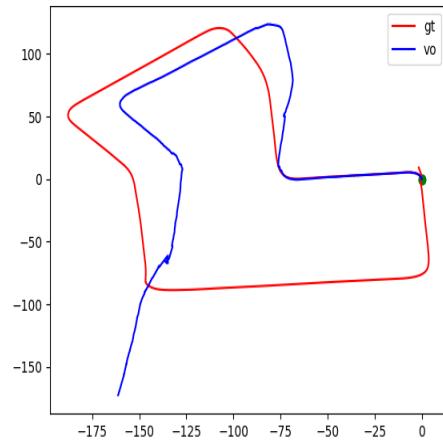
(e) Set 4



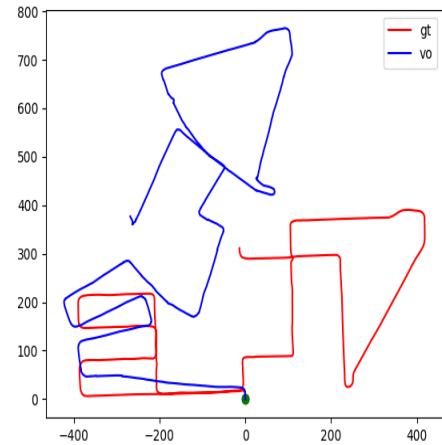
(f) Set 5



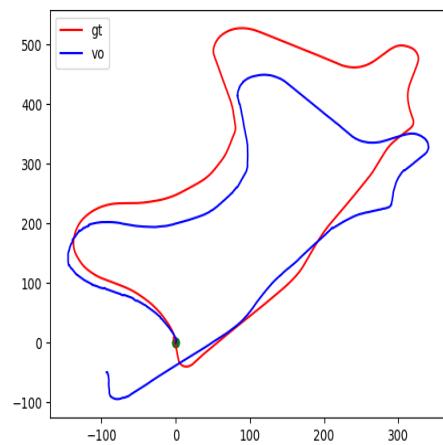
(g) Set 6



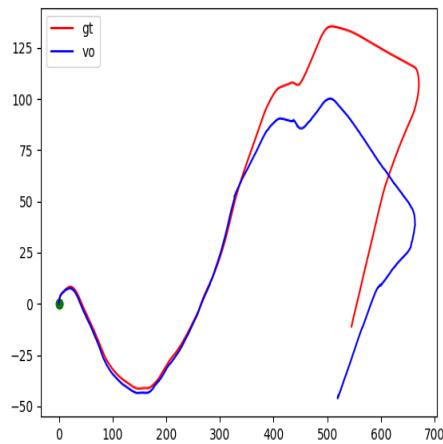
(h) Set 7



(i) Set 8

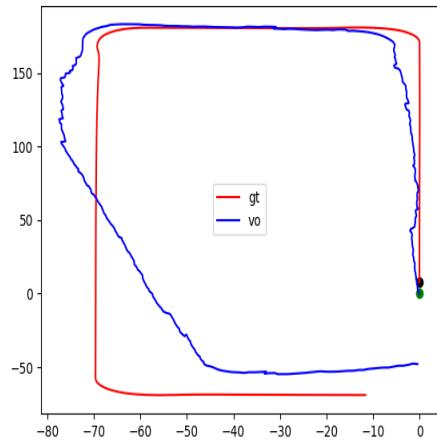


(j) Set 9

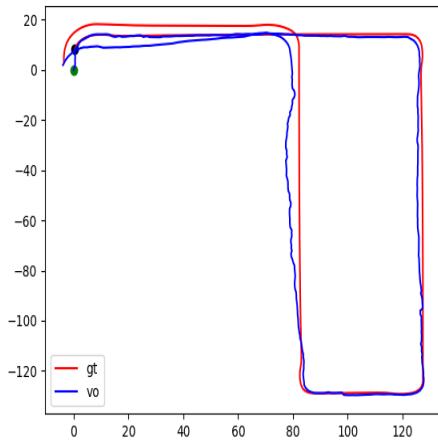


(k) Set 10

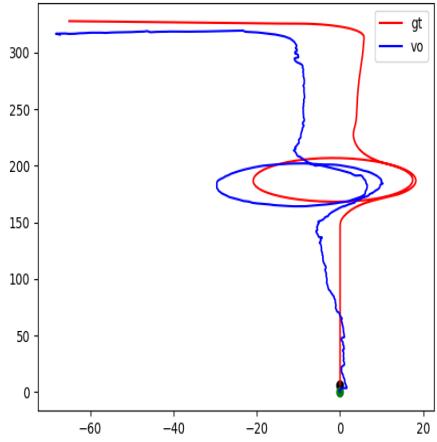
Slika 4.3. Rezultati za KITTI set podataka koristeći ORB i 2D-2D korespondenciju tačaka



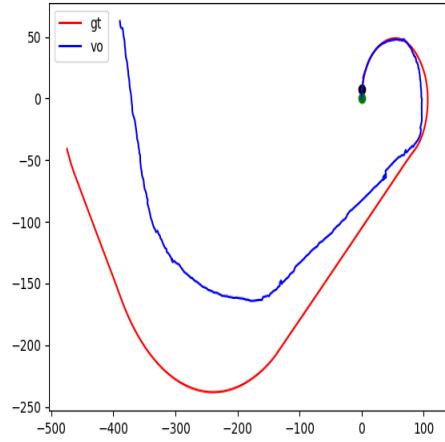
(a) Set 1



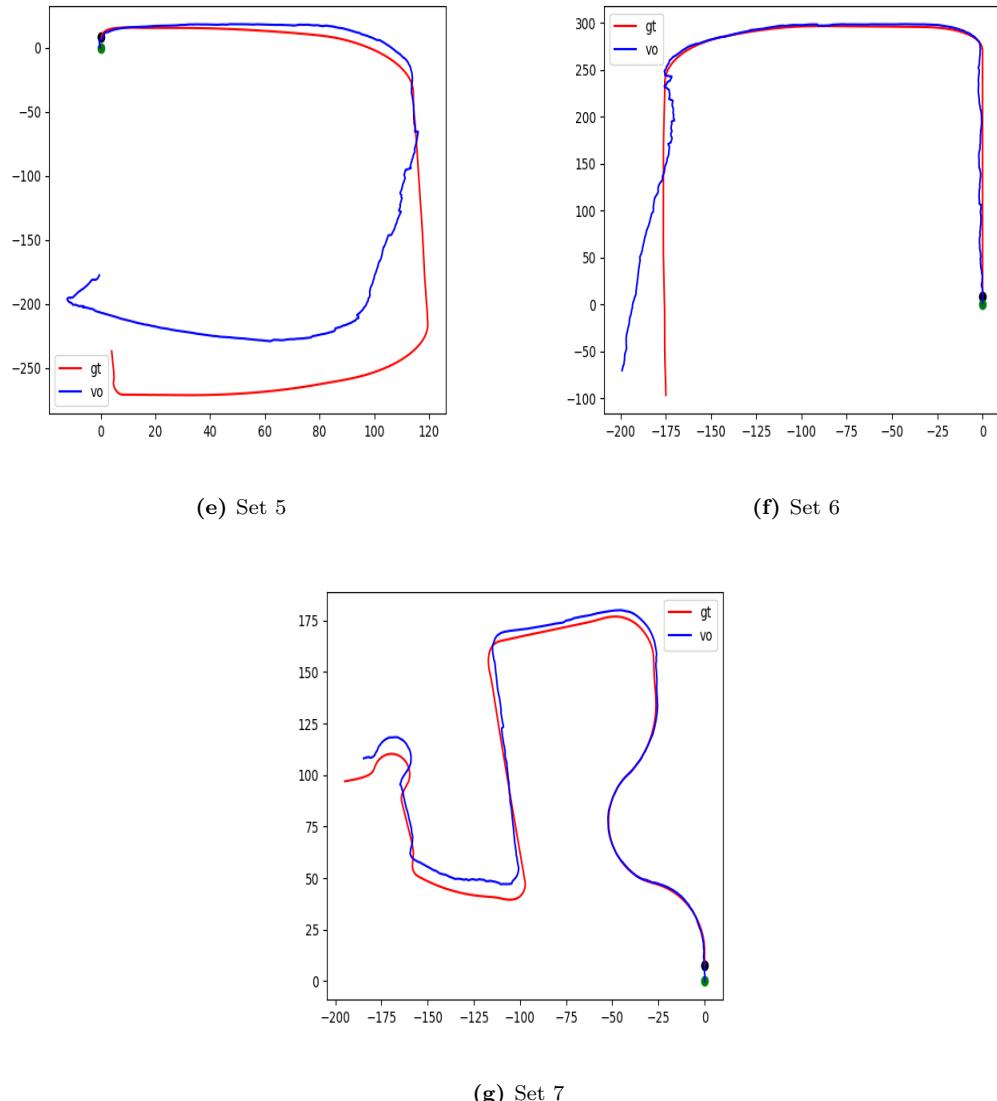
(b) Set 2



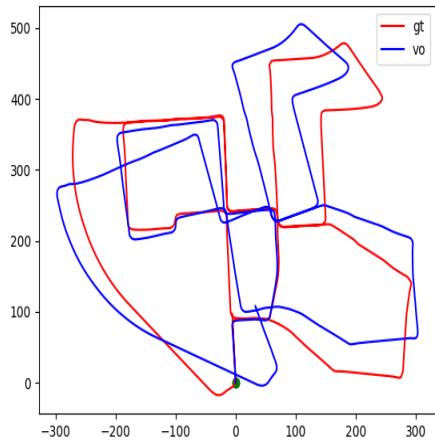
(c) Set 3



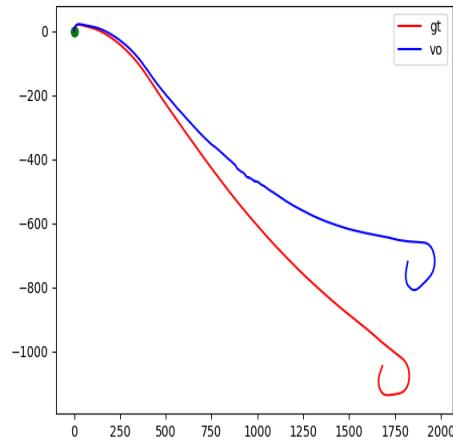
(d) Set 4



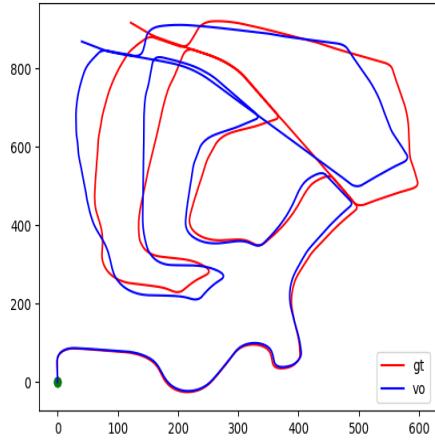
Slika 4.4. Rezultati za CARLA set podataka koristeći ORB i 2D-2D korespondenciju tačaka



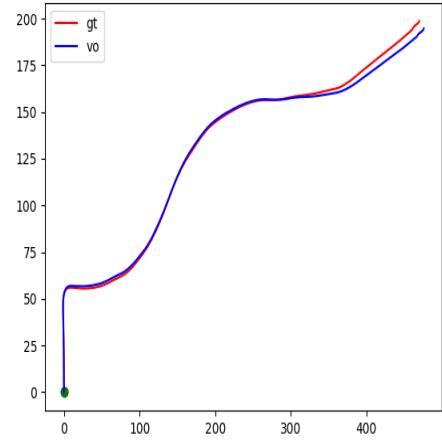
(a) Set 0



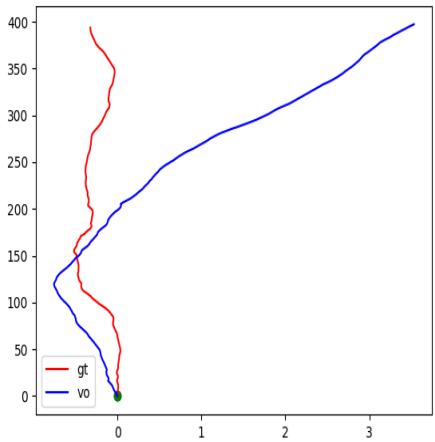
(b) Set 1



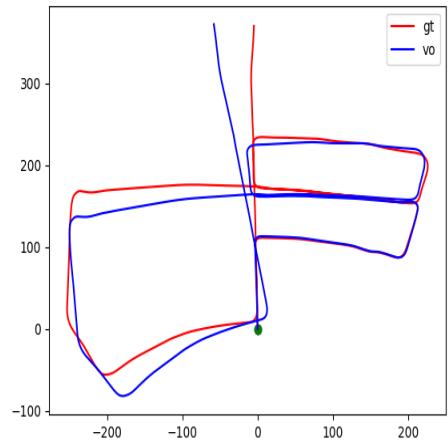
(c) Set 2



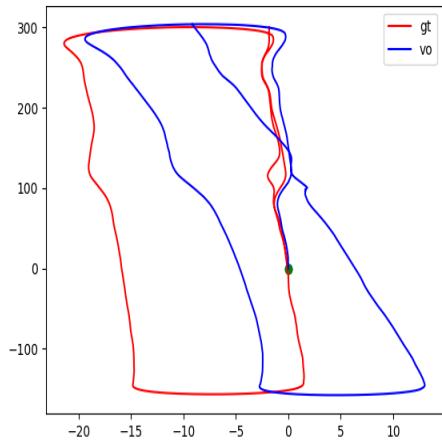
(d) Set 3



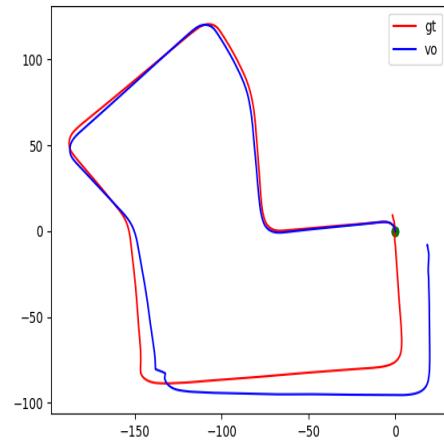
(e) Set 4



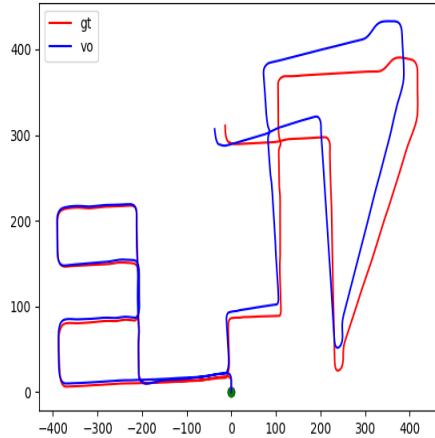
(f) Set 5



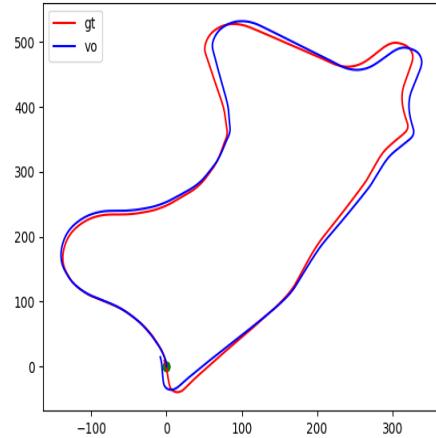
(g) Set 6



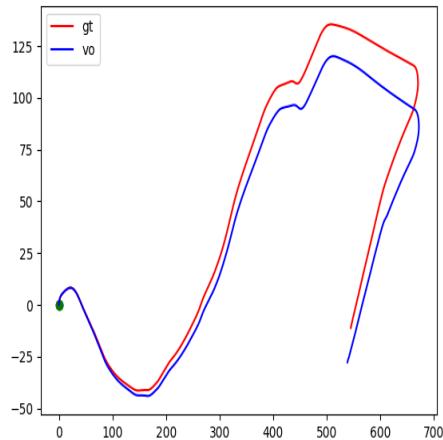
(h) Set 7



(i) Set 8

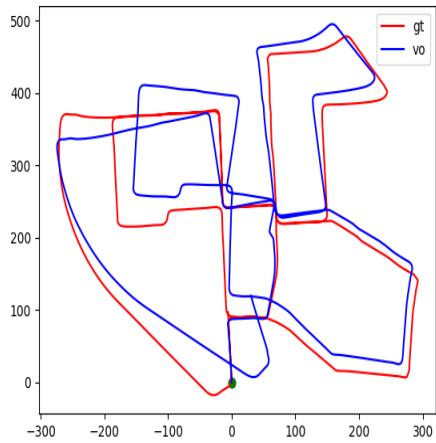


(j) Set 9

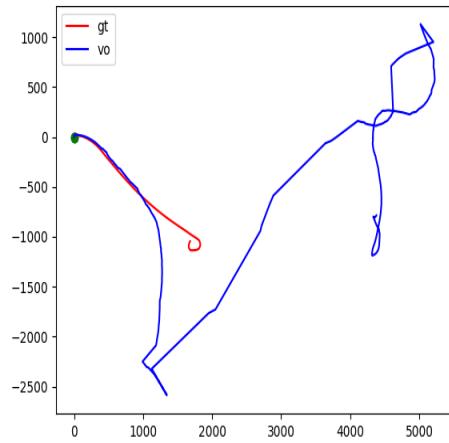


(k) Set 10

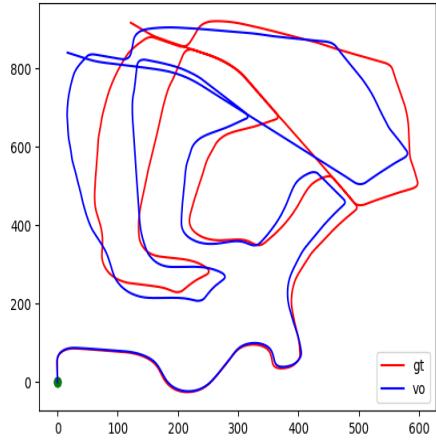
Slika 4.5. Rezultati za KITTI set podataka koristeći SIFT i 3D-2D korespondenciju tačaka



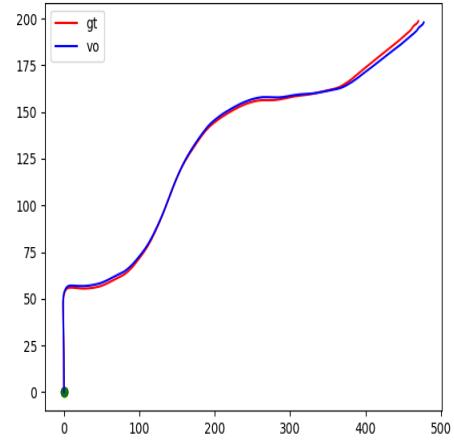
(a) Set 0



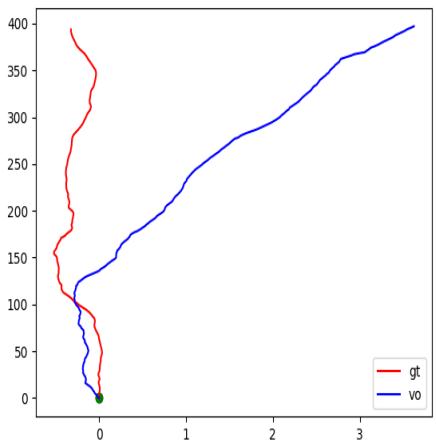
(b) Set 1



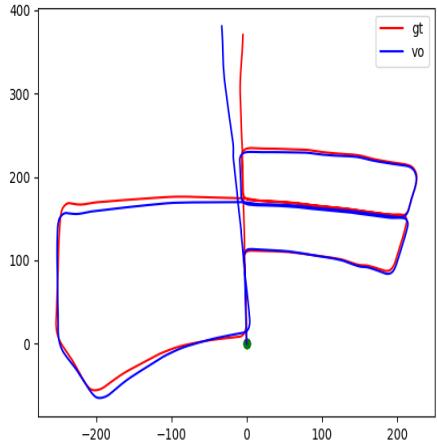
(c) Set 2



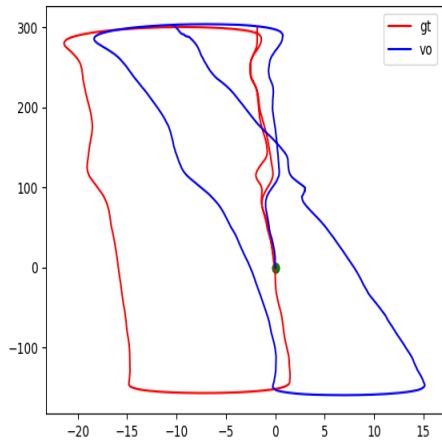
(d) Set 3



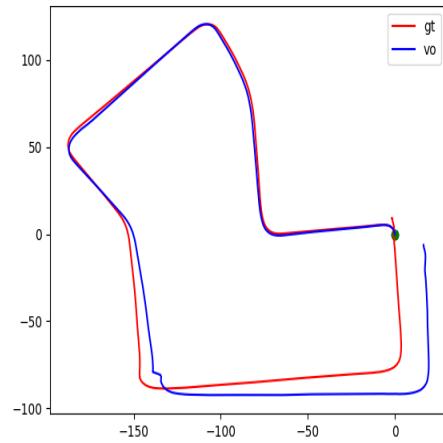
(e) Set 4



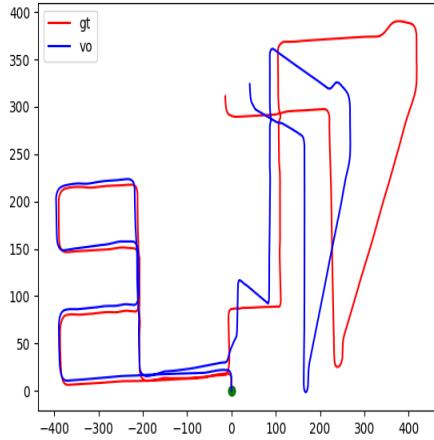
(f) Set 5



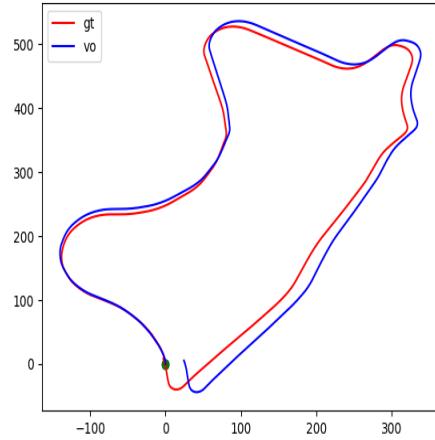
(g) Set 6



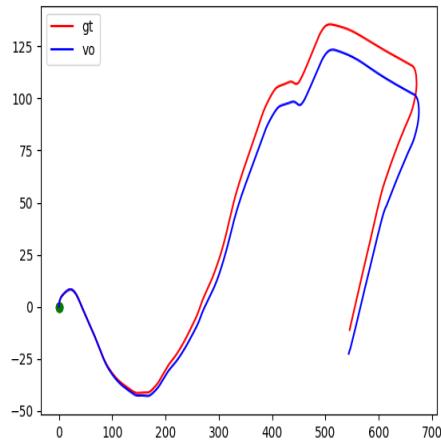
(h) Set 7



(i) Set 8



(j) Set 9



(k) Set 10

Slika 4.6. Rezultati za KITTI set podataka koristeći ORB i 3D-2D korespondenciju tačaka

4.2. Analiza performansi

4.2.1. 2D-2D estimacija kretanja

Mapa	Broj slika	Vrijeme izvršavanja [s]	
		SIFT	ORB
00	4541	734	179
01	1101	158	39
02	4661	807	188
03	801	128	33
04	271	39	10.75
05	2761	413	107
06	1101	163	41
07	1101	163	43
08	4071	665	162
09	1591	263	62
10	1201	192	48

Tabela 4.1. Ukupno vrijeme izvršavanja za 2D-2D estimaciju na osnovu KITTI seta podataka

Mapa	Broj slika	Vrijeme izvršavanja [s]	
		SIFT	ORB
1	449	107	26
2	437	108	26
3	428	99	23
4	438	99	23
5	431	108	25
6	429	96	24
7	436	111	25

Tabela 4.2. Ukupno vrijeme izvršavanja za 2D-2D estimaciju na osnovu CARLA seta podataka

4.2.2. 3D-2D estimacija kretanja

Mapa	Broj slika	Vrijeme izvršavanja [s]	
		SIFT	ORB
00	4541	962	508
01	1101	199	117
02	4661	1104	550
03	801	187	83
04	271	56	27
05	2761	600	272
06	1101	216	106
07	1101	220	111
08	4071	789	420
09	1591	301	155
10	1201	234	121

Tabela 4.3. Ukupno vrijeme izvršavanja za 3D-2D estimacij na osnovu KITTI seta podataka

Etapa (jedna iteracija)	Vrijeme izvršavanja [ms]
Detekcija znacajki - SIFT	52 ± 5
Detekcija znacajki - ORB	12 ± 2
Povezivanja znacajki - BF	0.805 ± 0.1
Povezivanje znacajki - FLANN	4.4 ± 0.3
Filtriranje parova znacajki - Lowe pravilo	0.06 ± 0.01

Tabela 4.4. Vrijeme izvršavanja jedne iteracije zajedničkih etapa oba načina estimacije kretanja

Etapa (jedna iteracija)	Vrijeme izvršavanja [ms]
Skala	0.017 ± 0.003
Esencijalna matrica	15 ± 5
Dekompozicija esencijalne matrice	0.96 ± 0.1

Tabela 4.5. Vrijeme izvršavanja jedne iteracije etapa za 2D-2D estimaciju kretanja

Etapa (jedna iteracija)	Vrijeme izvršavanja [ms]
Mapa dispariteta	47 ± 5
Mapa dubine	1.0 ± 0.2
3D-2D korespondenti	4.0 ± 0.8
PnP algoritam	0.63 ± 0.15

Tabela 4.6. Vrijeme izvršavanja jedne iteracije etapa za 3D-2D estimaciju kretanja

4.3. Analiza kvaliteta

Greska se može izračunati pomoću MSE (*Mean Squared Error*) metode.

$$MSE = \frac{1}{N} \sum_{i=1}^N (P_r - P_e)^2 \quad (4.1.)$$

gdje je P_r stvarna vrijednost a P_e estimirana vrijednost. U nastavku će za svaku koordinatu trajektorije (x,y,z) biti izracunata MSE.

4.3.1. 2D-2D estimacija kretanja

Mapa	SIFT			ORB		
	MSE - x	MSE - y	MSE - z	MSE - x	MSE - y	MSE - z
00	63.42	26.63	56.24	45570	6938	52216
01	45.41	168.69	510.62	9659	270	17939
02	57.29	57.93	36.41	81989	2578	100786
03	0.019	0.767	0.628	30	513	5800
04	0.203	0.0121	0.00	2.95	7.77	0.28
05	3.72	20.90	15.33	1216	5020	4818
06	13.86	5.15	0.064	9288	372	3791
07	1.52	3.02	1.25	5726	2845	3018
08	15.56	55.98	79.32	38486	1201	64710
09	2.79	9.66	5.16	1954	1077	7375
10	0.481	4.57	5.36	119	306	1138

Tabela 4.7. MSE po svim koordinatama za 2D-2D estimaciju na osnovu KITTI seta podataka

Mapa	SIFT			ORB		
	MSE - x	MSE - y	MSE - z	MSE - x	MSE - y	MSE - z
1	38.68	2.68	58.62	66.37	11.23	166.10
2	37.51	1.22	39.53	31.85	11.51	45.69
3	40.95	11.06	49.08	79.84	4.11	93.17
4	254.21	15.88	371.33	805.30	26.34	3121.93
5	132.54	2.56	100.43	172.31	39.51	1051.92
6	50.12	5.20	130.38	80.84	3.57	233.64
7	65.23	2.11	80.14	38.16	4.54	89.45

Tabela 4.8. MSE po svim koordinatama za 2D-2D estimaciju na osnovu CARLA seta podataka

4.3.2. 3D-2D estimacija kretanja

Mapa	SIFT			ORB		
	MSE - x	MSE - y	MSE - z	MSE - x	MSE - y	MSE - z
00	1000.5	416.21	1145.5	479.8	3058.9	500.2
01	7355	9263	45569	5513366	12106038	413158
02	1622.7	465.54	784.37	2715.2	481.9	1156.8
03	9.63	6.70	4.05	15.73	2.65	1.69
04	2.20	2.94	4.74	2.63	0.61	1.95
05	156.2	18.3	180.5	31.86	91.81	37.56
06	46.67	15.31	3.95	66.52	12.17	3.81
07	110.23	10.64	75.36	94.53	12.31	49.42
08	240.86	42.10	347.08	2874	23884	656.3
09	49.92	63.15	27.02	182.2	104.1	35.3
10	11.68	1070.1	168.26	9.96	168.4	84.9

Tabela 4.9. MSE po svim koordinatama za 3D-2D estimaciju na osnovu KITTI seta podataka

4.4. Zaključak

Na osnovu dobijenih rezultata mogu se izvesti odredjeni zaključci. Iz tabele 4.4., vidimo da je vrijeme izvršenja ORB-a znatno krace nego SIFT-a, tako da je za očekivati da će i ukupna vremena izvršavanja biti manja kada se koristi ORB. Oba ova podatka su dobivena kada se detektovalo oko 500 znacajki. U KITTI setu podataka, kamere su imale 10 FPS-a, tj. slikano je 10 frejmova po sekundi.

U tabeli 4.1. i 4.2. su navedena vremena izvršavanja za 2D-2D korespondenciju koristenjem SIFT-a i ORB-a. Ako uzmemo mapu 00 vidimo da ona ima 4541 frejmove. To znači da je ova putanja trajala oko 454 sekunde. Vrijeme izvršavanja koristeci SIFT za ovu mapu je bilo 734 [s], dok je koristenjem ORB-a bilo 179 [s]. To nam govori da je ORB dosta brzi od SIFT-a, i uporedjivanjem ovih vremena

sa stvarnim vremenom trajanja putanje, može se zaključiti da, za 2D-2D estimaciju kretanja, SIFT se ne bi mogao koristiti u stvarnom vremenu, dok ORB bi. Na osnovu tabele 4.7. i 4.8., može se zaključiti da je rekonstrukcija putanja tacnija koristenjem SIFT algoritma.

U tabeli 4.3. su navedena vremena izvršavanja za 3D-2D korespondenciju koristenjem SIFT-a i ORB-a. Vrijeme izvršavanja koristeci SIFT za ovu mapu je bilo 962 [s], dok je koristenjem ORB-a bilo 508 [s]. Vidimo da je 3D-2D korespondencija sporija. Razlog stoji u tome sto se racunala mapa dispariteta, pa se onda, nakon rekonstrukcije 3D tacaka, vrsilo ponovno povezivanje rekonstruisanih 3D tacaka prethodnog para slika, sa njihovim 2D korespondentima trenutnog para slika, jer uvijek za neke 2D korespondente prethodnog para slika nije moguce odrediti dubinu 3D tacke koja se projicirala u tu 2D tacku. Tek nakon ovoga, ovi 3D i 2D korespondenti se dalje proslijeduju funkciji za racunanje relativne transformacije koristeci PnP algoritam. Tako da u ovom slučaju, SIFT algoritam se ne bi mogao koristiti u stvarnom vremenu, dok je ORB na granici da se koristi u stvarnom vremenu. Kao i za 2D-2D estimaciju, i u ovom slučaju su bolji rezultati postignuti koristenjem SIFT algoritma, sto je prikazano u tabeli 4.9.. Medjutim u ovom slučaju, rezultati dobijeni koristenjem ORB-a su sličniji onima dobijenim koristenjem SIFT-a, nego sto je to slučaj za 2D-2D estimaciju.

U oba slučaja, koristen je BF Matcher za povezivanje znacajki, te nakon toga su se ti parovi filtrirali koristenjem *Lowe*-ovog pravila za odbacivanje losih parova znacajki. Koristen je i FLANN algoritam za povezivanje znacajki, medjutim iako bi po pravilu trebao biti brzi od BFM-a, on je bio nesto sporiji. Razlog tome je vjero-vatno njegova implementacija u *OpenCV*-u kao i podešavanje parametara FLANN Matcher-a, jer je moguce koristiti razlicite algoritme kada se on podešava.

Poglavlje 5.

Zaključak

S obzirom na to da su procesuirani veliki brojevi frejmova greška je ocekivana. Također, greška se akumulira tokom vremena, te to rezultuje većim odstupanjima aproksimirane od stvarne putanje. Većina trajektorija prate stvarnu trajektoriju, što govori da je estimacija kretanja agenta relativno uspjessna. Rezultati koji su dobijeni, nisu perfektni, međutim treba uzeti u obzir da nije izvršena optimizacija putanje (*bundle adjustment*), koja bi sigurno značajno popravila rezultate. Dakle za daljnja poboljšanja, potrebno bi bilo izvršiti optimizaciju putanje, kao zadnji korak vizualne odometrije. Za izradu ovog rada, koristene su funkcije iz *OpenCV-a*, što je značajno olakšalo samo pisanje koda.

Kroz ovaj rad pređeni su svi koraci vizualne odometrije (bez optimizacije) te je pokazan princip rada vizualne odometrije i principi na kojima se ona zasniva. Ovaj rad je bio uvodnog karaktera u vizualnu odometriju, a dobijeni rezultati se mogu poboljsati.

Bibliografija

- [1] R. Hartley, A. Zissermann: "Multiple view geometry in computer vision", New York, University of Cambridge, 2004.
- [2] R. Szeliski: "Computer Vision: Algorithms and Applications", 2021.
- [3] D. Osmanković: "Robotska vizija", predavanja, Sarajevo, ETF Sarajevo, 2020.
- [4] B. Lacevic: "Robotika 1", predavanja, Sarajevo, ETF Sarajevo, 2020.
- [5] Xiang Gao, Tao Zhang, Qinrui Yan and Yi Liu: "Basic Knowledge on Visual SLAM From Theory to Practice", 2021.
- [6] Davide Scaramuzza, Friedrich Fraundorfer: "Visual Odometry (Part I:The First 30 Years and Fundamentals)", 2011.
- [7] Davide Scaramuzza, Friedrich Fraundorfer: "Visual Odometry (Part II:The Matching, Robustness, Optimization, and Applications)", 2011.
- [8] Andrey Kudryavtsev, Sounkalo Dembele, Nadine Piat: "Stereo-image Rectification for Dense 3D Reconstruction in Scanning Electron Microscope", 2020.
- [9] Stephan Manthe, Adrian Carrio, Frank Neuhaus, Pascual Campoy, Dietrich Palus : "Combining 2D to 2D and 3D to 2D Point Correspondences for Stereo Visual Odometry", 2018. Visual Odometry"
- [10] Ethan Rublee,Vincent Rabaud,Kurt Konolige,Gary Bradski: "ORB: an efficient alternative to SIFT or SURF", 2011.
- [11] David G. Lowe: "Distinctive Image Features from Scale-Invariant Keypoints", 2004.

- [12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool: "SURF: Speeded Up Robust Features", 2006.
- [13] Michael Calonder, Vincent Lepetit, Christoph Strecha, Pascal Fua: "BRIEF: Binary Robust Independent Elementary Features", 2010.