SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 599

Napredno estimiranje strukture i gibanja kalibriranim parom kamera

Ivan Krešo

Zagreb, srpanj 2013.

Zahvala mentoru dr. sc. Siniši Šegviću na stručnoj pomoći i sugestijama pri izradi ovog diplomskog rada.

SADRŽAJ

Popis slika **vi**

1. Uvod **1**

2. Korišteni algoritmi i matematičke metode **3**

2.1. Model kamere 3

2.1.1. Idealna kamera 3

2.1.2. Kalibracija stereo sustava 8

2.1.3. Epipolarna geometrija 10

2.1.4. Esencijalna i fundamentalna matrica 11

2.1.5. Triangulacija 13

2.1.6. Rektifikacija stereo sustava 15

2.1.7. Triangulacija kod rektificiranog sustava 17

2.2. Vizualna odometrija - biblioteka Libviso2 18

2.2.1. Pronalaženje korespondencija značajki 18

2.2.2. Estimiranje gibanja 21

2.2.3. Filtriranje značajki - RANSAC 24

3. Ispitni skupovi **26**

3.1. Ispitna snimka Bumblebee 26

3.2. Ispitna snimka KITTI 28

3.3. Umjetni ispitni skup podataka 28

3.4. Kalibracijski skup 30

4. Programska izvedba i vanjske biblioteke **31**

4.1. Biblioteka OpenCV 31

4.2. Kalibracija 32

4.3. Rektifikacija 36

4.4. Biblioteka Libviso 39

4.5. Praćenje značajki 40
4.6. Monokularni SBA 43
4.7. Organizacija koda 44

5. Eksperimentalni rezultati **45**
5.1. Ispitna snimka Bumblebee 45
5.1.1. Sinkronizacija odometrije i GPS-a 45
5.1.2. Usporedba stare i nove kalibracije 49
5.1.3. Modifikacija biblioteke Libviso2 50
5.2. Ispitna snimka KITTI 52
5.3. Umjetni ispitni skup podataka 53

6. Zaključak **55**

Literatura **57**

POPIS SLIKA

1.1. Curiosity rover na Marsu 2

2.1. Geometrija idealne kamere [9] 3
2.2. Translacija centralne točke slike 5
2.3. Utjecaj leće na fokus kamere 6
2.4. Nejednolik lom svjetlosti uzrokuje efekt izobličenja 7
2.5. Učinak distorzije slike uzrokovan lećom 7
2.6. Ekstrinsični parametri $\mathbf{R}$ i $\mathbf{T}$ stereo sustava 9
2.7. Epipolarna geometrija 10
2.8. Triangulacija 14
2.9. Minimizacijja geometrijske pogreške 14
2.10. Rektifikacija stereo sustava 15
2.11. Uklanjanje distorzije i rektifikacija 16
2.12. Model stereo sustava nakon rektifikacije 16
2.13. Rektificirana triangulacija 18
2.14. Ovisnost dispariteta o udaljenosti točke 19
2.15. Nelinearnost između dispariteta i udaljenosti 19
2.16. Maske za detekciju značajki 20
2.17. Deskriptor korišten za usporedbu značajki 20
2.18. Ciklička detekcija značajki 21
2.19. Gornja slika prikazuje optički tok detektiranih značajki, a donja odabrane značajke nakon filtriranja, bliže značajke teže u crvenu boju, dok su one udaljenije obojene u zeleno 22

3.1. Bumblebee2 stereo kamera 26
3.2. Putanja vožnje 27
3.3. Lijeva i desna slika iz ispitne snimke Bumblebee 27
3.4. Lijeva i desna slika ispitnog skupa KITTI 28

3.5. Distribucija generiranih točaka u 3D prostoru 29
3.6. Spojene umjetna lijeva (crvena) i desna (plava) slika 30
3.7. Lijeva i desna slika iz kalibracijskog skupa 30

4.1. Kalibracijske slike prije i nakon rektifikacije 39

5.1. Usporedba odometrije s GPS podacima 46
5.2. Inkrementalni zakret Φ dobiven odometrijom i GPS-om 48
5.3. Inkrementalni translacijski pomak: $\Delta s(t) = \|\mathbf{p}(t) - \mathbf{p}(t-1)\|$ 49
5.4. Pogreška u skali $E_s(t)$ 50
5.5. Usporedba rezultata vizualne odometrije 50
5.6. Usporedba inkrementalnog translacijskog pomaka 51
5.7. Usporedba rezultata vizualne odometrije 51
5.8. Rezultati Libviso algoritma na KITTI ispitnom skupu, crveno - *groundtruth*, plavo - odometrija 52
5.9. Rezultati vizualne odometrije na umjetnom ispitnom skupu za pravocrtnu vožnju uz variranje razmaka između kamera 53
5.10. Uprosječene pogreške triangulacije 54

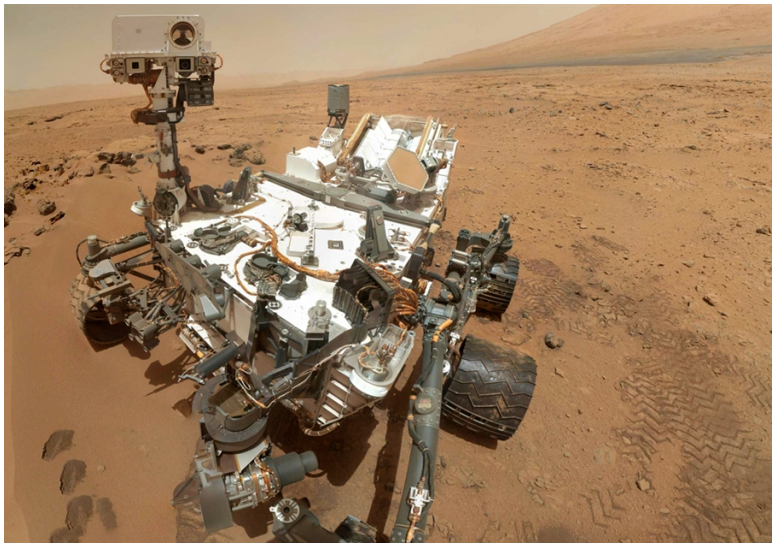
1. Uvod

Čovjek vidom prima veliku većinu informacija o svojoj okolini što vid čini najznačajnijim ljudskim osjetilom. Iako su načini na koje ljudski mozak obrađuje vizualne informacije još uvijek misterij, to nas ne sprječava da osmislimo vlastite algoritme obrade vizualnih informacija dobivenih digitalnim okom - kamerom. Ne čudi stoga da je upravo računalni vid područje intenzivnog istraživanja danas.

Ovaj rad bavi se stereo sustavima računalnog vida s naglaskom na problem vizualne odometrije. Vizualna odometrija je proces estimiranja kretanja (stereo) kamere kroz okolinu utvrđivanjem korespondencija značajki između parova vremenskih okvira uz nikakvo prethodno znanje o strukturi scene i kretanju. Trenutno najveće potencijalne primjene vizualne odometrije su u područjima robotike, automobilizma i sigurnosti u prometu općenito te proširene stvarnosti. Estimacija kretanja kamere, a pogotovo stereo kamere, je važan zadatak u robotici i naprednim sustavima za pomoć pri vožnji. Rješavanje ovog problema je i preduvjet za mnoge aplikacije poput onih za detekciju prepreka, robotskog upravljanja automobilom i mnogih drugih.

Posljednjih godina vizualna odometrija pokazala je visok potencijal kao rješenje za problem navigacije robota. Poznatiji primjer jest da ju svemirska agencija NASA koristi u misijama Mars Exploration Rover u kojoj su na Mars najprije poslana dva robota Spirit i Opportunity, a zatim i treći imena Curiosity koji je na Mars sletio 2012. godine [2]. Vizualna odometrija se na Marsu pokazala iznimno važnom s obzirom na nedostatak GPS satelita te posebno u slučajevima kada bi klasični inercijski sustav odometrije zakazao proklizavanjem kotača na strmom i rahlom terenu.

Termin *vizualna odometrija* skovao je 2004. god. Nister u svom istoimenom članku [7]. Naziv je izveden iz općeg pojma odometrije koji predstavlja problem estimacije gibanja nekog senzora iz podataka koje nam taj senzor daje. U vizualnoj odometriji taj senzor je kamera i estimacija gibanja se vrši inkrementalno između susjednih vremenskih okvira video sekvence. Bitni faktori za uspješan rad vizualne odometrije su dovoljno osvjetljenje scene te dovoljan udio statičnosti u sceni uz raznolikost tekstura kako bi se pronašao zadovoljavajuć broj statičnih značajki u slikama.



Slika 1.1: Curiosity rover na Marsu

Širi kontekst rada je određivanje gibanja kamere i strukture scene analizom slijeda pribavljenih slika. Razmatramo specifičan slučaj u kojem je kalibrirani par perspektivnih kamera montiran u smjeru kretanja vozila. U prvoj fazi određuju se specifični markeri u okolišu (točke interesa) i optički tok unutar sekvence slika. Za to se koriste metode za detekciju i praćenje značajki. Druga faza određuje strukturu scene u stvarnim koordinatama te gibanje kamere u odnosu na statičnu scenu iz podataka dobivenih tijekom prve faze. U okviru rada proučeni su različiti postupci praćenja značajki, kalibracije kamera, rektifikacije stereo sustava te vizualne odometrije. Na kraju rada su dani eksperimentalni rezultati evaluacije postupaka na više ispitnih snimaka.

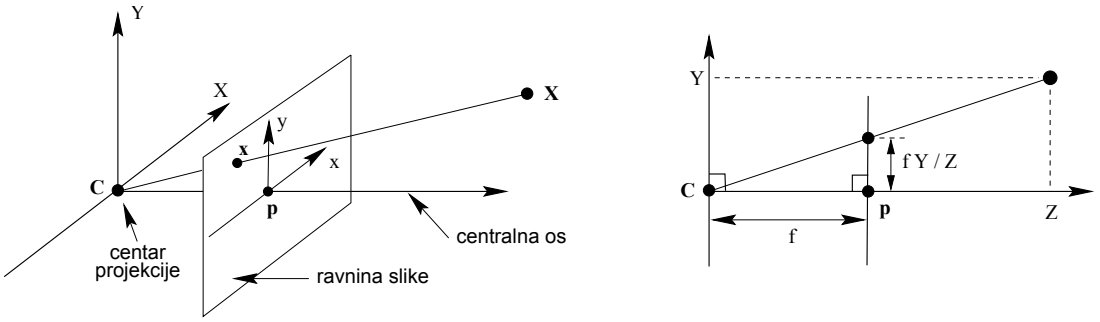
2. Korišteni algoritmi i matematičke metode

2.1. Model kamere

Kamera je zasigurno najvažniji senzor korišten u procesu vizualne odometrije. Iz tog razloga ključno je imati dobro razumijevanje o tome kako kamera generalno radi i kako fizičke pojave koji se u kameri zbivaju možemo modelirati matematički. U ovom poglavlju govorit će se o modelu kamere, kalibraciji stereo kamere i rektifikaciji slike, a potom o epipolarnoj geometriji te modelu stereo kamere.

2.1.1. Idealna kamera

Kamera je uređaj koji projicira točke iz 3D prostora u 2D sliku. Matematički opis tog projiciranja daje nam model idealne kamere (engl. *pinhole camera*). Idealna kamera je pojednostavljen matematički model fizičke kamere bez leća koja se sastoji od kutije koja na jednoj strani ima malenu rupicu. Svjetlost iz scene prolazi kroz rupicu i projicira obrnutu sliku na suprotnoj strani kutije. Kako bi izbjegli nepotrebnu obrnutu sliku u modelu se obično ravnina projekcije postavlja ispred centra projekcije umjesto iza. Takav model dan je na slici 2.1.



Slika 2.1: Geometrija idealne kamere [9]

$\mathbf{C}$ je projekcijski centar kamere. Žarišna duljina f je udaljenost ravnine slike od centra projekcije koje se u modelu postavlja u ishodište koordinatnog sustava svijeta. Centralna os je pravac okomit na ravninu slike koji prolazi točkom $\mathbf{C}$ Sjecište centralne osi i ravnine slike naziva se centralna točka $\mathbf{p}$. Perspektivnom projekcijom neka točka svijeta $\mathbf{X} = (X, Y, Z)^T$ projicira se u točku $\mathbf{x} = (fX/Z, fY/Z)$ slikovne ravnine.

Matematički ovaj model u homogenim koordinatama opisujemo projekcijskom funkcijom (2.1):

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Z \begin{bmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{bmatrix}$$

gdje su u i v koordinate 2D točke u koordinatnom sustavu slike, a s je informacija o udaljenosti ($s = Z$) koja se gubi projekcijom. Primijetimo da parametar žarišta f igra ulogu skaliranja točaka slike što se manifestira kao uvećavanje (engl. *zoom*) slike. U ovako normaliziranom 2D sustavu slike prikazanom na slici 2.1 centralna točka slike $\mathbf{p}$ nalazi se na ishodišnoj koordinati $(0, 0)^T$. No u realnoj kameri fotoosjetljivi senzor obavlja diskretizaciju analognog signala i slika se konačno sastoji od konačnog broja piksela gdje je sada praktičnije ishodišnu koordinatu postaviti u jedan u kutova slike pa se obično piksel u gornjem lijevom kutu slike označava koordinatama $(0, 0)^T$. Proširimo sada trenutni model kamere tako da translatiramo sliku kako bi se gornji lijevi piksel slike našao na centralnoj osi projekcije uvodimo translacijske parametre c_x i c_y:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Z \begin{bmatrix} \frac{fX}{Z} + c_x \\ \frac{fY}{Z} + c_y \\ 1 \end{bmatrix} \tag{2.2}$$

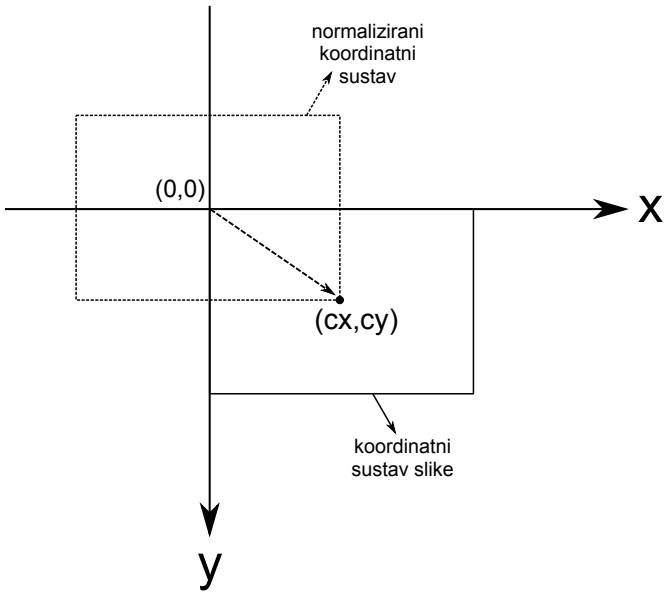
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{fX}{Z} + c_x \\ \frac{fY}{Z} + c_y \\ 1 \end{bmatrix}$$

Još bitnije od ove više praktične modifikacije jest činjenica da u realnoj kameri centar fotosenzora obično zbog nepreciznosti izvedbe nije savršeno poravnat s central-

nom osi leće. Ovu nepreciznost možemo također modelirati pomoću netom uvedenih parametara c_x i c_y. To znači da za model realne kamere centralna točka u normaliziranim koordinatama slike $\mathbf{p}$, a time i parametri c_x i c_y obično ne moraju biti točno na mjestu centralnog piksela slike već mogu biti pomaknuti za neku manju vrijednost ovisno u nepreciznosti izrade kamere. Na slici 2.2 prikazana je ova translacija slikovnih koordinata iz normaliziranog sustava kamere u koordinatni sustav slike.

Matrica modela koja sadrži intrinsične parametre kamere poznata je i kao matrica kamere $\mathbf{K}$ i ona opisuje linearnu transformaciju normaliziranih koordinata projekcije u koordinate slike.

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$



Slika 2.2: Translacija centralne točke slike

Dodatno, u model bismo htjeli proširiti kako bismo uključiti mogućnost gibanja kamere u odnosu na točke ili obrnuto, gibanje točaka u odnosu na kameru. Uvedimo stoga transformacijsku matricu $\mathbf{Rt}$ u model koja će rotirati i traslatirati točke prije njihovog projiciranja na ravninu slike.

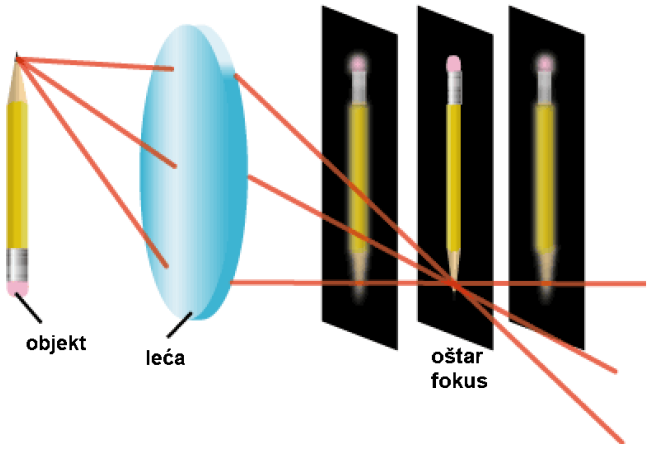
$$\mathbf{Rt} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Novi model kamere sada poprima i novi izgled:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

Parametri rotacije i translacije uvedene matrice $\mathbf{Rt}$ nazivaju se i ekstrinsičnim parametrima kamere. Bitno je napomenuti da uvedena matrica $\mathbf{Rt}$ modelira isključivo gibanje točaka u odnosu na kameru koja se uvijek nalazi u ishodištu koordinatnog sustava svijeta i gleda u pozitivnom smjeru z osi. Ako želimo modelom opisati gibanje kamere tada na mjesto matrice $\mathbf{Rt}$ upisujemo inverznu transformaciju gibanja kamere. Budući da je kamera fiksirana u ishodištu koordinatnog sustava, na taj način ćemo promatrane točke pomaknuti u položaj u kojem bi se našle ako se kamera giba sa koordinatnim sustavom svijeta.

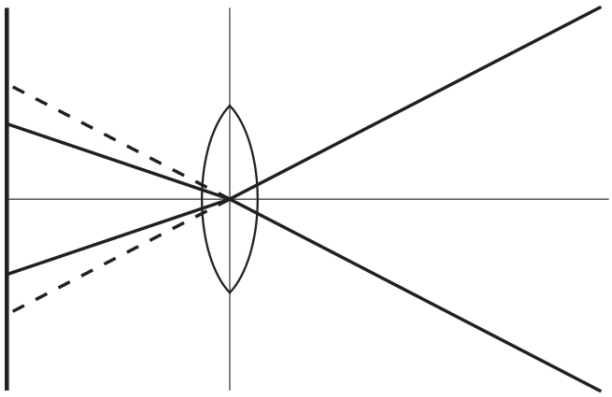
Trenutni model je pojednostavljena verzija rada realne kamere budući da ne modelira neke fizikalne pojave uzrokovane lećom. Jedna od tih pojava je fokus ili oštrina slike. Svi objekti projicirani idealnim modelom kamere će biti oštrog fokusa bez obzira na njihovu udaljenost od kamere. U realnom slučaju kamera će proizvesti oštru sliku samo za one objekte koji su u određenom rasponu udaljenosti od kamere gdje je taj raspon određen žarištem i svojstvima leće. No fokus nas ne brine pretjerano jer ga uvijek možemo podesiti unaprijed na dovoljan raspon udaljenosti ovisno o zahtjevima problema kako bi svi objekti u slici bili razumne oštrine. Stoga je zaključak da nema potrebe uključivati fokus u ovaj model kamere.



Slika 2.3: Utjecaj leće na fokus kamere

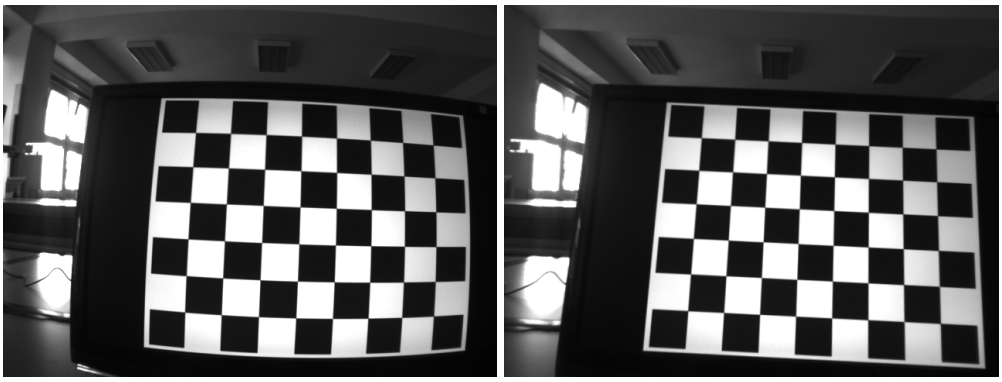
Ono što nas daleko više zanima je fizička pojava distorzije leće do koje dolazi kod praktičnih izvedbi kamera s lećama i nju moramo uključiti u naš model kamere.

Naime, leće su u kamerama potrebne kako bi se dobila oštra slika jer one savijaju zrake svjetlosti kako bi ih fokusirale na projekcijsku površinu slike. Sposobnost leće da prelama zrake ovisi o tome kako zraka pogodi leću. Upravo zato što ovo svojstvo nije konstantno i ovisi o mjestu upada zrake na leću dolazi do problema koji se manifestira kao efekt radijalne distorzije ili izobličenja slike.



Slika 2.4: Nejednolik lom svjetlosti uzrokuje efekt izobličenja

Efekt radijalne distorzije simetričan je s obzirom na udaljenost od optičkog centra, tj. savijanje svjetla je slabije ili jače (ovisno da li je riječ o konveksnoj ili konkavnoj leći) na rubovima leće nego na mjestima bližim optičkom centru leće. Slika 2.5 prikazuje učinak izobličenja uzrokovan konveksnom lećom.



(a) S distorzijom **(b)** Nakon uklanjanja distorzije i rektifikacije

Slika 2.5: Učinak distorzije slike uzrokovan lećom

Pored radijalne postoji i tangencijalna distorzija koja je uzrokovana greškom u poravnanju leće i senzora kamere koji u realnoj fizičkoj izvedbi nisu međusobno savršeno paralelni.

Distorzija se najčešće modelira kao suma radijalne i tangencijalne komponente izobličenja. Kako je u pravilu učinak radijalne distorzije znatno više izražen od učinka tangencijalne distorzije, obično se radijalna distorzija modelira s većim brojem parametara. Sljedeći izrazi opisuju relaciju između izobličenih (x_d, y_d) i neizobličenih (x_u, y_u) koordinata slike u kojima je efekt distorzije uklonjen:

$$x_d = x_u(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) + dx \tag{2.6}$$

$$y_d = y_u(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) + dy \tag{2.7}$$

$$r^2 = x_u^2 + y_u^2 \tag{2.8}$$

gdje su parametri tangencijalne distorzije dx i dy dani sa:

$$dx = 2\tau_1 x_u y_u + \tau_2(r^2 + 2x_u^2) \tag{2.9}$$

$$dy = 2\tau_2 x_u y_u + \tau_1(r^2 + 2y_u^2) \tag{2.10}$$

Ovaj model distorzije određen je s tri parametra za radijalnu distorziju $(\kappa_1, \kappa_2, \kappa_3)$ te dva parametra za tangencijalnu distorziju (τ_1, τ_2) Ovi parametri također spadaju u intrinsične parametre kamere s matricom kamere $\mathbf{K}$. Bitna razlika je jedino u tome da matrica $\mathbf{K}$ sadrži linearne intrinsične parametre dok parametri distorzije spadaju u nelinearne intrinsične parametre pa ih kao takve ne možemo ugraditi u matricu $\mathbf{K}$.

2.1.2. Kalibracija stereo sustava

Sada kada smo se upoznali s modelom kamere potrebno je provesti kalibraciju kamere kojom ćemo zapravo procijeniti intrinsične parametre modela (linearne i nelinearne).

Parametri modela koje određujemo kalibracijom:

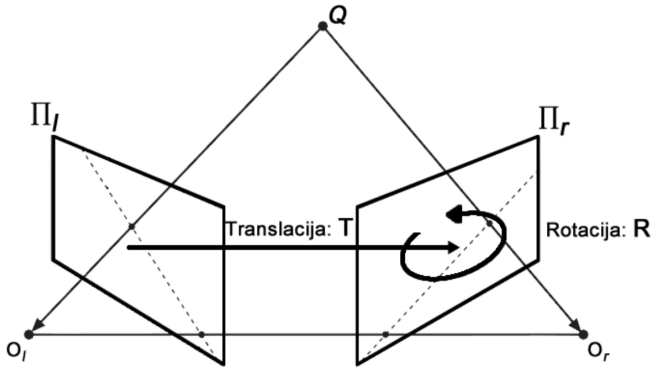
- intrinsični parametri:
  - linearni: matrica kamere $\mathbf{K}$
  - nelinearani: radijalna $(\kappa_1, \kappa_2, \kappa_3)$ i tangencijalna (τ_1, τ_2) distorzija leće
- ekstrinsični parametri:
  - međusobni položaj dviju kamera: rotacijska matrica $\mathbf{R}$ i translacijska matrica $\mathbf{T}$

Jedan od načina kalibriranja kamere je da se kamerom promatra neki objekt čija je geometrija poznata. U ovom radu kao kalibracijski objekt korišten je uzorak šahovske ploče koji se često koristi u različitim implementacijama kalibracijskog postupka. Taj

je uzorak pogodan jer se kutovi šahovskih polja lako detektiraju bazičnim metodama detekcije značajki pa se time bitno umanjuje mogućnost da neki kut ostane nepronađen. Algoritmu je na ulaz također potrebno dati i duljinu stranice jednog šahovskog kvadratnog polja u proizvoljnoj mjernoj jedinici. Bitno je samo paziti onda da će i izlazni parametri kalibracije tada biti u istoj mjernoj jedinici.

Potrebno je snimiti veći broj kalibracijskih uzoraka poželjno i što više različitih pozicija kako bi ekstrinsični parametri odnosa kamere i plohe uzorka što više varirali. Obično je sasvim dovoljno snimiti oko 20 takvih slika. Sa tim ulaznim podacima metoda kalibracije sada može iskoristiti činjenicu da se sve značajke nalaze u istoj ravnini, da tvore kvadratnu rešetku i da je udaljenost između susjednih značajki poznata. S tim znanjem se sada pronalaze oni intrinsični parametri same kamere te ekstrinsični parametri međusobnog odnosa kamere i plohe kalibracijskog uzorka za svaku sliku iz snimljenog kalibracijskog skupa s kojma se 3D točke kutova kalibracijskog uzorka optimalno projiciraju na poznate 2D koordinate značajki u slikama skupa.

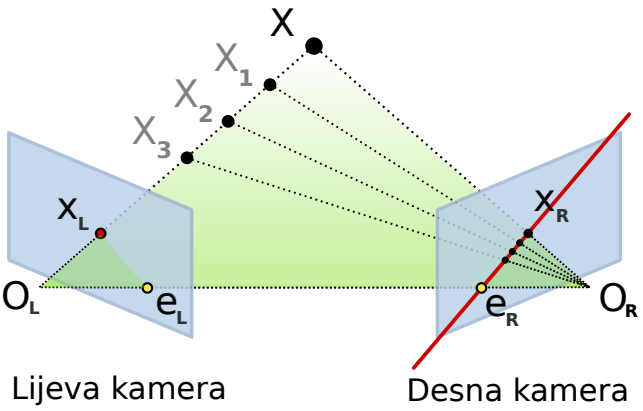
Kod kalibracije stereo sustava ili općenito sustava više kamera potrebno je još dodatno kalibracijom odrediti i međusoban odnos položaja svih kamera u prostoru, odnosno ekstrinsične parametre matrica $\mathbf{R}$ i $\mathbf{T}$ koje opisuju taj odnos između dvije kamere što je ilustrirano na slici 2.6. Ovo ne predstavlja dodatni problem u postupku jer ako kalibriramo lijevu i desnu kameru zasebno na istom kalibracijskom skupu snimljenom istovremeno tada znamo ekstrinsične parametre međusobnog odnosa lijeve, odnosno desne kamere i plohe kalibracijskog uzorka za svaku sliku iz snimljenog kalibracijskog skupa. No budući da također znamo da lijeva i desna kamera u istom vremenskom okviru gledaju isti položaj kalibracijske plohe tada zapravo znamo i međusobni položaj kamera, odnosno matrice $\mathbf{R}$ i $\mathbf{T}$.



Slika 2.6: Ekstrinsični parametri $\mathbf{R}$ i $\mathbf{T}$ stereo sustava

2.1.3. Epipolarna geometrija

Epipolarna geometrija opisuje geometriju stereo vida. Kada dvije kamere snimaju 3D scenu s dvije različite lokacije postoje geometrijske relacije između 3D točaka i njihovih projekcija na sliku koje nam daju informaciju o položaju točke u svijetu.



Slika 2.7: Epipolarna geometrija

Na slici 2.7 prikazane su dvije idealne kamere koje gledaju u smjeru točke X. $\mathrm{O_L}$ i $\mathrm{O_R}$ su točke centara projekcije za postavljene dvije kamere. $\mathrm{x_L}$ i $\mathrm{x_R}$ su projekcije točke X na lijevu, odnosno desnu ravninu. Točke $\mathrm{e_L}$ i $\mathrm{e_R}$ nazivaju se epipolovima i predstavljaju sjecište pravca određenog centralnim točkama $\mathrm{O_L}$ i $\mathrm{O_R}$ te lijeve i desne ravine projekcije. Lijeva kamera vidi pravac $\mathrm{O_L - X}$ kao točku, dok ga desna kamera vidi kao pravac. Taj pravac $(\mathrm{e_R - x_R})$ naziva se epipolarni pravac. Simetrično desna kamera pravac $\mathrm{O_R - X}$ vidi kao točku, a na lijevoj projekcijskoj ravnini on se vidi kao pravac $\mathrm{e_L - x_L}$.

Jasno je da epipolarni pravci ovise o položaju točke X. Uočimo još da svi epipolarni pravci jedne slike uvijek prolaze epipolarnom točkom te slike tj. projekcijske ravnine. Dodatno, alternativna vizualizacija bi bila ako bi se promatralo točke X, $\mathrm{O_R}$ i $\mathrm{O_L}$ koje određuju ravninu koja se naziva epipolarna ravnina.

Ako je relativna translacija i rotacija između dvaju kamera poznata, ranije definirana epipolarna geometrija dovodi do dva važna opažanja:

- Ako je poznata točka projekcije $\mathrm{x_L}$ tada je poznat i epipolarni pravac $\mathrm{e_R - x_R}$, a točka X se projicira na desnu sliku na točku $\mathrm{x_R}$ koja mora ležati na spomenutom epipolarnom pravcu.
- Ako su poznate točke projekcije $\mathrm{x_L}$ i $\mathrm{x_R}$, tada su poznati i njihovi projekcijski pravci. Ako su dvije točke na lijevoj i desnoj slici nastale projekcijom iste

točke X iz 3D prostora, tada se njihovi projekcijski pravci sijeku točno u točki X. Ovaj zaključak je jako bitan jer nam omogućuje da iz dvije točke na slikama izračunamo točku X u postupku koji se naziva triangulacija.

2.1.4. Esencijalna i fundamentalna matrica

Svojstvo epipolarne geometrije da se korespondentne značajke nalaze na istim pravcima naziva se popularno epipolarnim ograničnjem. Da bismo matematički opisali epipolarno ograničenje potrebno je uvesti pojmove esencijalne i fundamentalne matrice. Esencijalna matrica $\mathbf{E}$ sadrži informacije o translaciji i rotaciji koje vežu dvije kamere u prostoru, odnosno daje nam vezu između točaka $\hat{x}_L$ i $\hat{x}_R$ u normiranom koordinatnom sustavu slike (u kojem je žarišna duljina $f = 1$). Fundamentalna matrica $\mathbf{F}$ sadrži, uz informacije koje sadrži esencijalna matrica, i intrinsične parametre obje kamere. Ona povezuje točke x_L i x_R iz lijeve i desne slike.

Izvedimo najprije esencijalnu matricu $\mathbf{E}$. Imamo dvije normalizirane kamere ($\mathbf{K} = [\mathbf{I}|\tilde{\mathbf{0}}]$) koje projiciraju istu točku $\mathbf{P}$ iz 3D svijeta na svoje ravnine slike. Neka koordinate točke $\mathbf{P}$ budu $\mathbf{x}_1 = (x_1, y_1, z_1)$ u koordinatnom sustavu prve kamere i $\mathbf{x}_2 = (x_2, y_2, z_2)$ u sustavu druge kamere. Kako su kamere normalizirane, korespondentne koordinate točke u slikama nakon perspektivne projekcije su:

$$\mathbf{y}_1 = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \frac{1}{z_1} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{2.11}$$

$$\mathbf{y}_2 = \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \frac{1}{z_2} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{2.12}$$

Još jedno svojstvo normaliziranih kamera je da su njihovi koordinatni sustavi u relaciji preko translacije i rotacije pa se jednostavnom transformacijom točke mogu prebacivati iz jednog sustava u drugi na sljedeći način:

$$\mathbf{x}_2 = \mathbf{R}(\mathbf{x}_1 - \mathbf{t}) \tag{2.13}$$

gdje je $\mathbf{R}$ rotacijska matrica dimenzija 3×3, a $\mathbf{t}$ translacijski vektor dimenzije 3.

Esencijalna matrica je sada definirana kao:

$$\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times \tag{2.14}$$

gdje je $[\mathbf{t}]_\times$ matrična reprezentacija vektorskog produkta s vektorom $\mathbf{t}$. Matrična reprezentacija vektorskog produkta je uvijek antisimetrična matrica i detaljnije je opisana u izrazu (2.15).

$$\mathbf{t} \times \mathbf{a} = [\mathbf{t}]_\times \mathbf{a} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{2.15}$$

Dokažimo sada da definicija dana izrazom (2.16) zbilja vrijedi.

$$\begin{aligned} \mathbf{x}_2^T \mathbf{E} \mathbf{x}_1 &= (\mathbf{R}(\mathbf{x}_1 - \mathbf{t}))^T \mathbf{R}[\mathbf{t}]_\times \mathbf{x}_1 && (2.16) \\ &= (\mathbf{x}_1 - \mathbf{t})^T \mathbf{R}^T \mathbf{R}[\mathbf{t}]_\times \mathbf{x}_1 \\ &= (\mathbf{x}_1 - \mathbf{t})^T [\mathbf{t}]_\times \mathbf{x}_1 && (2.17) \\ &= \mathbf{x}_1^T [\mathbf{t}]_\times \mathbf{x}_1 - \mathbf{t}^T [\mathbf{t}]_\times \mathbf{x}_1 && (2.18) \\ &= 0 && (2.19) \end{aligned}$$

Korištene relacije i svojstva u izvodu:

1. $\mathbf{x}_2 = \mathbf{R}(\mathbf{x}_1 - \mathbf{t})$
2. $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ jer je $\mathbf{R}$ ortogonalna matrica.
3. Svojstva matrične reprezentacije vektorskog produkta:
   - $\mathbf{a}^T [\mathbf{b}]_\times \mathbf{a} = 0$
   - $\mathbf{b}^T [\mathbf{b}]_\times = 0$

Konačno, možemo pretpostaviti da su i z_1 i z_2 uvijek veći od nule jer inače točke se ne bi ni vidjele u kameri. Uz tu pretpostavku vrijedi uvijek sljedeće:

$$\mathbf{x}_2^T \mathbf{E} \mathbf{x}_1 = \frac{1}{z_2} \mathbf{x}_2^T \mathbf{E} \frac{1}{z_1} \mathbf{x}_1 = \mathbf{y}_2^T \mathbf{E} \mathbf{y}_1 = 0 \tag{2.20}$$

gdje je $\mathbf{y}_2^T \mathbf{E} \mathbf{y}_1 = 0$ epipolarno ograničenje.

Esencijalna matrica sadrži sve informacije o geometriji jedne kamere u odnosu na drugu, no ne sadrži nikakve informacije o samim kamerama. Međutim, u praksi nas zanimaju koordinate u pikselima slike. Kako bismo mogli pronaći vezu između piksela jedne slike s odgovarajućom epipolarnom linijom u drugoj slici, moramo iskoristiti intrinsične parametre kamera. Zato točku $\mathbf{y}$ u normaliziranim koordinatama supstituiramo točkom $\mathbf{y}'$ u koordinatnom sustavu kamere s intrinsičnim parametrima

$\mathbf{K}$ te vrijedi: $\mathbf{y}' = \mathbf{K}\mathbf{y}$, odnosno $\mathbf{y} = \mathbf{K}^{-1}\mathbf{y}'$. Jednadžbu za esencijalnu matricu sada možemo proširiti za rad s nenormaliziranim koordinatama slike:

$$\mathbf{y'}_2^T(\mathbf{K}_2^{-1})^T\mathbf{E}\mathbf{K}_1^{-1}\mathbf{y'}_1 = 0 \tag{2.21}$$

Upravo ovo proširenje opisuje fundamentalna matrica $\mathbf{F}$:

$$\mathbf{F} = (\mathbf{K}_2^{-1})^T\mathbf{E}\mathbf{K}_1^{-1} \tag{2.22}$$

Jednadžba (2.21) tada postaje:

$$\mathbf{y'}_2^T\mathbf{F}\mathbf{y'}_1 = 0 \tag{2.23}$$

2.1.5. Triangulacija

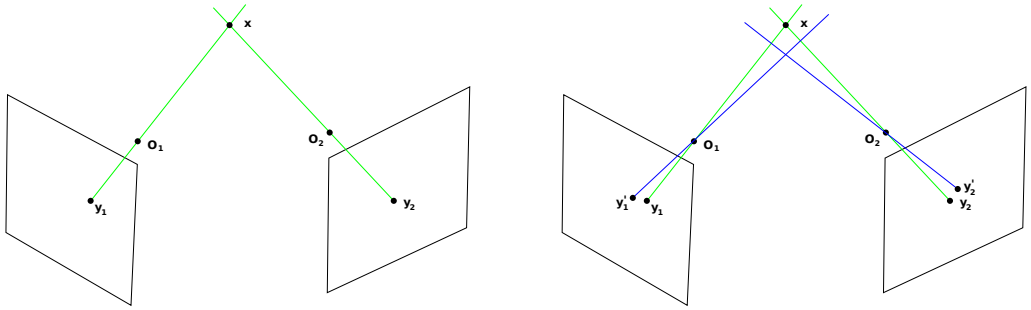
Triangulacija predstavlja postupak određivanja koordinata točke u 3D prostoru iz njenih projekcija na dvije ili više slika. Da bi se problem riješio nužno je poznavati parametre projekcijske funkcije kamere koja točku preslikava iz 3D prostora u 2D prostor. Ove parametre dobivamo pomoću prethodno opisanog procesa kalibracija kamere.

Idealan slučaj triangulacije, prikazan slikom 2.8a, bi bio kada bismo za kameru uzeli idealan model. Točka $\mathbf{x}$ se dobiva kao sjecište pravaca određenih točkom na slici i pripadajućim žarištem za svaku kameru. Međutim, u praksi se točke $\mathbf{y_1}$ i $\mathbf{y_2}$ ne mogu izmjeriti savršeno precizno. To je jasno jer realne digitalne kamere stvaraju diskretiziranu sliku što znači da je preciznost preslikavanja određena rezolucijom slike. Zbog takve prirode stvari mi, umjesto točnih koordinata točke, imamo cjelobrojne koordinate koje zapravo predstavljaju lokaciju izmjerenog piksela na slici. Slika 2.8 prikazuje realan slučaj triangulacije. Dakle, umjesto točnih projekcija $\mathbf{y_1}$ i $\mathbf{y_2}$ dobivaju se diskretizacijom zašumljene pomaknute projekcije $\mathbf{y_1}'$ i $\mathbf{y_2}'$ čiji se pravci projiciranja označeni zelenom bojom u općenitom slučaju uopće ne sijeku već se mimoilaze.

Ako se pravci ne sijeku tada je potrebno što točnije pokušati procijeniti položaj točke $\mathbf{x}$. Postoji više načina kako to ostvariti. Jedna intuitivno jasna metoda bi bila odabrati onu točku koja je najmanje udaljena od oba pravca. U tom slučaju htjeli bismo minimizirati funkciju sume kvadratnih udaljenosti tražene točke $\mathbf{x}$ i poznatih projekcijskih pravaca koji se mimoilaze:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(\|\mathbf{L_1}', \mathbf{x}\|^2 + \|\mathbf{L_2}', \mathbf{x}\|^2 \right)$$

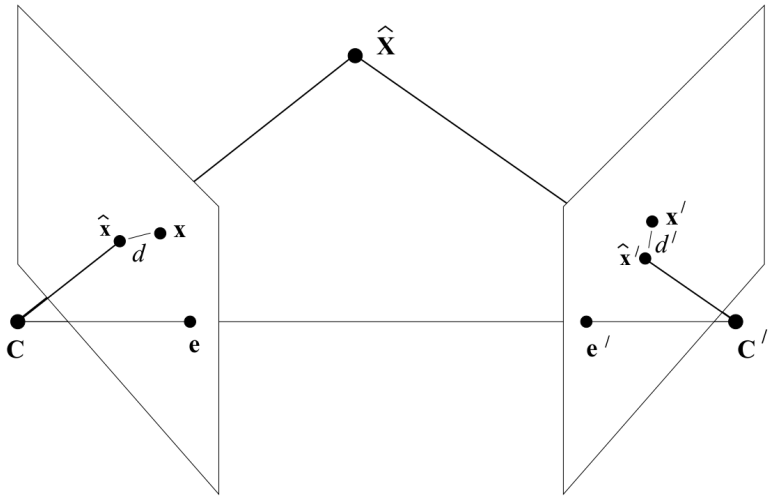
gdje su $\mathbf{L_1}'$ i $\mathbf{L_2}'$ projekcijski pravci točaka $\mathbf{y_1}'$ i $\mathbf{y_2}'$, $\mathbf{x}$ je slobodni parametar koji se minimizira, $\hat{\mathbf{x}}$ je tražena točka, a $\|\mathbf{L_1}', \mathbf{x}\|$ označava Euklidsku udaljenost između projekcijskog pravca i tražene točke.



(a) Idealan slučaj **(b)** Realan slučaj

Slika 2.8: Triangulacija

Postoji više pristupa rješavanju ovog problema, prethodno opisana triangulacija minimizira udaljenost 3D točke od dvaju 3D pravaca, međutim triangulaciju možemo provesti i minimiziranjem geometrijske pogreške.



Slika 2.9: Minimizacijja geometrijske pogreške

Na slici 2.9 je prikazana estimirana točka $\hat{\mathbf{X}}$ koja se na slikama projicira u točkama $\hat{\mathbf{x}}$ i $\hat{\mathbf{x}}'$ koje zadovoljavaju epipolarno ograničenje (2.23) za razliku od kamerom izmjerenih točaka $\mathbf{x}$ i $\mathbf{x}'$ koje ga zbog šuma u pravilu ne zadovoljavaju. Cilj je odabrati točku $\hat{\mathbf{X}}$ takvu da je reprojekcijska pogreška $d^2 + (d')^2$ minimizirana. To je ekvivalentno problemu pronalaženja točaka $\hat{\mathbf{x}}$ i $\hat{\mathbf{x}}'$ koje minimiziraju sljedeći izraz:

$$\underset{\hat{\mathbf{x}},\hat{\mathbf{x}}'}{\operatorname{argmin}}\, d(\mathbf{x},\hat{\mathbf{x}})^2 + d(\mathbf{x}',\hat{\mathbf{x}}')^2 \tag{2.24}$$

s obzirom na epipolarno ograničenje $\mathbf{y'}_2^T \mathbf{F} \mathbf{y'}_1 = 0$, gdje je $d(*,*)$ Euklidska udaljenost

između točaka.

Prednost prve opisane metode je da se može analitički izračunati, nađe se pravac okomit na dva projekcijska pravca koja se mimoilaze i onda možemo na primjer triangulirati točku između točaka gdje taj pravac sječe projekcijske pravce.

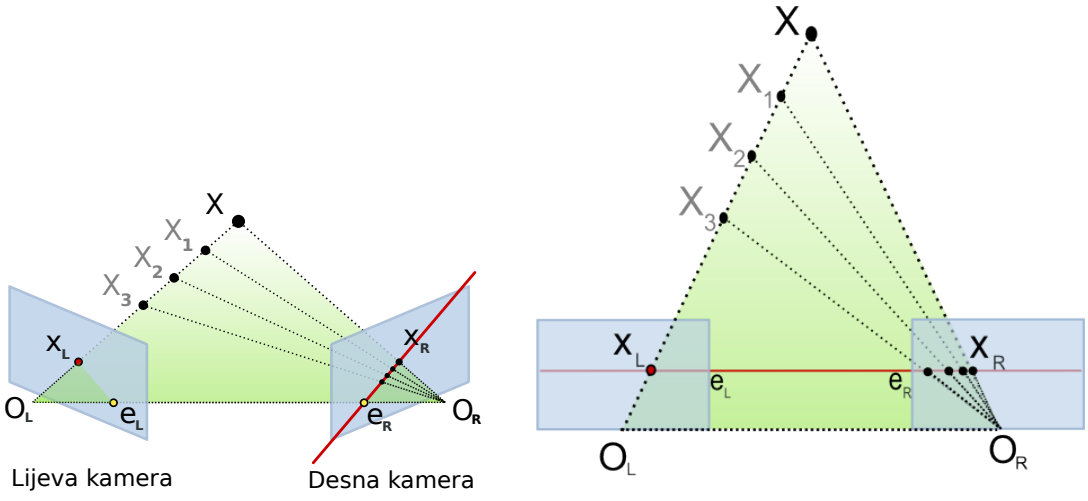
Druga opisana metoda se mora rješavati iterativnom minimizacijom i zbog toga će biti sporija od prve. Prednost druge metode je da možemo očekivati kako će minimiziranje reprojekcijske pogreške s obzirom na epipolarno ograničenje daji bolje estimacije 3D točke jer pretražuje točke $\hat{\mathbf{x}}$ i $\hat{\mathbf{x}}'$ u potpikselskoj preciznosti.

Postupak triangulacije možemo bitno olakšati ukoliko slike (ili značajke) najprije rektificiramo što je opisano u iduća dva poglavlja.

2.1.6. Rektifikacija stereo sustava

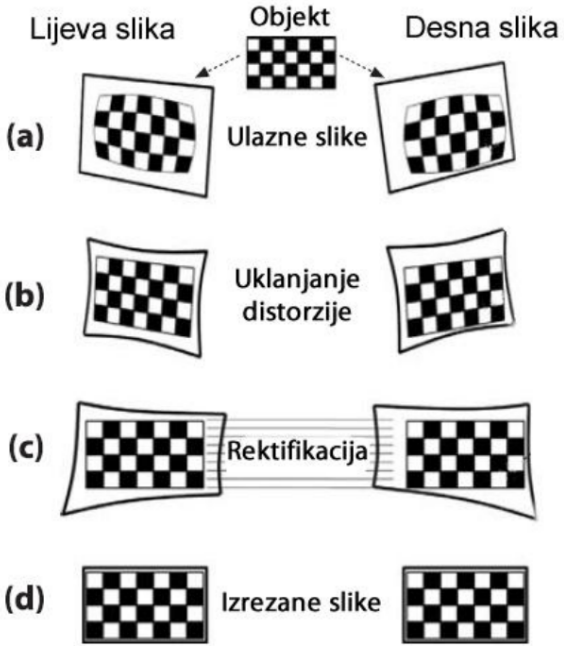
Nakon što smo kalibracijom doznali intrinsične parametre lijeve i desne kamere te ekstrinsične parametre odnosa njihovog međusobnog položaja možemo provesti postupak rektifikacije slike. Da bi se slike rektificirale potrebno je najprije ukloniti efekt distorzije, a zatim odrediti transformaciju slike koja bi se dobila ako bi projekcijske ravnine kamera ležale u istoj ravnini paralelno jedna uz drugu na istim visinama (slika 2.12). To se postiže tako da se epipolovi lijeve i desne kamere postave u beskonačnost na $\mathbf{x}$ osi.

Na slici 2.10 prikazana je ilustracija stereo sustava u općenitom slučaju te nakon rektifikacije.

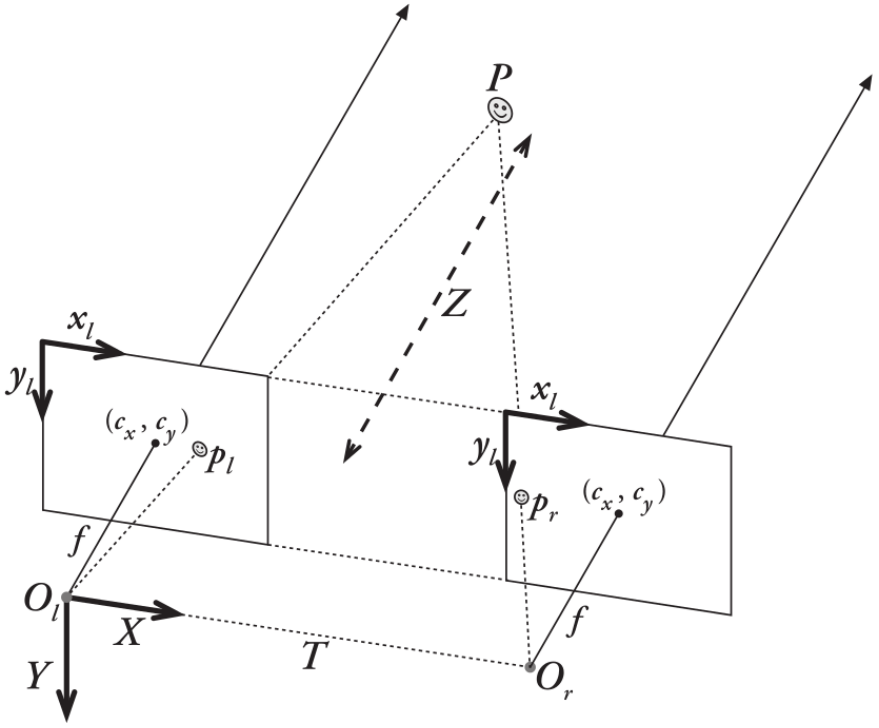


(a) Općenit slučaj **(b)** Nakon rektifikacije

Slika 2.10: Rektifikacija stereo sustava



Slika 2.11: Uklanjanje distorzije i rektifikacija



Slika 2.12: Model stereo sustava nakon rektifikacije

Nakon provođenja rektifikacije intrinsični parametri lijeve i desne kamere su potpuno jednaki budući da žarišna udaljenost f mora biti jednaka za obje kamere pošto smo postavili kamere u istu ravninu okomitu na centralnu os, a centralna točka (c_x, c_y) također mora biti jednaka kako bi se neka točka svjetovnog objekta projicirala u obje

slike na istoj visini. Uzimajući to u obzir model kamera idealnog stereo sustava možemo opisati na idući način:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} - \begin{bmatrix} t_b \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Jedini dodatak u odnosu na ranije opisan model jedne kamere je parametar t_b i on je jedini parametar u modelu koji se razlikuje za lijevu i desnu kameru. Odlučimo li lijevu kameru postaviti u ishodište koordinatnog sustava svijeta tada je $t_b = 0$ za lijevu kameru. Za desnu kameru će tada parametar t_b biti jednak odmaku između kamera prethodno označenim s b (engl. *baseline*) jer će za desnu kameru prije same projekcije biti potrebno translatirati sve točke na x osi za tu vrijednost pošto se samu kameru ne može pomicati jer je u modelu kamera uvijek fiksirana i postavljena u ishodište sustava, odnosno projicira se uvijek s obzirom na ishodište kao centar projekcije.

Prednost rektificiranog sustava u odnosu na nerektificirani jest u dvije stvari. Prva je da korespondencije značajki pretražujemo samo u jednoj dimenziji i to na istoj visini slike (istom retku piksela) te ne moramo računati epipolarne pravce za svaku značajku kao u slučaju nerektificiranog sustava što je ilustrirano na slici 2.10. Druga prednost je da je sada postupak triangulacije bitno jednostavniji što je opisano u idućem odjeljku.

2.1.7. Triangulacija kod rektificiranog sustava

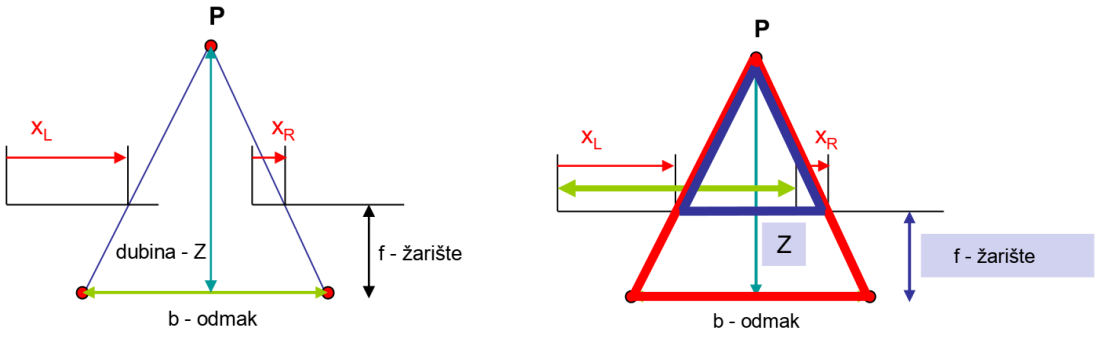
U slučaju rektificiranog sustava postupak triangulacije provodi se jednostavno u zatvorenoj formi. Slika 3.4 prikazuje ilustraciju triangulacije kod rektificiranog sustava. Iz sličnosti crvenog i plavog trokuta dobivamo dubinu Z, odnosno koordinatu 3D točke P na **z** osi (2.26). Na sličan način pomoću sličnosti drugih trokuta dobivamo i koordinate X i Y (2.27) i (2.28). U izrazu (2.26) uvedena je i nova oznaka $d = x_L - x_R$ koja predstavlja horizontalnu udaljenost između korespondentnih značajki u lijevoj i desnoj slici, a nazivat ćemo je ubuduće disparitet.

$$P = [X, Y, Z]^T \tag{2.25}$$

$$\frac{b}{Z} = \frac{b + x_R - x_L}{Z - f}; \;\; Z = \frac{f \cdot b}{x_L - x_R} = \frac{f \cdot b}{d} \tag{2.26}$$

$$\frac{X}{Z} = \frac{x_L}{f}; \;\; X = \frac{x_L}{f} \cdot \frac{f \cdot b}{d} = \frac{x_l \cdot b}{d} \tag{2.27}$$

$$\frac{Y}{Z} = \frac{y_L}{f}; \;\; Y = \frac{y_L}{f} \cdot \frac{f \cdot b}{d} = \frac{y_L \cdot b}{d} \tag{2.28}$$



Slika 2.13: Rektificirana triangulacija

Bitno svojstvo kod diskretizacije perspektivne projekcije koje u realnoj kameri obavlja senzor jest da vrijednosti dispariteta nelinearno ovise o trianguliranoj udaljenosti tj. za manje vrijednosti dispariteta udaljenost triangulirane značajke eksponencijalno raste što je vidljivo na slici 2.14. Možemo si to vizualizirati lako budući da znamo da će se sve značajke istog dispariteta triangulirati u istu ravninu na nekoj udaljenosti što je ilustrirano na slici 2.15. Na slici vidimo da je udaljenost između u koju će se triangulirati sve značajke dispariteta 12 i ravnine kod dispariteta 13 veća od udaljenosti između ravnine dispariteta 13 i ravnine 14. To je svojstvo perspektivne projekcije koje ne možemo izbjeći, naprosto se bliži objekti kod perspektivne projekcije projiciraju u lijevu i desnu sliku s većim disparitetima pa je stoga i informacija o njihovoj dubini sačuvana uz manje šuma. To si lako možemo vizualizirati na slici 2.12.

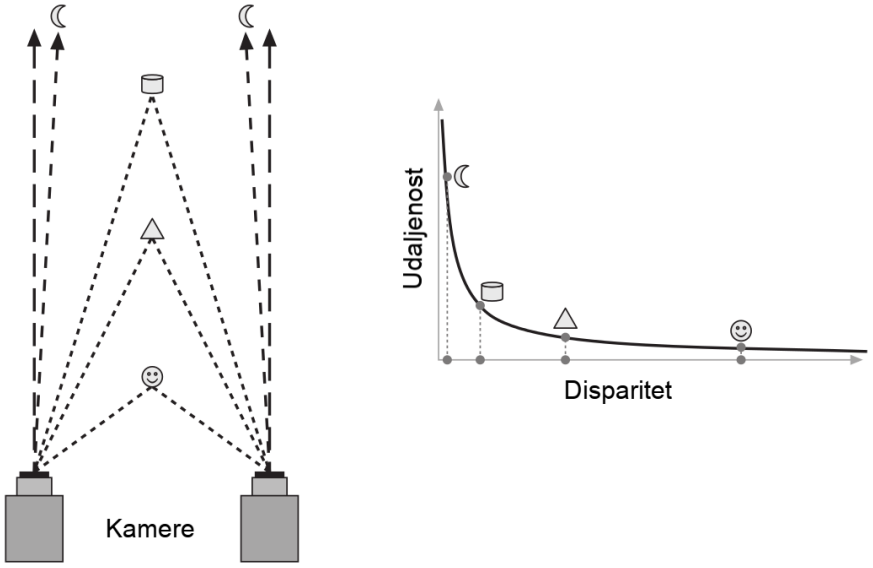
To znači da će i greška triangulacije biti veća za udaljenije značajke i da će izravno ovisiti o odmaku između kamera stereo sustava te širini rezolucije same kamere.

2.2. Vizualna odometrija - biblioteka Libviso2

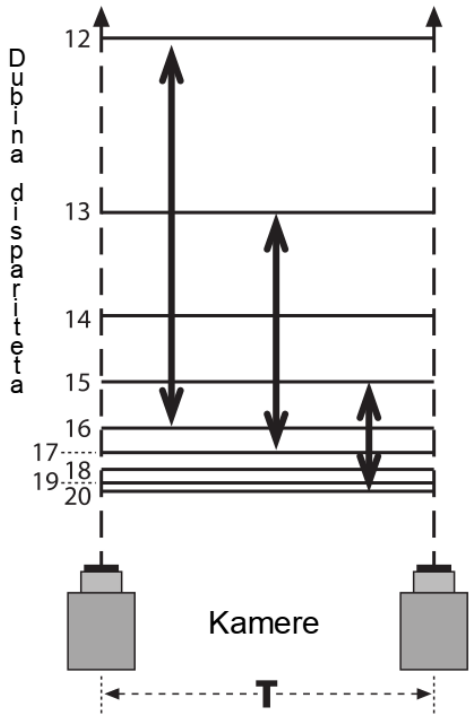
2.2.1. Pronalaženje korespondencija značajki

LIBVISO2 (Library for Visual Odometry 2) je vrlo brza multiplatformska C++ biblioteka za računanje 6 DOF (*degres of freedom*) gibanja mono ili stereo kamere u pokretu. Stereo verzija biblioteke zasniva se na minimizaciji projekcijskih grešaka rijetko podudarnih značajki. Glavna prednost ove biblioteke je što provodi postupak vizualne odometrije u stvarnom vremenu što znači da je sposobna obrađivati podatke brzinom 25 fps (*frames per second*). Biblioteka zahtijeva da su ulazne slike prethodno rektificirane.

Algoritam vizualne odometrije kojeg ova biblioteka ostvaruje na ulaz prima značajke dobivene cikličkim podudaranjem između četiri slike, trenutne lijeve i desne te



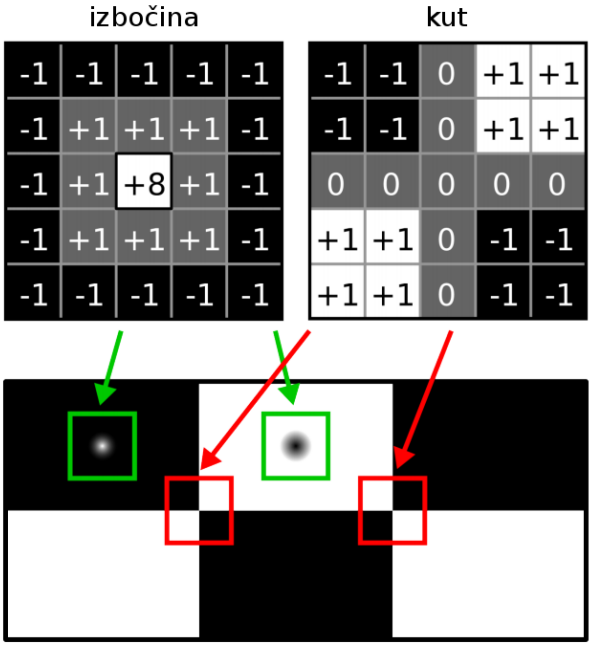
Slika 2.14: Ovisnost dispariteta o udaljenosti točke



Slika 2.15: Nelinearnost između dispariteta i udaljenosti

prethodne lijeve i desne kako prikazuje slika 2.18. Kako bi se pronašle lokacije stabilnih značajki, najprije se ulazne slike filtriraju s maskama dimenzija 5×5 za detekciju kuteva i izbočina. Korištene maske prikazane su na slici 2.16.

Sljedeći korak je potiskivanje odziva koji nisu maksimalni ni minimalni, čime preostale značajke pripadaju jednoj od dvije skupine što je označeno i na slici 2.16 crve-

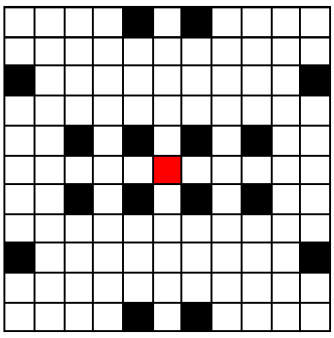


Slika 2.16: Maske za detekciju značajki

nim (kut) i zelenim (izbočina) bojama:

- izbočina: maksimum ili minimum.
- kut: maksimum ili minimum.

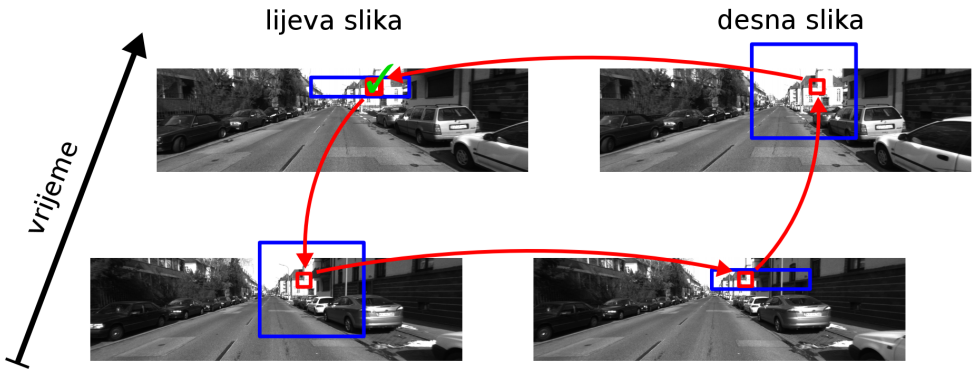
Potom se ulazne slike konvoluiraju Sobelovim filtrom te se oko značajki uzimaju područja veličine 11×11 piksela. Radi ubrzanja postupka, odziv Sobelovog filtra se kvantizira na 8 bitova te se razmatra samo 16 lokacija odziva, kao što prikazuje slika 2.17.



Slika 2.17: Deskriptor korišten za usporedbu značajki

Kao mjera sličnosti između dvije značajke koristi se suma apsolutnih razlika istaknutih lokacija u odzivu. Algoritam očekuje da se ista značajka pojavljuje u lijevoj i desnoj slici te u dva uzastopna vremenska okvira. To se postiže kružnim pronalaže-

njem korespondencija (engl. *quad matching, circular matching*). Počevši od značajke u trenutnoj lijevoj slici, u prozoru veličine $M \times M$ traži se korespondencija u prijašnjoj lijevoj slici, zatim prijašnjoj desnoj, trenutnoj desnoj te na kraju još jednom u trenutnoj lijevoj. Kružna korespondencija se prihvaća ako zadnja značajka odgovara prvoj. Dodatno, prilikom pronalaska korespondencija između lijevih i desnih slika, koristi se epipolarno ograničenje rektificiranog sustava s tolerancijom pogreške od jednog piksela. To znači da se za svaku značajku jedne slike pretražuje prozor veličine $M \times 3$ u drugoj slici. Među pronađenim korespondencijama, odbacuju se one čiji disparitet ili optički tok prelaze pragove τ_{disp} , odnosno τ_{flow}.



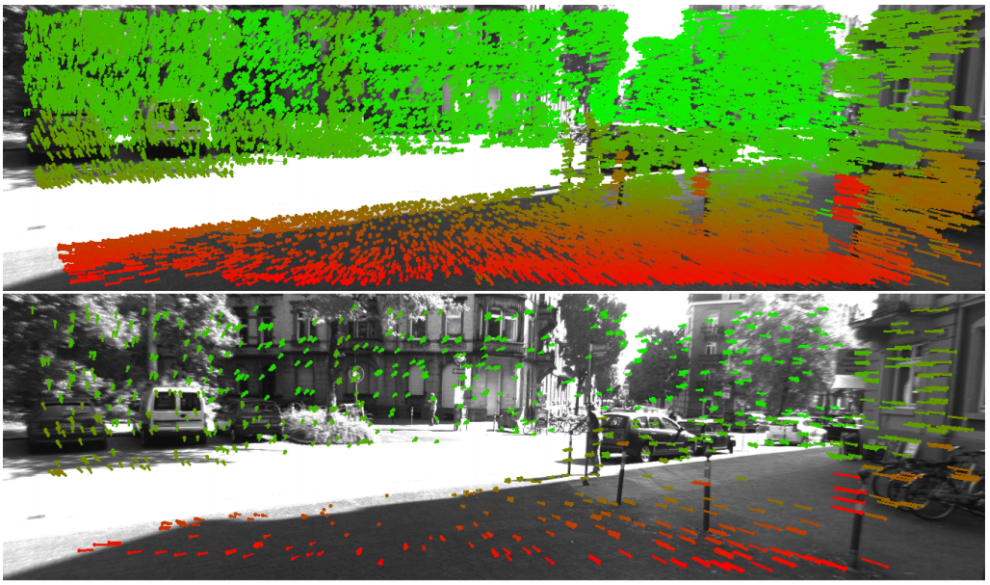
Slika 2.18: Ciklička detekcija značajki

2.2.2. Estimiranje gibanja

Obično se u sadržajem bogatoj slici raznolikih tekstura prethodno opisanim postupkom detekcije značajki može pronaći poveći broj istih. Biblioteka Libviso2 se ograničava na rad s fiksnim brojem značajki u svakoj iteraciji i to fiksiranje obavlja uniformnom selekcijom značajki. Naprosto ova biblioteka sliku podijeli na više pravokutnih područja i tada se u svakom području odabire fiksan broj značajki s najboljim minimalnim ili maksimalnim odzivima. Slika 2.19 prikazuje optički tok prije i poslije ovakve uniformne selekcije značajki.

Problem selekcije značajki je uvijek diskutabilan i može se provesti na mnogo načina no za početak je najbolje najprije probati jednostavno uzimati fiksan broj značajki s najboljim odzivima u čitavoj slici jer se ovakvom prisilnom uniformnom selekcijom lako može dogoditi da se odaberu jako loše značajke u područjima gdje dovoljno značajki jednostavno niti ne može biti.

Iz, sada selekcijom reduciranog, broja od N značajki se prethodno opisanom metodom triangulacije izračunavaju 3d koordinate točaka $\mathbf{X}_i$, $i \in \{1 \ldots N\}$. Dobivene



Slika 2.19: Gornja slika prikazuje optički tok detektiranih značajki, a donja odabrane značajke nakon filtriranja, bliže značajke teže u crvenu boju, dok su one udaljenije obojene u zeleno

3d koordinate točaka mogu se iskoristiti kako bi se izračunao pomak kamere u prostoru tako da se minimizira suma projekcijskih pogrešaka. Prisjetimo se sada ranije izvedenog modela kamere za rektificirani slučaj koji je opisan formulom (2.29).

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} - \begin{bmatrix} t_b \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \tag{2.29}$$

Izraz (2.29) možemo skraćeno napisati ako transformacijsku matricu ekstrinsičnih parametara modela označimo s $[\mathbf{Rt}]$ i sada poprima oblik (2.30).

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} \mathbf{Rt} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} - \begin{bmatrix} t_b \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \tag{2.30}$$

Forumula (2.30) sadrži:

- homogene koordinate slike $(u, v, 1)^T$
- žarišnu udaljenost f
- glavnu točku (c_x, c_y)
- rotacijsku matricu $\mathbf{R} = \mathbf{R_x}\ \mathbf{R_y}\ \mathbf{R_z}$

- translacijski vektor $\mathbf{t} = (t_1, t_1, t_1)^T$
- 3d koordinate točke $\mathbf{X} = (X, Y, Z)^T$ dobivene triangulacijom
- pomak za lijevu sliku $t_b = 0$, a za desnu sliku $t_b = b$ (engl. *baseline*)

Označimo sa $\pi^{(l)}(\mathbf{X}, \mathbf{R}, \mathbf{t}) : \mathbb{R}^3 \to \mathbb{R}^2$ projekciju opisanu formulom (2.30) koja uzima 3d točku $\mathbf{X}$ i projicira je na ravninu lijeve slike. Slično tako uvedimo funkciju $\pi^{(r)}(\mathbf{X}, \mathbf{R}, \mathbf{t})$ koja projicira točku na ravninu desne slike.

Iterativnom minimizacijom izraza kojeg ćemo ubuduće nazivati reprojekcijskom pogreškom:

$$\sum_{i=1}^{N} \|\mathbf{x_i}^{(l)} - \pi^{(l)}(\mathbf{X_i}, \mathbf{R}, \mathbf{t})\|^2 + \|\mathbf{x_i}^{(r)} - \pi^{(r)}(\mathbf{X_i}, \mathbf{R}, \mathbf{t})\|^2 \tag{2.31}$$

s obzirom na parametre $(\mathbf{R}, \mathbf{t})$ dobivamo one vrijednosti parametara za koje je pogreška projekcije minimalna. Čitatelj se može zapitati zašto minimiziramo baš reprojekcijsku pogrešku, a ne recimo izravno pogrešku u triangulaciji nakon pomaka kamere tj. udaljenosti između trianguliranih 3D točaka. Razlog leži u ranije opisanom svojstvu perspektivne projekcije kada je bila opisana nelinearna ovisnost pogreške triangulacije i razlučivosti dispariteta značajki (slika 2.15). Jednostavno zbog te nelinearne pogreške kod triangulacije dubine imamo veću nesigurnost u 3 dimenzije svijeta nego u 2 dimenzije slike. Zbog toga se reprojekcijska pogreška pokazala robusnijom na šum u podacima i njenom se minimizacijom dobivaju bolji rezultati.

Bitno je znati da su točke $\mathbf{X}_i$ dobivene triangulacijom iz lijeve i desne slike koje su prethodile onima koje se trenutno obrađuju, a čije su koordinate značajki na slikama jednake $\mathbf{x_i}^{(l)}$ za lijevu i $\mathbf{x_i}^{(r)}$ za desnu sliku. Na ovaj način dobivene vrijednosti rotacije $\mathbf{R}$ i translacije $\mathbf{t}$ su upravo one koje transformiraju točku $\mathbf{X_i}$ iz sustava prethodne kamere u sustav trenutne kamere. Sada vrijedi da ako je matrica $[\mathbf{Rt}]$ transformacijska matrica koja pomiče 3D točke iz koordinatnog sustava prethodnog okvira u koordinatni sustav trenutnog okvira, tada je pomak kamere u koordinatnom sustavu prethodnog okvira jednak inverzu ove matrice $[\mathbf{Rt}]^{-1}$ i opisuje pomak kamere u prostoru koji se dogodio između vremenski prethodnog i trenutnog para slika.

S obzirom na to da je pomak koji se optimira nelinearan u odnosu na parametre, za minimizaciju gornjeg izraza koristi se Gauss-Newtonova optimizacija koja u ovom radu neće biti detaljno opisana. Kada se u skupu značajki nikada ne bi pojavljivali *outlieri*, gornja metoda dala bi dovoljno dobro konačno rješenje. Međutim, pretpostavka o statičnoj sceni ne vrijedi uvijek, stoga je potrebno odabrati robusniji pristup koji će filtrirati značajke detektirane na pokretnim objektnima u sceni, a ostaviti samo

one iz statičnog dijela scene te uz to filtrirati i eventualne pogrešne korespondencije ako do njih dođe prilikom podudaranja značajki.

2.2.3. Filtriranje značajki - RANSAC

Već je spomenuto da metode najmanjih kvadrata poput Gauss-Newton metode ne mogu riješiti problem *outliera*. Jedan od pristupa kojim se problem *outliera* može napasti je metoda RANSAC. Algoritam RANSAC (engl. Random Sample Consensus) razvili su Fischler i Bolles 1981. godine kao robusnu alternativu metodi najmanjih kvadrata [3]. Općenito, RANSAC generira N modela koristeći nasumično odabrane podskupove ulaznog skupa, uspoređuje ih i vraća najbolji.

Za razliku od metoda najmanjih kvadrata, koje koriste što više mogućih primjera točka iz skupa za učenje, RANSAC radi upravo suprotno pokušavajući naučiti model na što manje primjera moguće. Na primjer, ako želimo povući optimalni pravac kroz set točaka tada će RANSAC u svakoj iteraciji uzimati nasumično po dvije točke iz skupa pošto je pravac određen već s dvije točke. Te dvije točke tada čine nasumično odabran uzorak u trenutnoj iteraciji. Za ostatak točaka iz skupa se sada provjerava konzistentnost s pravcem određenim s dvije odabrane točke, odnosno algoritam računa broj onih točaka koje su od pravca udaljene ispod prethodno definirane vrijednosti granične greške. Postupak se ponavlja u svakoj iteraciji algoritma. Nakon zadanog broja iteracija najbolja hipoteza je ona koja sadrži najveći broj konzistentnih točaka iz skupa i ona se kao robusno rješenje vraća na izlazu algoritma. Sada se sve one točke koje su konzistentne s tom hipotezom nalaze u skupu *inliera*, a one koje nisu konzistentne tretiramo kao *outliere*, tj. odbacujemo ih.

Kako bi se postupak dodatno poboljšao, možemo sada dodatno provesti metodu najmanjih kvadrata poput Gauss-Newton metode na skupu *inliera* koristeći robusnu hipotezu dobivenu RANSAC-om kao početno rješenje. U nastavku su dani koraci RANSAC algoritma:

1. Odaberi nasumičan uzorak točaka i odredi hipotezu modela.
2. Evaluiraj tu hipotezu pomoću funkcije dobrote - na primjer funkcija dobrote može biti broj točaka čija je udaljenost od pravca ispod određene vrijednosti.
3. Ponavljaj od koraka 1 zadani broj iteracija.
4. Odaberi najbolju hipotezu (onu s najboljom dobrotom) i poboljšaj ju nekom od metoda najmanjih kvadrata na skupu *inliera*.

Primijenjen na problem vizualne odometrije, algoritam glasi:

1. Nasumično odaberi tri značajke i koristeći ranije opisani algoritam pronađi rotacijsku matricu $\mathbf{R}$ i translacijski vektor $\mathbf{t}$.

2. Primijenimo pronađene $\mathbf{R}$ i $\mathbf{t}$ na sve značajke. Odaberemo značajke čija je udaljenost reprojekcije od stvarnih koordinata u trenutnom okviru manja ili jednaka nekom iznosu granične greške. Te značajke nazivaju se potpornim skupom trenutne hipoteze (engl. *inliers*) jer su konzistentne s njom.

3. Ako generirani model ima više *inliera* od svih prethodno generiranih modela, trenutni model pamtimo kao najbolji i nastavljamo algoritam od koraka 1. Ako dva modela imaju isti broj *inliera*, bolji je onaj čija je prosječna vrijednost udaljenosti manja. Kriterij zaustavljanja je unaprijed zadan broj iteracija.

3. Ispitni skupovi

U radu su za testiranje sustava korištene dvije ispitne snimke snimljene različitim kamerama te jedan umjetni ispitni skup. Njihovi opisi su slijede u idućim poglavljima teksta.

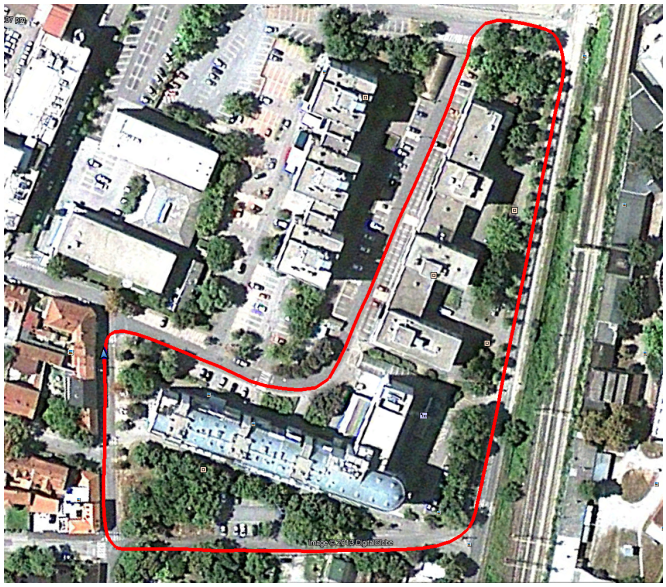
3.1. Ispitna snimka Bumblebee

Ovaj ispitni skup snimljen je kamerom Bumblebee2 [1] u blizini fakulteta. Slika 3.2 prikazuje referentnu putanju vožnje dobivenu GPS uređajem na Google Maps karti.

Slika 3.1: Bumblebee2 stereo kamera

Kamere su međusobno udaljene 0.12 m te daju sive slike rezolucije 640x480. Kamera se spaja preko *firewire* priključka, a slike su pribavljene korištenjem biblioteke PointGrey FlyCapture2. Biblioteke je konfigurirana tako da stereo parove slika ne sprema u zasebne slike već u jednu datoteku formata pgm (engl. *portable graymap*), tako da je svaki piksel predstavljen sa 16 bitova. Prvih 8 bitova određuju sivu razinu piksela lijeve slike, a zadnjih 8 bitova piksel desne slike (engl. *byte-interleaved format*). Kako bi se lijeva i desna slika razdvojile u zasebne datoteke za lakše korištenje u budućnosti, napisan je program koji razdvaja stereo png sliku u dvije png slike i nalazi u izvornom kodu na lokaciji `src/image_tools/split_stereo_pgm.cpp`. Da bi se proces automatizirao za proizvoljan broj slika napisana je skripta koja poziva program za razdvajanje za sve slike u zadanom direktoriju i nalazi se u izvornom kodu na lokaciji `src/scripts/split_stereo_pgm/split_images.py`.

[1] http://ww2.ptgrey.com/stereo-vision/bumblebee-2



Slika 3.2: Putanja vožnje

Slika 3.3: Lijeva i desna slika iz ispitne snimke Bumblebee

GPS mjerenja imamo u svakoj sekundi što znači da je frekvencija pribavljanja GPS koordinata 1 Hz, a u svakoj stereo slici pribavljenoj iz kamere nalazi se zapisana vrijednost internog brojača kamere (engl. *timestamp*) i to u prvom pikselu lijeve i desne slike što znači da se za spremanje broja ciklusa koriste 2 okteta. Na slici 3.3 prikazani su primjeri slika iz ove ispitne snimke.

3.2. Ispitna snimka KITTI

Skup ispitnih snimki za evaluaciju vizualne odometrije pod akronimom KITTI objavio je Karlsruhe Institute of Technology [4][2]. Skup se sastoji od više vožnji snimljenih pomoću dvije Flea2 kamere [3] montirane na krov automobila za koje, pored ostalih senzorskih mjerenja, imamo referentne položaje u trenutku svakog video okvira zapisane u GPS geografskim koordinatama i koordinatama lokalnog Euklidskog koordinatnog sustava. Slike su već rektificirane s danim parametrima kalibracije i rektifikacije. Dane slike su tijekom rektifikacije još i izrezane u visini da bi se smanjila složenost obrade. Razlog zašto su samo izrezane u visini je taj što gornji dio slike ionako dobrim dijelom sadrži nebo gdje su najmanje šanse algoritam detekcije uspije pronaći kvalitetne značajke pa se tako izgubilo najmanje vrijednih informacija u slici. Bitno je još spomenuti da je rezolucija slika ovog skupa jednaka 1241x376, a da je razmak između kamera jednak 0.537 m.

Slika 3.4: Lijeva i desna slika ispitnog skupa KITTI

3.3. Umjetni ispitni skup podataka

Umjetni ispitni skup napravljen je u sklopu ovog rada kao pomoćni alat za testiranje sustava. Skup se generira pomoću skripti napisanih u Pythonu (NumPy) koje

[2]http://www.cvlibs.net/datasets/kitti/eval_odometry.php

[3]http://www.ptgrey.com/products/flea2/flea2_firewire_camera.asp

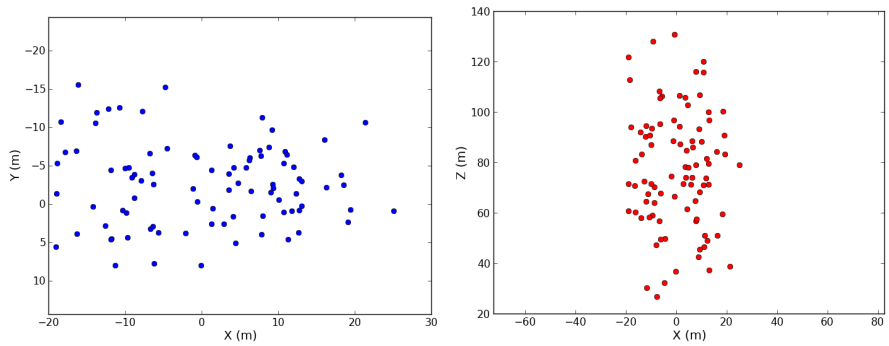
modeliraju rektificirani stereo sustav kamera proizvoljnih intrinsičnih i ekstrinsičnih parametara te simuliraju realne uvjete pojavljivanja značajki ovisno o primjeni.

Ulazni podaci za generator umjetnog skupa su kamera matrica rektificiranog stereo sustava, odmak između kamera te ekstrinsični parametri kretanja stereo sustava.

Program najprije generira skup 3D točaka u svijetu koje će se projicirati u značajke slike. Točke se generiraju Gaussovom razdiobom oko zadanih centroida s danim vrijednostima disperzije σ za svaki centroid i svaku dimenziju. Pozicije centroida i vrijednosti disperzije odabrane su tako da distribucija opisuje mjesta pojavljivanja značajki u realnoj vožnji na otvorenom. Slika 3.5 prikazuje distribuciju generiranih točaka na XY i XZ osima. Y os je invertirana jer je gornji lijevi piksel u ishodištu koordinatnog sustava pa negativna strana gleda prema gore. Prisjetimo se translacije slike pomoću parametra centralne točke (slika 2.2). Ako bolje pogledamo tu sliku vidimo da je jedini način da postavimo gornji lijevi piksel slike ishodište sustava ako postavimo pozitivnu stranu Y osi prema dolje.

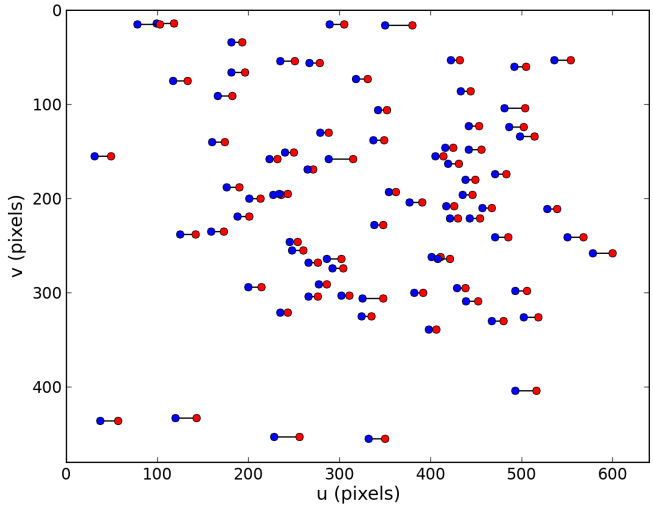
Generator sada ovako generirane točke projicira na sliku za trenutni vremenski okvir bez te zatim točke transformira za ekstrinsične parametre matrice pomaka kamere $\mathbf{Rt}$ u idućem okviru i takve ih iznova projicira na sliku koja predstavlja izgled istih točaka nakon pomaka kamere. Nakon ovog koraka lokacije značajki se spremaju u datoteku te se iznova generira novi skup 3D točaka s istom razdiobom i postupak se ponavlja za idući inkrementalni pomak kamere.

Na kraju su cikličke korespondencije značajki spremljene u zasebne datoteke za svaki inkrementalni pomak spremne da bude poslane na ulaz algoritma vizualne odometrije u svrhu evaluacije. Slika 3.6 prikazuje primjer stvorene lijeve i desne slike u jednom vremenskom okviru s pripadajućim disparitetima značajki. Implementacija se nalazi na lokaciji `/scripts/stereo_model/`.



(a) XY osi **(b)** XZ osi

Slika 3.5: Distribucija generiranih točaka u 3D prostoru

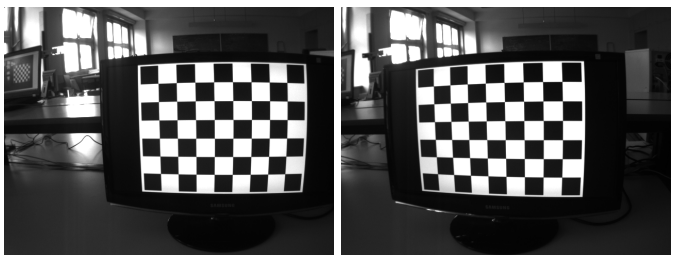


Slika 3.6: Spojene umjetna lijeva (crvena) i desna (plava) slika

3.4. Kalibracijski skup

U sklopu rada snimljena su i dva nova kalibracijska skupa za Bumblebee2 kameru, na jednom je kalibracijski uzorak bio isprintan na A4 papiru dok je u slučaju drugog kalibracijski uzorak prikazan na LCD monitoru. Budući da se u eksperimentima kalibracija na skupu snimljenom pomoću LCD monitora pokazala preciznijom, na slici 3.7 prikazane su lijeva i desna slika jednog primjera iz kalibracijskog skupa u kojem ih je ukupno 20.

Model monitora na kojem je prikazan kalibracijski uzorak je Samsung SyncMaster 2233SW. Rezolucija je 1920x1080 (HD), a u službenoj tehničkoj dokumentaciji stoji da su dimenzije piksela 0.248mm x 0.248mm[4]. S obzirom na to da je izmjereno kako je veličina jednog kvadrata uzorka bila jednaka 150 piksela, realna veličina kvadrata u metrima je tada $150 \cdot 0.248 = 37.2mm$



Slika 3.7: Lijeva i desna slika iz kalibracijskog skupa

[4]http://downloadcenter.samsung.com/content/UM/200810/20081024105827812/BN59-00803A-Eng.pdf

4. Programska izvedba i vanjske biblioteke

U sljedećim poglavljima opisana je implementacija postupaka ostvarenih tijekom ovog rada te korištene biblioteke.

4.1. Biblioteka OpenCV

Biblioteka OpenCV (engl. Open Computer Vision library[1] je biblioteka otvorenog koda (BSD licenca) koja zasad uključuje implementacije nekoliko stotina algoritama računalnog vida. Podržana je na svim značajnijim operacijskim sustavima: Linux, Windows, OSX, Android te iOS. Ima modularnu strukturu što znači da uključuje više dijeljenih i statičkih biblioteka. Neki od glavnih modula su:

- core - funkcionalnost jezgre, osnovne strukture i algoritmi.
- imgproc - algoritmi procesiranja slika.
- video - obrada videa, estimacija gibanja
- calib3d - algoritmi geometrije više pogleda, kalibracija (monokularna i stereo).
- features2d - detektori značajki.
- highgui - korisničko grafičko sučelje za prikaz slika i rezultata.
- ml - modul za strojno učenje.
- gpu - sadrži neke od algoritama iz različitih modula implementirane za izvođenje na GPU.
- ...

Instalacija OpenCV biblioteke na Arch Linux operacijskom sustavu je vrlo jednostavna budući da je samo potrebno instalirati paket opencv iz službenog repozitorija.

[1] http://opencv.org

Dodatno se može instalirati i paket opencv-samples koji sadrži programske primjere korištenja različitih metoda. Dovoljno je izvršiti naredbu:

```
sudo pacman -S opencv opencv-samples
```

Sva zaglavlja se nalaze u direktoriju `/usr/include/opencv2/`, a statičke i dinamičke biblioteke u direktoriju `/usr/lib/`.

4.2. Kalibracija

Kalibracija je izvedena pomoću OpenCV modula `calib3d`[2]. Taj modul podržava kalibraciju monokularne, ali i stereo kamere. Implementacija se nalazi na lokaciji `src /stereo_calib/calibrator.cpp`. U nastavku je dan popis korištenih metoda iz modula i objašnjenja parametara.

```
bool findChessboardCorners(InputArray image, Size patternSize,
         OutputArray corners, int flags=CALIB_CB_ADAPTIVE_THRESH +
         CALIB_CB_NORMALIZE_IMAGE)
```

Funkcija pronalazi kutove u kalibracijskom uzorku.
`image` - ulazna kalibracijska slika uzorka šahovske ploče.
`patternSize` - dimenzije kutova u šahovskom uzorku (broj kutova u retku, broj kutova u stupcu).
`corners` - izlazni niz pronađenih kutova.
`flags` - kontrolne zastavice.

- `CALIB_CB_NORMALIZE_IMAGE` - provedi normalizaciju histograma slike.
- `CALIB_CB_ADAPTIVE_THRESH` - koristi adaptivnu binarizaciju slike umjesto fiksnog praga.

```
void cornerSubPix(InputArray image, InputOutputArray corners,
          Size winSize, Size zeroZone, TermCriteria criteria)
```

Funkcija koja poboljšava preciznost lokacija niza značajki kutova iz cjelobrojnih koordinata slike u potpikselske decimalne koordinate.
`image` - slika na kojoj su pronađene značajke.
`corners` - niz početno pronađenih lokacija kutova u kojem će nakon izvršavanja funkcije biti spremljene pronađene preciznije lokacije.
`winSize` - veličina prozora oko kuta u kojem se pretražuju gradijenti prilikom izračuna. Korištena veličina (11,11).

[2] http://docs.opencv.org/trunk/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

zeroZone - ovaj parametar nije korišten.
criteria - TermCriteria(CV_TERMCRIT_ITER+CV_TERMCRIT_EPS, 30, 0.01). Kriterij zaustavljanja algoritma je kada dosegne 30 iteracija ili kada greška padne ispod 0.01.

```
double stereoCalibrate(InputArrayOfArrays objectPoints,
    InputArrayOfArrays imagePoints1, InputArrayOfArrays imagePoints2,
    InputOutputArray cameraMatrix1, InputOutputArray distCoeffs1,
    InputOutputArray cameraMatrix2, InputOutputArray distCoeffs2,
    Size imageSize, OutputArray R, OutputArray T, OutputArray E,
    OutputArray F, TermCriteria criteria, int flags)
```

Funkcija provodi kalibraciju stereo sustava i vraća pronađene intrinsične i ekstrinsične parametre.
Ulazni parametri:
objectPoints - 2D vektor koordinata točaka kalibracijskog uzorka u ravnini Z=0 u realnoj veličini kvadrata kalibracijskog uzorka. Prva dimenzija vektora su uzorci po slikama iz kalibracijskog skupa što omogućuje da se u skupu snime i uzorci različitih dimenzija.
imagePoints1, imagePoints2 - 2D vektori pronađenih kutova uzorka u lijevim, odnosno desnim slikama.
imageSize - dimenzije ulaznih slika.
criteria - kriterij zaustavljanja algoritma.
flags - kontrolne zastavice.
Izlazni parametri:
cameraMatrix1, cameraMatrix2 - izlazne kamere matrice lijeve i desne kamere.
distCoeffs1, distCoeffs2 - izlazni parametri distorzije lijeve i desne kamere.
R, T - ekstrinsični parametri rotacije i translacije između dvaju kamera.
E, F - esencijalna i fundamentalna matrica.

Naknadno je preciznost kalibracije poboljšana po savjetu iz dokumentacije gdje stoji da metoda stereoCalibrate, zbog visoke dimenzionalnosti prostora parametara te šuma u ulaznim podacima, postavlja određene pretpostavke u prostoru pretraživanja i zbog toga može divergirati od točnog rješenja. Kako bi se ta mogućnost izbjegla potrebno je najprije zasebno kalibrirati svaku kameru pomoću metode calibrateCamera i dobivene intrinsične parametre poslati kao ulazne parametre metodi stereoCalibrate uz postavljenu zastavicu CV_CALIB_FIX_INTRINSIC koja će algoritmu dati uputu da fiksira intrinsične parametre pošto su već unaprijed izračunati pa

će sada metoda optimirati samo ekstrinsične parametre $\mathbf{R}$ i $\mathbf{T}$. Ova preciznija implementacija kalibracije nalazi se na lokacijama: `src/stereo_calib/calibrator_mono.cpp` za kalibraciju svake kamere pojedinačno te `src/stereo_calib/calibrator_stereo_extr.cpp` za stereo kalibraciju koja na ulaz prima rezultate pojedinačnih kalibracija.

```
double calibrateCamera(InputArrayOfArrays objectPoints,
         InputArrayOfArrays imagePoints, Size imageSize,
         InputOutputArray cameraMatrix, InputOutputArray distCoeffs,
         OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs,
         int flags=0, TermCriteria criteria=TermCriteria(
         TermCriteria::COUNT + TermCriteria::EPS, 30, DBL_EPSILON))
```

Funkcija provodi kalibraciju kamere i vraća pronađene intrinsične parametre te ekstrinsične parametre odnosa položaja kamere i kalibracijskog uzorka za svaku sliku iz skupa.

Ulazni parametri:

`objectPoints` - 2D vektor koordinata točaka kalibracijskog uzorka u ravnini Z=0 u realnoj veličini kvadrata kalibracijskog uzorka. Prva dimenzija vektora su uzorci po slikama iz kalibracijskog skupa što omogućuje da se u skupu snime i uzorci različitih dimenzija.

`imagePoints` - 2D vektor pronađenih kutova uzorka u slikama kalibracijskog skupa.

`imageSize` - dimenzije ulaznih slika.

`criteria` - kriterij zaustavljanja algoritma.

`flags` - kontrolne zastavice.

Izlazni parametri:

`cameraMatrix` - izračunata matrica kamere.

`distCoeffs` - izlazni parametri distorzije uzrokovane lećom kamere.

`rvecs` - vektor rotacija koje prebacuju točke uzorka za pojedinu sliku iz lokalnog koordinatnog sustava dane u `objectPoints` u koordinatni sustav svijeta u kojem je kamera u ishodištu i gleda u pozitivnom smjeru Z osi.

`tvecs` - analogno kao u slučaju `rvecs`, da bi se prebacilo točke u koordinatni sustav svijeta (kamere) nakon rotacije potrebno je provesti i translaciju.

Dok su sve ulazne datoteke s popisima slika u XML formatu zapisa, izlazne datoteke su zapisane u YAML formatu koji je vizualno čitkiji kada je potrebno prikazati veći broj različitih parametara. U nastavku su prikazani rezultati kalibracije.

Rezultati kalibracije lijeve kamere:

```
%YAML:1.0
```

```
calibration_time: "Sun Jul  7 01:30:08 2013"
image_width: 640
image_height: 480
board_width: 8
board_height: 6
square_size: 3.7200000137090683e-02
flags: 0
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 5.2833185757717388e+02, 0., 3.3154718872578667e+02, 0.,
       5.2805898512142016e+02, 2.4715286799886439e+02, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 5
   cols: 1
   dt: d
   data: [ -3.6741362671312772e-01, 2.2325453986826649e-01,
       6.4976303880169897e-05, -4.7243220861350396e-05,
       -1.0126739516793781e-01 ]
avg_reprojection_error: 8.8213250074813523e-02
```

Rezultati kalibracije desne kamere:

```
%YAML:1.0
calibration_time: "Sun Jul  7 01:29:47 2013"
image_width: 640
image_height: 480
board_width: 8
board_height: 6
square_size: 3.7200000137090683e-02
flags: 0
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 5.2616050305656336e+02, 0., 3.1944493972372129e+02, 0.,
       5.2609980650911166e+02, 2.4690154538244661e+02, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 5
   cols: 1
   dt: d
   data: [ -3.5653234977247711e-01, 1.6877754145751542e-01,
       3.2215731574417495e-04, -4.3713604548299874e-04,
```

```
        -1.4171227738545872e-02 ]
avg_reprojection_error: 1.2202076142961658e-01
```

Rezultati stereo kalibracije:

```
R: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.9997610875271337e-01, -1.5246967558597516e-03,
       6.7421972371165002e-03, 1.4983412765048867e-03,
       9.9999122417168340e-01, 3.9123590117796725e-03,
       -6.7481032298440970e-03, -3.9021634282283284e-03,
       9.9996961765014580e-01 ]
T: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ -1.1974079489358994e-01, 8.3406748087582459e-04,
       2.1423614889845281e-04 ]
```

4.3. Rektifikacija

Rektifikacija je ostvarena pomoću metoda iz OpenCV modula `calib3d` i `imgproc` koje su opisane u nastavku. Implementacija se nalazi na lokaciji `src/stereo_calib/rectificator.cpp` i provodi rektifikaciju skupa slika iz XML datoteke popisa dane kao ulazni parametar programa. Implementacija koristi Bouguetov algoritam rektifikacije koji je ostvaren u biblioteci OpenCV te korišten u ovom radu. Detaljan opis Bouguetovog algoritma iz OpenCV biblioteke može se pronaći u knjizi [1].

```
void stereoRectify(InputArray cameraMatrix1, InputArray distCoeffs1,
        InputArray cameraMatrix2, InputArray distCoeffs2,
        Size imageSize, InputArray R, InputArray T, OutputArray R1,
        OutputArray R2, OutputArray P1, OutputArray P2,
        OutputArray Q, int flags=CALIB_ZERO_DISPARITY,
        double alpha=-1, Size newImageSize=Size(),
        Rect* validPixROI1=0, Rect* validPixROI2=0)
```

Funkcija računa transformacijske (rotacijske) matrice R1 i R2 koje prebacuju koordinate u rektificirani sustav.

Ulazni parametri:

`cameraMatrix1, cameraMatrix2` - matrice lijeve i desne kamere prethodno dobivene

kalibracijom.
distCoeffs1, distCoeffs2 - parametri distorzije leće dobiveni kalibracijom.
imageSize - dimenzije ulaznih slika.
R, T - ekstrinsični parametri stereo sustava dobiveni kalibracijom.
Izlazni parametri:
R1, R2 - 3x3 matrice transformacije (rotacije) u rektificirani sustav za lijevu i desnu kameru.
P1, P2 - 3x4 projekcijske matrice u novom (rektificiranom) koordinatnom sustavu za lijevu i desnu kameru.
flags - kontrolne zastavice algoritma.
alpha - slobodni parametar skaliranja. alpha = 0 znači da će rektificirana slika biti izrezana u području interesa koji se skalira i pomiče tako da samo važeći pikseli ostanu vidljivi.
newImageSize - konačna dimenzija izlaznih rektificiranih slika.
validPixROI1, validPixROI2 - pravokutne regije interesa u lijevoj i desnoj rektificiranoj slici u kojima su sve vrijednosti piksela važeći pikseli originalne slike. Ako je alpha = 0 tada područja interesa odgovaraju području cijele slike.

```
void initUndistortRectifyMap(InputArray cameraMatrix,
          InputArray distCoeffs, InputArray R,
          InputArray newCameraMatrix, Size size, int m1type,
          OutputArray map1, OutputArray map2)
```

Funkcija izračunava pomoćne transformacijske mape za uklanjanje distorzije te rektifikaciju za jednu kameru.
Ulazni parametri:
cameraMatrix - matrica kamere prethodno dobivena kalibracijom.
distCoeffs - parametri distorzije leće dobiveni kalibracijom.
R - matrica transformacije piksela u rektificirani sustav (R1 ili R2 iz stereoRectify).
newCameraMatrix - projekcijske matrice kamere u rektificiranom sustavu (P1 ili P2 iz stereoRectify).
size - dimenzije izlazne slike.
m1type - tip podataka korišten za map1 i map2 (CV_16SC2 - signed short ili CV_32FC1 - float).
Izlazni parametri:
map1 - izlazna transformacijska mapa za x os.
map2 - izlazna transformacijska mapa za y os.

```
void remap(InputArray src, OutputArray dst, InputArray map1,
           InputArray map2, int interpolation)
```

Provodi generičku geometrijsku transformaciju slike.

Ulazni parametri:

`src` - ulazna slika.

`map1` - izlazna transformacijska mapa za x os

`map2` - izlazna transformacijska mapa za y os

`interpolation` - metoda interpolacije, koristi se linearna (`CV_INTER_LINEAR`).

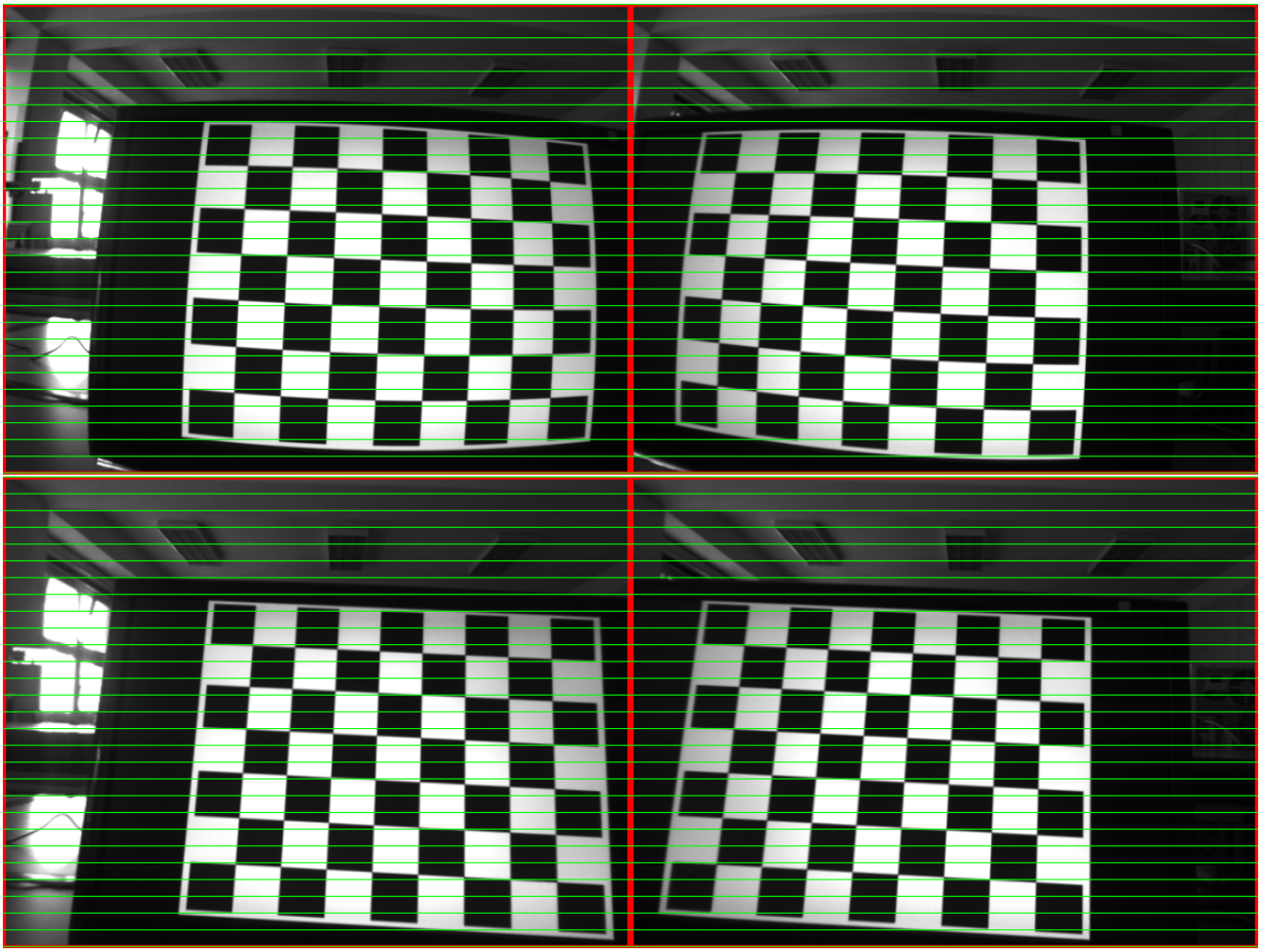
Izlazni parametri:

`dst` - izlazna slika.

U nastavku je dana izlazna datoteka programa za rektifikaciju s korištenim parametrima rektifikacije:

```
%YAML:1.0
R1: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.9995188654189981e-01, -8.4830429880356373e-03,
       4.9257063411204689e-03, 8.4733422377752336e-03,
       9.9996212652623540e-01, 1.9869535602215242e-03,
       -4.9423751999771743e-03, -1.9451207654235696e-03,
       9.9998589461681442e-01 ]
R2: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.9997414059591139e-01, -6.9654282245882127e-03,
       -1.7891196485706213e-03, 6.9618972215665020e-03,
       9.9997382066438212e-01, -1.9723014359652605e-03,
       1.8028107346963525e-03, 1.9597947663150036e-03,
       9.9999645453267938e-01 ]
P1: !!opencv-matrix
   rows: 3
   cols: 4
   dt: d
   data: [ 4.9121998717700109e+02, 0., 3.2886573791503906e+02, 0.,
      0.,
```

```
        4.9121998717700109e+02, 2.4943164825439453e+02, 0., 0., 0.,
           1.,
        0. ]
P2: !!opencv-matrix
   rows: 3
   cols: 4
   dt: d
   data: [ 4.9121998717700109e+02, 0., 3.2886573791503906e+02,
        -5.8820592797671075e+01, 0., 4.9121998717700109e+02,
        2.4943164825439453e+02, 0., 0., 0., 1., 0. ]
```



Slika 4.1: Kalibracijske slike prije i nakon rektifikacije

4.4. Biblioteka Libviso

Biblioteka Libviso na ulazu, osim rektificiranih slika, prima i sljedeće parametre:

```
param.calib.f;      // zarisna duljina rektificiranog sustava
param.calib.cu;     // osnovna tocka (u-koordinata) u pikselima
param.calib.cv;     // osnovna tocka (v-koordinata) u pikselima
param.base;         // udaljenost izmedu dviju kamera u metrima
```

Implementacija vizualne odometrije koja obuhvaća i ovu biblioteku nalazi se na lokaciji `src/stereo_odometry/`. U nastavku je dan primjer koda koji poziva biblioteku.

```
Matrix pose = Matrix::eye(4);
Mat img_left, img_right;
for(unsigned i = 0; i < imagelist.size(); i+=2) {
  img_left = imread(imagelist[i], CV_LOAD_IMAGE_GRAYSCALE);
  img_right = imread(imagelist[i+1], CV_LOAD_IMAGE_GRAYSCALE);
  if(viso.process(img_left.data, img_right.data, dims)) {
    pose = pose * Matrix::inv(viso.getMotion());
  }
}
```

Metoda `getMotion()` vraća transformaciju iz koordinatnog sustava kamere prethodnog vremenskog okvira u koordinatni sustav kamere trenutnog vremenskog okvira. Budući da ta transformacija zapravo opisuje kretanje točaka u odnosu na kameru da bismo dobili kretanje kamere u odnosu na točke potrebno je uzeti inverz te matrice. U varijabli `pose` sada će biti spremljena akumulirana transformacija kretanja kamere, odnosno ta transformacijska matrica će u i-tom vremenskom okviru prebacivati koordinate iz trenutnog lokalnog koordinatnog sustava u koordinatni sustav svijeta definiram položajem kamere u početnom vremenskom okviru. To znači da ako želimo saznati trenutni položaj kamere dovoljno je pomnožiti matricu `pose` s točkom ishodišta $[0, 0, 0, 1]^T$ budući da je u trenutnom lokalnom koordinatnom sustavu položaj kamere uvijek u ishodištu.

4.5. Praćenje značajki

U sklopu rada osmišljena su i dva sučelja za praćenje, jedno za monokularno praćenje te jedno za stereo praćenje. Zatim su iz tih sučelja izvedene različite implementacije praćenja. Implementacija se nalazi u direktoriju `src/tracker/`. Sučelje za monokularno praćenje naziva je `TrackerBase` i najvažnije metode koje definira su `config()` za konfiguraciju parametara algoritma, `init()` za početnu inicijalizaciju algoritma u početnom vremenskom okviru te `track()` za provođenje postupka praćenja u trenutnom vremenskom okviru. Iz ovog razreda izvedena su dva konkretna postupka praćenja,

jedan je implementacija Kanade-Lucas-Tomasi[3] praćenja iz rada [11] koja se nalazi u `TrackerBrich` razredu, dok je druga implementacija Lucas-Kanade[4] koju implementira OpenCV metoda `calcOpticalFlowPyrLK()`[5]. i nalazi se u `TrackerOpencv` razredu.

```
class TrackerBase {
public:
   struct Point {
      double x_;
      double y_;
      Point(double x=-1,double y=-1): x_(x), y_(y) {}
   };
   struct FeatureInfo {
      Point curr_;
      Point prev_;
      int age_;
      int status_;
      FeatureInfo(): age_(-1), status_(0) {}
   };
public:
   virtual ~TrackerBase() {}
   virtual void config(std::string conf) = 0;
   virtual int init(const TrackerImage& img) = 0;
   virtual int track(const TrackerImage& img) = 0;
   virtual std::string getConfig() = 0;
   virtual std::string getConfigDocs() = 0;
   virtual int countTracked() = 0;
   virtual int countFeatures() = 0;
   virtual FeatureInfo& feature(int i) = 0;
};
```

`TrackerImage` je jednostavan razred koji sadrži sliku. Ostvaren je tako da omogući proizvoljan broj referenci na istu sliku. Zato koristi brojač referenci kako bi znao očistiti zauzetu memoriju u trenutku kad zadnja referenca izađe iz dosega.

```
class TrackerImage {
public:
   TrackerImage(int rows=16, int cols=16);
   TrackerImage(uint8_t* data, int rows, int cols);
   TrackerImage(const TrackerImage& other);
   TrackerImage(TrackerImage&& other);
```

[3] http://www.ces.clemson.edu/~stb/klt/

[4] http://robots.stanford.edu/cs223b04/algo_tracking.pdf

[5] http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowpyrlk

```
   ~TrackerImage();
   TrackerImage& operator=(TrackerImage other);
   uint8_t& operator()(int row, int col);
   uint8_t operator()(int row, int col) const;
   void dealloc();
   void alloc(int rows, int cols);
public:
   int rows_;
   int cols_;
   int szBits_;
   uint8_t* data_;
   int* refcount_;
};
```

Sučelje za praćenje stereo značajki naziva se `StereoTrackerBase` i bitnije metode koje definira su `init()` i `track()` koje su analogne prethodnom sučelju samo pokrivaju stereo slučaj, stoga dobivaju dvije slike na ulazu.

```
class StereoTrackerBase
{
public:
   struct Point {
      double x_;
      double y_;
      Point(double x=-1,double y=-1): x_(x), y_(y) {}
   };
   struct FeatureInfo {
      Point curr_;
      Point prev_;
      int age_;
      int status_;
      FeatureInfo(): age_(-1), status_(0) {}
   };
public:
   virtual ~StereoTrackerBase() {}
   virtual void init(TrackerImage& img_left,
                     TrackerImage& img_right) = 0;
   virtual void trackTemporal(TrackerImage& img_left,
                              TrackerImage& img_right) = 0;
   virtual int countTracked() = 0;
   virtual int countFeatures() = 0;
   virtual FeatureInfo& featureLeft(int i) = 0;
   virtual FeatureInfo& featureRight(int i) = 0;
protected:
```

```
    virtual void matchEpipolar() = 0;
    virtual void checkEpipolar() = 0;
};
```

Iz ovog sučelja su zasad izvedena dva razreda. Prvi razred koji je izveden iz `StereoTrackerBase` jest `StereoTrackerLibviso` i on je napravljen kako bi se biblioteka Libviso proširila za rad s proizvoljnom implementacijom metode za praćenje značajki upravo preko sučelja `StereoTrackerBase`. Konkretno ovaj razred samo služi kao omotač za interni razred za praćenje značajki kojeg Libviso koristi. U nastavku je dan primjer kako se nakon izmjene instancira novi objekt za stereo odometriju.

```
StereoTrackerBase* tracker = new StereoTrackerLibviso(param);
VisualOdometryStereo viso(param, tracker);
```

Razlog zašto je Libviso proširen za rad s proizvoljnim algoritmom za praćenje značajki jest testiranje. Stoga se drugi razred koji izvodi `StereoTrackerBase` sučelje naziva `StereoTrackerSim` i zapravo jedini posao koji obavlja jest da simulira algoritam za praćenje tako što na izlaz daje značajke prethodno opisanog umjetnog ispitnog skupa za testiranje u trenutnom vremenskom okviru. Algoritam jednostavno svakim pozivanjem `track()` metode učitava skup značajki idućeg okvira iz datoteke umjetnog ispitnog skupa. U nastavku je prikazano kako se jednostavno Libviso može konfigurirati za rad s umjetnim ispitnim skupom.

```
string infolder = "stereo_model/points_170m_bbcam_base_0.03/";
string inlist = "datasets/stereo_model/stereosim_170m.xml";
StereoTrackerBase* tracker = new StereoTrackerSim(infolder, inlist);
VisualOdometryStereo viso(param, tracker);
```

4.6. Monokularni SBA

U direktoriju `src/sba_mono/` nalazi se implementacija Sparse Bundle Adjustment[6] algoritma koji trenutno radi samo za monokularni slučaj kamere. U planu je proširivanje ove implementacije za rad sa stereo sustavom i povezivanje SBA optimizacije s trenutnim postupkom vizualne odometrije. Prednost SBA metode naspram trenutne jest da optimira položaje kamere minimizacijom reprojekcijske pogreške kroz više okvira. Da bismo uopće mogli računati reprojekcijsku pogrešku kroz više okvira potrebne su nam i korespondencije značajki u više okvira. Budući da Libviso značajke prati samo između svaka dva susjedna vremenska okvira zasebno, koristit će se prethodno opisano

[6] http://users.ics.forth.gr/~lourakis/sba/

praćenje značajki kroz više okvira KLT metodom čija je dodatna prednost naspram trenutnog praćenja koje koristi Libviso da radi s potpikselskom preciznošću.

4.7. Organizacija koda

Izvorni kod te kalibracijski i ispitni skupovi na kojima je testiran nalaze se na DVD-u priloženom u radu. Glavni dio implementacije napisan je u C++ jeziku dok su pomoćni moduli napisani u jezicima Python[7] i Matlab. Primjeri pozivanja izvršnih datoteka dani su u izvornom kodu.

Sav C++ kod je povezan pomoću CMake[8] sustava za izgradnju izvršnih datoteka. Preporučuje se najprije napraviti jedan direktorij za izvršne datoteke i u njemu stvoriti dva zasebna direktorija, jedan za *release*, a drugi za *debug* verziju. Tada se pomoću cmake naredbe jednostavno generiraju Makefile skripte za kompajliranje *release* ili *debug* verzija u stvorenim direktorijima. Sljedeći redoslijed naredbi to slikovitije opisuje:

```
mkdir build build/debug build/release
cd build/debug
cmake src_dir -DCMAKE_BUILD_TYPE=Debug
cd ../release
cmake src_dir -DCMAKE_BUILD_TYPE=Release
```

gdje je `src_dir` put do direktorija s kodom gdje se nalazi CMakeLists.txt konfiguracijska datoteka za CMake. Nakon toga sve je spremno i kompajliranje izvršnih datoteka se može jednostavno pozivati sa `make` naredbom u `release` ili `debug` direktorijima.

[7] uz Python su korištene NumPy i Matplotlib biblioteke

[8] http://www.cmake.org

5. Eksperimentalni rezultati

U ovom poglavlju biti će prikazani rezultati provedenih eksperimenata tijekom rada. Budući da je sustav odometrije testiran na tri različita ispitna skupa rezultati su tako i podijeljeni.

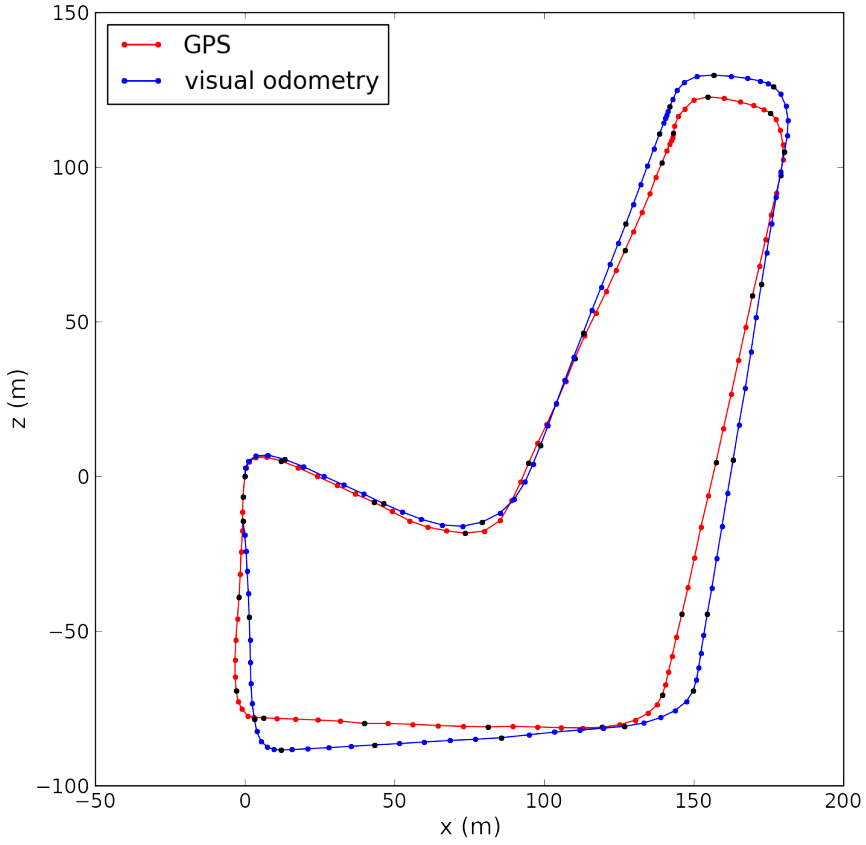
5.1. Ispitna snimka Bumblebee

Prvi ispitni skup na kojem su testiranja provedena je onaj snimljen kamerom Bumblebee predstavljen ranije u poglavlju o ispitnim skupovima. Rezultati odometrije dobiveni pokretanjem biblioteke Libviso nad ovom snimkom i njihova usporedba s referentnim GPS mjerenjem dani su na slici 5.1. Na slici je jasno da postoji vidljiva pogreška i u translaciji i u rotaciji. Dva su moguća glavna razloga za to. Jedan je ako kalibracija nije dovoljno precizna. Naime iako je kamera kalibrirana na uzorku prikazanom na LCD monitoru što su dosta idealni uvjeti, ostaje problem toga da je monitor mali i uzorak se mora snimati izbliza te je tu sasvim moguć slučaj da fokus napravi neželjeni utjecaj i malo zamuti sliku pri toj blizini. Često se kamere s fiksiranim fokusom namještaju tako da se u razuman fokus dovedu udaljenosti od par metara pa do beskonačnosti[1]. Budući da je kamera od monitora pri snimanju kalibracijskog uzorka bila odmaknuta svega otprilike pola metra, izgubljena oštrina slike svakako može biti jedan od razloga nepreciznosti u kalibraciji. Drugi mogući razlog je pogreška triangulacije koja bi u slučaju kamere Bumblebee mogla biti jednostavno prevelika da bi rezultati za ovaj postupak vizualne odometrije bili bolji.

5.1.1. Sinkronizacija odometrije i GPS-a

Problem koji se javio kod ove ispitne snimke je taj da u početku snimanja podaci iz kamere nisu bili sinkronizirani s podacima iz GPS-a. Stoga kako bi mogli uspoređivati rezultate odometrije s GPS podacima bilo je potrebno najprije provesti njihovu među-

[1] http://en.wikipedia.org/wiki/Hyperfocal_distance



Slika 5.1: Usporedba odometrije s GPS podacima

sobnu sinkronizaciju. Implementacija sinkronizacije ostvarena je u Pythonu i nalazi se na lokaciji `src/scripts/gps_sync/map_gps_to_vodometry.py` U nastavku je prikazan popis pribavljenih podataka iz kamere Bumblebee te iz GPS uređaja.
GPS podaci:

- trajanje vožnje: 111 s, frekvencija dohvata: 1 Hz

Bumblebee:

- *timestamp* zapisan u pikselima
- broj okvira: 2786, nakon poduzorkovanja: 929
- ukupan *timestamp* vožnje: 57032 ciklusa
- trajanje jednog ciklusa = $\frac{111}{57032}s \approx 2ms$

Kako bismo podatke iz odometrije povezali s podacima iz GPS-a potrebno je interpolirati odometriju u vremenima GPS točaka. To se postiže tako da se najprije pronalaze okviri u kojima je broj akumuliranih ciklusa najbliži trenutku GPS-a, a potom se položaj kamere u trenutcima GPS očitanja određuje interpolacijom.

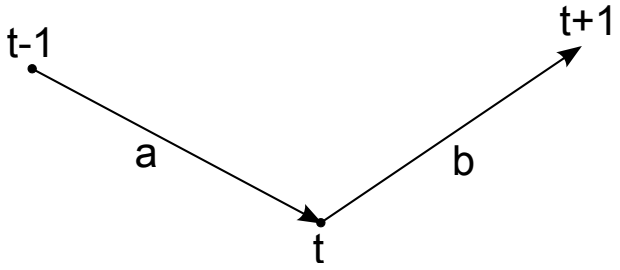
No da bi se interpolacija provela u pravim točkama najprije se mora provesti sinkronizacija početne točke GPS-a i odometrije. Da bismo to uspjeli prvo moramo odrediti mjere po kojima ćemo se ravnati prilikom sinkronizacije početne točke vizualne odometrije s GPS-om.

Prva ideja je provesti sinkronizaciju videa i GPS-a poklapanjem rotacijskih pomaka pa je mjera koju uvodimo kut Φ između vektora pomaka kroz tri uzastopne točke. Kut Φ ćemo računati pomoći geometrijske interpretacije vektorskog umnoška između vektora translacijskog pomaka kamere u prethodnom vremenskom okviru i vektora pomaka u trenutnom vremenskom okviru videa. Budući da trenutno put mapiramo na 2D kartu Z koordinatu možemo zanemarivati i postaviti na 0. Vektorski umnožak je odabran naspram skalarnog umnoška iz razloga što nam vektorski umnožak daje dodatnu informaciju o tome skreće li kamera u lijevo ili u desno. Ako je kut Φ pozitivan to znači da kamera skreće u lijevo, a ako je negativan tada kamera skreće u desno.

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_x \\ a_y \\ 0 \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ 0 \end{bmatrix} = \begin{vmatrix} i & j & k \\ a_x & a_y & 0 \\ b_x & b_y & 0 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ a_x b_y - b_x a_y \end{bmatrix} \tag{5.1}$$

$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\|\|\mathbf{b}\| \sin \Phi = a_x b_y - b_x a_y \tag{5.2}$$

$$\Phi = \arcsin\left(\frac{a_x b_y - b_x a_y}{\|\mathbf{a}\|\|\mathbf{b}\|}\right) \tag{5.3}$$

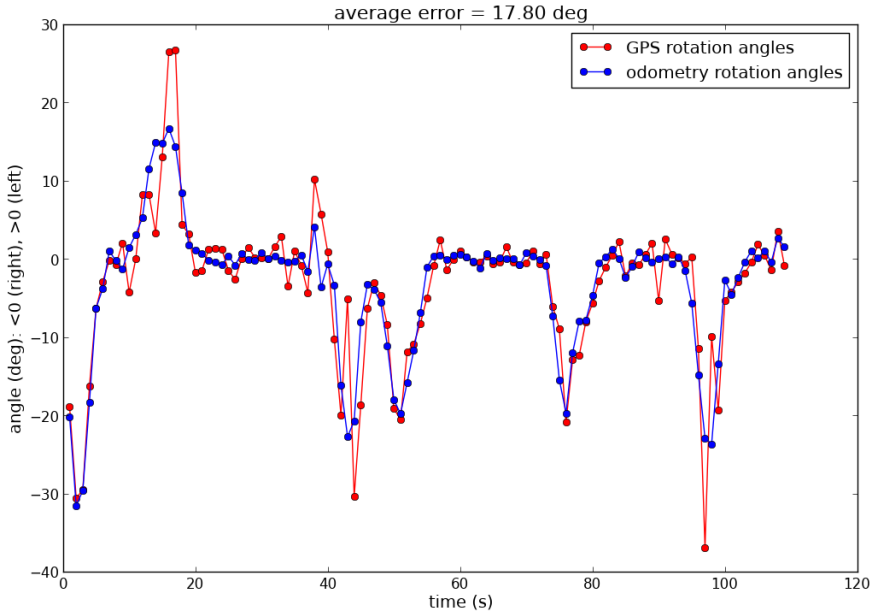


Sada možemo definirati funkciju pogreške koju je potrebno minimizirati kako bismo dobili vrijeme pomaka (engl. *offset*) početnog vremena odometrije u odnosu na UTC vrijeme iz GPS koje ćemo označiti sa $\hat{t_s}$.

$$t_s = \underset{t_s}{\operatorname{argmin}} \sum_{t=0}^{111} (\Phi_{vo}(t + t_s) - \Phi_{gps}(t))^2 \tag{5.4}$$

Slika 5.2 prikazuje usporedbu inkrementalnog zakreta dobivenog odometrijom i GPS-om nakon minimizacije ove funkcije, vidimo da se mjere dobro poklapaju. Definirajmo odmah i uprosječenu kvadratnu pogrešku ove mjere koju računamo kao:

$$E_{rot} = \frac{1}{N} \sum_{t=0}^{N} (\Phi_{vo}(t + \hat{t_s}) - \Phi_{gps}(t))^2 = 17.8 \tag{5.5}$$



Slika 5.2: Inkrementalni zakret Φ dobiven odometrijom i GPS-om

Prednost odabira kuta inkrementalnog zakreta je u tome da će takav pristup biti robustan na translacijske greške algoritma te ovisiti isključivo o rotacijskom gibanju. Druga ideja bi bila napraviti suprotno, uzeti mjeru inkrementalnog translacijskog pomaka Δs koja će ovisiti o preciznosti translacije, a biti robusna na rotacijsko gibanje kamere. Uvedimo stoga mjeru inkrementalnog translacijskog pomaka:

$$\Delta s(t) = \|\mathbf{p}(t) - \mathbf{p}(t-1)\| \tag{5.6}$$

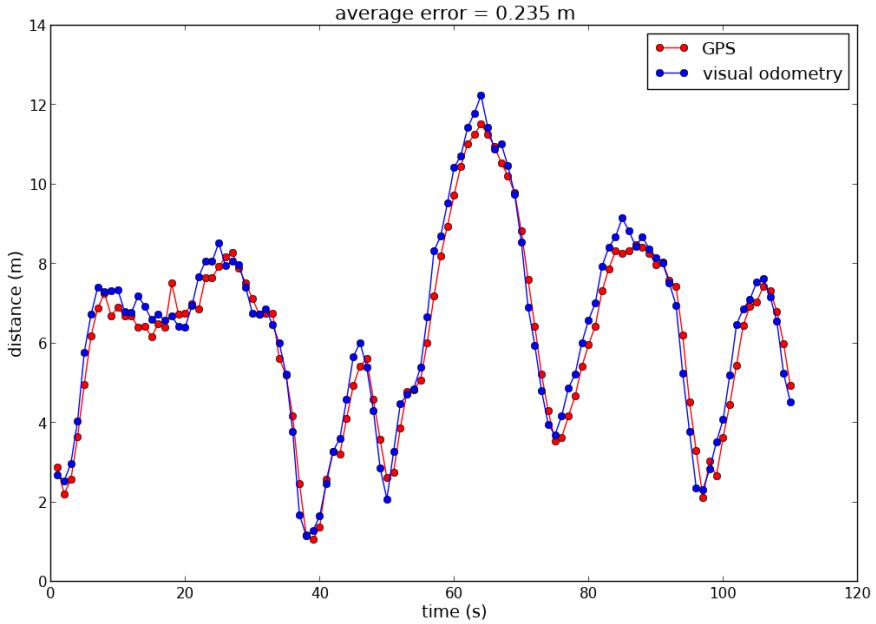
Vidimo da je ova mjera naprosto norma translacijskog vektora pomaka između dva vremenska okvira videa. Funkciju pogreške opet definiramo analogno kao u prethodnom slučaju:

$$\hat{t_s} = \underset{t_s}{\operatorname{argmin}} \sum_{t=0}^{111} (\Delta s_{vo}(t + t_s) - \Delta s_{gps}(t))^2 \tag{5.7}$$

Slika 5.3 sada prikazuje usporedbu inkrementalnog translacijskog pomaka dobivenog odometrijom i GPS-om nakon minimizacije ove funkcije gdje vidimo da se grafovi i u ovom slučaju dobro poklapaju. Definirajmo odmah i uprosječenu kvadratnu pogrešku ove mjere koju računamo kao:

$$E_{trans} = \frac{1}{N} \sum_{t=0}^{N} (\Delta s_{vo}(t + \hat{t_s}) - \Delta s_{gps}(t))^2 = 0.235 \tag{5.8}$$

Dodatni graf koji bi bilo zanimljivo vidjeti je graf pogreške vizualne odometrije u skali u pojedinom vremenskom okviru. Pogrešku u skali možemo definirati preko



Slika 5.3: Inkrementalni translacijski pomak: $\Delta s(t) = \|\mathbf{p}(t) - \mathbf{p}(t-1)\|$

prethodno definirane mjere inkrementalnog translacijskog pomaka kao omjer pomaka GPS-a i pomaka vizualne odometrije:

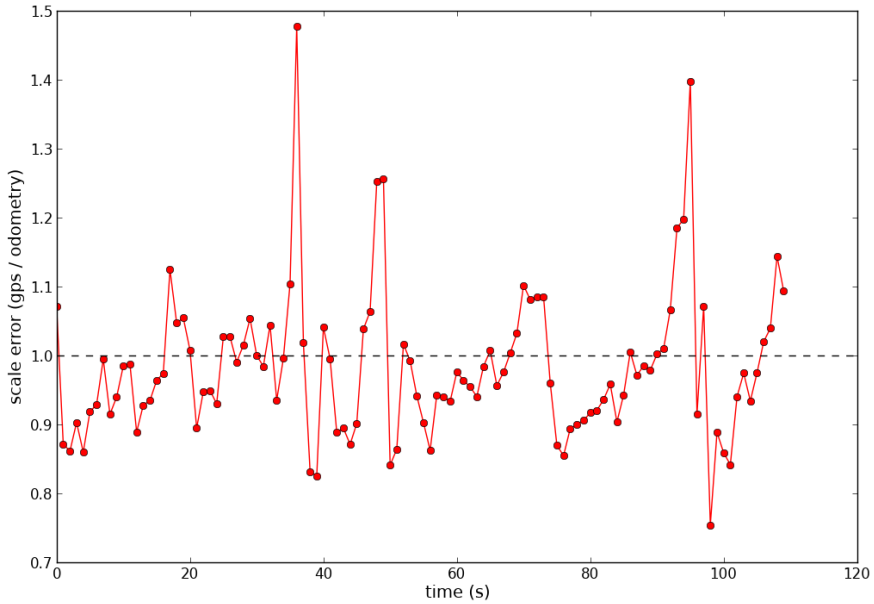
$$E_s(t) = \frac{\|\mathbf{p}_{gps}(t) - \mathbf{p}_{gps}(t-1)\|}{\|\mathbf{p}_{vo}(t) - \mathbf{p}_{vo}(t-1)\|} \tag{5.9}$$

Slika 5.4 prikazuje graf pogreške u skali $E_s(t)$. Na njemu možemo vidjeti da pogreška nije konstantna već varira i to ispod i iznad 1.0 što znači da je nekada translacijski pomak odometrije umanjen u odnosu na GPS, a nekada uvećan.

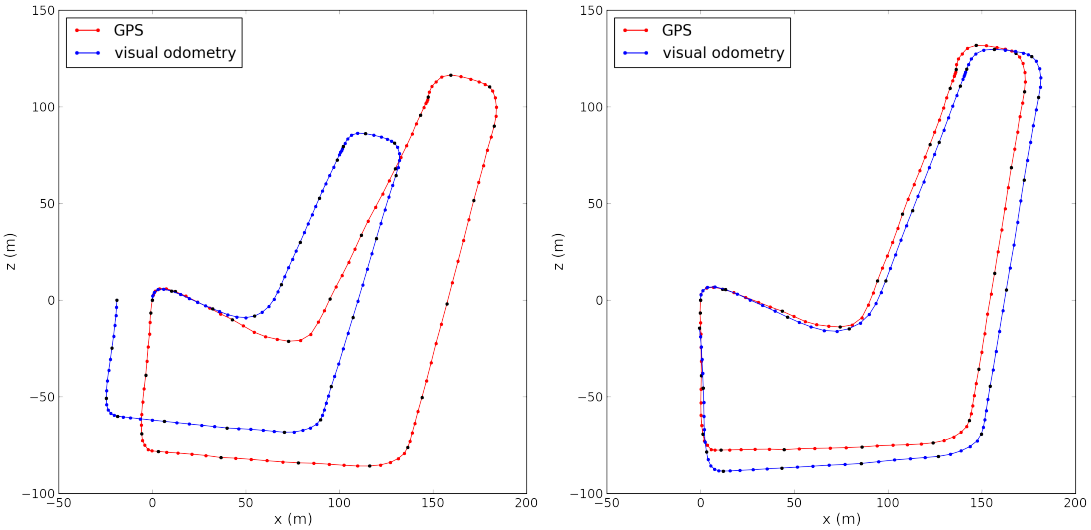
5.1.2. Usporedba stare i nove kalibracije

U početku rada za kalibraciju je korišten kalibracijski skup slika iz rada [10]. Budući da je bila riječ o neprecizno izvedenim slikama gdje je kalibracijski uzorak snimljen na neporavnatnom A4 papiru, snimljen je novi kalibracijski skup. Kalibracijski uzorak sniman je na LCD monitoru i preciznost kalibracije na novom skupu slika se znatno poboljšala. Koliko je kalibracija bitna za vizualnu odometriju, tj. preciznost triangulacije, govore nam sljedeći rezultati.

Na slici 5.5 je vidljivo da je sa starom kalibracijom translacijska pogreška bitno veća nego s novom. Na slici 5.6 dana je i usporedba inkrementalnih translacijskih pomaka prije i nakon unaprjeđenja kalibracije. Vidimo da je i razlika u uprosječenoj kvadratnoj pogrešci osjetna, $E_{trans} = 2.945$ u slučaju stare kalibracije, a $E_{trans} = 0.235$ u slučaju nove.



Slika 5.4: Pogreška u skali $E_s(t)$



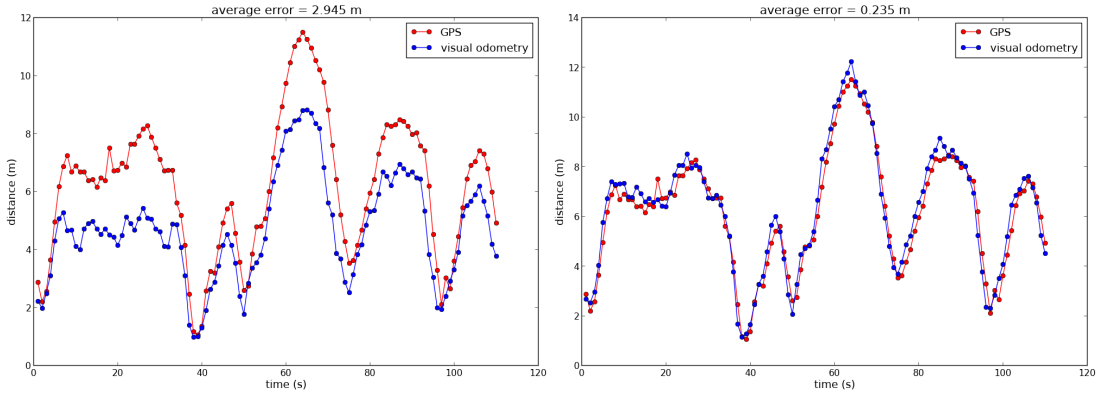
(a) Stara kalibracija

(b) Nova kalibracija

Slika 5.5: Usporedba rezultata vizualne odometrije

5.1.3. Modifikacija biblioteke Libviso2

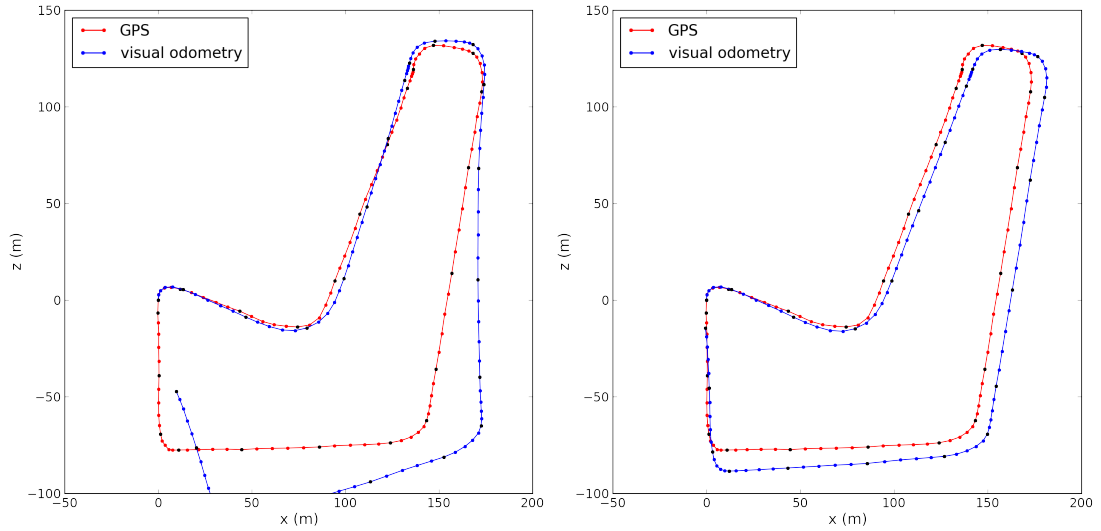
Dodatno unaprjeđenje je napravljeno ispravkom koda u Libviso2 biblioteci. Naime, ova biblioteka sve disparitete značajki manje od 1.0 zaokružuje na 1.0. To u slučaju Flea2 kamere na kojoj je izvorno testirana ne uzrokuje probleme jer su odmak između kamera i rezolucija same kamere dovoljno veliki da je slučaj kada je disparitet značajki jednak nuli jako rijetko događa. No u slučaju Bumblebee2 kamere, kojoj je



(a) Stara kalibracija **(b)** Nova kalibracija

Slika 5.6: Usporedba inkrementalnog translacijskog pomaka

odmak bitno manji i rezolucija slabija, takvo rješenje problema točaka u beskonačnosti nije dobro jer se takve točke pojavljuju puno češće i zbog manjeg odmaka među kamerama ravnina na koju se trianguliraju točke dispariteta 1 je dosta bliža kameri. Zbog toga je napravljena modifikacija u kojoj se takve točke zaokružuju na vrijednost 0.001 umjesto originalnih 1.0. Time smo ravninu u kojoj će se triangulirati te najudaljenije točke bitno udaljili od kamere. Usporedba rezultata dana je na slici 5.7 gdje se vidi osjetna razlika u pogreški rotacijskog gibanja.

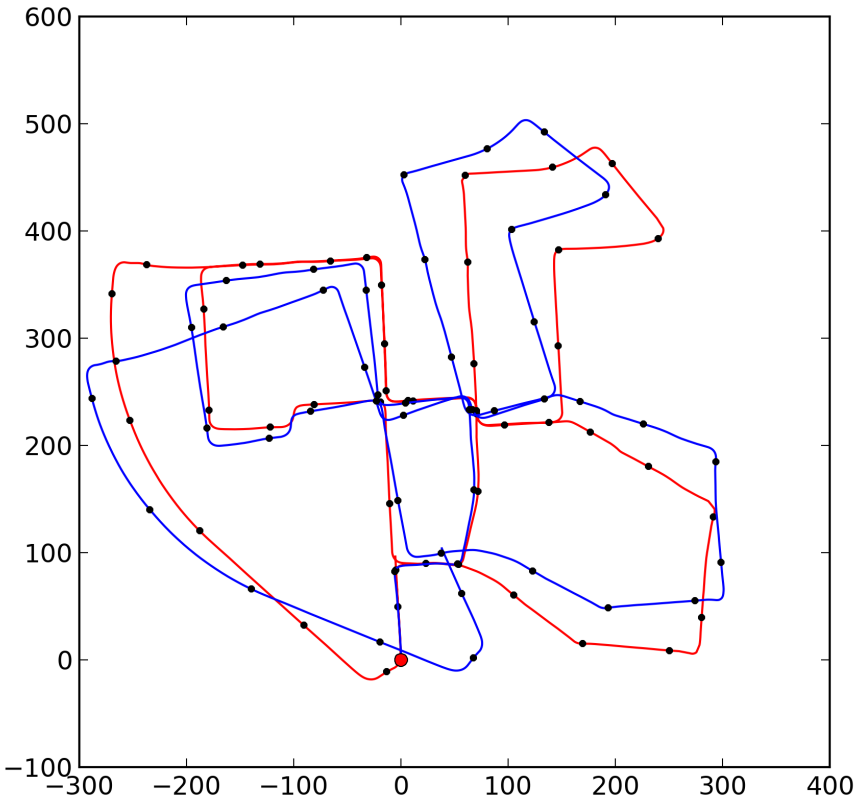


(a) Originalni Libviso **(b)** Nakon modifikacije

Slika 5.7: Usporedba rezultata vizualne odometrije

5.2. Ispitna snimka KITTI

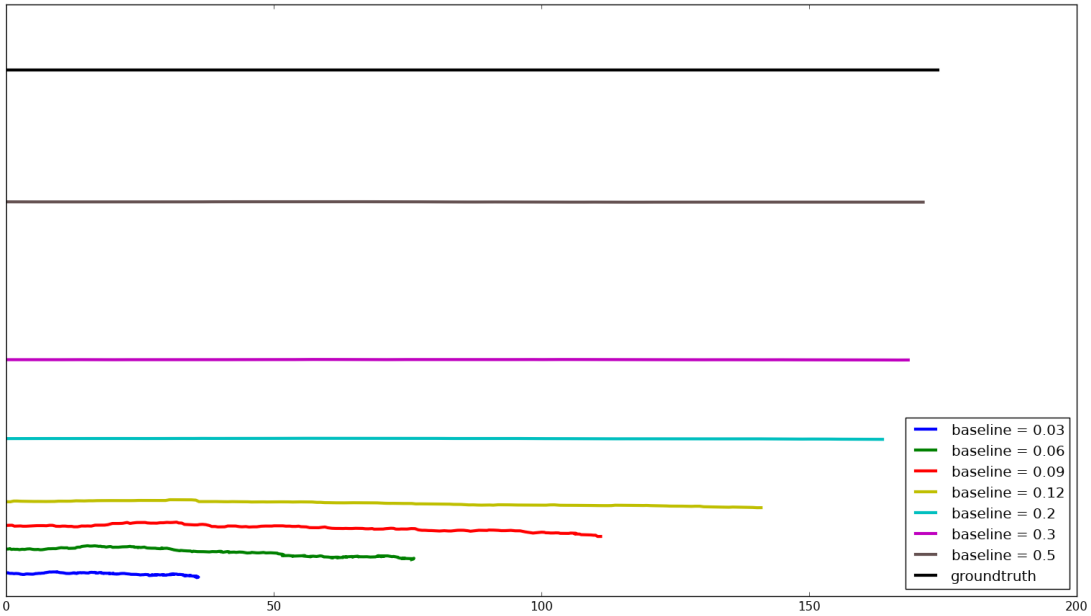
Vizualna odometrija testirana je i na jednoj od ispitnih snimki iz ranije opisanog KITTI ispitnog skupa. Rezultati su prikazani na slici 5.8. Crvenom bojom označen je istiniti put (engl. *groundtruth*) kojim se kamera kreće pribavljen preciznim GPS-om. Vožnja na kojoj je algoritam testiran u skupu KITTI numerirana je pod brojem 00. Put koji je dobiven vizualnom odometrijom je prikazan plavom bojom. Crvena točka predstavlja početnu lokaciju kamere. Na grafu se vidi da je najizraženiji problem akumulacije rotacijske pogreške. Akumuliranje pogreške je i glavni nedostatak vizualne odometrije općenito. Problem je u tome da, jednom kada algoritam pogriješi u nekom trenutku, ta pogreška se propagira čitav ostatak puta. Ovaj problem može predstavljati prepreku, ali i ne mora, ovisno o tome koliko dugačak put vožnje je potreban u visokoj preciznosti da bismo uspješno upotrijebili vizualnu odometriju u nekoj od brojnih primjena.



Slika 5.8: Rezultati Libviso algoritma na KITTI ispitnom skupu, crveno - *groundtruth*, plavo - odometrija

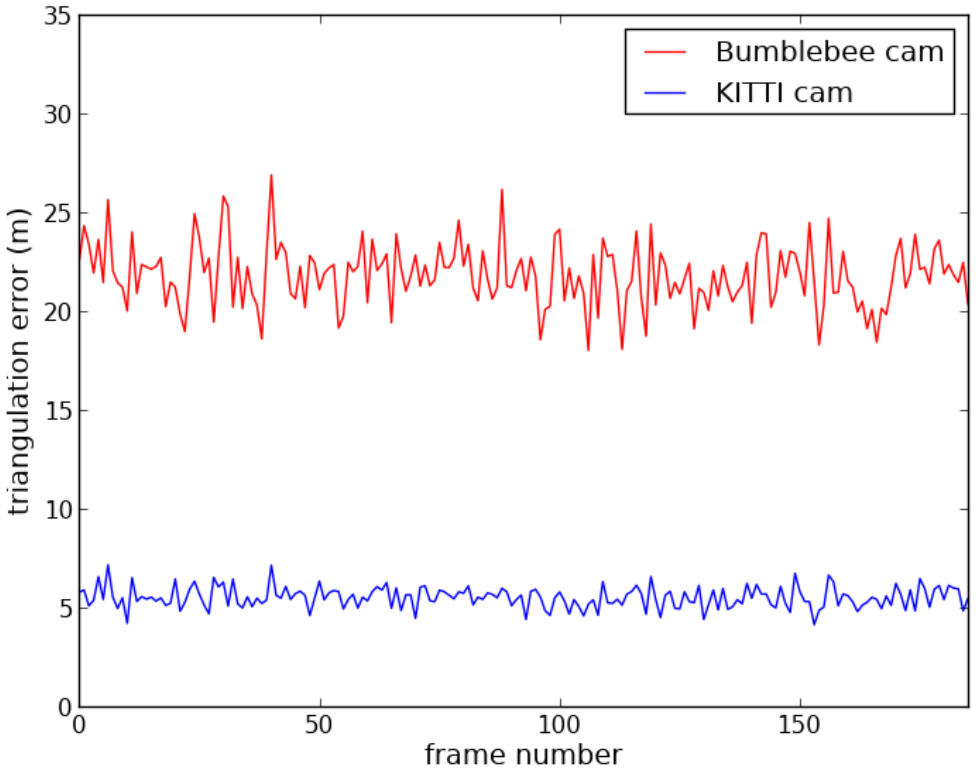
5.3. Umjetni ispitni skup podataka

Testiranje je provedeno i na prethodno opisanom umjetnom ispitnom skupu preko razreda `StereoTrackerLibviso`. Glavna ideja je bila testirati ponašanje vizualne odometrije za različite odmake između kamera. Većina značajki je postavljena na veću udaljenost što se vidi na slici 3.5 i te udaljenosti često pojavljuju prilikom vožnje na otvorenom. Rezultati se mogu vidjeti na slici 5.9. Prilikom generiranja ispitnog skupa variran je samo razmak između kamera, a intrinsični parametri kamera su odgovarali onima za kameru Bumblebee. Slika prikazuje rezultate vizualne odometrije na pravocrtnoj vožnji kroz nekih 175m. Postavljanjem većine značajki na veću udaljenost od kamere htjelo se testirati ponašanje algoritma vizualne odometrije u slučajevima kada imamo malo bliskih značajki i mnogo udaljenih za različite odmake kamera. Iz rezultata se može zaključiti kako odmak između kamera igra veliku ulogu u kvaliteti Razlog leži u tome da veći odmak između kamera povećava preciznost triangulacije, posebice za udaljene značajke zbog nelinearne ovisnosti dispariteta o pogreški triangulacije (slika 2.15). Dodatno je na slici 5.10 prikazan i graf usporedbe uprosječene pogreške triangulacije za Bumblebee2 kameru i kameru korištenu u KITTI datasetu (Flea2). Vidljiva je znatna razlika u slučaju ovako udaljenih značajki. Pogreška triangulacije pokazuje i veće skokove u slučaju Bumblebee kamere jer je šum uzrokovan pikselizacijom veći zbog slabije rezolucije kamere.



Slika 5.9: Rezultati vizualne odometrije na umjetnom ispitnom skupu za pravocrtnu vožnju uz variranje razmaka između kamera

Sustav je također testiran na umjetnom ispitnom skupu koji nije bio zašumljen rasterizacijom prilikom projiciranja točaka na slike. Ti rezultati nisu prikazani jer sustav s tako preciznim podacima daje vrlo točne estimacije bez obzira na razmak između kamera (engl. *baseline*) pa se sve trajektorije sa slike 5.9 savršeno poklapaju s referentnom (engl. *groundtruth*) trajektorijom. To je logično jer tada preciznost triangulacije može biti ograničena samo s preciznosti zapisa s jednostrukom (*float*) ili dvostrukom preciznošću (*double*). Bitno je još jednom naglasiti nedostatak bliskih značajki u distribuciji 3D točaka te da bi u realnom slučaju razlike u greškama odometrije za kameru Bumblebee u većini slučajeva bile manje. Iz ovih rezultata se da zaključiti da bi optimalan razmak između kamera stereo sustava u slučaju kretanja kamere na otvorenom terenu trebao biti negdje između 0.3 i 0.6 metara. Treba naglasiti da ne možemo ni pretjerivati s povećavanjem razmaka između kamera jer tada gubimo područje preklapanja između lijeve i desne slike pa nam za prevelik razmak među kamerama stereo sustav može prostati praktički slijep. Zato je bitno pronaći optimalnu vrijednost razmaka za koju će sustav imati dovoljno preciznu triangulaciju i dovoljno veliko područje preklapanja stereo slike.



Slika 5.10: Uprosječene pogreške triangulacije

6. Zaključak

Stereo vizualna odometrija je postupak određivanja pozicije i orijentacije stereo sustava analizom sekvence parova slika. Prije provođenja estimacije gibanja najprije je potrebno obaviti kalibraciju kamera, odnosno odrediti intrinsične i ekstrinsične parametre stereo sustava. Nakon toga se ulazne slike rektificiraju, čime se znatno smanjuje složenost problema pronalaska korespondencija značajki i olakšava postupak njihove triangulacije. Bitno je ovdje napomenuti da nije nužno da se čitave slike rektificiraju što može biti računski skupo, mogu se rektificirati samo značajke što će biti spomenuti i malo kasnije u predlaganju budućih poboljšanja. Iz pronađenih parova značajki se metodom triangulacije određuje 3D pozicija svake značajke te se na temelju tih 3D točaka i korespondencija značajki kroz dva ili više parova vremenskih okvira minimizacijom reprojekcijske funkcije pogreške određuje vizualna odometrija.

U sklopu ovog rada korišten je postupak iz biblioteke Libviso optimiran za rad u stvarnom vremenu, čiji rezultati uvelike ovise o točnosti kalibracije i rektifikacije te kvaliteti kamere i odmaka između dvaju kamera. U radu su detaljno opisani postupci kalibracije stereo sustava i rektifikacije slika te metode koje koristi biblioteka Libviso. Biblioteka Libviso proširena je za rad s proizvoljnom implementacijom detekcije i praćenja značajki preko osmišljenog i izvedenog sučelja za praćenje. Potom je testirana na dvije ispitne snimke i jednim umjetnim skupom za evaluaciju stereo sustava. Iz eksperimentalnih rezultata je vidljivo da sustav radi bolje na ispitnoj snimci koja se snimljena stereo kamerom bolje rezolucije i koja je imala veći odmak između kamera za razliku od kamere kojom je snimljen drugi ispitni skup. Iz rezultata se može zaključiti da i rezolucija i razmak između kamera izravno utječu na preciznost triangulacije, a samim time i točnost postupka vizualne odometrije.

U budućnosti je plan poboljšati trenutni postupak vizualne odometrije na više načina. Ideja je implementirati praćenje značajki kroz više okvira korištenjem Kanade-Lucas-Tomasi metode diferencijalnog praćenja. Zatim bi se estimacije dobivene trenutnom metodom koristile kao početne estimacije za Sparse Bundle Adjustment metodu koja optimira kretanje kamere kroz proizvoljan broj vremenskih okvira. Dodatno

ova metoda uz kretanje kamere može optimirati i strukturu scene pa čak i same intrinsične parametre kamere ako joj tako zadamo no u prvotnim budućim testiranjima te parametre ipak želimo fiksirati na one dobivene kalibracijom kamere i triangulacijom značajki te optimirati samo gibanje kamere. Glavni problem kod ove metode je tehničke prirode jer trenutne implementacije rade samo sa monokularnim kamerama pa će ih biti potrebno proširiti za rad sa stereo kamerom.

Dodatno se postupak može optimirati i postići ubrzanje tako da se ukloni proces rektifikacije čitavih slika i da se rektificiraju samo pronađene značajke. Zasad su tu dvije zanimljive ideje. Prva bi bila da se rektificira odvojen skup značajki zasebno detektiranih u lijevoj i desnoj slici te se zatim među njima može pronaći podskup korespondentnih značajki podudaranjem po horizontalnim pravcima. Druga ideja ovdje je da se značajke podudaraju po epipolarnim osima te rektificiraju tek nakon podudaranja. Ovdje je prednost naspram prve ideje to što ne moramo detektirati skup značajki u obje slike već samo u jednoj no zato je podudaranje skuplje jer treba računati epipolarne pravce za svaku značajku.

Primjene metode vizualne odometrije ostvarene u ovom radu su brojne. Zasad se ova metoda planira upotrijebiti za razvoj sustava za praćenje oznaka na kolniku koji bi za početak mogao poslužiti kao sustav za upozoravanje vozača na nepropisno ili opasno kretanje automobila na cesti (engl. *lane departure warning*). Osim toga planira se metodu upotrijebiti za napredne postupka detekcije i filtriranja objekata, konkretno u našoj primjeni prometnih znakova. Ideja je da znamo da su svi prometni znakovi planarni i da će se sve značajke detektirane na prometnim znakovima u 3D prostoru triangulirati u istoj ravnini pa tu informaciju možemo iskoristiti kako za detekciju znakova tako i za napredno filtriranje lažnih detekcija dobivenih nekom drugom metodom. Dosadašnji rad polučio je dobre rezultate i pruža opravdanje za daljnje unaprjeđenje postupka.

LITERATURA

[1] Gary Bradski i Adrian Kaehler. *Learning OpenCV*. O'Reilly Media, 2008.

[2] Yang Cheng, Maimone M., i Matthies L. Visual odometry on the mars exploration rovers. *Systems, Man and Cybernetics, 2005 IEEE International Conference*, 1: 903–910, 2005.

[3] M. A Fischler i R. C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association for Computing Machinery 24*, 1981.

[4] A. Geiger, P. Lenz, C. Stiller, i R. Urtasun. Vision meets robotics: The KITTI dataset. 2013. URL `http://www.cvlibs.net/publications/Geiger2013IJRR.pdf`.

[5] Andreas Geiger, Julius Ziegler, i Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. 2011. URL `http://www.cvlibs.net/publications/Geiger2011IV.pdf`.

[6] Bernd Kitt, Andreas Geiger, i Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. 2010. URL `http://www.cvlibs.net/publications/Kitt2010IV.pdf`.

[7] David Nistér, Oleg Naroditsky, i James R. Bergen. Visual odometry. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1:652–659, 2004.

[8] Simon J.D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[9] Andrew Zisserman Richard Hartley. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[10] Alan Sambol. Diplomski rad: Estimiranje strukture i gibanja stereo parom kamera. *FER*, 2012.

[11] S. Segvic, A. Remazeilles, i F. Chaumette. Enhancing the point feature tracker by adaptive modelling of the feature support. *Proceedings of the 9th European Conference on Computer Vision*, stranice 112–124, 2006. URL `http://www.zemris.fer.hr/~ssegvic/pubs/eccv06.pdf`.

Napredno estimiranje strukture i gibanja kalibriranim parom kamera

Sažetak

Širi kontekst rada je određivanje gibanja kamere i strukture scene analizom slijeda pribavljenih slika. Razmatramo specifičan slučaj u kojem je kalibrirani par perspektivnih kamera montiran u smjeru kretanja vozila. U okviru rada proučeni su postupci kalibracije i rektifikacije stereo sustava, postupci praćenja značajki u stereo slikama te postupci vizualne odometrije ostvareni u biblioteci Libviso i napredniji postupci kojima se prethodni mogu poboljšati. Preciznost postupaka je evaluirana na dvije georeferencirane ispitne snimke. Biblioteka je proširena za rad s proizvoljnom implementacijom sustava za detekciju i praćenje značajki pa je evaluacija provedena i na umjetnom ispitnom skupu koji je generiran pomoću ostvarenog simulatora stereo sustava. Razvijen je poboljšan modul za praćenje značajki kroz više vremenskih okvira koji se planira iskoristiti za napredniju metodu vizualne odometrije. Dobiveni rezultati su prikazani i analizirani u poglavlju s eksperimentima.

Ključne riječi: vizualna odometrija, stereo kamera, estimacija gibanja, kalibracija kamere, stereo rektifikacija, 3D rekonstrukcija, triangulacija, epipolarna geometrija, izlučivanje značajki, praćenje značajki, sparse bundle adjustment, biblioteka OpenCV, biblioteka Libviso

Estimation of scene structure and camera movement with calibrated stereo camera system

Abstract

Broader context of this work is to analyse sequence of acquired stereo images from camera and determine the motion of the camera and scene structure. We consider the specific case in which the calibrated stereo camera is mounted on the vehicle in the direction of movement. In this master thesis we studied the methods of stereo camera calibration and rectification, methods for features tracking and methods for visual odometry implemented in the Libviso library. We also consider new advanced methods which can improve visual odometry estimation. System is evaluated on two georeferenced stereo videos taken with two different stereo cameras. Libviso library has been expanded for work with custom implementation of feature detection and tracking methods. Also, the evaluation was performed with artificial dataset which was generated by developing stereo system simulator. Module for differential feature tracking through multiple frames was developed which is planed to be used for improved visual odometry method. The results obtained are presented and analyzed in the section with experiments.

Keywords: visual odometry, stereo camera, movement estimation, camera calibration, stereo rectification, 3D reconstruction, triangulation, epipolar geometry, feature detection, feature tracking, sparse bundle adjustment, OpenCV library, Libviso library