SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 356

Ocjena učinkovitosti postupaka za procjenu geometrije dvaju pogleda

Ivan Šakić

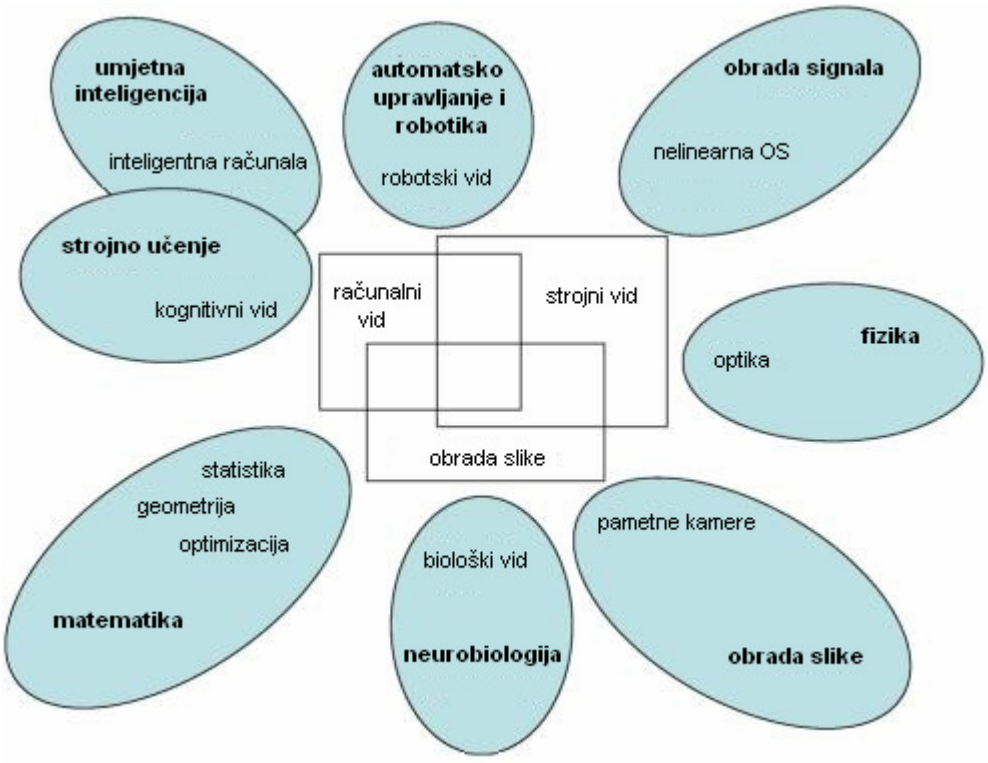
Zagreb, srpanj 2008

Sadržaj

1. Uvod

Računalni vid je podgrana računalne znanosti koja je relativno mlada i još uvijek nedovoljno istražena. Standardna formulacija problema računalnog vida još ne postoji. Računalni vid je imao mnogo napretka u posljednje vrijeme. Postoji više razloga za to, ali najvažniji su: ubrzavanje i poboljšavanje performansi računala, kontinuiran napredak teoretskih istraživanja na tom području [piccardi03].

Mnoge znanstvene discipline se dotiču ove problematike kao što su: umjetna inteligencija, strojno učenje, matematika, neurobiologija, digitalna obrada slike, fizika, digitalna obrada signala, automatsko upravljanje i robotika (slika 1.1.).



Slika 1.1. Odnos različitih znanstvenih disciplina i računalnog vida

Ubrzavanjem i poboljšavanjem performansi računala vizualizacija 3D objekata u posljednje vrijeme se sve više razvija. Javljaju se zahtjevi za kompleksnijim i realističnijim 3D modelima. Potrebe su različite: video nadzor, multimedija, sučelje čovjek-računalo, poboljšavanje percepcije, sigurnosne provjere i dr. [piccardi03]. Iako su alati za trodimenzionalno modeliranje sve moćniji, sintetiziranje realističnih modela je složeno i vremenski zahtjevno, stoga i skupo.

U ovom radu bit će opisani osnovni teoretski aspekti računalnog vida s fokusom na geometriji više pogleda u kalibriranom slučaju te rezultati i postignuća nekih postojećih implementacija.

1.1 Opis problema

Teorija računalnog vida je vrlo složena. Stoga ideja ovog rada nije bila upuštati se u detaljnu teoretsku razradu nego svladavanje teorije na informativnoj razini. Fokus rada je na povezivanju postojećih implementacija pojedinačnih postupaka koje čine rješenje problema, te na procjeni učinkovitosti rješenja. Dakle rad je na presjeku domene računalnog vida i programskog inženjerstva.

Širi kontekst je procjena geometrijskih odnosa strukture scene i pokretne kamere, na temelju slijeda slika pribavljenih kalibriranom kamerom iz vozila u pokretu. Razmatra se poopćeni stereo problem, gdje se na temelju dvije slike istovremeno utvrđuje relativna orijentacija kamera i 3D rekonstrukcija scene. U okviru rada se proučavaju sastavni postupci rješenja problema: pronalaženje korespondencija, procjena relativne orijentacije te trijangulacija strukture scene. Potrebno je povezati postojeće programske implementacije navedenih postupaka u funkcionalan sustav. Analizirati ponašanje implementiranog sustava na ispitnim sljedovima slika, te prikazati i ocijeniti ostvarene rezultate.

2. Geometrija dvaju pogleda

Kao što su mnoge pojave i zakoni u prirodi u posljednje vrijeme inspiracija istraživačima koji pokušavaju riješiti probleme u računarskoj znanosti na novi, učinkovitiji i pametniji način, tako i rješenja problema na području računalnog vida pokušavaju naći analogiju u prirodi. Analogiju je samu po sebi vrlo lako naći sa načinom ljudskog (ali i mnogih vrsta iz životinjskog svijeta) poimanja svijeta koji se opisuje kao stereo vid (eng. stereo vision).

Dva oka=tri dimenzije(3D)

Svako oko izdvojeno ima svoju posebnu sliku. Kada se te slike šalju u mozak na obradu istovremeno one se pokušavaju spojiti u jednu sliku. Da bi se mogla dobiti jedna slika u mozgu se odvijaju operacije uočavanja sličnosti i razlika na dobivenim slikama koje nam daju mnogo više informacija i na osnovu njih se dobije jedna trodimenzionalna slika.

Naravno, ne postoje jaki razlozi da bismo implementaciju računalnog vida oblikovali kao onog kojeg nalazimo u prirodi (biološkog), prije svega jer još uvijek je mnogo nepoznanica vezanih za vizualnu percepciju kod ljudi i životinja. No, treba uzeti u obzir da ni pokušaji stvaranja umjetnog modela percepcije (eng. silicon-based vision) nisu prošli nimalo uspješno [hartley04]. „Plodno tlo“ kao što obično biva se nalazi negdje u sredini, između biološkog i umjetnog modela, na korist obje discipline znanosti.

U geometriji dvaju pogleda imamo tri međusobno isprepletena cilja: (i) određivanje korespodentnih točaka, (ii) određivanje pomaka kamera te (iii) određivanje 3D strukture.

Također treba navesti dva konteksta koja imaju mnogo toga zajedničkog:

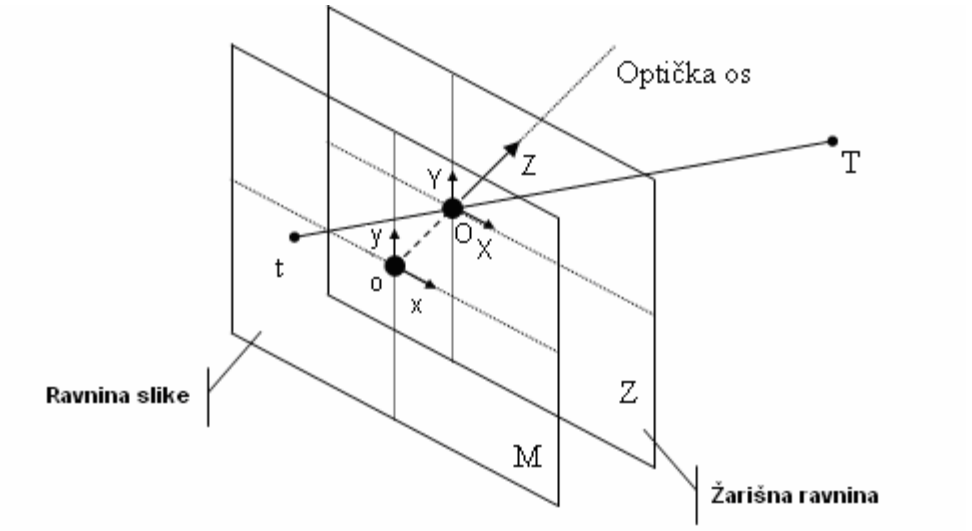
- stereo, gdje su slike pribavljene istovremeno i
- kretanje, gdje su slike pribavljene slijedno u vremenu jednom kamerom.

2.1. Osnove stereo vida

Osnovni motiv kod snimanja trodimenzionalne scene jest da iz dobivenih dvodimenzionalnih slika mjerenjima možemo odrediti poziciju, orijentaciju ili duljinu objekta u 3D sceni. Područje koje se bavi ovim problemom mjerenja iz slika se naziva fotogrametrija. Da bismo mogli izvršiti mjerenja 3D svijeta iz 2D slika potrebno je izvršiti kalibraciju kamere. Kalibracija kamere definira vezu između koordinatnog sustava svijeta i koordinatnog sustava kamere. Na temelju te veze i više pogleda na istu scenu može se onda za svaku točku u 3D prostoru odrediti polažaj na 2D slici.

2.1.1. Model kamere

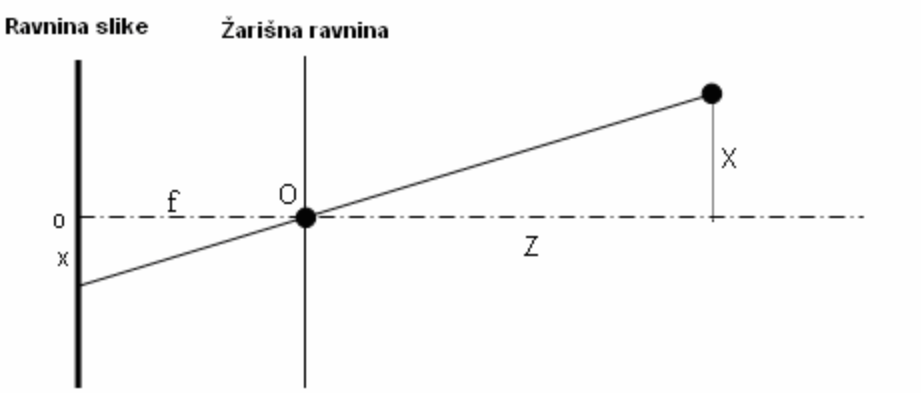
Dubinska slika predstavlja udaljenost od prizora do kamere. Elementi dubinske slike su: udaljenost do objekta, slikovni kut kamere, osvjetljenje, gibanje kamere, gibanje objekta, kontrast i dr. Pri generiranju takvih slika koriste se kamere, što znači da se kamera koristi kao mjerni instrument. Da bi je mogli koristiti na taj način potrebno je postaviti model kamere koji precizno opisuje kvantitativne odnose pri stvaranju slike. Perspektivna projekcija je jednostavan model koji te odnose dobro opisuje. Taj model je prikazan na slici 2.1.



Slika 2.1. Model stvaranja slike perspektivnom projekcijom

Model se sastoji od dvije paralelne ravnine M i Z, udaljene za udaljenost *f* koju zovemo i fokalna dužina. Ravnina Z, koju zovemo žarišna ravnina, ima u točki C rupicu kroz koju prolazi svjetlost. Ta rupica se zove optički centar. Pravac koji prolazi kroz optički centar i okomit je na ravninu slike zove se optička os. Sva svjetlost koja prođe kroz optički centar na ravnini M, koju zovemo i ravnina slike, stvara invertiranu sliku. Slika pojedine točke iz prostora dobiva se na sjecištu pravca koji prolazi kroz tu točku i optički centar, te ravnine slike. Uvođenjem koordinatnih sustava u prostoru i ravnini slike moguće je računati koordinate projekcija točaka iz prostora.

Koordinatni sustav (O,X,Y,Z) je koordinatni sustav u prostoru, dok je koordinatni sustav (o,x,y) koordinatni sustav u ravnini slike. Ako su oni postavljeni tako da je ishodište koordinatnog sustava u prostoru C u optičkom centru, a ishodište koordinatnog sustava u ravnini slike c, na mjestu gdje optička os sječe ravninu slike, onda se ti koordinatni sustavi zovu standardni koordinatni sustavi. S tako postavljenim koordinatnim sistemima vrijede odnosi prikazani na slici 2.2.



Slika 2.2. Odnosi koordinatnog sustava u prostoru i koordinatnog sustava u ravnini slike

Iz sličnosti trokuta mogu se postaviti odnosi:

$$\frac{X}{Z} = -\frac{x}{f} \qquad \frac{Y}{Z} = -\frac{y}{f}$$

a iz toga slijedi:

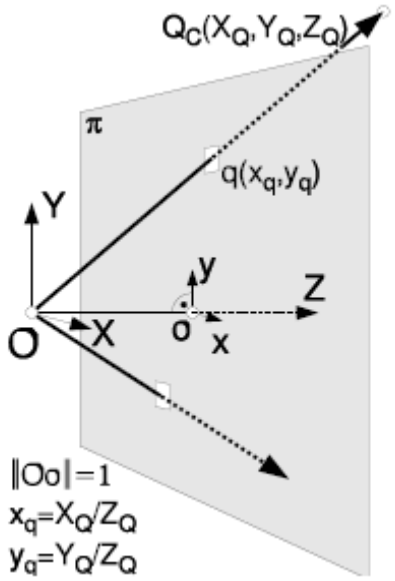
$$\Rightarrow -\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y}$$

Prikazani jednostavni model stvaranja slike dobro opisuje stvarne kamere koje se upotrebljavaju u svrhu dobivanja slika za potrebe računalnog vida. Često se koristi koordinatni sustav kod kojeg se ishodište nalazi u gornjem lijevom uglu, zbog načina pohrane slike u memoriji računala, te načina prisupa pojedinom elementu slike. Pored pomaka ishodišta, na slici se koriste drugačije jedinice od onih koje se koriste u stvarnom svijetu, pa je potrebno izvršiti pretvorbu jednih jedinica u druge.

Treba istaknuti da kamere posjeduju svoje ekstrinsične (vanjske) i intrinsične (unutarnje) parametre. Te parametre potrebno je uzeti u obzir prilikom transformacije točaka iz koordinatnog sustava svijeta u koordinatni sustav dobivene slike.

2.1.2. Model stvaranja slike i geometrijske transformacije

Zbog jednostavnosti ćemo u daljnjem tekstu slikovnu ravninu prikazivati ispred žarišta. Lako se pokaže da prednja slika odgovara stražnjoj slici zarotiranoj za 180°. U nastavku će biti prikazan primjer transformacije točke iz 3D scene i njena projekcija na ravninu slike (slika 2.3.).



Slika 2.3. Projekcija 3D točke na ravninu slike [segvic04]

Projekcijski, takvo stvaranje slike se opisuje linearnim izrazima (u opisu koristimo homogene koordinate [mihajlovic08]):

$$\lambda \cdot \begin{bmatrix} x_{\hat{q}} \\ y_{\hat{q}} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_Q \\ Y_Q \\ Z_Q \\ 1 \end{bmatrix} \qquad (2.1)$$

ili, kraće:

$$\lambda \cdot \hat{\mathbf{q}} = \mathbf{S} \cdot \mathbf{Q_C} \qquad (2.2)$$

gdje je:

- $\hat{\mathbf{q}}$ točka iz normaliziranog k.s. slike $(o, \hat{x}, \hat{y})$,
- $\mathbf{Q_C}$ točka iz k.s. kamere (O, X, Y, Z).

Ovaj osnovni model linearno proširujemo intrinsičnim parametrima kamere:

$$\begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{\hat{q}} \\ y_{\hat{q}} \\ 1 \end{bmatrix} \qquad (2.3.)$$

ili, kraće:

$$\mathbf{q} = \mathbf{K} \cdot \hat{\mathbf{q}} \qquad (2.4.)$$

gdje su:

- s_x i s_y dimenzije senzorskih elemenata,
- t_x i t_y pomak koordinatnog sustava slike,
- $s_\theta = \tan(\theta) * s_y$ odstupanje od pravog kuta među osima koorinatnog sustava slike (obično je s_θ približno 0).

Homogene koordinate omogućuju linearan opis Euklidskih geometrijskih transformacija sa šest stupnjeva slobode:

$$\begin{bmatrix} X_Q \\ Y_Q \\ Z_Q \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{t}_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} \cdot \begin{bmatrix} L_Q \\ M_Q \\ N_Q \\ 1 \end{bmatrix} \qquad (2.5.)$$

Ili, kraće:

$$\mathbf{Q_C} = \mathbf{M} \cdot \mathbf{Q_W} \qquad (2.6.)$$

gdje je:

- Q_C točka iz k.s. kamere (O, X, Y, Z),
- R rotacija k.s. kamere oko osi k.s. svijeta,

- t translacija k.s. svijeta u k.s. kamere

Q_W točka iz k.s. svijeta (O_w , L, M, N).

Ekstrinsični (vanjski) parametri kamere su položaj kamere u koordinatnom sustavu svijeta, opisuju se pomoću matrice rotacije R i vektora translacije t. Matrica rotacije R ima 3 stupnja slobode, može se opisati kao kombinacija 3 elementarne rotacije. Čest način dekompozicije R se temelji na Eulerovim kutevima (α, β, γ), kako slijedi:

-rotacija oko osi x za kut α, opisuje se pomoću matrice Rx

-rotacija oko osi y za kut β, opisuje se pomoću matrice Ry

-rotacija oko osi z za kut γ, opisuje se pomoću matrice Rz

Ukupna rotacija R se izražava pomoću umnoška:

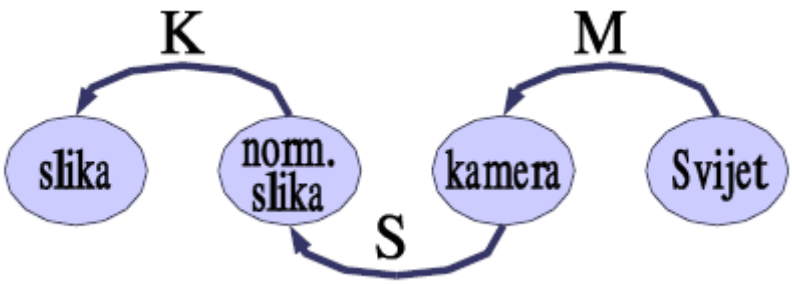
$$\mathbf{R} = \mathbf{R_z}(\gamma) \cdot \mathbf{R_y}(\beta) \cdot \mathbf{R_x}(\alpha) \quad (2.7.)$$

$$\mathbf{R} = \begin{bmatrix} \cos(\beta)\cos(\gamma) & \cos(\gamma)\sin(\alpha)\sin(\beta) - \cos(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\gamma)\sin(\beta) + \sin(\alpha)\sin(\gamma) \\ \cos(\beta)\sin(\gamma) & \cos(\alpha)\cos(\gamma) + \sin(\alpha)\sin(\beta)\sin(\gamma) & \cos(\gamma)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\alpha) & \cos(\alpha)\cos(\beta) \end{bmatrix} \quad (2.8.)$$

Te konačno ukupna transformacija:

$$\lambda \mathbf{q} = \mathbf{K} \cdot \mathbf{S} \cdot \mathbf{M} \cdot \mathbf{Q_W} = \mathbf{P}_{3\times 4} \cdot \mathbf{Q_W} \quad (2.9.)$$

Slikoviti prikaz postupka transformacija točaka iz koordinatnog sustava svijeta u koordinatni sustav dobivene slike nalazi se na slici 2.4.

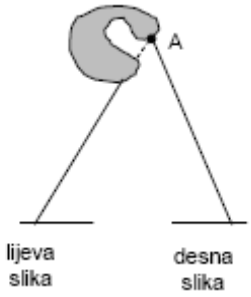


Slika 2.4. Postupak transformacija točaka iz koordinatnog sustava svijeta u koordinatni sustav dobivene slike [segvic04]

2.2. Pronalaženje korespodencija

Najvažniji i najteži problem za rekonstrukciju 3D objekata iz stereo slika je određivanje korespodentnih točaka. Korespodentne točke su točke na ravninama slika koje su nastale od iste fizičke točke na objektu kojeg promatramo. U nekim slučajevima se i ne može riješiti, npr.:

- ako imamo objekt koji ima uniformnu svjetlinu bez ikakvih karakterističnih točaka teško je naći korespodentan par točaka
- drugi mogući problem je zaklanjanje točaka (npr. točka A vidljiva iz jednog pogleda nije vidljiva iz drugog pogleda)(slika 2.5.)



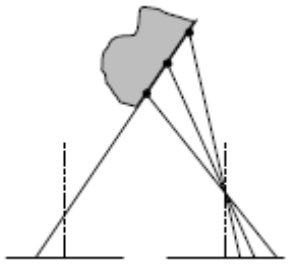
Slika 2.5. Točka A na objektu je zaklonjena u lijevom pogledu, tj.nije moguće naći točku koja korespondira točki A u desnoj slici [loncaric08]

Da bi se ovi problemi mogli riješiti često se koriste dodatni uvjeti tj. ograničenja:

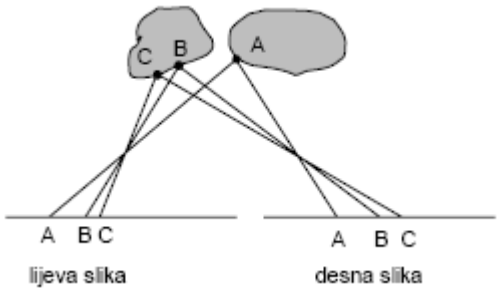
- ograničenje jedinstvenosti korespodencije (slika 2.6.)
- fotometrička kompatibilnost: pikseli imaju u oba pogleda slične svjetline (izobličenje koje se javlja je fotometrička distorzija koja se događa zbog različitog kuta refleksije svjetla s površine objekta)
- geometrijska sličnost: geometrijske strukture u oba pogleda su slične (npr. duljina ili orijentacija linija, regija ili kontura)
- očuvanje poretka točaka (slika 2.7.)

Dodatna ograničenja mogu biti npr.:

- epipolarna linija
- glatkoća dispariteta i dr.



Slika 2.6. U najvećem broju slučajeva točka u lijevoj slici može odgovarati samo jednoj točki u desnoj slici, ova slika nam pokazuje izuzetak ovog pravila [loncaric08]



Slika 2.7. Za površine podjednako udaljene od kamere, karakteristične točke imaju isti poredak u oba pogleda [loncaric08]

Na temelju navedenih i drugih ograničenja razvijen je veći broj metoda za rješavanje problema korespodentnih točaka, te možemo navesti dva glavna pristupa:

- korelacijski pristup
- pristup temeljen na invarijantnim značajkama slike

Kod korelacijskog pristupa najčešće se koriste prozori s kojim se pretražuje slika i računa se sličnost regija u lijevoj i desnoj slici dok se ne nađe najveći odziv. Najveći problem kod ovog pristupa je uzrokovan pojavom šuma.

Pristup temeljen na invarijantnim značajkama (algoritam SIFT) slike bit će u nastavku detaljnije pojašnjen.

2.2.1. Algoritam SIFT (engl. Scale-invariant feature transform)

Algoritam SIFT se koristi za otkrivanje i opis lokalnih značajki na slikama, te uparivanje korespodentnih točaka što je korisno za 3D rekonstrukciju scene. Algoritam je objavio David Lowe 1999.

Algoritam je dosta robustan jer pokazuje dobre rezultate usprkos promjeni osvijetljenosti, pojavi šuma, zaklanjanju te malim promjenama točke gledišta. Također ga karakteriziraju mala vjerojatnost pogrešnog uparivanja, uparivanje s velikim bazama podataka lokalnih značajki te procesiranje vrlo blizu radu u stvarnom vremenu.

Algoritam možemo podijeliti na četiri osnovne faze:

- detekcija ekstrema u prostoru mjerila
  (engl. scale-space extrema detection)
- lokalizacija interesnih točaka
  (engl. keypoint localization)
- pridruživanje orijentacije
  (engl. orientation assignment)
- određivanje opisnika interesnih točaka
  (engl. keypoint descriptor)

Detekcija ekstrema u prostoru mjerila

U ovoj fazi se detektiraju interesne točke. Algoritam prolazi kroz sve lokacije u svim veličinama slike da bi se detektirale potencijalne interesne točke. Za svaku veličinu slike potrebno je izmjeriti promjenu Gaussovog odziva tako što ćemo dobivene slike konvoluirati s Gaussovim filtrom (niskopropusni filtar) mjenjajući parametar σ.

Gaussova funkcija ima oblik:

$$G(u, v, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-(u^2+v^2)/(2\sigma^2)} \qquad (2.10)$$

Ulazna slika dana je sljedećom funkcijom:

$$I(x, y), \qquad (2.11.)$$

gdje su x i y koordinate točaka.

Slika se konvoluira s Gausovom funkcijom:

$$L(x,y,\sigma) = I(x, y) * G(x,y,\sigma) \qquad (2.12.)$$

gdje je σ parametar uvećanja u odnosu na originalnu sliku, a "*" operator konvolucije.

Razlika Gausijana (engl. difference of Gaussians) je prikazana na sljedeći način:

$$D(x,y,\sigma)= I(x, y)*G(x,y,k\sigma)- I(x, y)*G(x,y,\sigma)=$$

$$L(x,y,k\sigma)- L(x,y,\sigma) \qquad (2.13.)$$

Za dobivene razlike Gausijana, potrebno je svaki piksel iz slike jedne veličine usporediti sa susjednih 8 piksela na toj slici te odgovarajućih 9 piksela na slikama jednog stupnja veće ili manje veličine. Ako taj piksel iznosi minimum ili maksimum među svim uspoređenim pikselima, on postaje kandidat za interesnu točku.

Lokalizacija interesnih točaka

Budući da detekcija ekstrema u prostoru mjerila daje previše točaka kandidata, od kojih neke mogu biti nestabilne, podložne šumu naslici, imati nizak kontrast u odnosu na okolinu. Zbog toga je potrebno je dodatno filtrirati moguće interesne točke. U ovoj fazi je potrebno detaljnije podešenje susjednih informacija da bismo odredili trenutnu poziciju, veličinu i odnos mogućih kandidatnih točaka. U ovom koraku se obično izbacuju točke koje imaju slab kontrast i nisu raspoređene po rubovima.

Prvo je potrebno interpolirati lokacije maksimuma korištenjem razvoja Tavlorovog reda funkcije D (2.13.) do drugog stupnja:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \qquad (2.14)$$

gdje je x odmak od kandidatne točke. Ako je odmak veći od 0,5 tada se taj ekstrem nalazi bliže nekoj drugoj kandidatnoj točki. Zbog niskog kontrasta, odbaciju se točke gdje funkcija ima vrijednost manju od 0,03.

Pridruživanje orijentacije

Dodjeljivanje orijentacije je ključan korak u postizanju nepromjenljivosti na rotaciju koja se računa na sljedeći način:

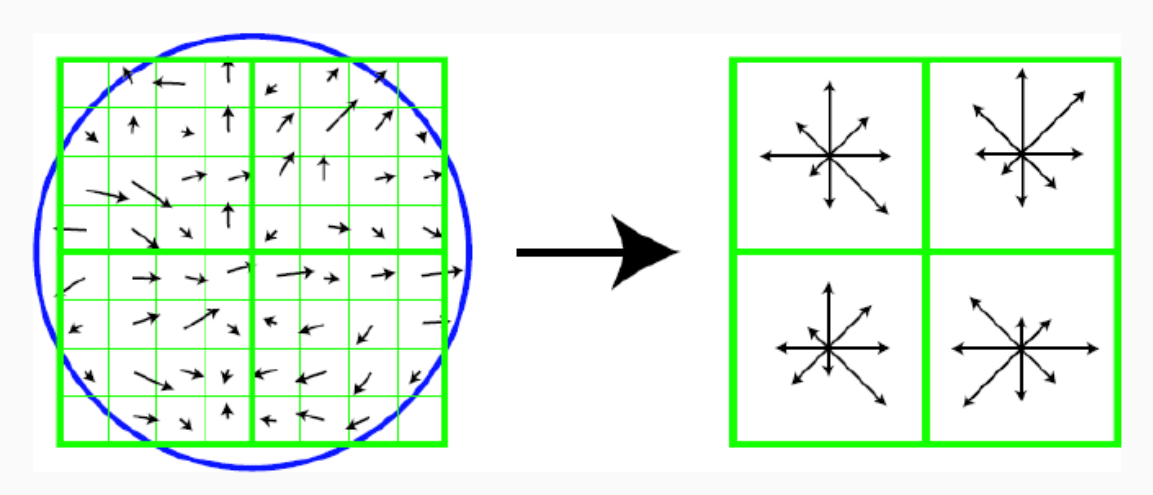
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.15.)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \quad (2.16.)$$

gdje m(x,y) predstavlja magnitudu, a θ(x,y) orijentaciju gradijenta a L(x,y) uzorak slike.

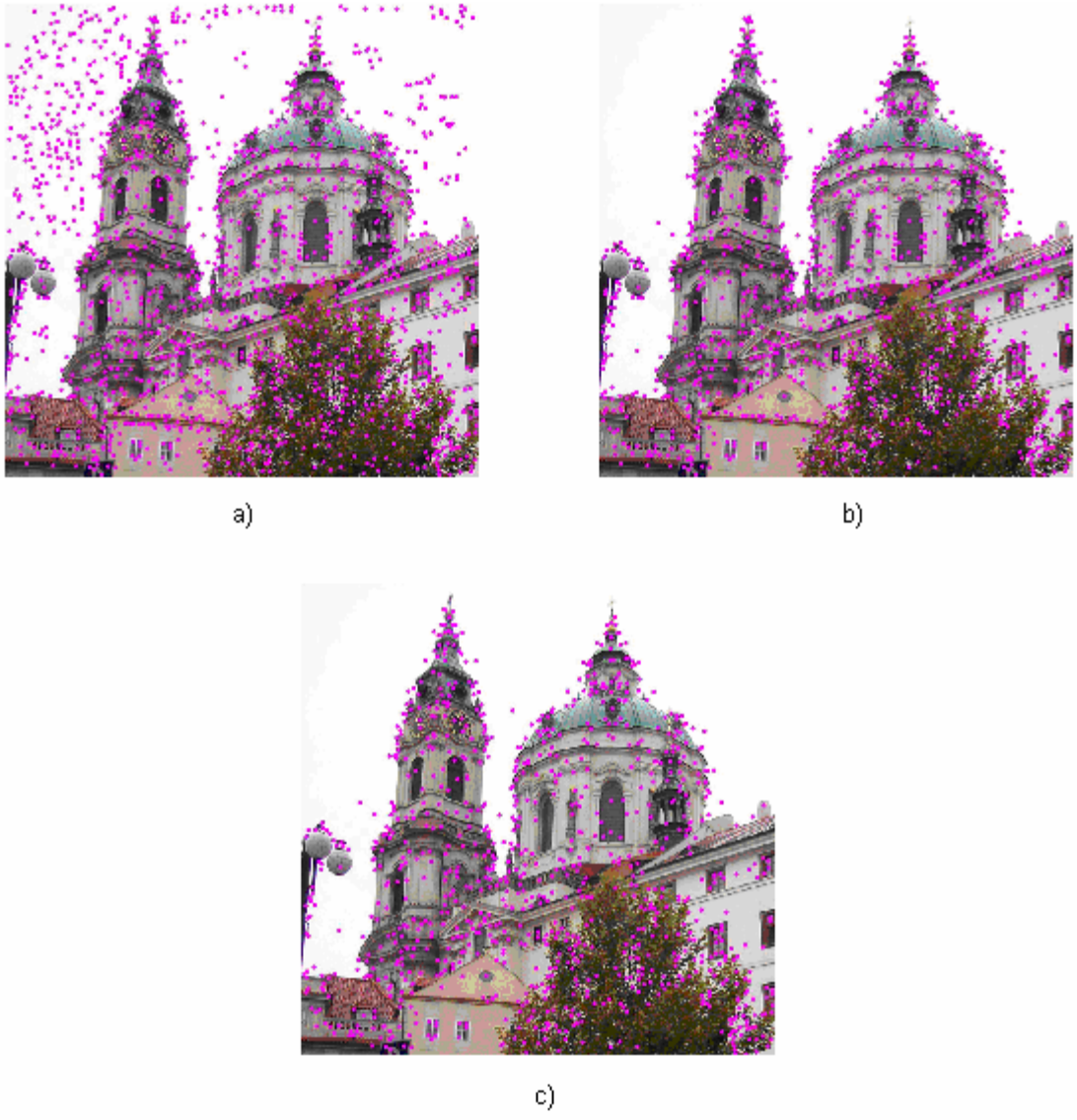
Određivanje opisnika interesnih točaka

U ovom koraku potrebno je dodijeliti opisni vektor svakoj interesnoj točki tako da se postigne invarijantnost na razlike u boji, osvjetljenju i točki gledišta. Uzimaju se područja od 4x4 piksela u okružju interesne točke te se predstavljaju kao skup orijentacijskih histograma (slika 2.8.). Od svakog od tih područja , stvara se vektor sa 8 potencijalnih smjerova, tj. 128 podataka za svaku točku.



Slika 2.8. Opisnik interesnih točaka

Na primjeru (slika 2.9.) slijedi demonstracija uklanjanja kandidatnih točaka nakon svih ovih navedenih ograničenja.



Slika 2.9. Na slici a) nalaze se sve potencijalne interesne točke, na slici b) odbacujemo točke koje imaju slab kontrast, te na slici c) može se uočiti odbacivanje točaka koje se nalaze na rubovima a posjeduju određenu nestabilnost[1]

2.3. Procjena relativne orijentacije

Kao što je već navedeno, jedno od ograničenja prilikom pretrage korespodentnih točaka može biti i epipolarna geometrija.

---

1 http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

U nastavku će biti opisane matematičke osnove epipolarne geometrije te određivanje relativne orijentacije u slučaju kada su kamere kalibrirane.

2.3.1. Epipolarna geometrija i esencijalna matrica

Uzmimo u obzir da promatramo dvije slike iste scene uzete sa različitih pozicija. Homogene koordinate točke na slici x i prostorne koordinate X točke p u sceni su povezane s izrazom:

$$\lambda \boldsymbol{x} = \Pi_0 \boldsymbol{X}, \qquad (2.17.)$$

gdje je $\Pi_0 = [\mathrm{I}, 0]$ tj. Uz homogenu točku x se nalazi parametar λ jer se u nju mogu preslikati pravci iz scene. Ako sada označimo sa x_1 i x_2 korespodentne točke na dvjema uzetim slikama tada postoji točno određena geometrijska veza među njima.

Ako je točka X_1 na sceni u koordinatnom sustavu prve kamere a točka X_2 ta ista točka u koordinatnom sustavu druge kamere tada vezu među njima možemo prikazati kao:

$$\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T. \qquad (2.18.)$$

Neka su x_1 i x_2 homogene koordinate projekcije točke p na dvije slikovne ravnine tada njihovu vezu izražavamo:

$$\lambda_2 \boldsymbol{x}_2 = R\lambda_1 \boldsymbol{x}_1 + T. \qquad (2.19.)$$

Ako pomnožimo obje strane jednadžbe s $\widehat{T}$ i s $\boldsymbol{x}_2^T$ na lijevoj strani ostat će 0 zbog množenja okomitih vektora, a na desnoj:

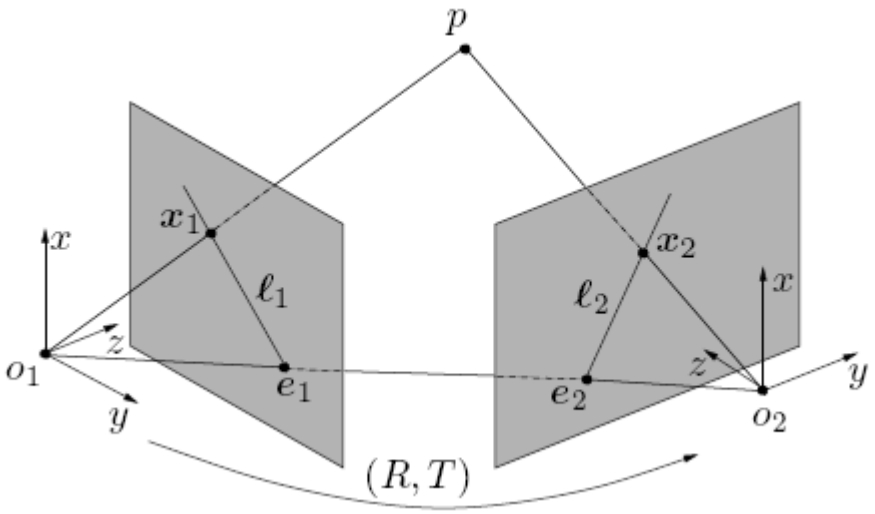
$$0 = \langle \boldsymbol{x}_2, T \times R\boldsymbol{x}_1 \rangle \quad \text{ili} \quad \boxed{\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0.} \qquad (2.20.)$$

Ova jednadžba se naziva epipolarno ograničenje, a matrica dana izrazom:

$$E = \widehat{T}R \quad \in \mathbb{R}^{3\times 3} \qquad (2.21.)$$

esencijalna matrica. Esencijalna matrica dakle sadrži informacije o rotaciji i translaciji, tj. relativnoj orijentaciji. Epipolarno ograničenje se također naziva i esencijalno ograničenje te bilinearno ograničenje.

Ovaj algebarski izvod geometrijski je prikazan na slici 2.16.:



Slika 2.10. Prikaz epipolarnog ograničenja

Vektor koji spaja centar prve kamere o_1 i točku p zatim vektor koji spaja centar druge kamere o_2 i točku p te vektor koji spaja optičke centre o_1 i o_2 tvore trokut i leže u istoj ravnini. Vektor koji spaja optičke centre o_1 i o_2 naziva se osnovna linija (engl. baseline).
Definiramo entitete epipolarne geometrije:

1. Epipolarna ravnina određena točkama o_1 , o_2 i p, te za svaku točku p iz scene postoji jedna takva ravnina
2. Projekcija optičkog centra jedne kamere o_1 (o_2) na slikovnu ravninu druge kamere e_2 (e_1) nazivamo epipol
3. Presjek epipolarne ravnine sa pojedinom slikovnom ravninom je pravac l_1 (l_2) koji se naziva epipolarna linija točke p

Veza između ovih entiteta i esencijalne matrice možemo prikazati kao:

- za epipolove e_1 i e_2 :

$$e_2^T E = 0, \quad Ee_1 = 0.$$

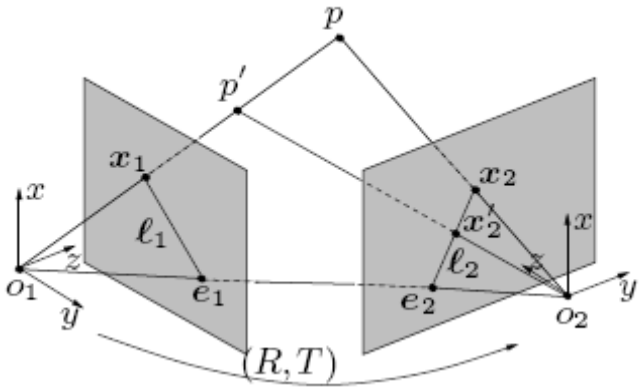
- za epipolarne linije l_1 i l_2 i točke x_1 i x_2:

$$\ell_2 \sim Ex_1, \quad \ell_1 \sim E^T x_2 \quad \in \mathbb{R}^3,$$

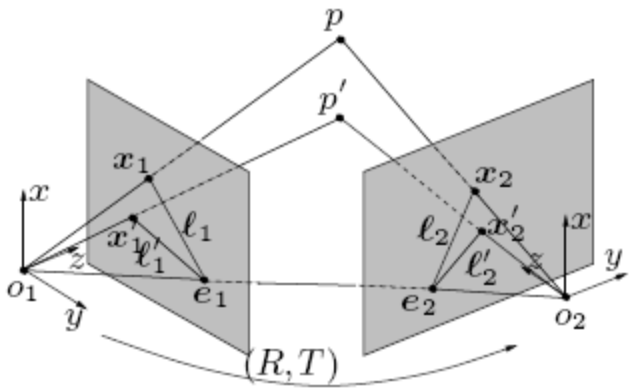
- na obje slike epipolovi e_1 i e_2 i točke x_1 i x_2 leže na epipolarnim linijama:

$$\ell_i^T e_i = 0, \quad \ell_i^T x_i = 0, \quad i = 1, 2.$$

Ilustracija ovih veza prikazana je na slikama 2.17. i 2.18.:

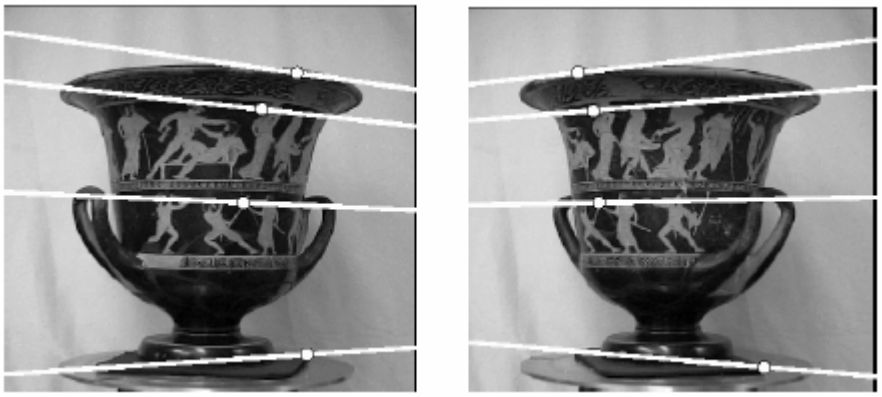


Slika 2.11. Entiteti epipolarne geometrije



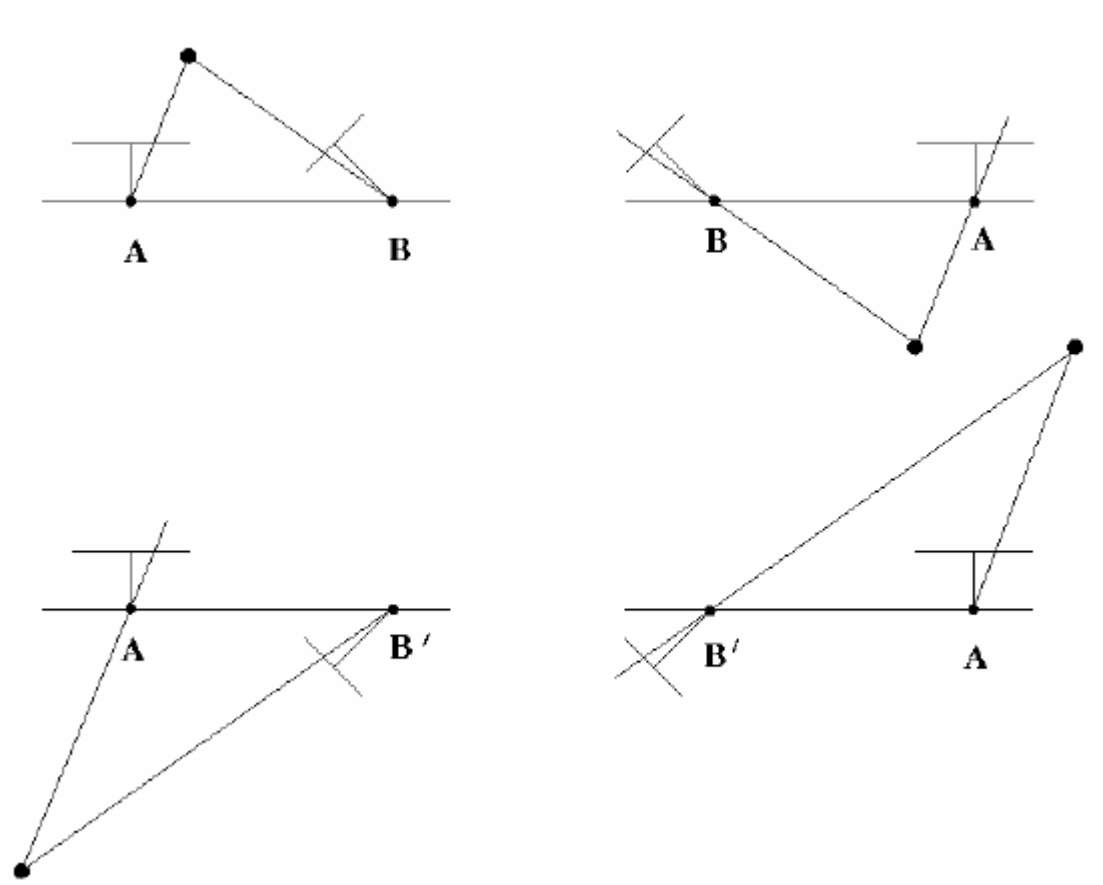
Slika 2.12. Entiteti epipolarne geometrije

Prikaz epipolarnih linija s nekim korespodentnim točkama na paru realnih slika dan je na slici 2.19.:



Slika 2.13. Epipolarne linije na realnim slikama [hartley04]

Kao što je već navedeno esencijalna matrica E sadrži podatke o relativnoj orijentaciji stoga ako nam je ona poznata iz nje možemo dobiti podatke o rotaciji i translaciji. Za svaku esencijalnu matricu možemo izračunati točno četiri relativne orijentacije. Jednoznačnost dekompozicije esencijalne matrice dobivamo zahtjevom da se rekonstrukcije korespodentnih točaka nalaze ispred obje kamere kao što je prikazano na slici 2.14.



Slika 2.14. Slučaj gore lijevo je jedini fizički moguć

2.3.2. Algoritam s osam točaka (engl. eight-point algorithm)

Neka je $E = \widehat{T}R$ esencijalna matrica sa prethodno definiranim epipolarnim ograničenjem (2.20.). Članove te matrice:

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \in \mathbb{R}^{3\times 3} \qquad (2.22.)$$

možemo prikazati vektorom $E^s \in \mathbb{R}^9$:

$$E^s = [e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}]^T \in \mathbb{R}^9 \qquad (2.23.)$$

Neka je ⍺ definiran Kroneckerovim produktom:

$$\boldsymbol{a} = \boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \qquad (2.24.)$$

gdje su $\boldsymbol{x}_1 = [x_1, y_1, z_1]^T \in \mathbb{R}^3$ i $\boldsymbol{x}_2 = [x_2, y_2, z_2]^T \in \mathbb{R}^3$ tada možemo zapisati:

$$\boldsymbol{a} = [x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2]^T \in \mathbb{R}^9 \quad (2.25.)$$

Te konačno se pokazuje da se epipolarno ograničenje (2.20.) može zapisati kao:

$$\boxed{\boldsymbol{a}^T E^s = 0.} \qquad (2.26.)$$

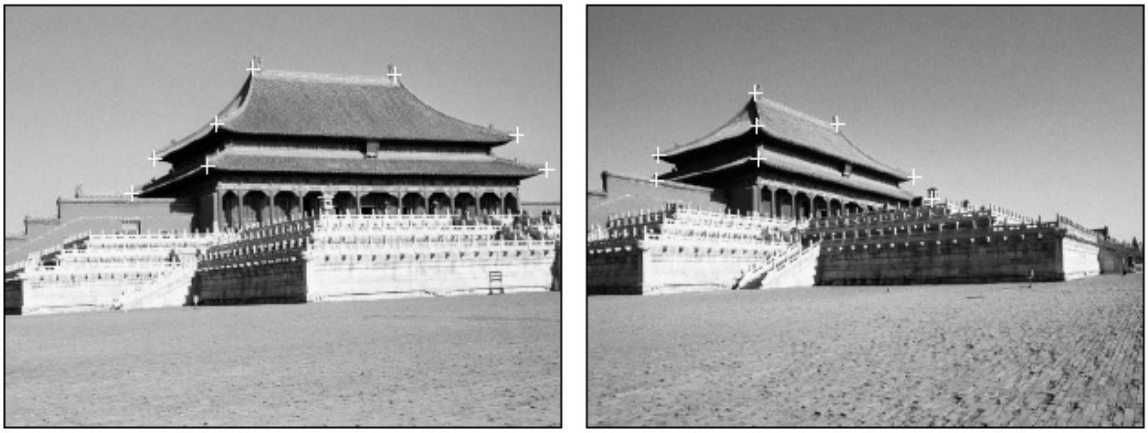
Za ulazni skup korespodentnih točaka $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j),\ j = 1, 2, \ldots, n$, definiramo matricu $\chi \in \mathbb{R}^{n\times 9}$:

$$\chi = [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T \qquad (2.27.)$$

gdje su retci matrice Kroneckerovi produkti svakog para točaka $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$. U odsustvu šuma vektor E^s zadovoljava:

$$\chi E^s = 0 \quad (2.28.)$$

Vektor E^s iz ove linerane jednadžbe može biti određen do na konstantu (engl. up to scale) samo ako rang matrice $\chi \in \mathbb{R}^{n\times 9}$ iznosi 8. To se može postići ako je n>=8 tj. ako imamo barem 8 „idealnih“ korespodentnih točaka (slika 2.21.). Za općeniti slučaj potrebno je računati minimalne kvadrate funkcije pogreške $\|\chi E^s\|^2$.



Slika 2.15. Idealne korespodentne točke [ma05]

Da bismo mogli odrediti relativnu orijentaciju iz E osim epipolarnog ograničenja potrebno je uvesti još jedno dodatno ograničenje. To ograničenje uslovljava pripadnost esencijalne matrice E esencijalnom prostoru ε:

$$\mathcal{E} \doteq \left\{ \widehat{T}R \,\middle|\, R \in SO(3), T \in \mathbb{R}^3 \right\} \quad \subset \mathbb{R}^{3\times 3}.$$

Sada algoritam s osam točaka možemo opisati [ma05]:

Za ulazni skup korespodentnih točaka:

$$(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j),\, j \;=\; 1, 2, \ldots, n \;\; (n \;\geq\; 8)$$

ovaj algoritam pronalazi R i T koji zadovoljavaju:

$$\boldsymbol{x}_2^{jT} \widehat{T} R \boldsymbol{x}_1^j = 0, \quad j = 1, 2, \ldots, n.$$

1. Računanje prve aproksimacije esencijalne matrice:
   Napravimo matricu $\chi = [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T \in \mathbb{R}^{n\times 9}$ iz korespodencija

   $$\boldsymbol{a}^j = \boldsymbol{x}_1^j \otimes \boldsymbol{x}_2^j \quad \in \mathbb{R}^9.$$

   Iz jedinstvene dekompozicije (SVD) $\chi = U_\chi \Sigma_\chi V_\chi^T$ vektor E^s je određen devetim stupcem matrice V_χ . U općenitom slučaju esencijalna matrica ne mora pripadati esencijalnom prostoru ε.

Vektor E^S zapišemo u standardnom obliku esencijane matrice E dimenzija 3x3.

2. Projekcija u esencijalni prostor:
   Računamo jedinstvenu dekompoziciju (SVD) esencijalne matrice:

$$E = U\,\mathrm{diag}\{\sigma_1, \sigma_2, \sigma_3\}V^T,$$

   gdje je $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$ i $U, V \in SO(3)$.

3. Određivanje R i T iz esencijalne matrice:
   U ovm koraku su nam potrebne samo matrice U i V za određivanje R i T:

$$R = UR_Z^T\left(\pm\frac{\pi}{2}\right)V^T, \quad \widehat{T} = UR_Z\left(\pm\frac{\pi}{2}\right)\Sigma U^T.$$

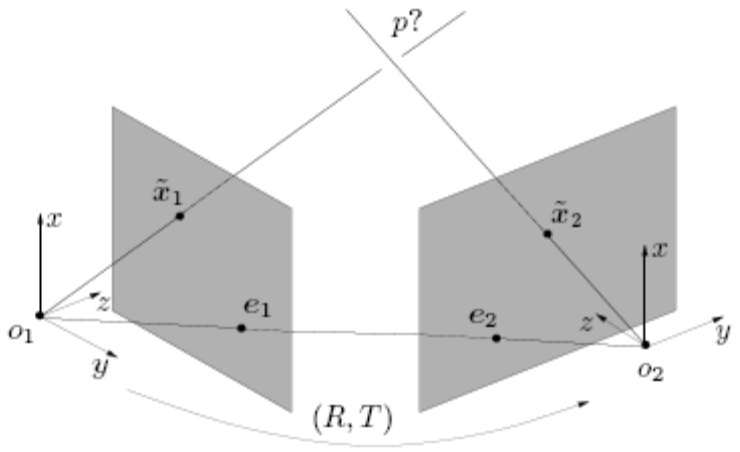
   Gdje je :

$$R_Z^T\left(\pm\tfrac{\pi}{2}\right) \doteq \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.4. Trijagulacija strukture scene

Trijagulacija strukture scene je obrnut postupak od postupka uzimanja 3D točaka iz scene iz dva različita gledišta i računanja relativne orijentacije među njima. Kod trijangulacije su nam poznati rotacija (R) i translacija(T) među točkama te koordinate tih točaka na slikovnoj ravnini a cilj nam je odrediti koordinate 3D točke u sceni.

Za dobivene R i T iz esencijalne matrice E zbog prisutnosti šuma ne postoji garancija da imaju točan iznos kao stvarna rotacija i translacija. Zbog toga postoji mogućnost da se reprojicirane točke ne sjeku u sceni. Ovaj problem se naziva trijangulacijski problem (slika 2.23.).



Slika 2.16. Trijangulacijski problem

Greška uzrokovana šumom računa se kao:

$$\phi(x) = \|\tilde{x}_1 - x_1\|^2 + \|\tilde{x}_2 - x_2\|^2$$

gdje su $\tilde{x}_1, \tilde{x}_2 \in \mathbb{R}^3$ točke sa šumom a $x_1, x_2 \in \mathbb{R}^3$ idealne točke na slikovnoj ravnini.

3. Opis korištenih vanjskih biblioteka i njihovo prevođenje

Sustav je izgrađen na operacijskom sustavu FreeBSD (engl. Free Berkeley Software Distribution) koji je baziran na Unixu a razvijan na Sveučilištu California u Berkeleyu. Razvijeni programski sustav je kao i vanjske biblioteke pisan u programskom jeziku C++. Za prevođenje je korišten prevodioc g++, koji je sastavni dio obitelji prevodioca gcc (engl. GNU Compiler Collection).

Razvoj programske podrške mnogo potpomažu vanjske biblioteke. U ovom radu su korištene sljedeće vanjske biblioteke:

- implementacija algoritma SIFT
- skup biblioteka VW (engl. Visual Workbench)
- skup biblioteka Boost
- algoritmi određivanja relativne orijentacije

U nastavku su opisane vanjske biblioteke koje su korištene u realizaciji rada i postupak njihovog prevođenja.

3.1. Implementacija algoritma SIFT

Korištena je implementacija algoritma SIFT koju je razvio A. Vedaldi sa Sveučilišta California u Los Angelesu[1]. Algoritam uzima jednu sliku na obradu, izračunava interesne točke i njihove deskriptore. Podatci za jednu točku opisuju: x i y koordinatu na slici (pikseli), veličinu gradijenta, orijentaciju i 128 brojeva (u rasponu od 0 do 255) koji opisuju deskriptor.

Algoritam SIFT je dodan u ljusku i nalazi se u direktoriju cvsh_src/ext/sift sa ostalim vanjskim dodanim algoritmima.

3.2. Biblioteka VW (engl. Vision Workbench)

VW sadrži skup biblioteka koje pomažu u razvoju programske podrške za

---

1 http://vision.ucla.edu/~vedaldi/code/siftpp/assets/siftpp/versions/

računalni vid. Njeni sastavni dijelovi su:

-VNL (engl. Vision Numeric Library): sadržava matrice, vektore i algoritme nad njima

-VW: glavna komponenta biblioteke, sadržava razrede slike, procesiranje slike i geometriju pogleda, te sa VNL-om daje osnovnu funkcionalnost

-ostale komponente daju dodatnu funkcionalnost poput dohvaćanje slika, kontroliranje hardvera, korištenje grafičkih sučelja: X, XForms, GTK, GLOW, GL, Wx, Firewire, Meteor, Yorick, GTI

Biblioteka VW je razvija se na Sveučilištu u Oxfordu pod licencom LGPL[1] .

3.2.1. Prevođenje biblioteke VW

Da bi preveli biblioteku VW na FreeBSD-u potrebno je učiniti nekoliko promjena u izvornom kodu. Izvršene promjene su navedene po datotekama:

-datoteka imageio_linux.cpp:

-potrebno je dodati:

```
#include<sys/stat.h>
```

-datoteka point2d.h:

-potrebno je u imenik (engl. namespace) dodati:

```
Point2d operator/(const Point2d &p1, const double a);
Point2d operator+(const Point2d &p1, const Point2d &p2);
```

---

1 http://www.doc.ic.ac.uk/~ajd/Scene/Release/vw34.tar.gz

-datoteka lineseg2d.h:

-potrebno je u imenik dodati:

```
std::ostream& operator<<(std::ostream&, const
      VW::LineSeg2d&);
std::ostream& operator>>(std::istream&, VW::LineSeg2d&);
```

-datoteka lineseg3d.h:

-potrebno je u imenik dodati:

```
double Norm2(const VW::LineSeg3d&);
double Norm(const VW::LineSeg3d&);
std::ostream& operator<<(std::ostream&, const
      VW::LineSeg3d&);
std::ostream& operator>>(std::istream&, VW::LineSeg3d&);
```

-datoteka sequencerbase.h:

-potrebno je zakomentirati linije 126 i 127, tj. :

```
template<class ImType>
void CopyImage(...);
```

-datoteka delunay.cpp:

-potrebno je izbrisati `#include<malloc.h>` i dodati

```
#include<stdlib.h>
```

-datoteka matchdata.h:

-potrebno je u imenik dodati:

```
std::ostream& operator<<(std::ostream&, const
      VW::MatchData&);
std::ostream& operator>>(std::istream&, VW::MatchData&);
```

-datoteka bootstrap:

-potrebno je zamijeniti redak automake_minor_version='echo $version | sed 's/\([0-9]*\).\([0-9]*\).\([0-9]*\)/\2/'' s automake_minor_version='echo $version | sed 's/\([0-9]*\).\([0-9]*\).*/\2/''
-zatim izbrisati redak automake_micro_version='echo $version | sed 's/\([0-9]*\).\([0-9]*\).\([0-9]*\)/\3/'' a na njegovo mjesto upisati echo $automake_minor_version

Osim navedenih promjena potrebno je i proširiti funkcionalnost u sljedećim datotekama:

-datoteka essentialmatrixcompute5pt.h:

-potrebno je dodati člansku funkciju:

```
void compute(
      const VNL::MatrixFixed<3,3, double>& e1,
      const VNL::MatrixFixed<3,3, double>& e2,
      const VNL::MatrixFixed<3,3, double>& e3,
      const VNL::MatrixFixed<3,3, double>& e4,
      std::vector<EssentialMatrix>& ematrices);
```

-datoteka essentialmatrixcompute5pt.cpp:

-potrebno je dodati člansku funkciju:

```
void compute(
      const VNL::MatrixFixed<3,3, double>& e1,
      const VNL::MatrixFixed<3,3, double>& e2,
      const VNL::MatrixFixed<3,3, double>& e3,
      const VNL::MatrixFixed<3,3, double>& e4,
      std::vector<EssentialMatrix>& ematrices)
      {
            ematrices.clear();
```

```
        SolveForEs (e1, e2, e3, e4, ematrices);
    }
```

Nakon ovih izmjena potrebno je izvesti sljedeće naredbe:

$./bootstrap

$./configure

$make

$make install

3.3. Biblioteka Boost

Boost je skup biblioteka namijenjen proširenju funkcionalnosti u jeziku C++. Iskoristivost ove biblioteke proteže se od upotrebe pametnih pokazivača (npr. smart_ptr.hpp), apstrakcija OS (npr. FileSystems), operacija linearne algebre (npr. uBLAS (engl. Basic Linear Algebra Subprograms)) te ostale biblioteke namijenjene naprednim korisnicima jezika C++.

Budući da je za programsku realizaciju ovog rada bilo potrebno raditi izračunavanja i operacije iz linearne algebre posebno nam je od koristi biblioteka uBLAS. uBLAS obuhvaća osnovne operacije s matricama i vektorima poput: indeksiranja (operator[]), zbrajanja (operator+), oduzimanja (operator-), množenja (`prod`) i transponiranja (`trans`). Budući da su funkcije generičke omogućen je rad s različitim dimenzijama vektora i matrica.

Da bismo mogli koristiti biblioteku potrebno ju je nakon raspakiranja smjestiti u direktorij /usr/local/include.

Izvorni kod biblioteke Boost može se slobodno preuzeti sa stranice projekta[1].

3.4. Algoritmi određivanja relativne orijentacije

Budući da se u ovom radu bilo potrebno odrediti relativnu orijentaciju iz

---

1 http://www.boost.org/users/download/

korespodentnih točaka iskorišten je izvorni kod pripremljen za [segvic07][1]. Realizirani su su različiti algoritmi: algoritam s pet točaka, algoritam s osam točaka te dekompozicija planarne homografije. Ti algoritmi koriste biblioteku VW stoga su ovisni o njoj.

3.4.1 Prevođenje algoritama

Potrebno se pozicionirati u direktorij build, te izvesti naredbu:

$make

Budući da algoritmi koriste vanjske biblioteke poput VW i Boost njih je također potrebno prevesti i smjestiti u odgovarajući direktorij. Upotreba vanjskih biblioteka zahtijeva promjenu u datoteci *makefile* a izvršena promijena je:

-u retku gdje piše CXXFLAGS= -03 –Wall –w –I$(VWDIR)/include/ treba staviti CXXFLAGS= -03 –Wall –w –I$(VWDIR)/include/ –I/usr/local/include

1 http://www.zemris.fer.hr/~ssegvic/pubs/evalpose.tar.gz

4. Programska realizacija

Zbog složenog matematičkog modela računalnog vida postupak programske realizacije je također složen stoga će u ovom radu biti opisane i prikazane postojeće realizacije, nadogradnja postojećeg sustava novim algoritmima te prikaz korištenja novoizgrađenog algoritma.

4.1.Ljuska cvsh (engl. computer vision shell)

Za izvedbu završnog rada korišteno je programsko okruženje (ljuska) cvsh. Ona je razvijana na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave (ZEMRIS) da bi olakšala eksperimentiranje i razvoj novih postupaka računarskog vida. Zbog određene sličnosti s korisničkom ljuskom koja je uobičajena na operacijskim sustavima UNIX, program je nazvan ljuskom računarskog vida. Program se izvodi u jednom ili više izvedbenih ciklusa, koji se zadaju ili pri pozivu programa, u naredbenom retku, ili interaktivnim dijalogom. Jedna od njenih karakteristika je prenosivost i kompatibilnost s različitim operacijskim sustavima. Za specifične pozive operacijskog sustava (grafika, pristup sklopovlju, rad s datotečnim sustavom i dr.) razvijene su paralelne komponente za Windows, koje imaju sufiks _w32 i Unix, sa sufiksom _unix. Također zbog različitosti OS zasnovanih na Unixu također su razvijane paralelne implementacije i to za Linux (sufix _linux) te za MacOS (sufiks _macx).

Izvorni kod ljuske je dostupan na www stranicama autora[1].

4.1.1. Prevođenje ljuske cvsh

Da bismo preveli ljusku cvsh na operacijskom sustavu Unix potrebno je uraditi sljedeće:

-u direktoriju gdje se nalazi izvorni kod ljuske paralelno napraviti direktorij

---

1 http://www.zemris.fer.hr/~ssegvic/cvsh2.zip

build te u njemu direktorij *release*

-pozicionirati se u direktorij *release* i staviti u njega datoteku *project.pro*

-izvesti naredbe:

$tmake project.pro >makefile

$make

Tmake je alat za stvaranje i održavanje datoteka koje određuju tok prevođenja programskog sustava (tzv. *makefileova*). Također je potrebno namjestiti varijable PATH i TMAKEPATH prije korištenja alata tmake[1]:

```
export
PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/loca
l/sbin:

export TMAKEPATH=/usr/local/tmake-1.13/lib/linux-g++/
```

Prilikom prevođenja kompajler će nam javiti upozorenje (engl. warning) stoga je potrebno u datoteci grid_draw_x.cpp u liniji 127 i 137 umjesto char* staviti const char*.

4.2. Dodavanje novog algoritma u ljusku

Zbog modularne programske arhitekture, koja je postignuta objektno orijentiranom paradigmom u jeziku C++ [segvic04] te korištenjem oblikovnog obrasca metode tvornice olakšan je način dodavanja novih funkcionalnosti u ljusku. Navedeni obrazac podatnost postiže nasljeđivanjem (engl. inheritance) osnovnog razreda *alg_base* koji pruža sljedeće sučelje:

```
class alg_base{
      protected:
```

1 http://tmake.sourceforge.net/

```
  alg_base();
public:
  virtual ~alg_base();

//identification
public:
  virtual char const* name() =0;

//mandatory operations
public:
  virtual void process(
   const img_vectorAbstract& src,
   const win_event_vectorAbstract& events,
   int msDayUtc, int frame);
  virtual void process(
   const img_vectorAbstract& src,
   const win_event_vectorAbstract& events,
   int msDayUtc);
//discretionary operations
public:
  virtual void config(cli_wrapAbstract& cli);
  virtual void profile(ui_reportAbstract& );

//output
protected:
  void checkId(int id, char const* context);
public:
  virtual int countDst() =0; // >=1!
  virtual char const* captionDst(int id);
  virtual const img_fmt& fmtDst(int id);
  virtual const img_data& imgDst(int id) =0;
  virtual win_ann_abstract const* pAnnDst(int id);

// alg_base construction
public:
// concrete classes ctor argument
class CtorArg{
 public:
```

```
  img_fmt fmt;//image format
  char const* cfg;//configuration string
 CtorArg(const img_fmt& f, char const* c):fmt(f),cfg(c){}
 CtorArg(const img_fmt& f):fmt(f),cfg(""){}
};

//factory for the concrete classes
static alg_base* create(
      const char* name,
      const CtorArg& arg);
};
```

Budući da je navedeno sučelje za većinu korisnika preopširno, definiran je razred *alg_mono* koji pojednostavljuje sučelje osnovne klase (korisno kada algoritam koji dodajemo ne treba kompletno sučelje osnovnog razreda (kreatora)). Razred *alg_mono* koristimo kad radimo nad samo jednom ulaznom slikom:

```
class alg_mono:
       public alg_base
      {
      protected:
        alg_mono(){}

      private:
      //operations
        virtual void process(
         const img_data& src,
         const win_event_vectorAbstract& events,
         int msDayUtc) =0;

      public:
      //template method
      virtual void process(
      const img_vectorAbstract& src,
      const win_event_vectorAbstract& events,
```

```
int msDayUtc);
};
```

Nasljednici razreda *alg_mono* ipak moraju implementirati i neke metode iz razreda *alg_base.*

Dodavanje novog algoritma (konkretnog proizvoda u obrascu metode tvornice) se ostvaruje nasljeđivanjem razreda *alg_mono* i implementacijom sučelja koje taj razred definira:

```
class alg_relor:
  public alg_mono
{
public:
  alg_relor(const CtorArg&);
  virtual ~alg_relor(){}

  virtual char const* name(){return s_name();}
  static char const* s_name(){return "relor";}

//alg_mono interface
public:
  virtual void process(
    const img_data& src,
    const win_event_vectorAbstract& events,
    int msDayUtc);
  virtual void config(cli_wrapAbstract& cli);
  virtual void profile(ui_reportAbstract& pd);
public:
  virtual int countDst();
  virtual char const* captionDst(int id);
  virtual const img_data& imgDst(int id);
  // from alg_mono:
  //   virtual const img_fmt& fmtDst(int id);
  virtual win_ann_abstract const* pAnnDst(int id);

//alg_relor interface
private:
  double compute_err(Vector3 Q3Dt,
```

```
                        math::Point2D Qref,
                        math::Point2D Qsrc,
                        Matrix3x3& R, Vector3 t);
private:
  void config(char const *);
  void dumpParams(
    std::ostream& os,
    char const* intro);

//data
private:
  boost::shared_ptr<ext_match> imp_;

  img_data src_;
  img_data ref_;
  std::string fnRef_;
  std::vector<math::Point2D> Psrc_;
  std::vector<math::Point2D> Pref_;

  win_ann annSrc_;
  win_ann annRef_;
  win_ann ann3DR_;

  std::vector<int> t_;
};
```

4.3. Implementirani algoritam *alg_relor*

Osnovna funkcionalnost implementiranog algoritma je određivanje korespodencija, 3D rekonstrukcija trijangulacijom te prikaz rezultata. Algoritmu se proslijeđuju polja korespodentnih točaka: `std::vector<math::Point2D> Psrc_` i `std::vector<math::Point2D> Pref_` dobivenih iz referentne i ispitne slike koje obrađuje algoritam SIFT. Dobivene točke treba prvo normirati u koordinatni sustav kamere (`image2norm(qImage)`) a zatim ih proslijediti postupku za uspostavljanje geometrije dvaju pogleda. U biblioteci VW postoji više takvih postupaka a korišten je `vwEssentialMLESAC(…)`. Navedeni postupak računa

esencijalnu matricu iz korespodentnih točaka. Esencijalna matrica sadrži informacije o relativnom položaju kamera tj. matricu rotacije R i vektor translacije t koji se dobiju funkcijom `decomposeEssential()`. Nakon toga slijedi iscrtavanje korespodentnih točaka na referentnoj i ispitnoj slici te trijanguliranih točaka, kamera i njihovog međusobnog odnosa u sceni na tlocrtu.

U članskoj funkciji `virtual void process (...)` implementira se rad algoritma kojeg dodajemo:

```
void alg_relor::process(
  const img_data& src,
  const win_event_vectorAbstract&, int)
{
  /*...*/
  img_access_copy(src, src_);
  int n=imp_->process(ref_,src_, Pref_,Psrc_);
  //Pref_ Psrc_ vektori korespodentnih tocaka

  //normiranje korespodentnih tocaka u k.s. kamere
  ccam_par* pcam=ccam_par::create("fixed",
      ccam_par::CtorArg("cycab-wide"));

  std::vector<math::Point2D> Nsrc_(n);
  std::vector<math::Point2D> Nref_(n);
  for(int j=0; j<n; ++j){

      math::Point2D qNorm_src=pcam->image2norm(Psrc_[j]);
      Nsrc_[j]=qNorm_src;

      math::Point2D qNorm_ref=pcam->image2norm(Pref_[j]);
      Nref_[j]=qNorm_ref;

  }
  //racunanje esencijalne matrice
  Matrix3x3 Ess;
  vwEssentialMLESAC(Nref_, Nsrc_, Ess);

  //racunanje projekcijske matrice P i trijangulacija
  //tocaka
  Matrix3x4 P;
  int tp;
  std::vector<VectorN<4> > Qs_h(n);

  tp=ep::decomposeEssential(Ess, Nref_, Nsrc_, P, Qs_h);

  //trijangulacija
  std::vector<Vector3> Qs_w(Qs_h.size());

  size_t g=Qs_h.size();
  for(unsigned int k=0; k<g; ++k){

    double h;
    h=Qs_h[k][3];
```

```
  Qs_w[k][0]=Qs_h[k][0]/h;
  Qs_w[k][1]=Qs_h[k][1]/h;
  Qs_w[k][2]=Qs_h[k][2]/h;
}

//racunanje R i t iz P
Matrix3x3 R;
Vector3 t;
R=subrange(P, 0,3, 0,3);
t=column(P, 3);

annSrc_.clear();
annRef_.clear();
//isrtavanje korespodentnih tocaka na slikama
for (int i=0; i<n; ++i){
  win_ann_abstract::PixelStyle style=
      win_ann_abstract::PS_BoldLarge;
  std::ostringstream oss; oss <<imp_->id(i);
  grid::Pixel psrc(math2grid(Psrc_[i]));
  annSrc_.addPixel(psrc, 0, oss.str().c_str(), style);
  annSrc_.addCircle(psrc, myround(imp_->scale2(i)), 0,
      "", win_ann_abstract::LS_Normal,
            win_ann_abstract::FS_None, 1);

  grid::Pixel pref(math2grid(Pref_[i]));
  annRef_.addPixel(pref, 0, oss.str().c_str(), style);
  annRef_.addCircle(pref, myround(imp_->scale1(i)), 0,
      "", win_ann_abstract::LS_Normal,
            win_ann_abstract::FS_None, 1);

}

ann3DR_.clear();
//iscrtavanje kamera (P[I|0], P'[R|t])
Trans3Dto2D transform(20, math::Point2D(100,10));

Vector3 Qc1(0, 0, 0); //referentna kamera
Vector3 Qc1il(-.25, 0, .5);
Vector3 Qc1ir(+.25, 0, .5);
math::Point2D qc1(transform(Qc1));
math::Point2D qc1il(transform(Qc1il));
math::Point2D qc1ir(transform(Qc1ir));
ann3DR_.addPixel(math2grid(qc1), 0, "",
    win_ann_abstract::PS_BoldLarge);
ann3DR_.addLine(grid::Line(math2grid(qc1il),
    math2grid(qc1ir)), 0, "",
          win_ann_abstract::LS_Normal, 1);

Matrix3x3 Rtrans(trans(R));
Vector3 Qc2=-prod(Rtrans, t);
Vector3 Qc2il(prod(Rtrans,Qc1il)+Qc2);
Vector3 Qc2ir(prod(Rtrans,Qc1ir)+Qc2);
math::Point2D qc2(transform(Qc2));
math::Point2D qc2il(transform(Qc2il));
math::Point2D qc2ir(transform(Qc2ir));
ann3DR_.addPixel(math2grid(qc2), 1, "",
    win_ann_abstract::PS_BoldLarge);
```

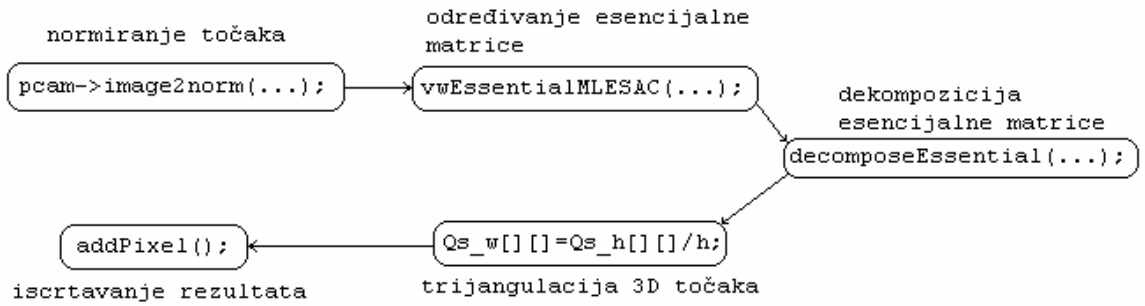
```
  ann3DR_.addLine(grid::Line(math2grid(qc2il),
       math2grid(qc2ir)), 1, "",
            win_ann_abstract::LS_Normal, 1);

  //iscrtavanje trijanguliranih tocaka (tlocrt)
  double global_err=0;
  for(unsigned int h=0; h<g; ++h){
    math::Point2D q_h(transform(Qs_w[h]));
    std::ostringstream oss;
    oss <<imp_->id(h);

    double err=compute_err(Qs_w[h], Pref_[h],
            Psrc_[h], R, t);
    global_err+=err;

    if(err<=5)
      ann3DR_.addPixel(math2grid(q_h), 2,
            oss.str().c_str(),
                  win_ann_abstract::PS_BoldSmall);
    else
      ann3DR_.addPixel(math2grid(q_h), 4,
            oss.str().c_str(),
                  win_ann_abstract::PS_BoldSmall);
  }
  global_err=sqrt(pow(global_err, 2));
  std::cerr<<"\n\n\n";
  std::cerr<<"Ukupna reprojekcijska greska:"<<global_err;
  std::cerr<<"Prosjecna reprojekcijska
            greska po tocki:"<<global_err/g;
  std::cerr<<"\n\n\n";

  /*...*/
}
```



Slika 4.1. Struktura algoritma

4.4. Korištenje ljuske i algoritma

Nakon pozicioniranja u direktorij sa izvršnom datotekom[1]

1 ../../cvsh_src/build/release

ljuska se pokreće preko tekstualnog korisničkog sučelja naredbom:

$./cvsh_match -sf=/../direktorij_sa_slikama/ -a=relor -i

Parametri koje proslijeđujemo ljuski su:

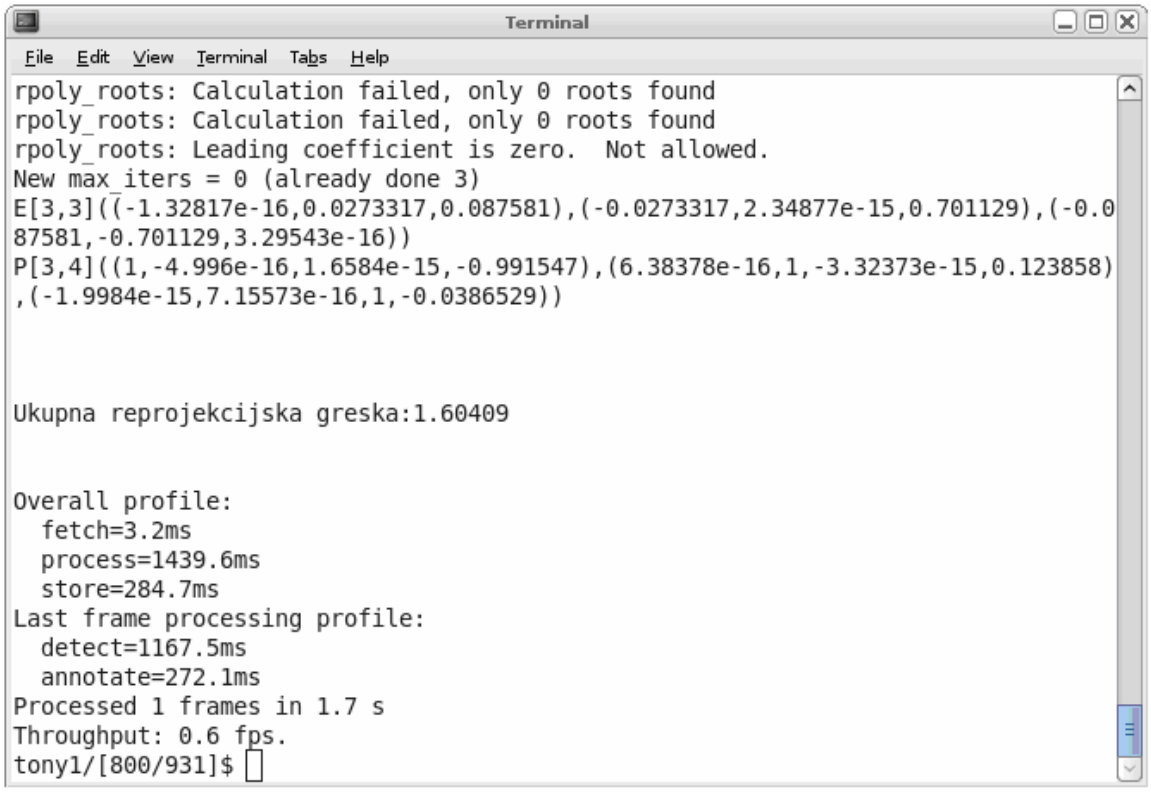
- -sf -(od engl. source file) direktorij uzorka slika
- -a -naziv algoritma obrade slike (u ovom slučaju *relor*)
- -i -interaktivni način rada

Prikaz obrade se odvija preko tri prozora:

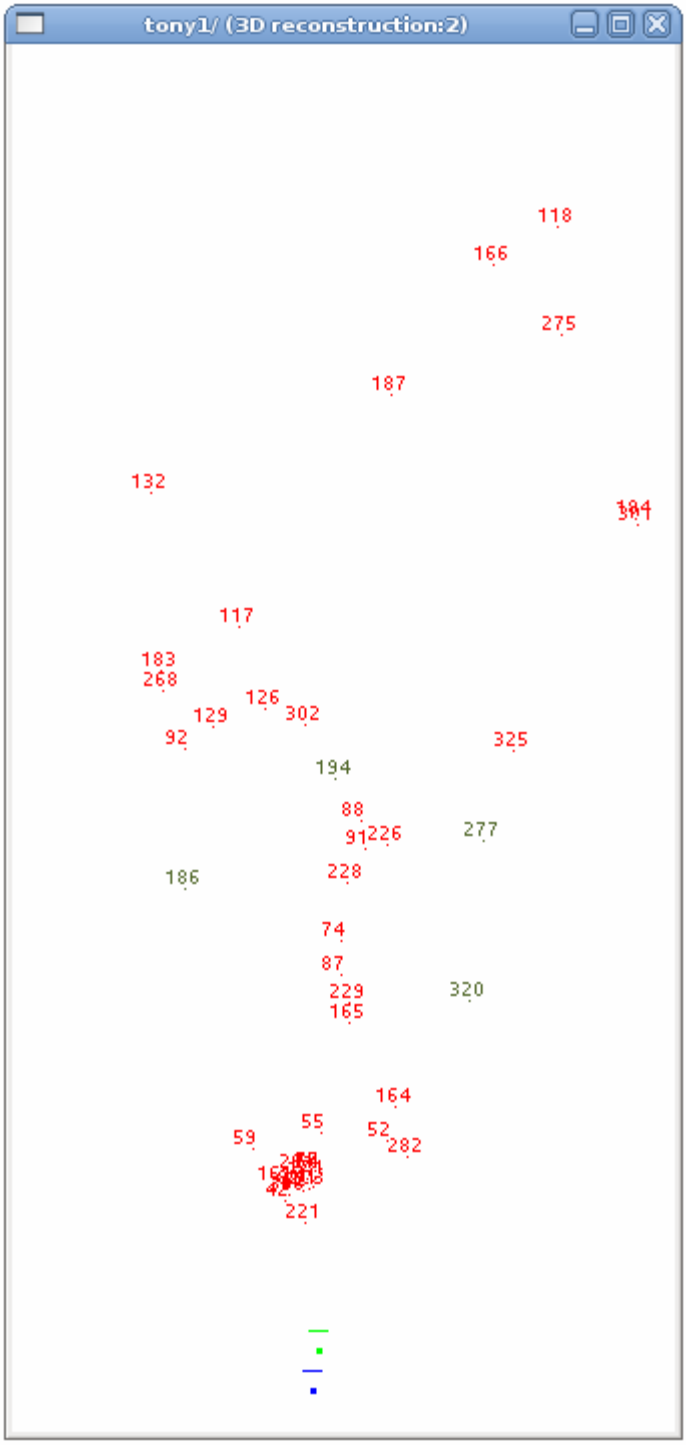
- prozor s referentnom slikom,
- prozor sa ispitnom slikom,
- prozor s tlocrtom trijanguliranih točaka i kamera,

Kontrola progama se vrši kroz tekstualno korisničko sučelje (slika 4.1., 4.2. i 4.3.).

Slika 4.1. Prozor s referentnom slikom (a)) i ispitnom slikom (b))



Slika 4.2. Kontrola programa iz terminala



Slika 4.3. Prozor na kojem se iscrtavaju kamere i trijangulirane točke

Na prozorima s referentnom i ispitnom slikom se iscrtavaju i pronađene korespodentne točke (slika 4.1.). Točke su obojene plavom bojom, svaka je

označena svojim identifikacijskim brojem i okružena plavim krugom čiji promjer govori o površini slike koju je algoritam SIFT obradio pri gradnji deskriptora.

Na prozoru s tlocrtom trijanguliranih točaka i kamera raferentna kamera je obojena plavom bojom a kamera s kojom uzimamo ispitne slike zelenom bojom. Kamere su prikazane s točkom optičkog centra i slikovnom ravninom ispred njih.

Nakon pokretanja ljuske u terminalu dajemo naredbu za obradu slike:

$p a <redni broj slike u uzorku>

Prva unesena slika je referentna a svaka sljedeća uzima se kao ispitna.

Promjena algoritma obrade slike može izvršiti naredbom:

$a <ime algoritma>

Osim algoritma *relor* možemo staviti npr. *sift* (algoritam SIFT)*, match* (algoritam uparivanja točaka) i dr. Za prikaz svih podržanih algoritama potrebno je upisati:

$h

Kada želimo prekinuti rad sa ljuskom potrebno je upisati:

$q

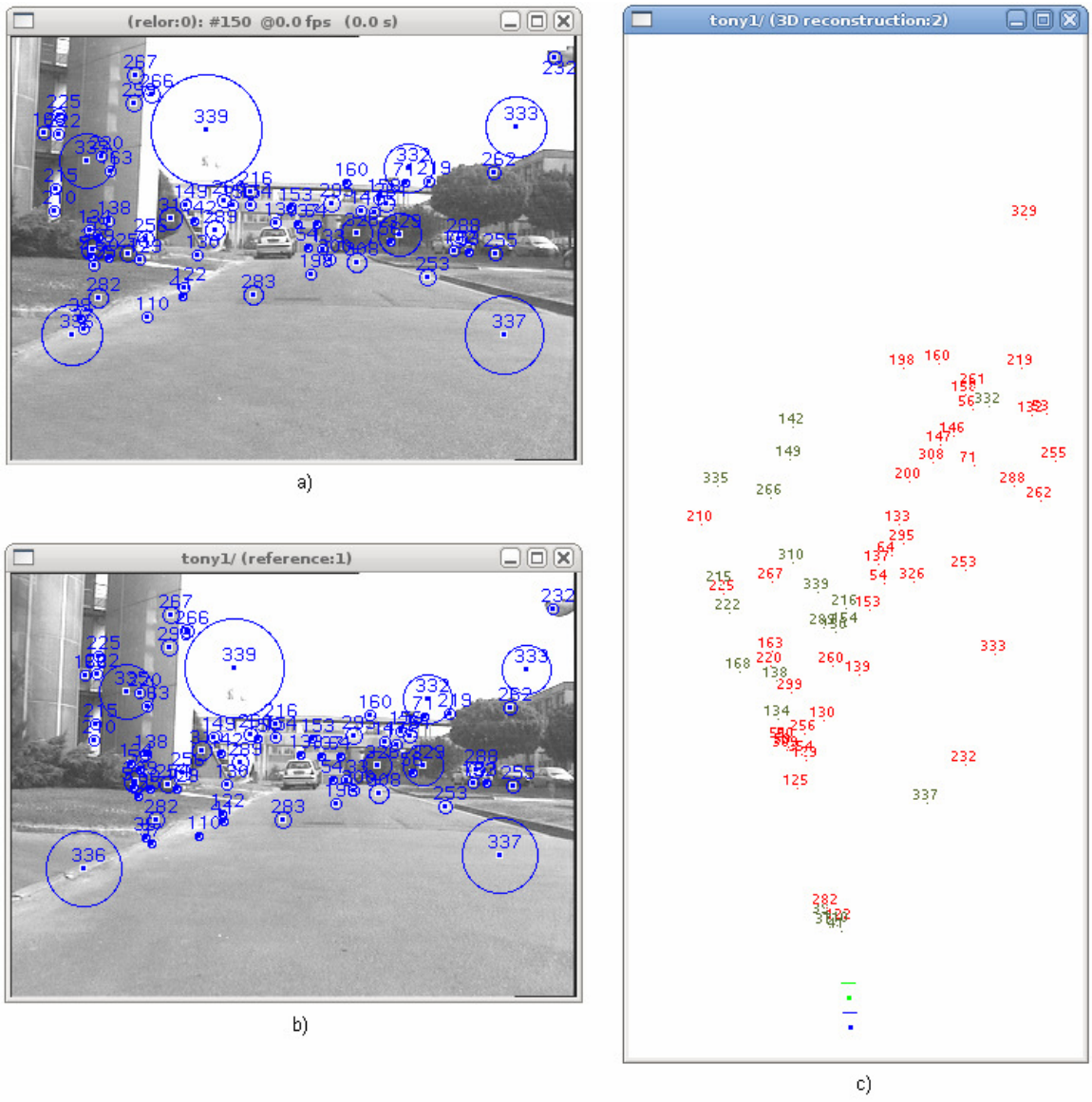
5. Ispitivanje algoritma i prikaz rezultata

Ispitivanjem se utvrđuje u kojim slučajevima algoritam ima dobar odziv a u kojim ne. Kao ocjena odziva uzimaju se reprojekcijske pogreške svih trijanguliranih točaka. Ispitivanje se vrši na slijedu slika pribavljenih iz jedne kamere u pokretu. Ispitivanje se može raditi sa slikama u formatu PGM (engl. portable graymap), BMP (engl. bitmap) te RAS (engl. raster).

5.1. Ispitni primjeri

Odziv algoritma ovisi o razlici udaljenosti scene na slikama i položaju korespodentnih točaka na sceni u odnosu na kamere.

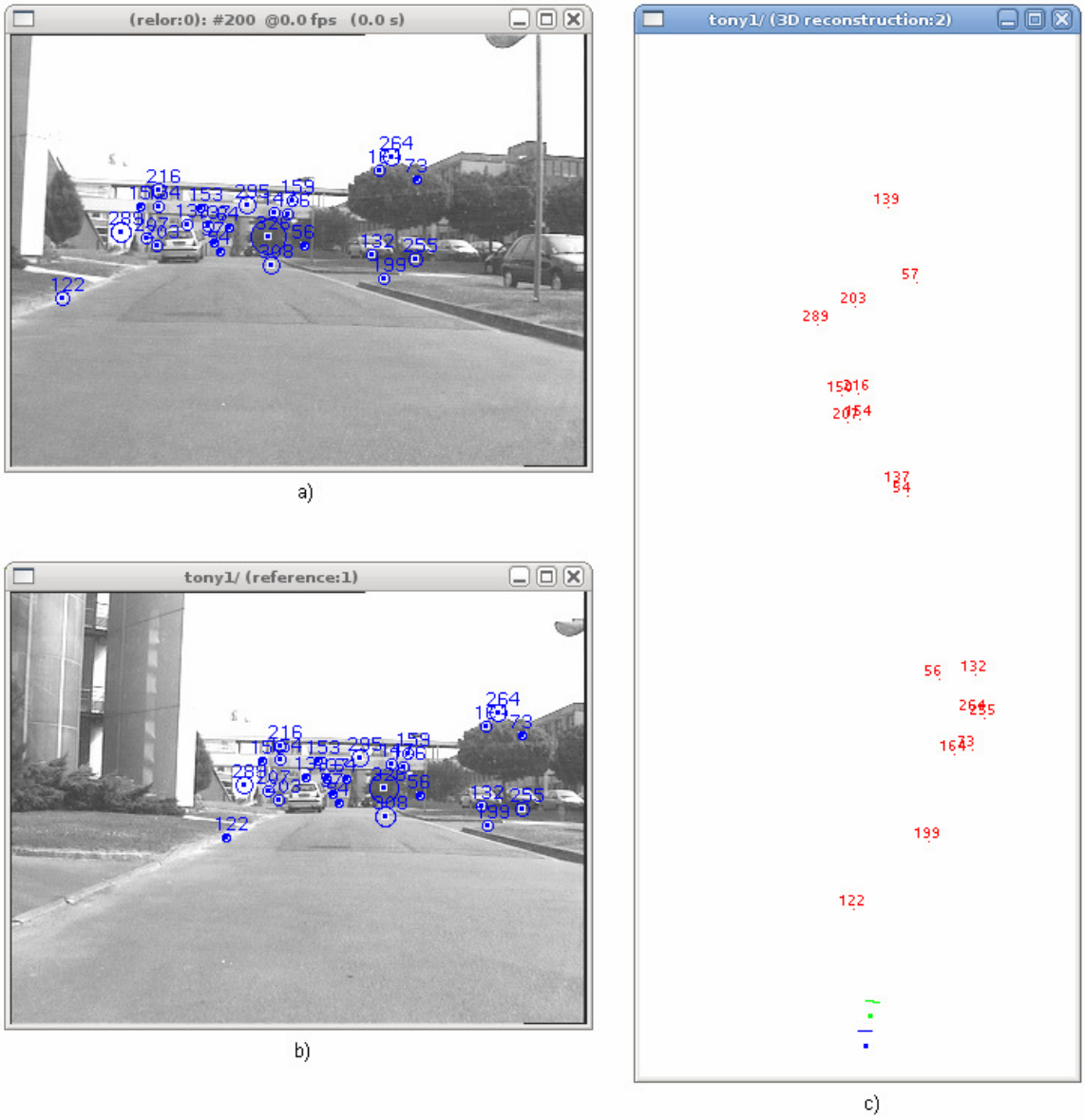
Povećanjem razlike udaljenosti scene na slikama algoritam pronalazi sve manje korespodentnih točaka (slike 5.1., 5.2., 5.3.) .



Slika 5.1. Razlika udaljenosti scene na slikama a) i b) je mala.

Na prozoru referentne i ispitne slike (slika 5.1. b) i a)) iscrtane su korespodentne točke a na prozoru s tlocrtom trijangulirane točke i kamere te njihov međusobni položaj u sceni. Položaj kamere kojom uzimamo ispitne slike (obojena zeleno) je ispred referentne kamere (obojena plavo), što odgovara stvarnom uzorku slika u ovom primjeru. Trijangulirane točke čija je reprojekcijska pogreška manja ili jednaka 5 piksela obojeni su crvenom bojom a ostale maslinasto zelenom. U danom primjeru broj pronađeni korespodentnih točaka je 69, ukupna reprojekcijska pogreška je iznosila 1118,52 piksela a prosječna

pogreška po točki 16,21 piksela. Algoritam nije pokazao dobar odziv ali je još uvijek reprojekcijska pogreška većine točaka ispod 5 piksela.



5.2. Razlika udaljenosti scene na slikama a) i b) je veća nego na slikama 5.1. a) i b)

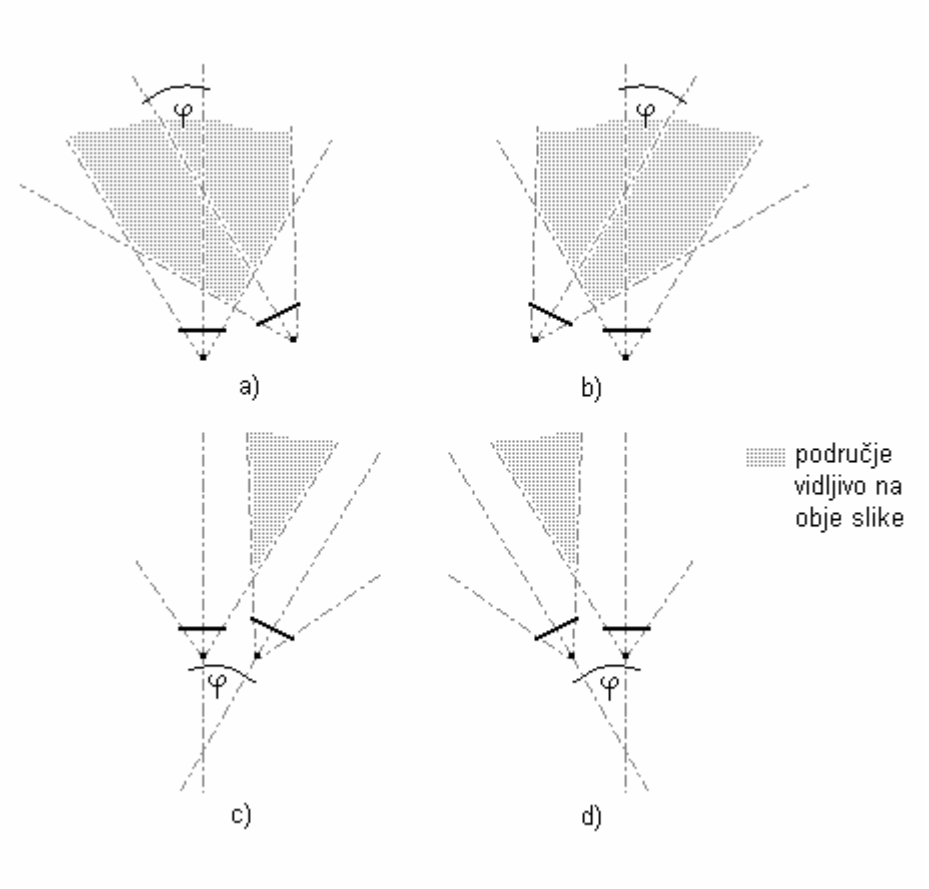
Broj korespodentnih točaka se bitno smanjio u odnosu na primjer sa slike 5.1. i iznosi 26 (slika 5.2.). Ukupna reprojekcijska pogreška je iznosila 12,5 piksela a prosječna pogreška po točki samo 0,48 piksela.

Slika 5.3. Razlika udaljenosti scene na slikama a) i b) je veća nego na slikama 5.1. a) i b) i 5.2. a) i b)

Broj korespodentnih točaka se smanjiva na slikama 5.3. a) i b) u odnosu na primjere sa slika 5.1. i 5.2. i iznosi samo 10. Ukupna reprojekcijska pogreška u ovom primjeru je iznosila 137,5 piksela a prosječna pogreška po točki 13,75 piksela.

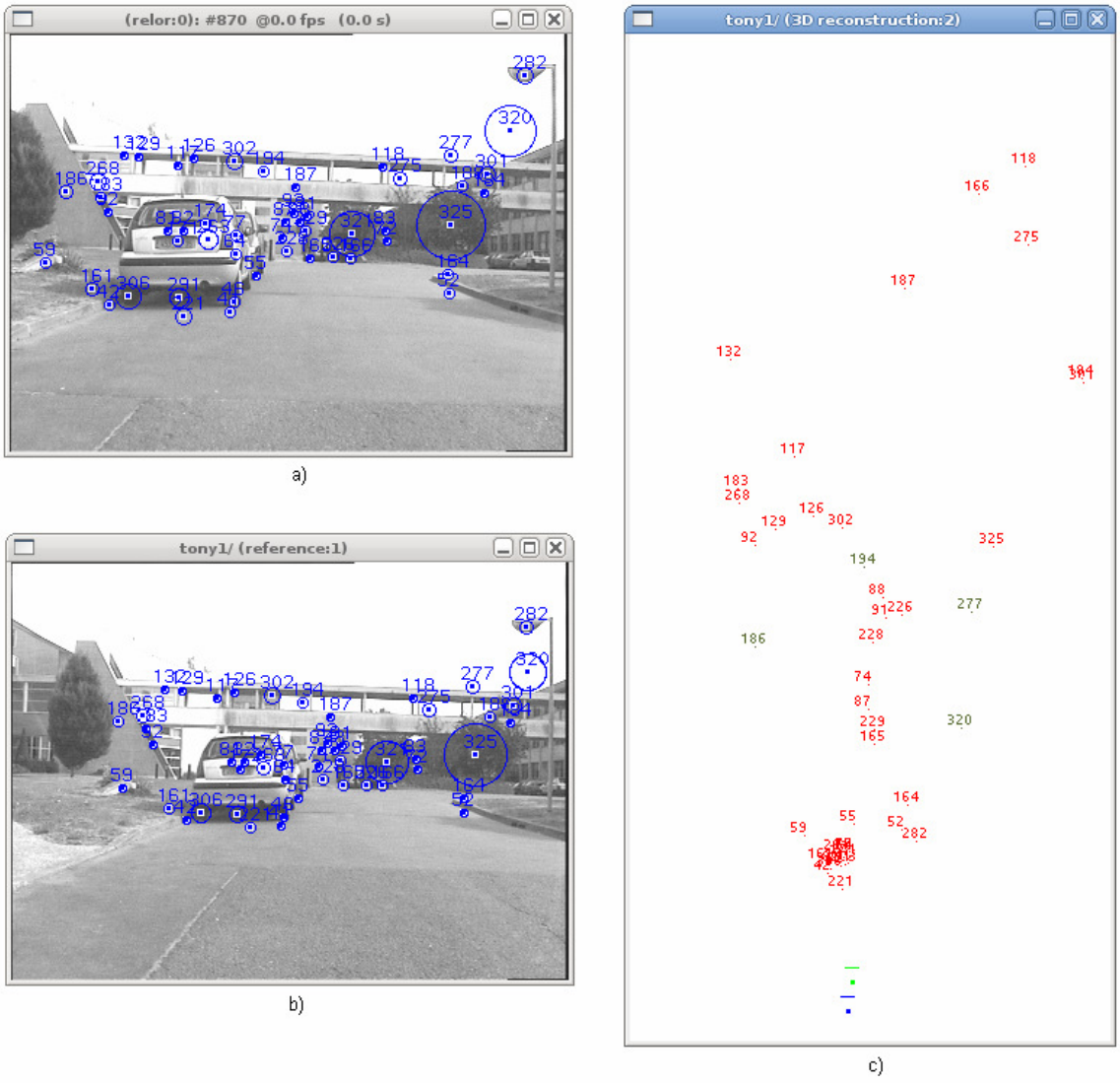
U primjeru sa slike 5.2. može se uočiti utjecaj drugog faktora na kvalitetu odziva algoritma tj. ovisnost o položaju korespodentnih točaka na sceni u odnosu

na kamere. Ako su korespodentne točke na sceni u blizini optičkih osi obje kamere (slika 5.4.) onda će njihova reprojekcijska pogreška biti manja.



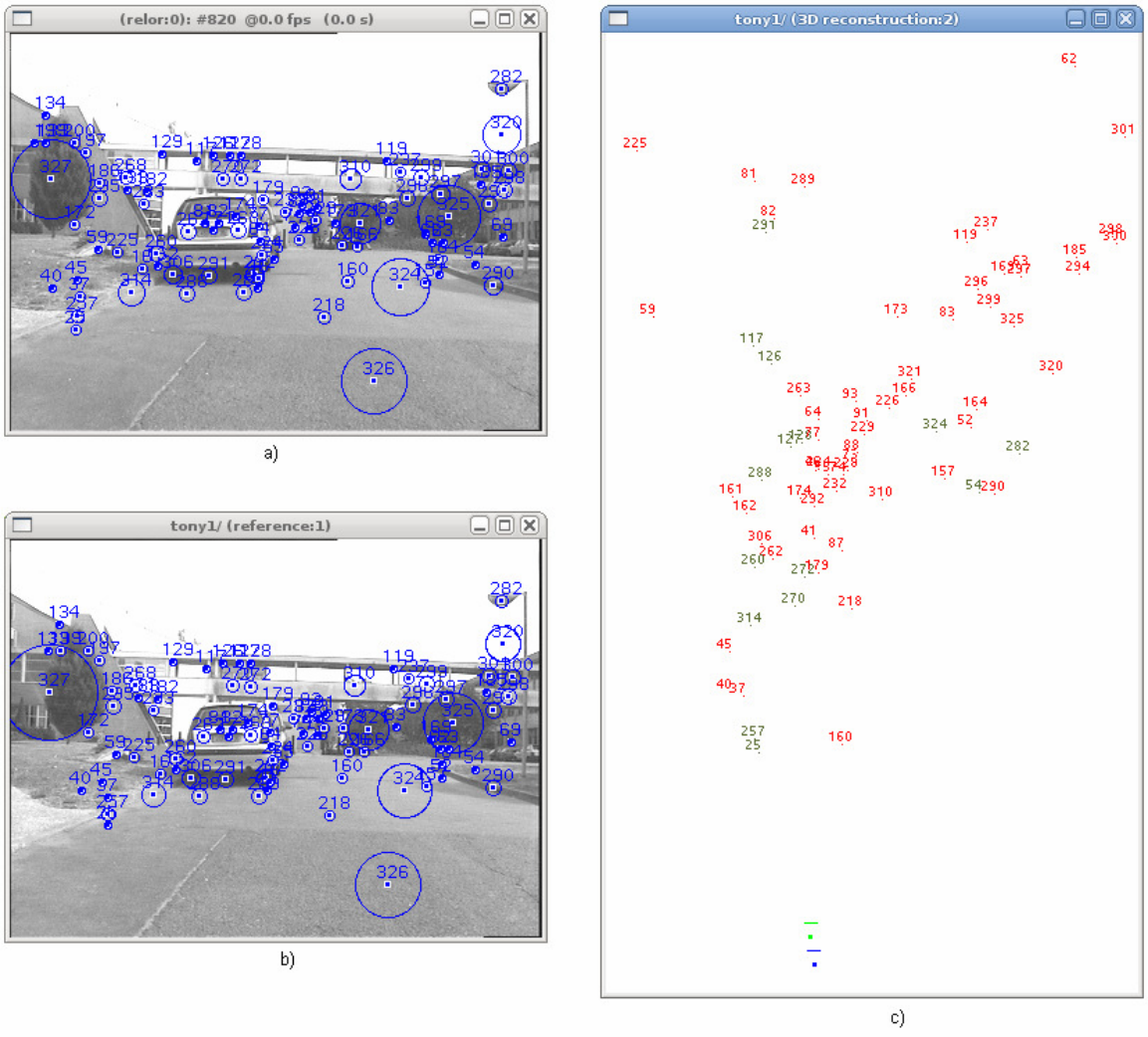
Slika 5.4. Položaj kamera u odnosu na scenu.

Položaj kamera u odnosu na scenu sa slike 5.4. a) i b) pokazuje kada će algoritam imati bolji odziv, tj. reprojekcijska pogreška trijanguliranih točaka će biti manja nego kada su položaji kamera u odnosu na scenu kao na slikama 5.3. c) i d). Utjecaj položaja kamera u odnosu na scenu je prikazan na slikama 5.5., 5.6., 5.7. i 5.8.



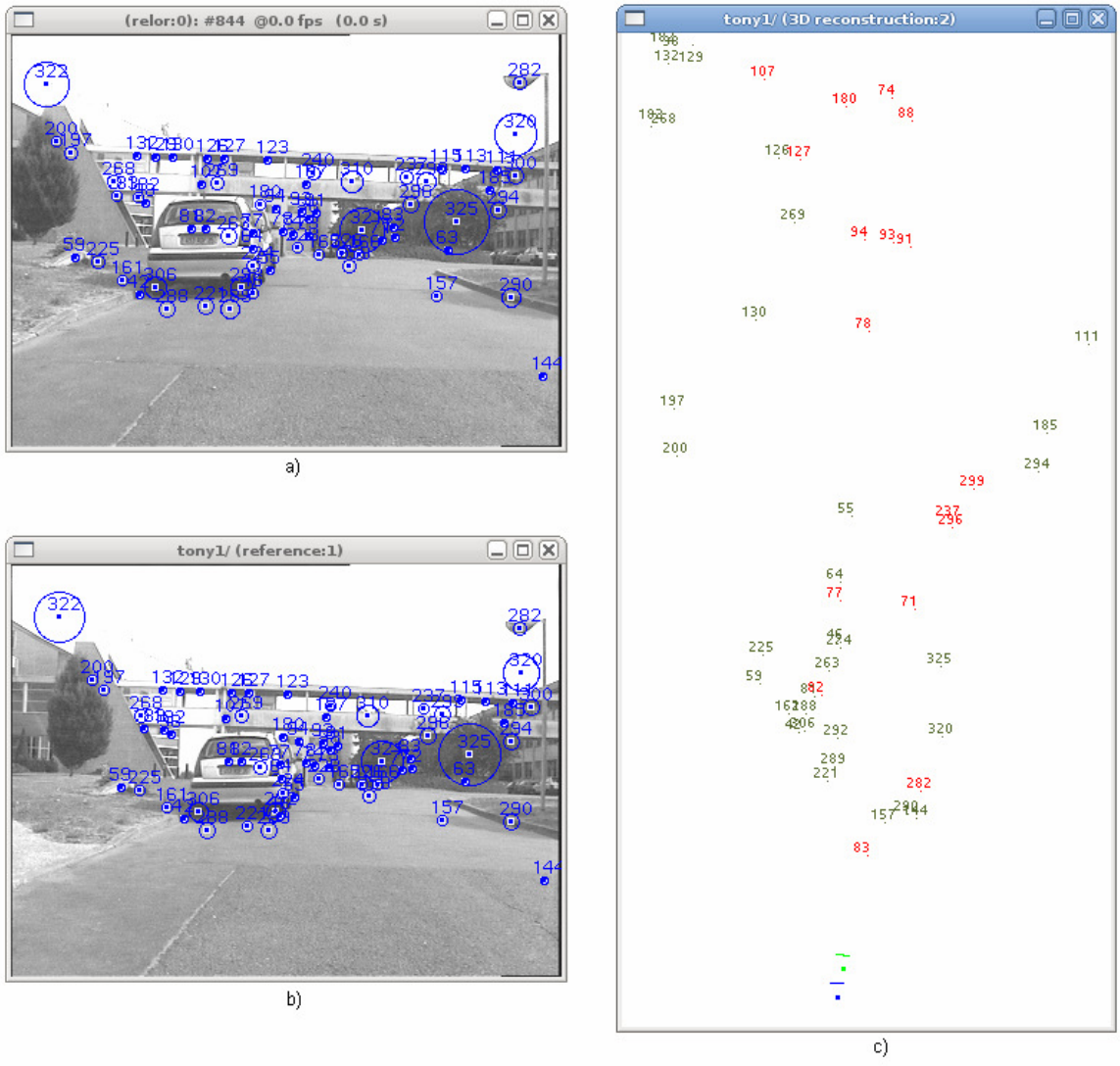
Slika 5.5. Položaj kamera u odnosu na scenu je sličan kao na slici 5.3 a)

Broj korespodentnih točaka je 51, ukupna reprojekcijska pogreška 190,84 a prosječna pogreška po točki 3,74 (slika 5.5.). Primjer pokazuje slučaj kada je položaj kamera sličan kao na slici 5.3. a) te algoritam daje dobar odziv.



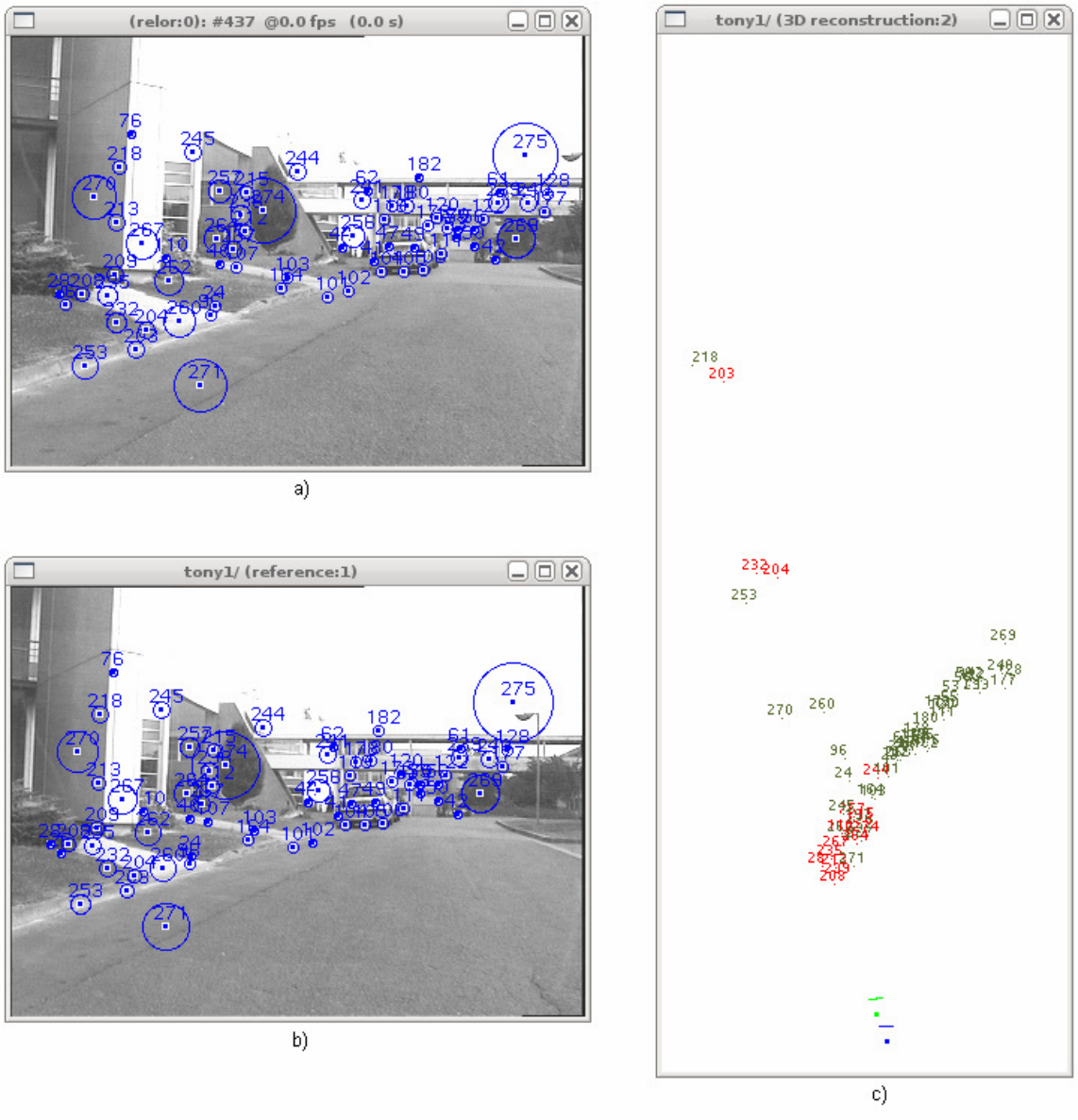
Slika 5.6. Položaj kamera u odnosu na scenu je sličan kao na slici 5.3 b)

Broj korespodentnih točaka je 90, ukupna reprojekcijska pogreška 374,41 a prosječna pogreška po točki 4,16 (slika 5.6.). Primjer pokazuje slučaj kada je položaj kamera sličan kao na slici 5.3. b). Algoritam daje dobar odziv.



Slika 5.7. Položaj kamera u odnosu na scenu je sličan kao na slici 5.3 c)

Broj korespodentnih točaka je 66, ukupna reprojekcijska pogreška 4427,4 a prosječna pogreška po točki 67,08 (slika 5.7.). Primjer pokazuje slučaj kada je položaj kamera sličan kao na slici 5.3. c). Algoritam daje loš odziv.



Slika 5.8. Položaj kamera u odnosu na scenu je sličan kao na slici 5.3 d)

Broj korespodentnih točaka je 66, ukupna reprojekcijska pogreška 5960,71 a prosječna pogreška po točki 87,65 (slika 5.8.). Primjer pokazuje slučaj kada je položaj kamera sličan kao na slici 5.3. d). Algoritam daje loš odziv.

6. Zaključak

Računalni vid je mnogo napredovao u zadnje vrijeme zahvaljujući složenim matematičkim modelima umjetnog vida te ubrzavanjem performansi računalnih sustava. Potrebe za sustavima sa sposobnošću računalnog vida su razne: video nadzor, multimedija, sučelje čovjek-računalo, poboljšavanje percepcije, sigurnosne provjere i dr. Broj implementacija takvih sustava u budućnosti će se povećavati, pogotovo što potrebna oprema postaje sve kvalitetnija i jeftinija.

U ovom radu su pokazani osnovni teorijski modeli računalnog vida, povezane postojeće implementacije, način nadogradnje sustava novim algoritmima i testiranje takvog sustava.

Pristupnik:

_____

Ivan Šakić

Literatura

[hartley04] Richard Hartley and Andrew Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, March 2004

[kalafatic08] Zoran Kalafatić i Siniša Šegvić, predavanja iz kolegija Dinamička analiza scena, 2008, postdiplomski studij računarstva

[loncaric08] Sven Lončarić, predavanja iz kolegija Digitalna analiza slike, 2008

[lowe04] David G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, 60, 2 (2004)

[ma05] Yi Ma, Stefano Soatto, Jana Kosecka and Shankar S.Sastry, An Invitation to 3-D Vision, Springer (June 17, 2005)

[mihajlovic08] Željka Mihajlović, predavanja iz kolegija Interaktivna računalna grafika, 2008

[piccardi03] M. Piccardi, T. Jan, Recent Advances in Computer Vision, Feb/Mar. 2003, American Institute of Physics

[segvic00] Siniša Šegvić, Uporaba projekcijske geometrije i aktivnog vida u tumačenju scena, Magistarski rad, Fakultet elektrotehnike i računarstva, Zagreb, Hrvatska, 2000.

[segvic04] Siniša Šegvić, Višeagentsko praćenje objekata aktivnim računarskim vidom, Doktorska disertacija, Fakultet elektrotehnike i računarstva, Zagreb, Hrvatska, 2004.

[segvic07] Siniša Šegvić, Gerald Schweighofer, Axel Pinz, Influence of numerical conditioning on the accuracy of relative orientation, 2007.

[segvic08] Siniša Šegvić, Gerald Schweighofer, Axel Pinz , Performance evaluation of closed-form approaches for recovering the relative orientation , 2008.

[vedaldi07] Andrea Vedaldi, An open implementation of the SIFT detector and descriptor, 2007., UCLA CSD Technical Report 070012

[vedaldi:www] Andrea Vedaldi , A lightweight C++ implementation of SIFT http://vision.ucla.edu/~vedaldi/code/siftpp/assets/siftpp/sift.html

Kratki sažetak

Opisuju se građevni elementi postupka za određivanje geometrije dvaju pogleda: pronalaženje korespodencija (algoritam SIFT), procjena relativne orijentacije (epipolarna geometrija, algoritam sa osam točaka, dekompozicija esencijalne matrice) i trijangulacija strukture scene. Navode se korištene vanjske biblioteke i njihovo prevođenje: implementacija algoritma SIFT, biblioteka VW, biblioteka Boost i biblioteka za evaluaciju postupaka relativne orijentacije. Prikazana je razvijena programska izvedba u okviru ljuske cvsh. Detaljno su opisani implementacijski detalji integriranja razvijenog postupka s ljuskom. Konačno, prikazani su eksperimenti na stvarnim scenama, te komentirani dobiveni rezultati.

Ključne riječi:
Računarski vid, epipolarna geometrija, esencijalna matrica, algoritam SIFT, algoritam sa osam točaka, određivanje relativne orijentacije, 3D rekonstrukcija scene

Summary

We describe building blocks for determining the two view geometry: finding correspondences (algorithm SIFT), estimation of relative orientation (epipolar geometry, eight point algorithm, decomposition of the essential matrix) and triangulation of the scene structure. The employed external libraried are specified, together with guidelines for their compilation: SIFT implementation, Boost library, VW library, and a pose evaluation library. The software implementation developed in the experimental computer vision framework cvsh is presented. Integration of the developed code with the framework is explained in detail. Finally, we present experimental results on real images discuss the obtained results.

Keywords:
Computer vision, epipolar geometry, essential matrix, algorithm SIFT, eight point algorithm, pose evaluation, 3D scene reconstruction

Privitak

Uz rad prilažem CD-ROM na kojem se nalaze izvorni kod ljuske cvsh sa svim algoritmima koji su korišteni i ispitni slijed slika nad kojim je vršeno testiranje.