SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 366

# ESTIMIRANJE STRUKTURE I GIBANJA STEREO PAROM KAMERA

Alan Sambol

Zagreb, lipanj 2012.

**Sadržaj**

## Popis slika

# 1. Uvod

Računalni vid je relativno mlada grana računarske znanosti koja se isprepliće s mnogim drugim disciplinama: umjetnom inteligencijom, strojnim učenjem, matematikom, digitalnom obradom slike, fizikom, robotikom.

Stereo računalni vid je posebno zanimljiva podgrana računalnog vida koja ima niz primjena, kao što su video nadzor, praćenje ljudi, autonomna navigacija vozila i robota, proširena stvarnost, mjerenje volumena. Temelj stereo vida je 3D percepcija kod ljudi, koja se bazira na triangulaciji zraka iz više pogleda.

Stereo vid se u ovom radu koristi u svrhu određivanja kretanja kamere i estimiranja strukture analizom slijeda slika. Razmatran je slučaj u kojem je kalibrirani stereo par kamera montiran u smjeru kretanja kamera.

U sljedećem poglavlju opisani su postupci i algoritmi za rektifikaciju, pronalaženje značajki, određivanje koordinata u 3D prostoru i vizualnu odometriju. U trećem poglavlju opisani su ispitni skupovi izvornih sekvenci i slika, pribavljeni pomoću kamera *Bumblebee2* [1] i *GoPro HD* [2]. U četvrtom poglavlju opisana je programska izvedba i korištene vanjske biblioteke za kalibraciju, rektifikaciju i vizualnu odometriju. U petom poglavlju prikazani su eksperimentalni rezultati.

# 2. Korišteni algoritmi i matematičke metode

Stereo vid bazira se na pronalaženju podudarnosti između točaka slike lijeve i desne kamere. Uz poznati razmak između kamera (engl. *baseline*), moguće je izračunati koordinate točaka u 3D prostoru. Slika 1 prikazuje par slika sa dvije horizontalno poravnate kamere. Točka $A_{left}$ lijeve slike odgovara točki $A_{right}$ desne slike. Ukoliko izmjerimo udaljenosti točaka do lijevog ruba slike, uočit ćemo da je na lijevoj slici ta udaljenost veća nego na desnoj. Na temelju razlike tih udaljenosti, odnosno dispariteta (engl. *disparity*), moguće je izračunati udaljenost točke *A* od kamere pomoću triangulacije. Svaki piksel digitalne kamere "skuplja" svjetlost koja dolazi do kamere jednom 3D zrakom. Ukoliko koristimo više kamera, imamo više zraka. Sjecište tih zraka predstavlja 3D lokaciju značajke.


Slika 1 - primjer uparivanja točaka lijeve i desne slike

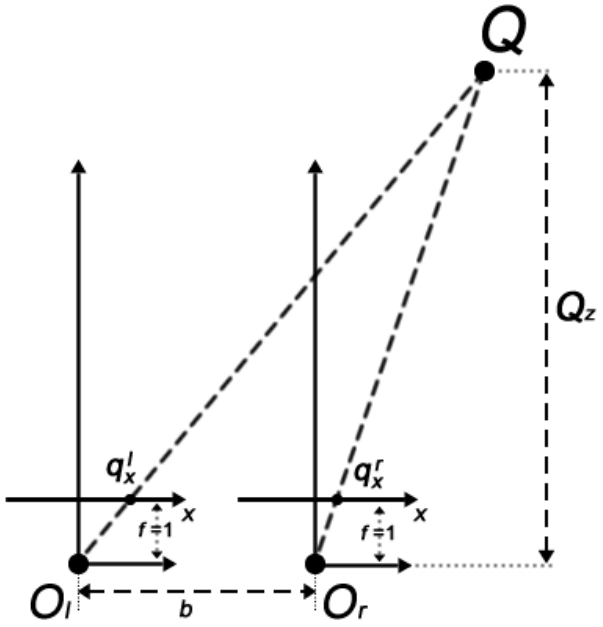
Stereo vid najčešće uključuje četiri koraka:

1. Matematičko uklanjanje radijalne i tangencijalne distorzije leće (engl. *undistortion*),
2. Rektifikacija, odnosno projekcija slika dviju kamera u zajedničku slikovnu ravninu (engl. *rectification*). Nakon ovog koraka slike su poravnate po retcima.
3. Pronalaženje korespondentnih značajki u lijevoj i desnoj slici (engl. *correspondence*). Nakon ovog koraka poznati su nam dispariteti značajki, odnosno razlike u iznosima *x*-koordinata značajki u lijevoj i desnoj slici: $x^l - x^r$.

4. Ukoliko su nam poznata geometrijska svojstva kamera, moguće je iz mape dispariteta izračunati udaljenosti točaka od kamere u 3D prostoru. Ovaj korak zove se reprojekcija (engl. *reprojection*), a njegov izlaz dubinska mapa (engl. *depth map*).

## 2.1. Triangulacija

Pretpostavimo da imamo idealni stereo sustav, bez distorzije, horizontalno poravnat, kakav prikazuje Slika 2. Slikovne ravnine obje kamere su međusobno koplanarne te imaju paralelne optičke osi, čiji razmak nam je poznat (sijeku se u beskonačnosti). Optička os (engl. *optical axis*, *principal ray*) je zraka koja izlazi iz centra projekcije *O* kroz osnovnu točku (engl. *principal point*) *c*. Osnovna točka je točka u kojoj optička os siječe slikovnu ravninu i ne mora nužno biti jednaka centru slike. Obje kamere imaju jednake žarišne duljine (engl. *focal length*) $f_l = f_r = f = 1$, te su im osnovne točke $c_x^{left}$ *i* $c_x^{right}$ na istim koordinatama u svojim slikama.




Slika 2 - shema idealnog stereo sustava

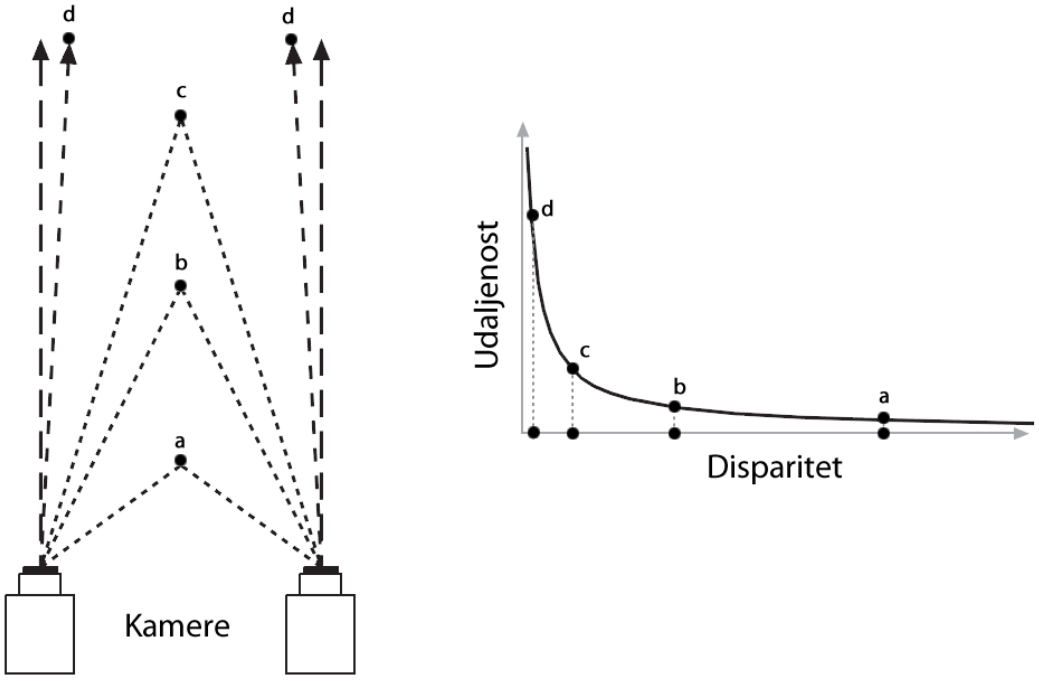
Također, pretpostavimo da točku *Q* iz stvarnog svijeta možemo pronaći u lijevoj i desnoj slici na pozicijama $q_l$ i $q_r$, čije horizontalne koordinate su $q_x^l$ i $q_x^r$. Koristeći sličnost trokuta, možemo pokazati da je dubina ($Q_z$) obrnuto proporcionalna disparitetu $d = q_x^l - q_x^r$, što je prikazano sljedećim jednadžbama:

$$\frac{Q_x^l}{Q_z^l} = q_x^l, \qquad \frac{Q_x^r}{Q_z^r} = q_x^r \tag{1}$$

$$\frac{Q_x^l - b}{Q_z} = q_x^r \tag{2}$$

$$\frac{Q_x^l - Q_x^l + b}{Q_z} = q_x^l - q_x^r \;\Rightarrow\; Q_z = \frac{b}{d} \tag{3}$$

Zahvaljujući toj činjenici, uz pretpostavku da je udaljenost između kamera fiksna, kada je disparitet blizu nule, male promjene u disparitetu  znače velike promjene u dubini, dok u slučaju kada je disparitet velik, male promjene u disparitetu ne utječu znatno na promjenu dubine. Direktna posljedica toga je da stereo sustavi postižu dobru dubinsku rezoluciju samo za objekte koji se nalaze relativno blizu kameri, što prikazuje Slika 3.




Slika 3 - obrnuta proporcionalnost dispariteta i dubine

U stvarnim primjenama, kamere su rijetko u odnosu kakav prikazuje Slika 2, stoga je potrebno pronaći projekcije slike i distorzijske mape (engl. *distortion map*) pomoću kojih možemo rektificirati lijevu i desnu sliku. Pri postavljanju kamera u stvarnom svijetu, ukoliko ih ne postavimo barem približno kao u idealnom slučaju,

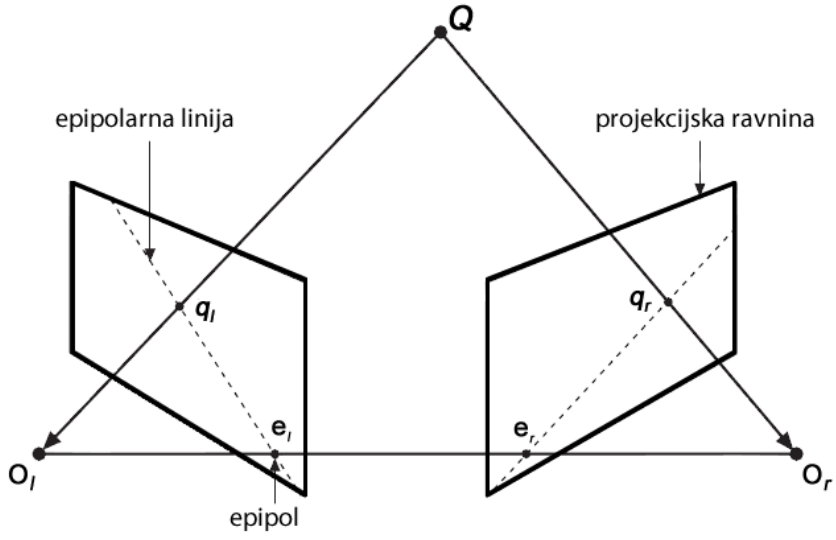
smanjit ćemo (ili čak eliminirati) područje prekrivanja (engl. *overlap area*) stereo slika. Iznimka su primjene gdje želimo veću rezoluciju na bližim razmacima. U tom slučaju je dobro blago okrenuti kamere jednu prema drugoj, tako da im se optičke osi sijeku na konačnim udaljenostima (vergencija). Također, vrlo je bitno i da kamere budu vremenski sinkronizirane, kako bismo izbjegli probleme s pomičnim objektima u sceni (kao i pomičnim kamerama). U suprotnom smo limitirani na korištenje stacionarnih kamera za stacionarne scene.

S obzirom da se mnoge operacije pojednostavljuju u idealnom rasporedu kamera, cilj je matematički (a ne fizički) poravnati dvije kamere u jednu ravninu, što se opisuje u sljedećem potpoglavlju.

## 2.2. Epipolarna geometrija

Epipolarna geometrija (engl. *epipolar geometry*) je unutarnja projekcijska geometrija između dva pogleda koja ne ovisi o strukturi prizora već samo o unutarnjim parametrima kamera te njihovoj relativnoj poziciji.

Slika 4 prikazuje shemu epipolarne geometrije. Za svaku kameru sada postoji zasebni centar projekcije, $O_l$ i $O_r$ te pripadajuće projekcijske ravnine $\Pi_l$ i $\Pi_r$. Točka *Q* iz svijeta ima projekcije $q_l$ i $q_r$. Epipol $e_l$ ($e_r$) je definiran kao slika centra projekcije druge kamere $O_r$ ($O_l$). Ravnina koju formiraju točke *Q*, $e_l$ i $e_r$ (ili *Q*, $O_r$ i $O_l$) naziva se epipolarna ravnina, a linije $q_l e_l$ i $q_r e_r$ epipolarne linije.




Slika 4 - epipolarna geometrija

Točka *Q* koju vidimo projiciranu na desnu (ili lijevu) projekcijsku ravninu može se u stvarnosti nalaziti bilo gdje uzduž linije koja izlazi iz točke $O_r$ i prolazi kroz $q_r$, s obzirom da s jednom kamerom nije moguće odrediti udaljenost do točke. Projekcija te linije na lijevu ravninu zapravo je epipolarna linija $q_l e_l$. Drugim riječima, projekcija svih mogućih lokacija točke jedne slike je linija koja prolazi kroz korespondentnu i epipolarnu točku druge slike.

Za bilo koju točku jedne slike vrijedi da njena korespondentna točka u drugoj slici leži na epipolarnoj liniji. Taj je uvjet poznat pod nazivom epipolarno ograničenje (engl. *epipolar constraint*) [3]. To pak znači da je umjesto dvodimenzionalne pretrage za korespondencijama dovoljna jednodimenzionalna pretraga, čime se znatno smanjenje složenost, a i omogućava odbacivanje velikog broja lažnih korespondencija.

## 2.3. Esencijalna i fundamentalna matrica

Kako bismo mogli pronaći epipolarne linije, potrebno je uvesti pojmove esencijalne i fundamentalne matrice. Esencijalna matrica *E* sadrži informacije o translaciji i rotaciji koje vežu dvije kamere u prostoru, odnosno daje nam vezu između točaka $\hat{q}_l$ i $\hat{q}_r$ u normiranom koordinatnom sustavu slike (u kojem je žarišna duljina *f = 1*). Fundamentalna matrica *F* sadrži, uz informacije koje sadrži esencijalna matrica, i intrinsične parametre obje kamere. Ona veže točke iz lijeve i desne slike $q_l$ i $q_r$.

Uzmimo za referentnu točku $O_l$. Lokacija te točke u slici je točka $Q_l$, a ishodište druge kamere nalazi se u točki *T*. Točka *Q* u desnoj kameri jest

$$Q_r = R(Q_l - T) \tag{4}$$

Jednadžbu epipolarne ravnine možemo predstaviti na više načina, a najjednostavnije je prisjetiti se da jednadžba svih točaka **x** na ravnini sa vektorom normale **n** i koja prolazi točkom **a** glasi:

$$(\mathbf{x} - \mathbf{a}) \cdot \mathbf{n} = 0 \tag{5}$$

Vektor normale **n** u našem slučaju možemo dobiti kao $Q_l \times T$. Stoga, jednadžba svih točaka $Q_l$ kroz točku *T* sa normalom $Q_l \times T$ glasi:

$$(Q_l - T)^T (T \times Q_l) = 0 \tag{6}$$

Jednadžba (2) se može zapisati i ovako:

$$(Q_l - T) = R^{-1} Q_r \tag{7}$$

Koristeći tu supstituciju nad jednadžbom (4), uz $R^T = R^{-1}$ dobivamo:

$$(R^T Q_r)^T (T \times Q_l) = 0 \tag{8}$$

Vektorski produkt zapišimo kao običan matrični produkt:

$$(T \times Q_l) = S Q_l \;\Rightarrow\; S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{9}$$

Koristeći tu supstituciju nad jednadžbom (8) dobivamo:

$$(Q_r)^T R S Q_l = 0 \tag{10}$$

Definicija esencijalne matrice tada glasi $E = RS$, a jednadžbu (8) možemo zapisati kao:

$$(Q_r)^T E Q_l = 0, \text{ odnosno } {q_r}^T E q_l = 0 \tag{11}$$

Esencijalna matrica sadrži sve informacije o geometriji jedne kamere u odnosu na drugu, no ne sadrži nikakve informacije o samim kamerama. Međutim, u praksi, zanimaju nas koordinate u pikselima slike. Kako bismo mogli pronaći vezu između piksela jedne slike sa odgovarajućom epipolarnom linijom u drugoj slici, moramo iskoristiti intrinsične parametre kamera. Zato točku $\hat{q}$ supstituiramo točkom $q$ i matricom intrinsičnih parametara kamere *K*, točnije $q = K\hat{q}$, odnosno $\hat{q} = K^{-1} q$. Jednadžba za esencijalnu matricu sada postaje:

$${q_r}^T ({K_r}^{-1})^T E {K_l}^{-1} q_l = 0 \tag{12}$$

Fundamentalna matrica se definira kao

$$F = ({K_r}^{-1})^T E {K_l}^{-1} \tag{13}$$

Jednadžba (12) tada postaje:

$${q_r}^T F q_l = 0 \tag{14}$$

## 2.4. Kalibracija stereoskopskog sustava

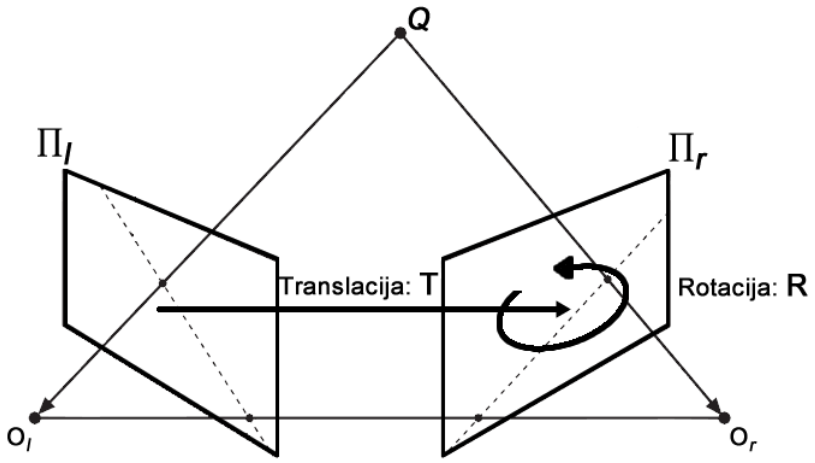
Kalibracija ili umjeravanje kamere je postupak određivanja unutarnje geometrije kamere, koja utječe na proces formiranja digitalne slike sastavljene od piksela. Svodi se na pronalaženje intrinsičnih parametara kamere:

- žarišna duljina, koja udaljenost slikovne ravnine od centra projekcije
- osnovna točka, kojom prolazi optička os
- koeficijent ukošenosti, kojim se modelira nejednaka horizontalana i vertikalna udaljenost slikovnih elemenata (pretpostavlja se da je jednak nuli)
- koeficijenti distorzije leće, kojima se opisuju promjene u slici nastale zbog optičkih nesavršenosti u leći

Matrica kamere *K*, koja sadrži intrinsične parametre glasi:

$$K = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Kalibracija stereoskopskog sustava je proces kojim se određuje geometrijski odnos dviju kamera u prostoru, a svodi se na pronalaženje rotacijske matrice *R* i translacijskog vektora *T* između kamera, što prikazuje Slika 5:




Slika 5 - stereo kalibracija

Moguće je iskoristiti kalibracije pojedinačnih kamera kako bi se proizvoljna točka *Q* predstavila u koordinatama lijeve ili desne kamere kao $Q_l = R_l Q + T_l$,

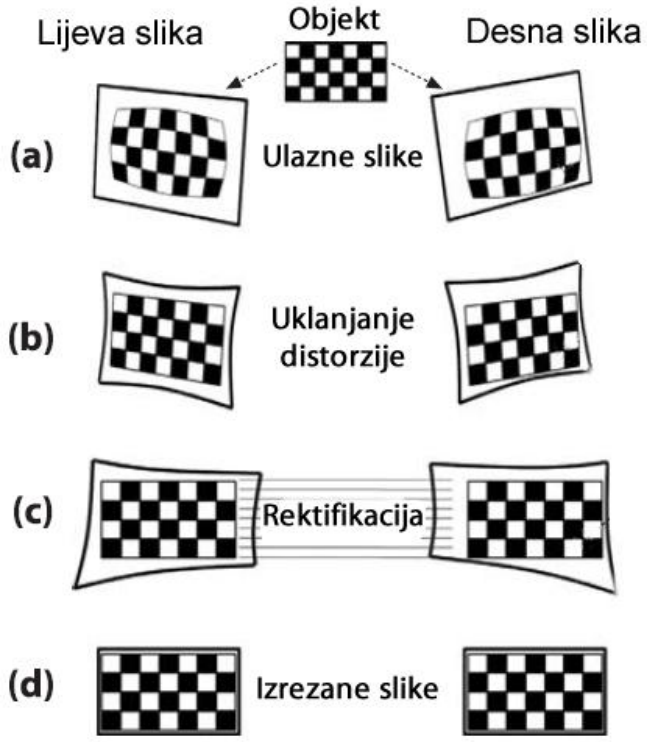
odnosno $Q_r = R_r Q + T_r$. Također vrijedi $Q_l = R^T(Q_r - T)$. Konačni izrazi za rotaciju i translaciju tada glase:

$$R = R_r(R_l)^T \tag{16}$$

$$T = T_r - RT_l \tag{17}$$

## 2.5. Stereo rektifikacija

Stereo rektifikacija je proces ispravljanja pojedinačnih slika s ciljem da one budu poravnate po retcima te da optičke osi njihovih virtualnih kamera budu paralelne, odnosno sijeku se u beskonačnosti. To nam garantira da se korespondentne točke nalaze na istom retku u obje slike, čime se znatno ubrzava proces njihovog pronalaženja, a povećava se i točnost.




Slika 6 - stereo rektifikacija

Nakon poravnavanja redaka, epipoli će se nalaziti u beskonačnosti, odnosno, slika centra projekcije u jednoj slici biti će paralelna drugoj slikovnoj ravnini. Budući da postoji beskonačno mnogo paralelnih ravnina koje možemo odabrati, potrebno je dodati još ograničenja, kao što su maksimizacija područja preklapanja slika ili

minimizacija distorzije. U ovom radu korišten je Bouguetov algoritam za rektifikaciju slika [4], opisan u nastavku.

### 2.5.1. Bouguetov algoritam

Bouguetov algoritam za rektifikaciju je pojednostavljena i dovršena metoda koju su prvi put predstavili Tsai [5] i Zhang [6]. Bouguet nikada nije objavio članak s algoritmom, osim njegove implementacije unutar alata *Camera Calibration Toolbox for Matlab*.

Za rad algoritma potrebna je prethodna kalibracija stereo sustava, odnosno rotacijska matrica $\boldsymbol{R}$ i translacijski vektor $\boldsymbol{T}$ između slika. Ideja algoritma je jednostavna - minimizacija promjene (distorzije) uzrokovane reprojekcijom uz maksimizaciju područja preklapanja slika.

Algoritam se sastoji od četiri koraka:

- rotacija lijeve kamere na način da epipol bude u beskonačnosti, uz horizontalnu os
- rotacija desne kamere za isti iznos
- rotacija desne kamere za iznos $\boldsymbol{R}$
- računanje koordinata odgovarajućih piksela u rektificiranim slikama

Kako bismo izračunali matricu $\boldsymbol{R}_{rect}$, koja pomiče epipol lijeve kamere u beskonačnost uz horizontalnu os, potrebna su tri ortogonalna vektora, $\boldsymbol{e}_1, \boldsymbol{e}_2$ i $\boldsymbol{e}_3$. Prvi vektor, $\boldsymbol{e}_1$, određen je epipolom te se podudara s translacijskim vektorom između centara projekcija dviju kamera:

$$\boldsymbol{e}_1 = \frac{\boldsymbol{T}}{\|\boldsymbol{T}\|} \tag{15}$$

Jedino ograničenje na vektor $\boldsymbol{e}_2$ je da mora biti ortogonalan u odnosu na vektor $\boldsymbol{e}_1$. Odabiremo smjer ortogonalan u odnosu na optičku os, uz slikovnu ravninu. Vektor $\boldsymbol{e}_2$ dobivamo skalarnim produktom vektora $\boldsymbol{e}_1$ sa smjerom optičke osi te ga normaliziramo kako bismo opet dobili jedinični vektor:

$$\boldsymbol{e}_2 = \frac{[-\boldsymbol{T_y T_x}\ \ \boldsymbol{0}]^{\boldsymbol{T}}}{\sqrt{\boldsymbol{T_x}^2 + \boldsymbol{T_y}^2}} \tag{16}$$

Treći vektor, $\boldsymbol{e}_3$, ortogonalan je u odnosu na $\boldsymbol{e}_1$ i $\boldsymbol{e}_2$ te se računa kao njihov vektorski produkt:

$$\boldsymbol{e}_3 = \boldsymbol{e}_1 \times \boldsymbol{e}_2 \tag{17}$$

Rotacijska matrica kojom pomičemo epipol lijeve kamere u beskonačnost sada glasi:

$$\boldsymbol{R}_{rect} = \begin{bmatrix} \boldsymbol{e}_1{}^T \\ \boldsymbol{e}_2{}^T \\ \boldsymbol{e}_3{}^T \end{bmatrix} \tag{18}$$

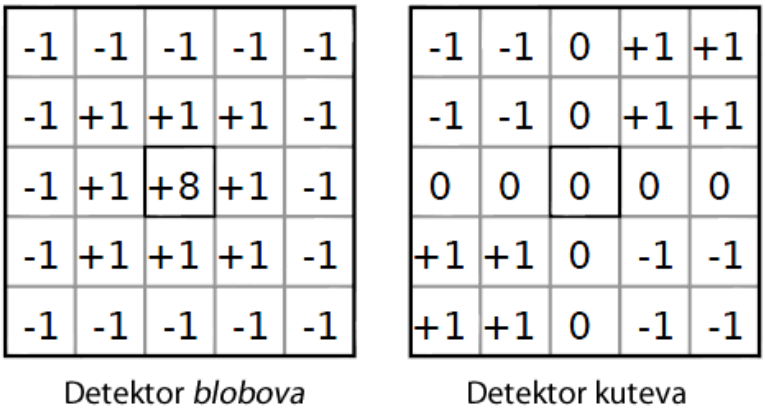
Ostatak algoritma glasi:

1. Izgradi matricu $\boldsymbol{R}_{rect}$ preko jednadžbe (18),
2. Postavi $\boldsymbol{R}_l = \boldsymbol{R}_{rect}$ i $\boldsymbol{R}_r = \boldsymbol{R}\boldsymbol{R}_{rect}$,
3. Za svaku točku lijeve kamere $q_l = [x \ \ y \ \ 1]^T$ izračunaj $\boldsymbol{R}_l q_l = [x' \ \ y' \ \ z']$ i koordinate pripadajuće rektificirane točke, $q_l' = \frac{1}{z'}[x' \ \ y' \ \ z']$,
4. Ponovi prethodni korak za desnu kameru koristeći $\boldsymbol{R}_r$ i $q_r$.

S obzirom da rezultirajući skup rektificiranih točaka ne obuhvaća sve moguće koordinate, u slučaju da želimo rektificirati cijelu sliku, a ne samo pojedine točke, potrebno je koordinate koje nedostaju popuniti korištenjem bilinearne interpolacije.

## 2.6. Pronalaženje značajki

Značajke koje se koriste u ovom radu inspirirane su radom [7] [8]. Prvi korak je filtriranje ulaznih slika sa maskama za pronalaženje kuteva i *blobova*. Slika 7 prikazuje te filtre.

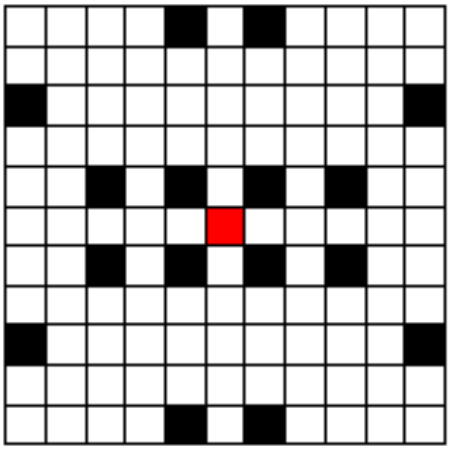



Slika 7 - filtri za pronalaženje *blobova* i kuteva

Sljedeći korak je potiskivanje odziva koji nisu maksimalni ni minimalni (engl. *non-maximum- and non-minimum-suppression*), čime preostale značajke pripadaju jednoj od četiri skupine:

1. *blob*, maksimum
2. *blob*, minimum
3. kut, maksimum
4. kut, minimum

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



Slika 8 - deskriptor korišten za značajke

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* pronalaženjem 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 sa 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 $\tau_{disp}$, odnosno $\tau_{flow}$.

## 2.7. Vizualna odometrija

Vizualna odometrija je proces određivanja pozicije i orijentacije kamere analizom slika pribavljenih tom kamerom. Algoritmi za vizualnu odometriju mogu se podijeliti na algoritme koji koriste uparivanje značajki između susjednih slika te algoritme koji prate značajke kroz slijed slika.

Nakon postupka triangulacije, poznate su nam 3D koordinate značajki, njihove korespondencije u prošlom vremenskom okviru te korespondencije u lijevoj, odnosno desnoj slici. Ukoliko je prisutno gibanje kamere, 3D točke u prošlom i sadašnjem vremenskom okviru će biti odmaknute za neki iznos. S obzirom da se radi o koordinatnom sustavu kamere, pretpostavljamo da se značajke kreću u odnosu na kameru. Taj slučaj je moguć ako postoji kretanje unutar scene, no može se pretpostaviti da je većina značajki stacionarna u koordinatnom sustavu svijeta. To znači da kretanje 3D točaka zapravo podrazumijeva kretanje kamere.

Pomak kamere opisuje se matricama ***R*** i ***T*** te sadrži ukupno šest stupnjeva slobode, a za njegovo izračunavanje, potrebne su minimalno tri točke sa svojim pozicijama prije i poslije pomaka. Problem se svodi na određivanje rotacije i translacije trokuta određenog s te tri točke s prve pozicije na drugu. Reprojekciju 3D točke iz prošle slike u trenutnu sliku opisuje sljedeća jednadžba:

$$\begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix} = K \left[ P \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} - \begin{pmatrix} s \\ 0 \\ 0 \end{pmatrix} \right] \tag{19}$$

odnosno:

$$\begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{pmatrix} \left[ (R(r) \quad t) \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} - \begin{pmatrix} s \\ 0 \\ 0 \end{pmatrix} \right] \tag{20}$$

gdje su:

- $(q_x \ q_y \ 1)^T$ - homogene koordinate u slici
- *f* - žarišna duljina
- $(c_u, c_v)$ - osnovna točka (engl. *principal point*)
- $R(r)$ - rotacijska matrica
- *t* - translacijski vektor
- $Q = (Q_x \ Q_y \ Q_z)^T$ - 3D koordinate točke
- *s* - pomak (0 za lijevu sliku, udaljenost između kamera za desnu sliku)

Neka $\pi^{(l)}(Q; r, t) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ označava projekciju koja 3D točku *Q* preslikava u piksel $q_i^{(l)} \in \mathbb{R}^2$ u lijevoj slici, te neka je $\pi^{(r)}(Q; r, t)$ projekcija na desnu sliku. Cilj je pronaći transformacijske parametre (***r, t***) za koje je razlika između projekcije 3D točaka na trenutni okvir i stvarnih koordinata tih točaka u trenutnom okviru minimalna, što opisuje sljedeća jednadžba:

$$\sum_{i=1}^{N} \left\| q_i^{(l)} - \pi^{(l)}(Q_i; r, t) \right\|^2 + \left\| q_i^{(r)} - \pi^{(r)}(Q_i; r, t) \right\|^2 \tag{20}$$

Za minimizaciju gornjeg izraza koristi se Gauss-Newtonova optimizacija, koja u ovom radu neće biti detaljno opisana.

Kada bi sve značajke bile idealne i bez šuma, gornja metoda dala bi konačno rješenje - odabirom bilo koje tri točke pronalazimo (***r, t***). Međutim, značajke sadrže šum te ranije izložena pretpostavka o statičnoj sceni ne vrijedi uvijek, stoga je potrebno je odabrati robusniji pristup.

### 2.7.1. Algoritam RANSAC

Algoritam RANSAC (engl. *Random Sample Consensus*) razvili su Fischler i Bolles 1981. godine kao robusnu alternativu metodi najmanjih kvadrata [9]. Općenito, RANSAC generira *n* modela koristeći nasumično odabrane podskupove ulaznog skupa, uspoređuje ih i vraća najbolji.

Primijenjen na problem vizualne odometrije, algoritam glasi:

1. Nasumično odaberi tri značajke i koristeći ranije opisani algoritam pronađi rotacijsku matricu ***R*** i translacijski vektor ***t***.
2. Primijenimo pronađene ***R*** i ***t*** na sve značajke. Odaberemo značajke čija je udaljenost reprojekcije od stvarnih koordinata u trenutnom okviru manja ili jednaka nekom iznosu *d.* Te značajke nazivaju se potpornim skupom trenutne hipoteze, odnosno unutarpopulacijske značajke (engl. *inliers*).
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 broj iteracija *k*.

U ovom radu korišten je algoritam RANSAC s $k = 50$ iteracija. Svi *inlieri* najboljeg modela koriste se za poboljšanje parametara, čime se dobiva konačna transformacija (***r, t***).

# 3. Ispitni skupovi izvornih slika

Ispitni skupovi izvornih pribavljeni su sa tri zasebna izvora. U sljedećim podpoglavljima opisani su pojedini skupovi.

## 3.1. Ispitni skup Karlsruhe

Ispitni skup Karlsruhe [10] sastoji se od sivih stereo sekvenci pribavljenih iz vožnje automobilom kroz grad Karlsruhe, no ne u sklopu ovog rada. Korištene su za provjeru rada biblioteke *libviso2*. Sekvence su pribavljene *firewire* kamerom Pointgrey Flea2 [11] kao rektificirane slike u formatu *png.* Kamere su odmaknute 57 cm, a rezolucija slika je oko 1300×400, ovisno o veličini interesne regije nakon rektifikacije. Ukupno je raspoloživo 20 različitih sekvenci, a u ovom radu korištena je sekvenca *2009_03_09_drive_0019*.

Za svaku od sekvenci dostupna je stereo kalibracija kamera te izlaz geolokacijskog (GPS) sustava za svaki vremenski okvir, sa sljedećim stupcima:

1. Vremenska oznaka
2. Geografska širina
3. Geografska dužina
4. Nadmorska visina
5. x, y i z koordinate
6. Zaokret u tri smjera (engl. *roll*, *pitch*, *yaw*)

Koordinate geolokacijskog sustava mogu se koristiti za usporedbu rezultata s vizualnom odometrijom, za što je potrebno poravnati oba koordinatna sustava.

Slika 9 prikazuje prvi vremenski okvir iz sekvence *2009_03_09_drive_0019* s iscrtanim horizontalnim linijama koje potvrđuju da se rektifikacijom iste značajke nalaze u istom retku u lijevoj i desnoj slici.

Slika 9 - rektificirane slike iz sekvence *2009_03_09_drive_0019*

## 3.2. Ispitni skup Bumblebee2

Drugi ispitni skup pribavljen je u sklopu ovog rada, kamerom *PointGrey Bumblebee2* [1]. Kamere su međusobno udaljene 12 cm te daju sive slike rezolucije 640×480. Na sljedećoj slici prikazana je kamera *Bumblebee2*:

Slika 10 - kamera Bumblebee2

Kamera se spaja preko *firewire* priključka, a slike su pribavljene korištenjem aplikacije *PointGrey FlyCapture2*. Karakteristika tog softvera je da stereo parove slika ne sprema u zasebne slike već u jednu datoteku formata *pgm* (engl. *portable*

*graymap*), na način 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 daljnjim algoritmima, napisana je skripta u jeziku *Python*:

Odsječak koda 1 - skripta za razdvajanje slika u zasebne datoteke

```
f=open('putanja/do/datoteke.pgm', mode='rb')
f.readline() # preskakanje zaglavlja
f.readline() # preskakanje zaglavlja
f.readline() # preskakanje zaglavlja
img=f.read()
imgl=img[0::2] # parni pikseli
imgr=img[1::2] # neparni pikseli
f2=open('putanja/do/lijeve/slike.pgm', mode='wb')
f2.write(b'P5\n') # pisanje zaglavlja
f2.write(b'640 480\n') # pisanje zaglavlja
f2.write(b'255\n') # pisanje zaglavlja
f2.write(imgl)
f2.close()
f2=open('putanja/do/desne/slike.pgm', mode='wb')
f2.write(b'P5\n') # pisanje zaglavlja
f2.write(b'640 480\n') # pisanje zaglavlja
f2.write(b'255\n') # pisanje zaglavlja
f2.write(imgr)
f2.close()
```

S obzirom da su parametri kamere nepoznati, pribavljene su sekvence za kalibraciju. Postupak kalibracije stereo sustava detaljno je opisan u sljedećem poglavlju.

Pribavljene su ukupno četiri ispitne sekvence: prva sekvenca snimljena je u hodniku stana te, s obzirom na duljinu *firewire* kabla od 4 metra, sadrži kretanje od nekoliko koraka u dva smjera.

Slika 11 prikazuje tlocrt s procijenjenim kretanjem te jedan vremenski okvir iz sekvence:

Slika 11 - tlocrt i jedan vremenski okvir sekvence *bumblebee_stan*

Druga, treća i četvrta sekvenca pribavljene su u prostorima Fakulteta elektrotehnike i računarstva, pomoću kolica na koja je montirana kamera spojena na računalo koje se također nalazi na kolicima. Slika 12 prikazuje kolica s računalom i montiranom kamerom *Bumblebee2* (na ista kolica montirana je i kamera korištena za sekvence iz sljedećeg potpoglavlja).


Slika 12 - kolica s montiranim kamerama

Druga sekvenca sastoji se od izlaska iz kabineta na hodnik fakulteta, vožnju kroz hodnik te zaokret za 180 stupnjeva. Zanimljiva je iz više razloga: hodnici fakulteta uglavnom su bijeli i ne sadrže puno značajki, na podu hodnika prisutna je

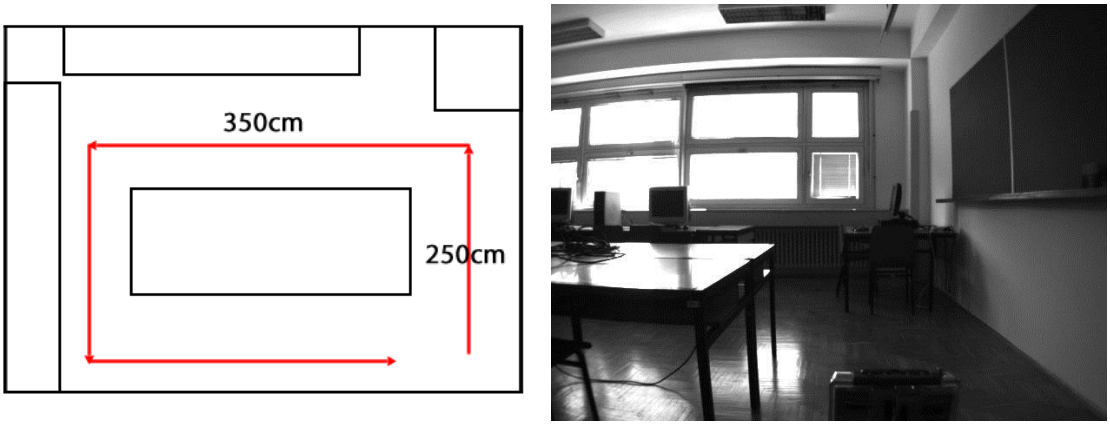
značajna refleksija svjetlosti te trenutak okreta za 180 stupnjeva sadrži isključivo pogled na bijeli zid, bez ikakvih točaka od interesa koje bi mogle predstavljati značajke. Slika 13 prikazuje procjenjeno kretanje po tlocrtu [12] te vremenski okvir koji prikazuje scenu u hodniku.


Slika 13 - tlocrt sekvence *bumblebee_zemris* i jedan vremenski okvir

Slika 14 prikazuje vremenski okvir s refleksijom svjetlosti od poda te vremenski okvir tik nakon okreta od 180 stupnjeva gdje je vidljiv samo bijeli zid.

Slika 14 - vremenski okviri sa refleksijom i okretom od 180 stupnjeva

Treća i četvrta sekvenca sastoje se od pravokutnog gibanja u prostoriji D337 Fakulteta elektrotehnike i računarstva. Na sljedećim slikama prikazan je njihov tlocrt i jedan vremenski okvir.




Slika 15 - tlocrt sekvence *bubmlebee_kabinet* i jedan vremenski okvir

## 3.3. Ispitni skup GoPro HD

Zadnji ispitni skup pribavljen je također u okviru ovog rada, pomoću dvije kamere *GoPro HD Hero2* kamere [2], međusobno udaljene 3.4 cm. Kamere daju slike u boji, rezolucije 1920×1080 te ih spremaju na SD karticu, zbog čega za snimanje sekvenci nije potrebna povezanost s računalom, za razliku od kamere *Bumblebee2*. Slika 16 prikazuje kamere te kućište koje ih povezuje.

Slika 16 - stereo sustav *GoPro HD*

Kamere snimaju sekvence u obliku *avi* datoteka koje su prebačene u slike formata *jpg* pomoću aplikacije *Avidemux 2*. Parametri kamera su nepoznati tako da je također bilo potrebno provesti postupak kalibracije, opisan u sljedećem poglavlju.

Kamera je bila montirana na računalo na istim kolicima kao i kamera Bumblebee2 te je pribavljena sekvenca *gopro_kabinet* koja odgovara sekvenci

*bumblebee_kabinet*. Specifičnost ove sekvence je prisutnost znatne količine trzanja uslijed kretanja kolica. Slika 17 prikazuje jedan vremenski okvir sekvence:

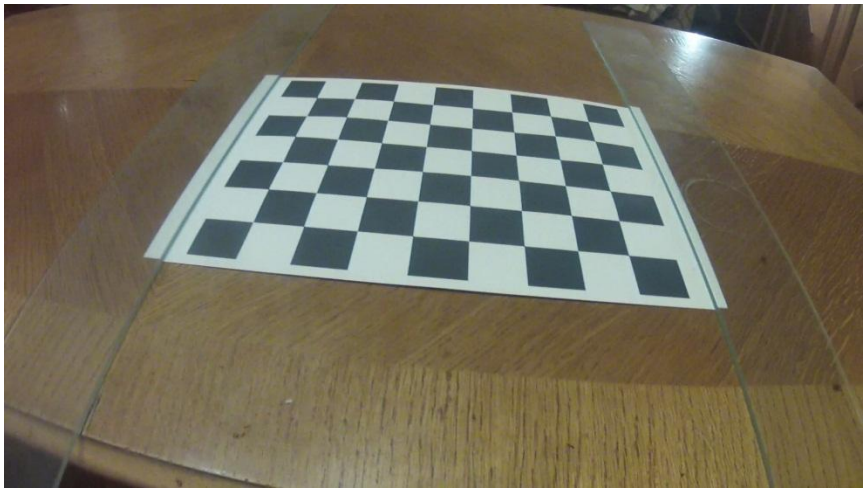
Slika 17 - vremenski okvir iz sekvence *gopro_kabinet*

# 4. Programska izvedba i vanjske biblioteke

U sljedećim potpoglavljima opisan je postupak kalibracije stereo sustava koristeći alat *Camera Calibration Toolbox for Matlab,* uz prilagodbu alata za rektifikaciju slika, biblioteka *libviso2* korištena za 3D rekonstrukciju i vizualnu odometriju zajedno s implementacijom podrške za rad s nerektificiranim slikama, biblioteka *OpenCV*, pomoćne skripte u programskom jeziku *Matlab* za iscrtavanje značajki i rezultata vizualne odometrije te su dane upute za instalaciju svihalata i komponenti.

## 4.1. Alat Camera Calibration Toolbox for Matlab

Alat *Camera Calibration Toolbox for Matlab* [4] omogućava kalibraciju monokularnih i stereoskopskih kamera uz modeliranje intrinsičnih i ekstrinsičnih parametara te nudi mogućnost grafičkog prikaza različitih koraka kalibracije. Slika 18 prikazuje planarni uzorak korišten pri kalibraciji kamera:

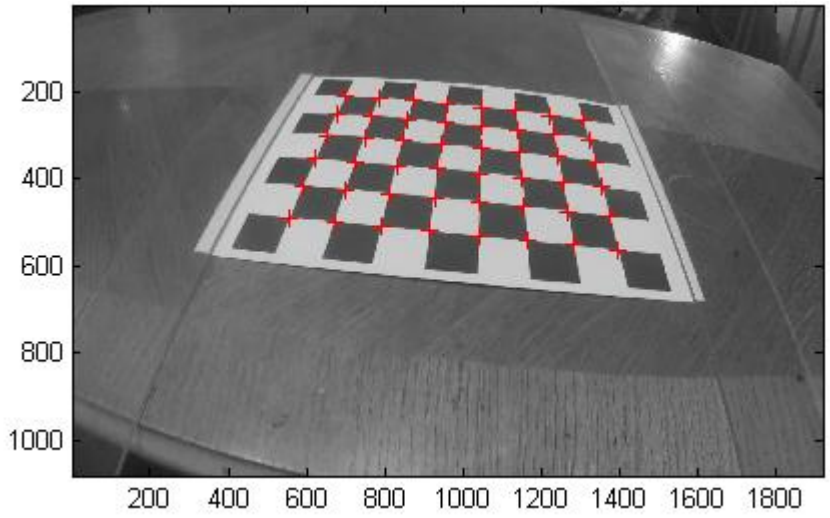


Slika 18 - planarni uzorak korišten pri kalibraciji kamere

Prvi korak u kalibraciji je fiksiranje planarnog uzorka na ravnu površinu te snimanje sekvence sa više pogleda na uzorak. Iz te sekvence potrebno je izdvojiti slike sa što više različitih kuteva prikaza. U ovom radu odabrano je 15 reprezentativnih slika za obje kamere.

Sljedeći korak je instalacija alata – potrebno je u *Matlabu* pozvati naredbu `addpath('putanja/do/mape/toolbox_calib')`. Nakon toga se naredbom

`calib_gui` otvara korisničko sučelje alata za kalibraciju pojedinačnih kamera. Potrebno je učitati slike odabirom opcije *Image names*, što pretpostavlja da su sve slike jedne kamere imenovane sa zajedničkim prefiksom i ekstenzijom te varijabilnim rednim brojem. U ovom radu, prva slika lijeve kamere nazvana je *left0000.jpg* a zadnja *left0014.jpg*. Nakon unošenja prefiksa, ukupnog broja slika i oznake za ekstenziju slike se učitavaju u memoriju te se odabirom opcije *Extract grid corners* otvara sučelje za označavanja kuteva na uzorku. Potrebno je na svim slikama označiti četiri vanjska kuta istim redoslijedom, na primjer, uvijek počevši od gornjeg lijevog kuta. Za prvu sliku potrebno je unijeti dimenzije kvadrata planarnog uzorka (u ovom slučaju iznose 28 × 28 milimetara). Alat nakon toga sam pronalazi preostale kuteve. Ukoliko pronađeni kutevi nisu blizu stvarnim lokacijama na slici, alat u kasnijoj fazi optimizacije sam nalazi parametre distorzije a i omogućava unos početne vrijednosti distorzije. Slika 19 prikazuje pronađene kuteve uz faktor distorzije ***kc*** iznosa `-0.2`:



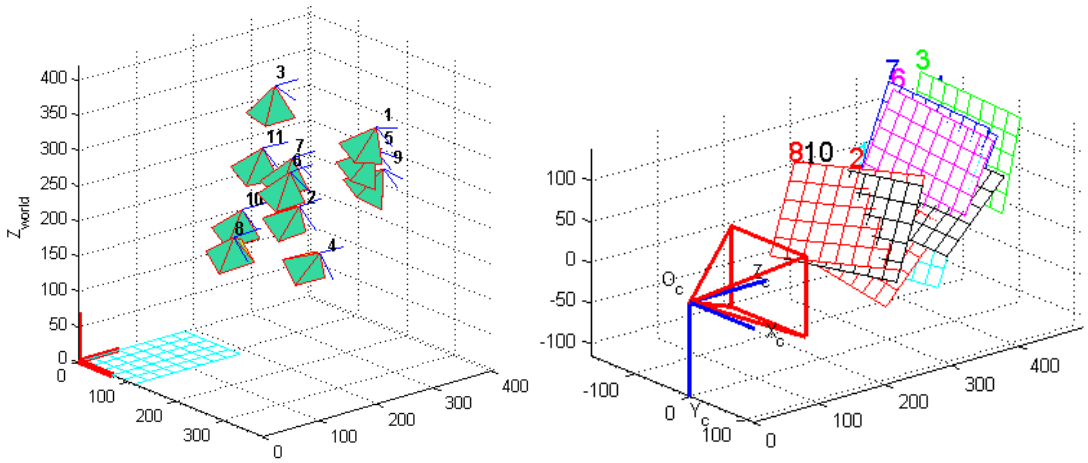

Slika 19 - označeni kutevi na planarnom uzorku

Kalibracija se pokreće odabirom opcije *Calibration*. Postupak konvergira u dvadesetak koraka te kao izlaz daje intrinsične i ekstrinsične parametre kamere: žarišnu duljinu, osnovnu točku, koeficijent ukošenosti (engl. *skew*) te koeficijente koji opisuju distorziju leće, uz mjere nesigurnosti za svaki parametar. Parametri dobiveni za lijevu GoPro kameru prikazani su u sljedećoj tablici:

Tablica 1 - intrinsični parametri lijeve *GoPro* kamere

| | |
|---|---|
| Žarišna duljina | [ 1227.06730 1238.36777 ] ± [ 3.13885 3.08437] |
| Osnovna točka | [ 985.41950 524.61385 ] ± [ 6.34657 5.07427 ] |
| Koeficijent ukošenosti | Ne evaluira se po pretpostavljenim postavkama, pretpostavlja se da je jednak 0 |
| Koeficijenti distorzije | [ -0.34811 0.14447 0.00051 0.00057 0.0 ] ± [ 0.00526 0.01168 0.00057 0.00041 0.0] |

Alat nudi i mogućnost prikaza ekstrinsičnih parametara koji su jedinstveni za svaku sliku, a određuju relativan položaj uzorka u odnosu na kameru. Odabirom opcije *Show extrinsics* prikazuje se sučelje u obliku 3D prikaza, s obzirom na koordinate svijeta ili koordinate kamere. Slika 20 prikazuje ta dva slučaja:




Slika 20 - ekstrinsični parametri lijeve *GoPro* kamere

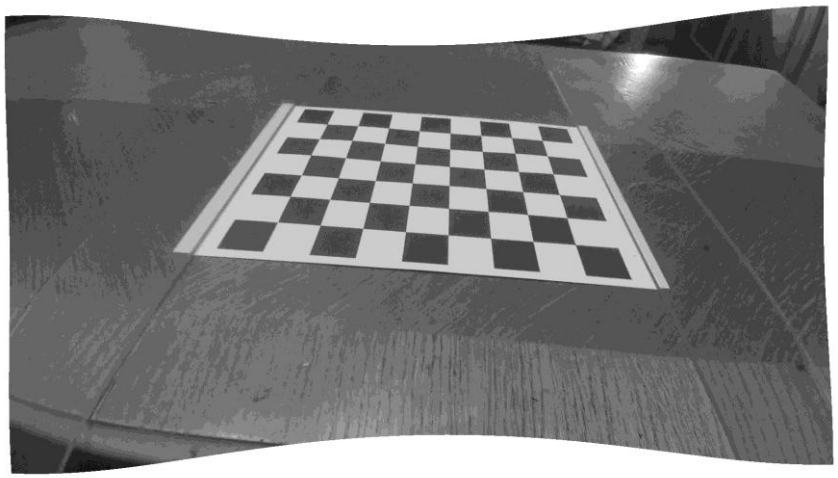
Ovime je postupak kalibracije lijeve kamere gotov te ga je potrebno ponoviti i za desnu kameru. Nakon toga potrebno je provesti postupak stereo kalibracije, odnosno određivanje ekstrinsičnih parametara stereo sustava – translacijskog vektora i rotacijske matrice između dviju kamera. Pozivom naredbe *stereo_gui* otvara se sučelje za stereo kalibraciju. Prvi korak je odabir opcije *Load left and right calibration files* čime se učitavaju intrinsični parametri dviju kamera, iz zasebnih datoteka, dobiveni ranije spomenutim postupkom. Sljedeći korak je odabir opcije *Run stereo calibration* čime se pokreće optimizacijski postupak

stereo kalibracije. Rezultati stereo kalibracije sustava *GoPro* prikazani su u sljedećoj tablici:

Tablica 2 - ekstrinsični parametri stereo sustava *GoPro*

| | |
|---|---|
| Rotacijski vektor | [ -0.00131 -0.00931 -0.00411 ]<br>±[ 0.00243 0.00438 0.00023 ] |
| Translacijski vektor | [ -34.17533 -0.26682 -0.37314 ]<br>± [ 0.11166 0.10309 0.44629 ] |

Rotacijska matrica ***R*** se iz rotacijskog vektora ***om*** može dobiti pozivom funkcije `R = rodrigues(om)`, koja implementira Rodriguesovu formulu [13]. Alat dodatno nudi opciju rektificiranja kalibracijskih slika. Slika 21 prikazuje primjer rektificirane lijeve kalibracijske slike:



Slika 21 - rektificirana kalibracijska slika

## 4.2. Biblioteka libviso2

Biblioteka libviso2 (engl. *Library for Visual Odometry 2*), pisana u jeziku *C++* s mogućnošću pozivanja iz programskog jezika *Matlab*, služi za izračun vizualne odometrije mono ili stereo sustava. U izvornom obliku zahtijeva da ulazne slike budu rektificirane, a u sklopu ovog rada prilagođena je da radi i s nerektificiranim slikama. Također, u izvornom obliku učitava slike pomoću biblioteka *libpng* i

*libpng++*, no u sklopu ovog rada prilagođena je da koristi biblioteku *OpenCV* za učitavanje i prikaz slika, iscrtavanje značajki te pomoćne funkcije za rektifikaciju.

Upute za instalaciju biblioteke *libviso2* na operacijskom sustavu *Windows* i *Microsoft Visual Studio*:

1. Pokrenuti alat *CMake GUI*
2. Postaviti direktorij na vršni direktorij iz ekstrahirane *libviso2* arhive
3. Odabrati opciju *configure*, ponovno *configure* i *generate*
4. Otvoriti novonastalu *.sln* datoteku pomoću *MS Visual Studia*
5. Prebaciti se u *Release* mod te *BUILD ALL*
6. Pokrenuti *viso2.exe*

Ukoliko se želi pokrenuti program s proizvoljnom sekvencom, potrebno je u datoteci *demo.cpp* promijeniti intrinsične i ekstrinsične parametre kamere stereo sustava. Za osnovne parametre potrebne za vizualnu odometriju s rektificiranim slikama dovoljno je popuniti sljedeću strukturu:

Odsječak koda 2 - struktura s osnovnim parametrima stereo sustava

```
param.calib.f;    // žarišna duljina u pikselima
param.calib.cu;   // osnovna točka (u-koordinata) u pikselima
param.calib.cv;   // osnovna točka (v-koordinata) u pikselima
param.base;       // udaljenost između dviju kamera u metrima
```

Za rad s nerektificiranim slikama, potrebno je postaviti `param.input_rectified = false` te dodati sve parametre dobivene postupkom stereo kalibracije iz *Matlaba*. Tada se, ovisno o parametru `param.rectify_features`, koristeći pomoćne funkcije biblioteke *OpenCV,* računaju pozicije rektificiranih značajki ili se rektificiraju cijele slike, što je opisano u sljedećem poglavlju.

Zatim slijedi instanciranje primjerka razreda *VisualOdometryStereo* te prolazak po svim vremenskim okvirima i računanje vizualne odometrije:

Odsječak koda 3 - instanciranje objekta *VisualOdometryStereo* i prolazak po svim vremenskim okvirima

```
VisualOdometryStereo viso(param);
for (int32_t i=1; i<frames; i++) {
   if (viso.process(left_img_data,right_img_data,dims)) {
       pose = pose * Matrix::inv(viso.getMotion());
   }
}
```

Razred *VisualOdometryStereo*, osim metode `getMotion()`, koja vraća matricu pomaka, nudi sljedeće metode:

- `bool process (uint8_t *I1,uint8_t *I2,int32_t* dims,bool replace=false)`
  - procesiranje novog vremenskog okvira
  - nakon drugog poziva daje ispravnu estimaciju gibanja
  - `I1` i `I2` su pokazivači na lijevu i desnu sliku, `dims` su dimenzije
- `std::vector<Matcher::p_match> getMatches ()`
  - vraća vektor svih korespondencija
- `int32_t getNumberOfMatches ()`
  - vraća broj korespondencija
- `int32_t getNumberOfInliers ()`
  - vraća broj unutarpopulacijskih značajki (engl. *inliera*)
- `std::vector<int32_t> getInlierIndices ()`
  - vraća indekse *inliera* unutar vektora svih korespondencija

Razred *Matcher* sadrži funkcionalnosti potrebne za pronalazak značajki, a unutar njega postoji struktura `p_match` koja predstavlja samu značajku:

Odsječak koda 4 - struktura `p_match` koja predstavlja pronađenu značajku

```
struct p_match {
    float   u1p,v1p; // u,v-koordinate u prošloj lijevoj slici
    float   u2p,v2p; // u,v-koordinate u prošloj desnoj slici
    float   u1c,v1c; // u,v-koordinate u trenutnoj lijevoj slici
    float   u2c,v2c; // u,v-koordinate u trenutnoj desnoj slici
 };
```

Nedostatak biblioteke *libviso2* je nemogućnost čitanja slika s kamere *Bumblebee2* u stvarnom vremenu. Slike je prvo potrebno pribaviti drugim softverom te razdvojiti lijeve i desne slike u zasebne datoteke.

Za prikaz rezultata estimacije gibanja koriste se skripte napisane u jeziku *Matlab* koje pozivaju omotače (engl. *wrapper*) funkcija biblioteke *libviso2* te su opisane u potpoglavlju 4.4.

## 4.3. Biblioteka OpenCV

Biblioteka *OpenCV* (engl. *Open Computer Vision library*) je biblioteka otvorenog tipa (engl. *open source*) sa preko 2500 funkcija vezanih uz računalni vid, obradu slike i strojno učenje. Podržana je na svim značajnijim operacijskim sustavima: *Windows*, *Linux*, *Mac OS*, *Android* te *iOS*. Neki od glavnih modula su:

- *cxcore* – funkcionalnost jezgre
- *cv* – procesiranje slika i računalni vid
- *highgui* – korisničko sučelje te čitanje i zapis slika
- *ml* – modul za strojno učenje

Instalacija je moguća preuzimanjem gotove, prevedene verzije za odgovarajući operacijski sustav ili preuzimanjem izvornog koda i prevođenjem za neku drugu arhitekturu, prateći upute sa službene stranice [14]. U sklopu ovog rada preuzeta je gotova biblioteka verzije 2.3 prevedena za *MS Visual Studio*. Integracija sa bibliotekom *libviso2* provodi se na sljedeći način:

1. U projektu *viso* unutar *solutionalibviso2.sln* odabrati *Properties*
2. Pod *Configuration* odabrati *All Configurations*
3. Unutar *Configuration Properties* odabrati sekciju *VC++ directories* i u polje *Include Directories* dodati sljedeće tri putanje unutar *OpenCV* direktorija:
    a. *build**include*
    b. *build**include**opencv*
    c. *build**include**opencv2*
4. U polje *Library Directories* dodati putanju do *build**x64**vc10**lib* direktorija unutar *OpenCV* direktorija (za 64-bitni sustav i *Visual Studio 2010*)

5. U kategoriji Linker – Input u polje Additional Dependencies dodati sljedeće module biblioteke:
    a. *opencv_core230.lib*
    b. *opencv_highgui230.lib*
    c. *opencv_calib3d230.lib*
    d. *opencv_imgproc230.lib*

Ovime je povezivanje biblioteka gotovo te je moguće korištenje funkcija biblioteke *OpenCV* unutar biblioteke *libviso2*. U sklopu ovog rada korištene su sljedeće *OpenCV* funkcije i razredi:

- `namedWindow("ime prozora", CV_WINDOW_AUTOSIZE)` – stvaranje novog prozora za prikaz slike
- `Mat(brojRedaka,brojStupaca,CV_64F,poljeZaInicijalizaciju)` - instanciranje nove matrice zadanih dimenzija i početnih vrijednosti iz polja
- `CvMat matricaCvMat = matricaMat` – plitko kopiranje primjerka razreda *Mat* u primjerak razreda *CvMat* iz razloga što neke funkcije zahtjevaju referencu na objekt tipa *CvMat*
- `int cvRodrigues2(const CvMat* src, CvMat* dst, CvMat* jacobian=0)`– funkcija za pretvaranje rotacijskog vektora u rotacijsku matricu pomoću Rodriguesove formule
- `CvSize velicina = cvSize(sirina, visina)` – stvaranje objekta koji predstavlja dimenzije slike ili prozora
- `cvStereoRectify()` – služi za računanje parametara potrebnih za rektifikaciju slika. Ulazi su matrice obiju kamera, koeficijenti distorzije, dimenzije slike te rotacijska i translacijska matrica stereo sustava, a izlazi su rotacijske i projekcijske matrice za lijevu i desnu sliku

```
void cvStereoRectify(
       const CvMat* cameraMatrix1, const CvMat* cameraMatrix2,
       const CvMat* distCoeffs1, const CvMat* distCoeffs2,
       CvSize imageSize,
       const CvMat* R,
       const CvMat* T,
       CvMat* Rl, CvMat* Rr,
       CvMat* Pl, CvMat* Pr,
       CvMat* Q=0,
       int flags=CV_CALIB_ZERO_DISPARITY
);
```

- `cvInitUndistortRectifyMap()` – služi za računanje mapa slikovnih elemenata koje se koriste za stvaranje rektificiranih slika. Ulazi su matrica kamere, koeficijenti distorzije te rotacijska matrica i projekcijska matrica koje se dobiju kao izlazi funkcije `cvStereoRectify()`. Izlazi su mape koje za svaki piksel rektificirane slike sadrže koordinate s kojih se iz originalne slike uzima vrijednost piksela.

```
void cvInitUndistortRectifyMap(
    const CvMat* M, const CvMat* distCoeffs,
    const CvMat* Rrect,
    const CvMat* Mrect,
    CvArr* mapx,  CvArr* mapy
);
```

- `void cvRemap(const CvArr* src, CvArr* dst, const CvArr* mapx, const CvArr* mapy)` – rektificira piksele polja src koristeći mape dobivene funkcijom `cvInitUndistortRectifyMap()`
- `IplImage* slika = cvLoadImage(ime,CV_LOAD_IMAGE_GRAYSCALE)` – učitavanje slike
- `imshow("identifikator_prozora", slika)` – prikaz slike

## 4.4. Skripte u programskom jeziku Matlab

Skripte u programskom jeziku *Matlab* korištene su za rektifikaciju slika uz prilagodbu skripti alata *Camera Calibration Toolbox for Matlab*, za prikaz rezultata estimacije gibanja iz biblioteke *libviso2* te za prikaz trajektorije dobivene geolokacijskim sustavom.

Alat *Camera Calibration Toolbox for Matlab* nudi funkcionalnost za rektifikaciju kalibracijskih slika, no ne i za rektifikaciju proizvoljnih slika sekvence. Kako bi se rektificirala proizvoljna slika, potrebno je u ljudki *Matlaba* pozicionirati se u direktorij koji sadrži rezultate kalibracije te pokrenuti skriptu *rectify_init.m* za učitavanje rezultata. Odsječak koda 5 prikazuje dio skripte za rektifikaciju skupa slika nakon učitavanja kalibracijskih parametara:

Odsječak koda 5 - dio skripte za rektifikaciju skupa slika

```
for i = 0 : broj_slika,
    imL = imread(['seq_left/left-' num2str(i,'%04d') '.pgm']);
    imR = imread(['seq_right/right-' num2str(i,'%04d') '.pgm']);
    imL = double(imL);
    imR = double(imR);
```

```
    [imL,imR] = rectify(imL, imR);

    imwrite(imL, ['rectified/left' num2str(i,'%04d') '.jpg'], 'jpg');
    imwrite(imR, ['rectified/right' num2str(i,'%04d') '.jpg'], 'jpg');
end
```

Kako bi se omogućilo pozivanje funkcija biblioteke *libviso2* iz *Matlaba*, potrebno je izgraditi omotače (engl. *wrapper*) tih funkcija. Prvo je potrebno naredbom `mex -setup` u *Matlabu* odabrati željeni prevodioc – u ovom slučaju *MS Visual Studio 2010 compiler*. Nakon toga potrebno je pozicionirati se u direktorij *matlab* unutar *libviso2* direktorija te pokrenuti skriptu *make.m*, koja će izgraditi potrebne omotače funkcija u četiri zasebne datoteke: *matcherMex.cpp*, *reconstructionMex.cpp*, *visualOdometryMonoMex.cpp* i *visualOdometryStereoMex.cpp*. Sada je moguće pokrenuti skriptu *viso_stereo.m*, koja je prikazana u sljedeća dva odsječka koda:

Odsječak koda 6 - inicijalizacija parametara skripte *viso_stereo.m*

```
% inicijalizacija parametara
img_dir     = 'D:\FER\diplrad\bumblebee1\seq2\rectified';
param.f     = 425.3849;
param.cu    = 325.9;
param.cv    = 244.54275;
param.base  = 0.12;
first_frame = 0;
last_frame  = 2412;

% inicijalizacija vizualne odometrije
visualOdometryStereoMex('init',param);

% inicijalizacija transformacijske matrice
Tr_total{1} = eye(4);

% stvaranje figure-a
figure('Color',[1 1 1]);
ha1 = axes('Position',[0.05,0.7,0.9,0.25]);
axis off;
ha2 = axes('Position',[0.05,0.05,0.9,0.6]);
set(gca,'XTick',-500:1:500);
set(gca,'YTick',-500:1:500);
axis equal, grid on, hold on;
```

Odsječak koda 7 - dio skripte *viso_stereo.m* za iteraciju po slikama, prikaz trajektorije i značajki

```
for frame=first_frame:last_frame

 % 1-index
 k = frame-first_frame+1;
```

```
 % čitanje slika
 I1 = imread([img_dir '/left' num2str(skip,'%04d') '.jpg']);
 I2 = imread([img_dir '/right' num2str(skip,'%04d') '.jpg']);

 % računanje gibanja, nakon drugog frame-a
 Tr = visualOdometryStereoMex('process',I1,I2);
if k>1
    Tr_total{k} = Tr_total{k-1}*inv(Tr);
end

% prikaz trajektorije
axes(ha2);
if k>1
    plot([Tr_total{k-1}(1,4) Tr_total{k}(1,4)], ...
         [Tr_total{k-1}(3,4) Tr_total{k}(3,4)],'-xb','LineWidth',1);
end
  pause(0.05); refresh;

 % ispis broja značajki i inliera
 num_matches = visualOdometryStereoMex('num_matches');
  num_inliers = visualOdometryStereoMex('num_inliers');
  disp(['Frame: ' num2str(frame) ...
', Matches: ' num2str(num_matches) ...
', Inliers: ' num2str(100*num_inliers/num_matches,'%.1f') ,' %']);


 % iscrtavanje značajki
if k>1
         p_matched = visualOdometryStereoMex('get_matches');
         axes(ha1); cla;
         plotMatch(I1,p_matched,2);
         axis off;
end
end
```

Skripta *gpsPlot.m* korištena je za prikaz trajektorije izlaza geolokacijskog sustava:

Odsječak koda 8 - skripta *gpsPlot.m*, za prikaz trajektorije *GPS* sustava

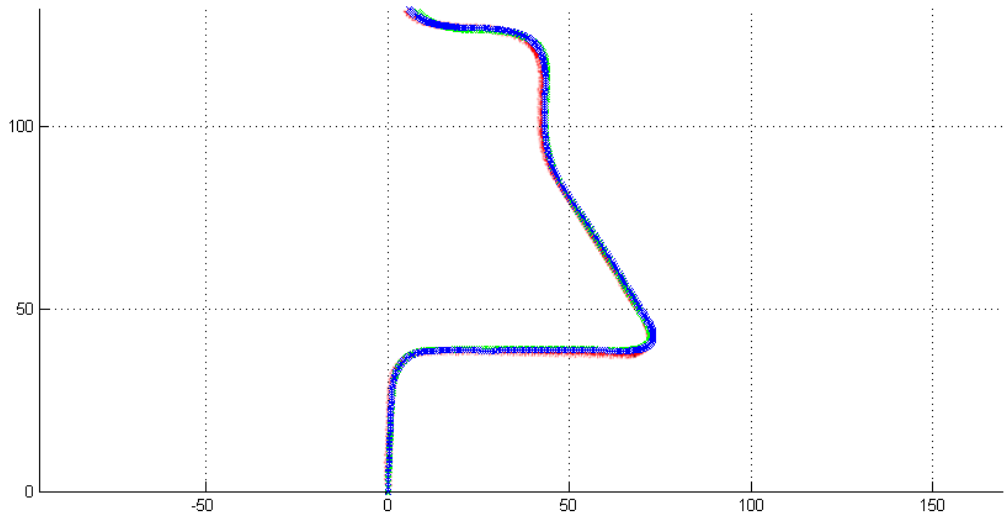
```
fid = fopen('putanja\do\insdata.txt');
[A count] = fscanf(fid, '%lf', 10);
B = A;
while true
[A count] = fscanf(fid, '%lf', 10);

% iscrtavanje linije
  plot([B(5) A(5)], [B(6) A(6)],'-xr','LineWidth',1);
  pause(0.000005);
  refresh;
  B = A;
end
```

# 5. Eksperimentalni rezultati

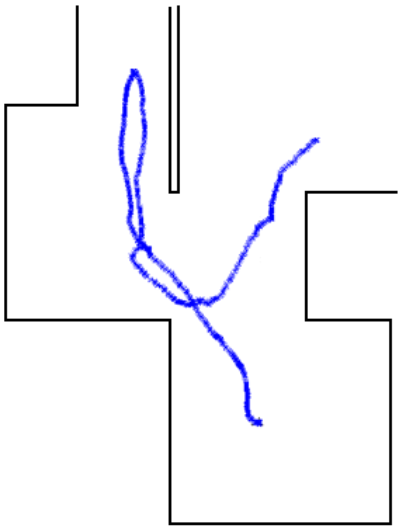
U ovom poglavlju prikazani su rezultati estimacije gibanja nad tri ispitna skupa slika te još nekoliko provedenih eksperimenata. Prvi ispitni skup (*Karlsruhe* skup) korišten je za provjeru rada biblioteke *libviso2*. Za taj skup postoje rezultati mjerenja geolokacijskog sustava (engl. *groundtruth*).

Za ovu sekvencu proveden je još jedan eksperiment. Budući da je veličina prozora za pronalazak značajki veličine 5×5 slikovnih elemenata, značajke koje se koriste u ovom radu odgovaraju mikroteksturama na slici te je pretpostavka bila da bi kompresija slike mogla znatno utjecati na kvalitetu rekonstrukcije. Izvorne slike ovog skupa su u formatu *png,* u kojem nema gubitaka na kvaliteti slike uslijed kompresije. U sklopu eksperimenta su sve slike prebačene u format *jpeg* uz maksimalnu kompresiju, čime se je znatno smanjila njihova kvaliteta. Plavom bojom prikazan je rezultat rekonstrukcije gibanja za originalne slike, crvenom bojom prikazan je izlaz geolokacijskog sustava, a zelenom bojom prikazan je rezultat rekonstrukcije gibanja za kompresirane slike. Vidljivo je da su rezultati gotovo identični rezultatima postignutima sa slikama u *png* formatu, čime je pokazano da je metoda pronalaska značajki vrlo robusna na jačinu kompresije u slici.



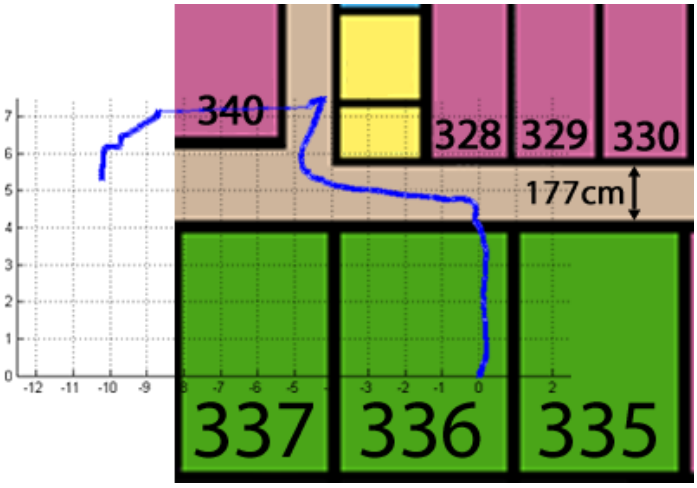

Slika 22 - rezultat vizualne odometrije sekvence *2010_03_09_drive_0019*

Drugi ispitni skup sadrži četiri sekvence čiji su rezultati prikazani na sljedećim trima slikama. Slika 23 prikazuje rezultat vizualne odometrije prve sekvence. S obzirom da je kretanje bilo ograničeno duljinom kabla od četiri metra te sadrži samo nekoliko koraka, rezultati su samo informativnog karaktera. Vidljivo je da rezultati otprilike odgovaraju procjeni kretanja koje prikazuje Slika 11.



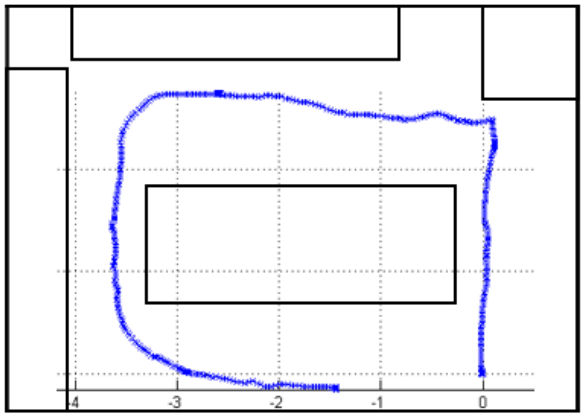
Slika 23 - rezultat vizualne odometrije sekvence *bumblebee_stan*

Druga sekvenca sadrži kretanje zavodskim hodnikom, a rezultati su prikazani na sljedećoj slici. Vidljivo je da je mjerilo dobro procijenjeno, kao i kretanje hodnikom, sve do trenutka okreta od 180 stupnjeva, gdje je vidljiv samo bijeli zid te je sustav izgubio sve značajke, zbog čega je estimacija kretanja u potpunosti pogrešna.



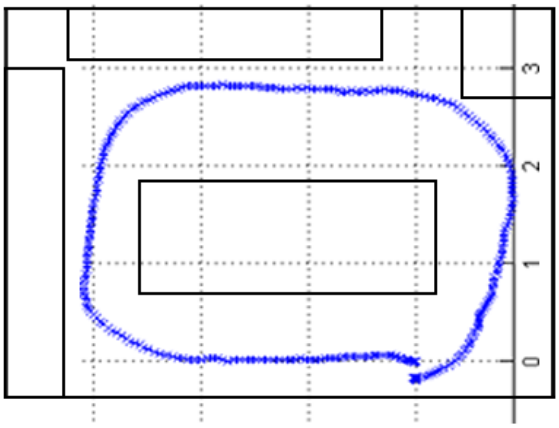

Slika 24 - rezultat vizualne odometrije sekvence *bumblebee_zemris*

Treća sekvenca sadrži pravokutno kretanje zavodskim kabinetom, a rezultati su prikazani na sljedećoj slici. Vidljivo je da je kretanje dobro procijenjeno.



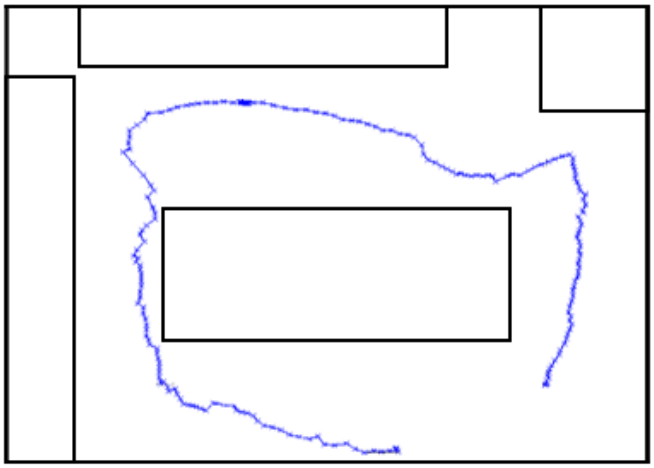

Slika 25 - rezultat vizualne odometrije sekvence *bumblebee_kabinet*

Četvrta sekvenca sadrži kretanje po istoj prostoriji kao i treća sekvenca, ali u obrnutom smjeru, s razlikom da su početna i završna točka približno jednake. Na sljedećoj slici prikazani su rezultati:




Slika 26 - rezultat vizualne odometrije *bumblebee_kabinet2*

Zadnji ispitni skup sadrži sekvencu s kretanjem jednakim kretanju sekvence *bumblebee_kabinet*, no snimljen je sustavom *GoPro HD* te sadrži znatne količine trzanja uslijed gibanja kolica. Na sljedećoj slici vidljivo je da je rezultat vizualne odometrije te sekvence znatno lošiji nego rezultat postignut kamerom *Bumblebee2*. Bolji rezultati mogli bi se postići dodavanjem podrške za stabilizaciju video sekvence.



Slika 27 - rezultat vizualne odometrije sekvence gopro_kabinet

Zadnji eksperiment je usporedba rezultata rektificiranja pojedinih značajki sa rezultatima rektificiranja cijelih slika. Slika 28 prikazuje jednu istaknutu značajku čija je lokacija nakon rektifikacije jednaka lokaciji u rektificiranoj slici s uklonjenom distorzijom leće. Isto vrijedi i za sve ostale značajke.

Slika 28 - usporedba rektificiranih značajki s rektificiranom slikom s uklonjenom distorzijom leće

# 6. Zaključak

Stereo vizualna odometrija je postupak određivanja pozicije i orijentacije stereo sustava analizom slijeda parova slika. Prije vizualne odometrije potrebno je obaviti kalibraciju kamera, odnosno odrediti intrinsične i ekstrinsične parametre stereo sustava. Zatim je potrebno ulazne slike rektificirati, čime se znatno smanjuje složenost problema pronalaska korespondencija. Druga mogućnost je rektifikacija samo pronađenih značajki. Iz pronađenih parova značajki se metodom triangulacije određuje 3D pozicija svake značajke te se na temelju 3D značajki određuje vizualna odometrija.

U sklopu ovog rada korišten je brz postupak iz biblioteke *libviso2* baziran na jednostavnim značajkama, čiji rezultati uvelike ovise o točnosti kalibracije i rektifikacije. Također, detaljno su opisani postupci kalibracije stereo sustava te rektifikacije slika. U biblioteku *libviso2* dodana je mogućnost za rad s nerektificiranim ulazima uz poznatu kalibraciju stereo sustava, na način da se rektificiraju cijele slike ili samo značajke.

Rezultati nad javno dostupnim sekvencama pribavljenim iz perspektive vozača su vrlo dobri, dok su rezultati nad sekvencama pribavljenim u sklopu ovog rada nešto lošiji, djelomično zbog manjka značajki u pojedinim dijelovima sekvenci (npr. bijeli zid hodnika) te zbog trzanja kamere za vrijeme snimanja. Može se zaključiti da postupak daje bolje rezultate u prirodnim okolinama koje su bogate značajkama, u odnosu na laboratorijske uvjete. Također, pokazano je da je postupak vrlo robustan na kvalitetu ulaznih slika.

Budući rad uključivao bi dodavanje postupka stabiliziranja slike u sustav, čime bi se poboljšali rezultati vizualne odometrije u stvarnim uvjetima. Još jedna mogućnst je dodavanje podrške za praćenje značajki kroz više vremenskih okvira, čime bi se dodatno eliminirale vanpopulacijske značajke. Dosadašnji rad polučio je dobre rezultate te pruža osnovu za daljnje istraživanje.

# 7. Literatura


1. Bumblebee2 CCD FireWire camera, Point Grey research, http://www.ptgrey.com/products/bumblebee2/
2. GoPro 3D HERO System, Woodman Labs Inc., http://gopro.com/hd-hero-accessories/3d-hero-system/
3. Bradski, G; Kaehler, A*: "Learning OpenCV," O'Reilly Media, Inc.*, rujan 2008.
4. Bouguet, J.-Y:"*Camera Calibration Toolbox for Matlab,*" 2010., http://www.vision.caltech.edu/bouguetj/calib_doc
5. Tsai, R. Y: "*A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV camera and lenses," IEEE Journal of Robotics and Automation 3*, 1987.
6. Zhang, R: *"Flexible camera calibration by viewing a plane from unknown orientations," Proceedings of the 7th International Conference on Computer Vision*, Corfu, rujan 1999.
7. Geiger, A; Ziegler, J; Stiller, C: "*LIBVISO2: C++ Library for Visual Odometry 2*," 2011., http://www.cvlibs.net/software/libviso2.html
8. Geiger, A; Ziegler, J; Stiller, C: "*StereoScan: Dense 3d Reconstruction in Real-time*," *IEEE Intelligent Vehicles Symposium*, Baden, lipanj 2011.
9. Fischler, M. A; Bolles, R. C: "*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.
10. Geiger, A: "*Karlsruhe Dataset: Stereo Video Sequences + rough GPS Poses,*" Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2010., http://www.cvlibs.net/datasets/karlsruhe_sequences.html
11. Flea2 CCD FireWire 800 Cameras, Point Grey research, http://www.ptgrey.com/products/flea2/
12. Vodič po Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave, tlocrt, http://www.fer.unizg.hr/zemris/o_nama
13. Belongie, S*: "Rodrigues' Rotation Formula," MathWorld - A Wolfram Web Resource created by Eric W. Weisstein.*, http://mathworld.wolfram.com/RodriguesRotationFormula.html
14. Emami, S: "*OpenCV Installation Guide,"* 18. svibnja 2012., http://opencv.willowgarage.com/wiki/InstallGuide
15. Georgiev, Y; Maxwell, B: "*Stereo visual odometry,"* 6. svibnja 2006.
16. Kitt, B; Geiger, A; Lategahn, H: *"Visual Odometry based on Stereo Image Sequences with RANSAC-based Outlier Rejection Scheme," IEEE Intelligent Vehicles Symposium*, San Diego, lipanj 2010.
17. Longuet-Higgins H. C: *"A computer algorithm for reconstructing a scene from two projections," Nature 293 (5828): 133–135.*, rujan 1981.
18. Nistér, D: *"An efficient solution to the five-point relative pose problem". IEEE Trans. on Pattern Analysis and Machine Intelligence 26 (6): 756–777.,* lipanj 2004.
19. Nistér, D; Naroditsky, O; Bergen, J: *"Visual odometry," IEEE Computer Society Conference Computer Vision and Pattern Recognition, vol. 1: 652 – 659.*, 2004.

20. Raykar, V. C: *"Motion detection,"* projektna dokumentacija, 11. svibnja 2004.

21. Šegvić, S: *"Uporaba projekcijske geometrije i aktivnog vida u tumačenju scena,"* magistarski rad, Zagreb, 2000.

22. Trucco, E; Verri, A: "Introductory Techniques for 3-D Computer Vision," *Prentice Hall,* 16. ožujka 1998.

# 8. Naslov, sažetak, ključne riječi

## Naslov

Estimiranje strukture i gibanja stereo parom kamera

## Sažetak

Osnovni cilj rada je estimiranje strukture i gibanja stereo parom kamera. Širi kontekst rada je određivanje kretanja kamere analizom slijeda pribavljenih slika. U okviru rada proučeni su algoritmi i postupci kalibracije stereo sustava, stereo rektifikacije slika, izlučivanja značajki, 3D rekonstrukcije i vizualne odometrije. Opisana je biblioteka *libviso2* korištena za vizualnu odometriju, u nju je dodana mogućnost rada s nerektificiranim slikama te je dodana mogućnost povezivanja s bibliotekom *OpenCV*. Na kraju su prezentirani eksperimentalni rezultati dobiveni nad sekvencama iz perspektive vozača te sekvencama snimljenim u sklopu ovog rada u prostorijama Fakulteta elektrotehnike i računarstva.

## Ključne riječi

Vizualna odometrija, estimacija gibanja, izlučivanje značajki, epipolarna geometrija, kalibracija kamere, stereo rektifikacija, 3D rekonstrukcija, triangulacija, biblioteka *libviso2*, biblioteka *OpenCV*

# 9. Title, abstract, key words

## Title

Structure and motion estimation using a pair of stereo cameras

## Abstract

The main goal of this thesis is structure and camera motion estimation by analysing a sequence of images obtained from a pair of stereo cameras. The following algorithms and procedures have been explained: stereo camera calibration, stereo rectification, feature extraction, 3D reconstruction and visual odometry. Library for visual odometry *libviso2* has been explained, with added support for computing visual odometry using unrectified stereo image pairs and has been linked with the *OpenCV* library. Finally, experimental results of sequences obtained from driver perspective and sequences obtained as a part of this thesis, inside the halls of Faculty of Electrical Engineering and Computing, are presented.

## Key words

Visual odometry, motion estimation, feature extraction, epipolar geometry, camera calibration, stereo rectification, 3D reconstruction, triangulation, *libviso2* library, *OpenCV* library