SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

POSLIJEDIPLOMSKI DOKTORSKI STUDIJ ELEKTROTEHNIKE I INFORMACIJSKE TEHNOLOGIJE

KVALIFIKACIJSKI ISPIT

Procjena položaja objekta korištenjem fiducijalnih markera na velikom rasponu udaljenosti

Jurica Teklić

Split, rujan 2022.

SADRŽAJ

1. UVOD

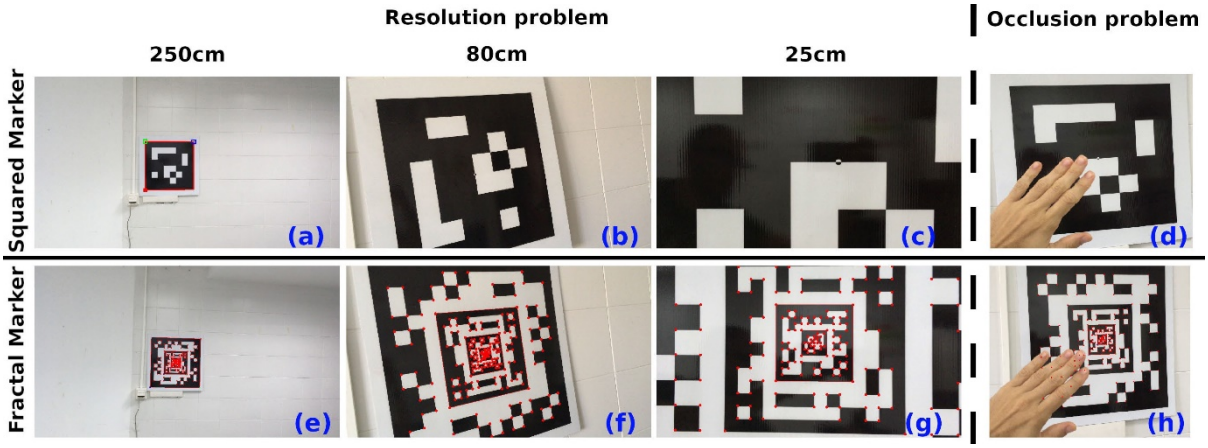
Na otvorenom prostoru nije uvijek jednostavno odrediti prirodne značajke prema kojima se može računati položaj senzora ili objekta koji ih promatra. Takve prilike utječu na složenost sustava za npr. slijetanje samostalne letjelice bez posade (UAV) [1]. Pouzdanije informacije o položaju objekta mogu pružiti umjetni orijentiri, kao što su fiducijalni markeri.

U medicini markeri mogu biti medicinski uređaji ili mali predmeti smješteni u ili na tijelu kojima se označava područje za zračenje ili operaciju [2]. Na taj način liječnik može primijeniti veće doze zračenja na npr. tumor s manje štete obližnjem zdravom tkivu [3]. Dakle, markeri predstavljaju referentne točke i primjenjuju se, između ostalog, kada ishod postupka jako ovisi o preciznosti. Primjene se mogu naći u svemirskim projektima [4], za istraživanja nad insektima [5], za pozicioniranje PCB-a [6] i mnoge druge. Jedan od primjera primjene markera je procjena položaja UAV letjelice tijekom slijetanja i uzlijetanja, naročito ako prenose vrijedan teret [7-9].

Kvadratni marker je vrsta fiducijalnog markera kvadratnog oblika koji se sastoji od vanjskog crnog obruba i unutarnjeg binarnog uzorka [10]. Oblik kvadratnog markera je definiran s četiri vanjska ugla što je dovoljno za procjenu položaja objekta. Spomenute karakteristike omogućavaju brzu detekciju, otpornost na osvjetljenje i perspektivnu projekciju, uz nisko korištenje CPU-a. Uzmu li se u obzir uglovi koda markera, procjena postaje još preciznija. Ukratko, pomoću ovih elemenata se postiže pouzdanija i jednostavnija detekcija, identifikacija i procjena položaja objekta u okolini.

Brojnost detektiranih uglova je posebno važna za procjenu položaja u prilikama u kojima dio markera neće biti vidljiv [11]. Takve prilike su za otvorene prostore neminovne i otežavaju detekciju fiducijalnih markera (slika 1d). Također, procjenom položaja kamere isključivo prema vanjskim uglovima se odbacuju važne informacije o unutarnjoj strukturi markera na osnovu kojih se može poboljšati preciznost procjene položaja [13]. Osim problema zaklanjanja, točna detekcija klasičnog markera je veoma ograničena rasponom udaljenosti kamere od markera, što odgovara problemu rezolucije (slika 1a-c). Takav marker, bio on većih ili manjih dimenzija, može biti koristan samo u nekom ograničenom periodu slijetanja ili polijetanja.

Fraktalni marker [12] je rekurzivna kompozicija kvadratnih fiducijalnih markera različitih veličina, smještenih jedan u drugi (slika 1f). Za razliku od ploče s markerima gdje su markeri pomaknuti, fraktalni markeri nemaju pomaka i dijele istu referentnu točku, zbog čega poprimaju rekurzivnu strukturu. Kao što je prikazano na slici 1(e-g), predloženi fraktalni marker može se detektirati iz šireg raspona udaljenosti od jednog markera. Također, ublažava problem djelomičnog zaklanjanja, budući da se položaj može procijeniti iz bilo kojeg markera, čak i ako je zaklanjanje prisutno na vanjskom markeru (slika 1(g, h)). Kako bi u potpunosti bio otporan na zaklanjanje, ovom metodom se može detektirati i procijeniti položaj markera detekcijom i klasifikacijom njegovih unutarnjih uglova. Dakle, ne samo da se može detektirati marker u slučaju zaklanjanja, već omogućava precizniju procjenu položaja iskorištavajući sve informacije o uglovima dostupnih u markeru.



Slika 1: Uobičajeni problemi kvadratnih markera: problem rezolucije (a-c) i problem zaklanjanja (d). Na slici (a-c) prikazan je kvadratni marker uočen na udaljenostima 250 cm, 80 cm i 25 cm od kamere i prekriven crvenim pravokutnicima detekcije ArUco [10] (detekcija samo u prvom slučaju). Pod istim uvjetima, slika (e-g) prikazuje dobivene rezultate prikazanog prijedloga (fraktalni marker) prekrivajući crvenom bojom detektirane unutarnje uglove markera. Slike (d, h) prikazuju rezultate u slučaju zaklanjanja obje metode. Kao što se može vidjeti, fraktalni markeri mogu se detektirati u više slučajeva od uobičajenih kvadratnih markera. [12]

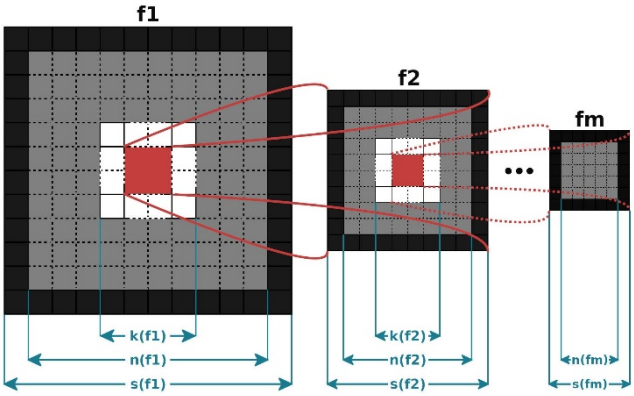
Namjera ovog rada je prezentirati fraktalni marker [12] kao prijedlog smjera rješavanja problema procjene položaja objekta na otvorenim prostorima npr. u prirodi za velike raspone udaljenosti. Odabir rješenja u vidu fraktalnog markera je usmjeren spomenutim kvadratnim fiducijalnim markerima koji se redovito koriste za pouzdanu procjenu položaja. Fraktalnim markerom se postiže širi raspon detekcije od tradicionalnih markera, te visoka otpornost na zaklanjanja, uz vrlo malo dodatnog vremena potrebnog za računanje.

2. DIZAJN FRAKTALNOG MARKERA

Kroz ovo poglavlje će biti opisan način generiranja fraktalnih markera proisteklih iz ArUco kvadratnih markera.

Fraktalni marker F se definira kao skup m kvadratnih markera $(f^1, f^2, \dots, f^m)$, postavljenih jedan u drugi na rekurzivni način (slika 2). U fraktalnom markeru svaki kvadratni marker f^i sadrži vanjski crni obrub (za brzu detekciju), područje rezervirano za identifikaciju (prikazano sivom bojom) i bijelo područje koje okružuje njegov unutarnji marker f^{i+1}, te ujedno olakšava detekciju crne granice unutarnjeg markera.

Neka su $s(f^i)$, $n(f^i)$ i $k(f^i)$ dužine stranica crnog područja (širina markera), identifikacijskog područja (prikazano sivom bojom) i bijelog područja (separator za lakšu detekciju unutarnjeg markera) za kvadratni marker f^i (slika 2). Postoji izuzetak za najunutarnjiji marker f^m, jer se unutar tog markera neće postavljati marker pa bijelo područje nije potrebno, tj. $k(f^m) = 0$. Vrijednosti se računaju s obzirom na referentni sustav s ishodištem u donjem lijevom vanjskom uglu unutarnjeg (promatranog) markera f^i.



Slika 2: Generička struktura fraktalnog markera F, u kojem je svaki marker sastavljen od skupa kvadratičastih elemenata (pločica) koje se mogu grupirati u tri kategorije. Crna traka odgovara granici markera, sive pločice konfiguriraju i jedinstveno određuju marker, i konačno, bijela traka olakšava detekciju unutarnjeg markera. [12]

Formalno govoreći, jedina ograničenja za vrijednosti od $s(f^i)$, $n(f^i)$ i $k(f^i)$ su:

$$s(f^{i+1}) < k(f^i) \forall i \neq m \tag{2.1}$$

i

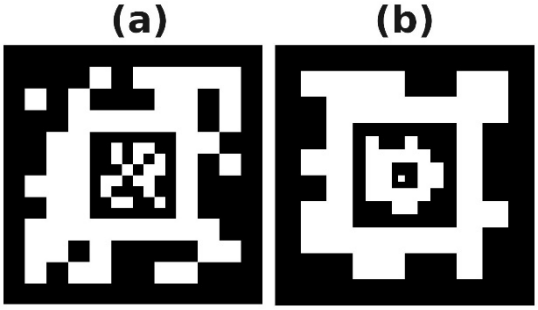
$$k(f^i) < n(f^i) < s(f^i) \forall i. \tag{2.2}$$

Svaki marker f^i može imati različit broj bitova za identifikaciju, ovisno o površini svoje identifikacijskog područja (duljine $n(f^i)$). Broj bitova u identifikacijskom području f^i je manji od tradicionalnog kvadratnog fiducijarnog markera.

Zatim, veličina kodiranja područja unutarnjih markera $f^i, i \in \{1, \dots, m\}$ je (slika 2):

$$S_R(f^i) = n(f^i)^2 - k(f^i)^2 \tag{2.3}$$

Slika 3 prikazuje dvije različite moguće kombinacije unutarnjih markera za fraktalni marker. Slika 3a prikazuje fraktalni marker sastavljen od dva unutarnja markera $s(f^1) = 12, n(f^1) = 10, k(f^1) = 6, S_R(f^1) = 64$ i $s(f^2) = 8, n(f^2) = 6, k(f^2) = 0, S_R(f^2) = 32$. Na slici 3b, fraktalni marker sastavljen je od tri unutarnja markera $(f^1) = 10,\ n(f^1) = 8,\ k(f^1) = 6,\ S_R(f^1) = 28;\ s(f^2) = 8,\ n(f^2) = 6,\ k(f^2) = 4,\ S_R(f^2) = 20\ i\ s(f^3) = 4,\ n(f^3) = 2,\ k(f^3) = 0, S_R(f^3) = 4$.



Slika 3: Primjeri različitih konfiguracija fraktalnog markera i identifikacijskih područja $S_R(f^i)$. (a) Fraktalni marker sastavljen od dva unutarnja markera $F = \{f^1, f^2\}$, čija su identifikacijska područja $S_R(f^1) = 64$ i $S_R(f^2) = 36$. (b) Fraktalni marker sastavljen od tri unutarnja markera $F = \{f^1, f^2, f^3\}$, čija su identifikacijska područja $S_R(f^1) = 28$, $S_R(f^2) = 20$ i $S_R(f^3) = 4$. [12]

Odabrana konfiguracija ovisi o potrebama aplikacije. Što je veći broj unutarnjih markera, to je veći radni raspon fraktalnog markera.

Označimo

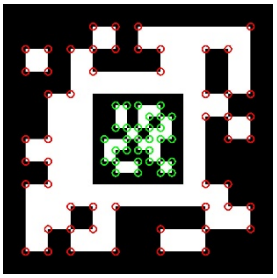
$$bits(f^i) = \left(b_1^i, . b_j^i, .. b_{S_R(f^i)}^i\right), \tag{2.4}$$

gdje su $b_j^i \in \{0,1\}, \forall j = 1, \dots, S_R(f^i)$ informacijski bitovi markera f^i. Niz bitova se slaže red po red počevši od gornjeg lijevog bita (slika 5). Unutarnji bitovi fraktalnog markera nasumično se generiraju korištenjem Bernoullijeve distribucije (tj. $b_j^i \sim Be(1/2)$). Međutim, niti jedna nasumično dobivena konfiguracija se ne može unaprijed smatrati valjanom jer će neke od njih će pod rotacijom biti identične. Zato se pseudoslučajno generirani marker smatra važećim tek kada je Hammingova udaljenost za tri moguće rotacije veća od nule, npr:

$$H\left(bits(f^i), bits\left(R_j(f^i)\right)\right) > 0, \forall j \in \left\{\frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}, \tag{2.5}$$

gdje je H Hammingova udaljenost između dva markera, a R_j funkcija koja rotira matricu markera f^i u smjeru kazaljke na satu ukupno j stupnjeva (slika 5). Ako uvjet jednadžbe 2.5 nije ispunjen, tada marker f^i nije valjan i postupak nasumičnog odabira bitova se ponavlja sve

dok se ne dobije valjani marker f^i. Fraktalni marker F vrijedi kada su valjani svi unutarnji markeri f^i.



Slika 4: Fraktalni marker sastavljen od dva unutarnja markera. Unutarnji uglovi markera f^1 i f^2 prikazani su crveno, odnosno zeleno. [12]

Detekcija i procjena položaja markera temelji se na detekciji i analizi projekcija uglova markera na slici. U nastavku su trodimenzionalne koordinate četiriju vanjskih uglova f^i u odnosu na središte markera:

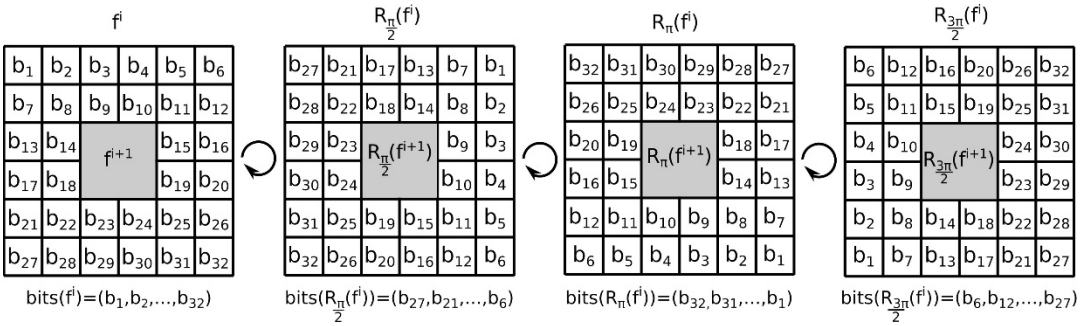
$$c_1^i = \left(s\left(f^i\right)/2\,, -s\left(f^i\right)/2\,, 0\right)$$

$$c_2^i = \left(s\left(f^i\right)/2\,, s\left(f^i\right)/2\,, 0\right)$$

$$c_3^i = \left(-s\left(f^i\right)/2\,, s\left(f^i\right)/2\,, 0\right)$$

$$c_4^i = \left(-s\left(f^i\right)/2\,, -s\left(f^i\right)/2\,, 0\right) \qquad (2.6)$$

Ako se pretpostavi da je marker ispisan na ravnoj površini, onda treća komponenta postaje nula za sve uglove.



Slika 5: Četiri moguće rotacije markera f^i. [12]

Pored četiri vanjska ugla $c_j^i \in R^3$ (jednadžba 2.6) svakog markera f^i, postoji niz unutarnjih uglova (vidi sliku 4) koji se mogu iskoristiti za praćenje markera u slučaju zaklanjanja, kao i za poboljšavanje procjene položaja.

Neka je W^i skup unutarnjih uglova markera F:

$$W^i = \left(w_1^i, \ldots, w_n^i\right), w_j^i \in R^3 \qquad (2.7)$$

gdje su w_j^i trodimenzionalne koordinate u odnosu na središte markera. Sl. 4 prikazuje primjer fraktalnog markera sastavljenog od dva markera f^1 i f^2 gdje su njihovi unutarnji uglovi

prikazani kao crveni i zeleno obojeni krugovi. Četiri vanjska ugla markera nisu uključena kao unutarnji uglovi bilo kojeg markera.

Na kraju, neka je

$$C^i = \left\{\{W^i\}, c_1^i, c_2^i, c_3^i, c_4^i\right\} \tag{2.8}$$

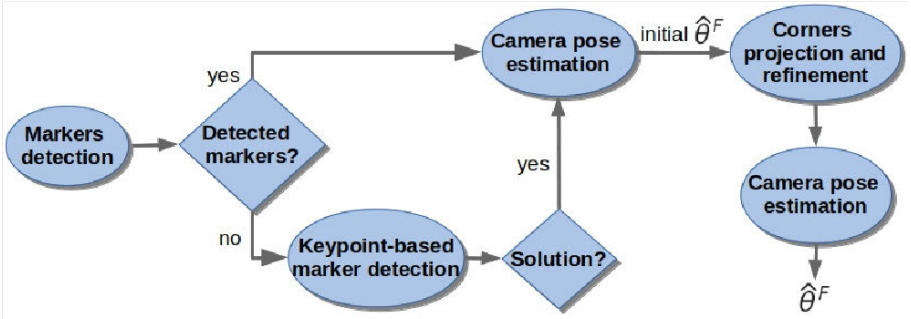
skup unutarnjih i vanjskih uglova svakog markera $f^i \in F$, i

$$C(F) = \left\{\{C^i\}/f^i \in F\right\} \tag{2.9}$$

na skup svih uglova markera fraktalnog markera F.

3. DETEKCIJA FRAKTALNOG MARKERA

U ovom poglavlju će biti obrađena metoda za detekciju i praćenje zaklonjenih fraktalnih markera, a slika 6 prikazuje tijek rada. U provom koraku se odvija detekcija markera (potpoglavlje 3.1.). Ako se detektira barem jedan marker, detektirani uglovi se koriste za dobivanje početne procjene položaja markera (potpoglavlje 3.2.), koji se koristi za projiciranje očekivanog mjesta fraktalnih uglova $C(F)$ fraktalnog markera. Projektirane lokacije upotrebljavaju se kao polazna točka procesa izoštravanja kako bi se točno pronašlo njihovo mjesto na slici. Čitav niz dorađenih uglova se zatim koristi za ponovno računanje položaja markera, koji sad sadrži više točaka na osnovu čega mu se preciznije određuje mjesto (potpoglavlje 3.3.).



Slika 6: Opći tijek rada metode za procjenu položaja markera. [12]

Ako markeri nisu detektirani u početnom koraku, predložena metoda nastoji pronaći mjesto markera koristeći prethodnu detekciju kao početnu točku pa se koristi detektor ugla FAST [15]. Uglovi su zatim razvrstani u tri kategorije (potpoglavlje 3.4.). Nakon toga se koristi metoda za usklađivanje promatranih uglova s uglovima markera $C(F)$ pomoću algoritma RANSAC. Kao rezultat, predloženom metodom se može odrediti početni položaj markera. U ovom se trenutku algoritma promatrana grana tijeka rada spaja s drugom u koraku „projekcije ugla“ kako bi dobio dorađeni položaj markera (potpoglavlje 3.4.).

3.1. Detekcija markera

Prvi korak postupka je pokušaj detekcije markera f^i koji tvore fraktalni marker. Taj je postupak isti kao u [4] i može izvući samo najudaljenije vanjske uglove c_j^i markera. U tu svrhu se primjenjuju sljedeći koraci:

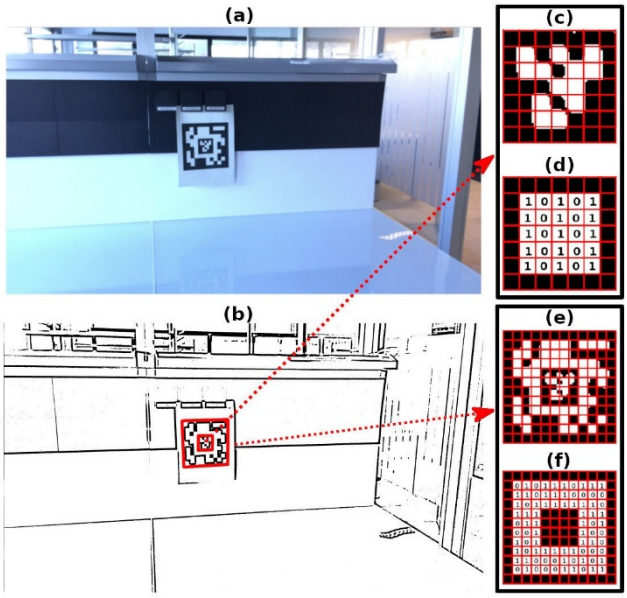
3.1.1. Segmentacija slike

Fraktalni marker sastoji se od više kvadratnih markera koji imaju crni okvir okružen bijelim prostorom koji olakšava njegovu detekciju. Metoda koristi lokalni adaptivni prag koji omogućava detekciju otpornu na svjetlosne uvjete (slika 7b).

Za primjenu lokalnog adaptivnog praga je potrebno odrediti najmanju i najveću veličinu prozora za konvoluciju, korak povećanja veličine prozora, te konstantu praga. Ako korisnik ne odredi eksplicitno parametre, podrazumijevaju se slijedeće vrijednosti:

```
int adaptiveThreshWinSizeMin = 3
int adaptiveThreshWinSizeMax = 23
int adaptiveThreshWinSizeStep = 10
double adaptiveThreshConstant = 7
```

Algoritam lokalnog adaptivnog praga započinje s konvolucijom, a vrijednosti rezultata se oduzimaju od originalne slike. Nakon toga se primjeni prag intenziteta što rezultira pikselima s vrijednostima 1 ili 0. Konačan rezultat ovog algoritma su invertirane vrijednosti svakog piksela prethodnog koraka.



Slika 7: Detekcija i identifikacija fraktalnih markera. (a) Izvorna slika. (b) Slika ograničenog intenziteta koja prikazuje rezultat izvlačenja i filtriranja kontura. (c i e) Kanonske slike pravokutnih kontura koje sadrže promatrane markere. (d i f) Binarne verzije kanonskih slika. [12]

3.1.2. Izvlačenje kontura i filtriranje

Izvlačenje kontura svakog unutarnjeg markera vrši se algoritmom Suzuki i Abe [16]. Pruža skup kontura od kojih mnoge odgovaraju neželjenim objektima. Algoritam topološki analizira binarnu sliku, određuje odnose između vanjskih i unutarnjih granica na osnovu kojih se mogu izvući značajke.

Proces filtriranja provodi se koristeći Douglas-Peuckerov algoritam [17] koji odabire samo one koji su najsličniji poligonu (slika 7b) usporedbom točaka promatranih i željenih kontura. Cilj algoritma je proizvesti pojednostavljenu poli-liniju koja ima manje točaka od originala, ali i dalje zadržava karakteristike i oblik izvornika. Kako samo četverokuti definiraju dijelove slike na kojima se može nalaziti marker, konture oblika ostalih mnogokuta bivaju odbačene.

3.1.3. Izvlačenje koda markera

Sljedeći korak je analiza unutarnjeg područja preostalih kontura kako bi se utvrdilo koji su od njih valjani markeri. Prvo, potrebno je ukloniti perspektivnu projekciju (pomoću homografske transformacije na osnovu četiri vanjska ugla) i potom filtrirati Otsuovom metodom [18].

Rezultirajuća slika podijeljena je pravokutnom rešetkom i svakom se elementu dodjeljuje vrijednost 0 ili 1, ovisno o vrijednostima većine piksela (slika 7 (c-f)) Na kraju je potrebno usporediti kandidata sa skupom važećih markera. Izvode se četiri moguće usporedbe svakog kandidata, što odgovara četiri moguće orijentacije.

Kao rezultat postupka dobiva se početni skup uglova vanjskih markera C' koji pripadaju vanjskim crnim okvirima. Od njih se može dobiti početni položaj kako je prikazano u idućem potpoglavlju.

3.1.4. Procjena položaja markera

Položaj markera $\theta \in \mathbb{R}^6$se definira s njegove tri rotacijske i translacijske komponente $r = (r_x, r_y, r_z)$i $t = (t_x, t_y, t_z)$:

$$\theta = (r, t) \vee r, t \in \mathbb{R}^3 \tag{3.1}$$

Koristeći Rodriguesovu formulu rotacije:

$$\mathbf{R} = \mathbf{I} + (\sin \varphi)\mathbf{V} + (1 - \cos \varphi)\mathbf{V}^2 \tag{3.2}$$

$$\varphi = \|\mathbf{r}\|;\ \mathbf{v} = \frac{\mathbf{r}}{\varphi};\ \mathbf{V} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_z \\ -v_y & v_x & 0 \end{bmatrix}$$

matrica rotacije **R** se može dobiti iz **r**.

Točka $p \in \mathbb{R}^3$ projicira se u ravninu kamere u piksel $u \in \mathbb{R}^2$. Pod pretpostavkom da su parametri kamere poznati, projekcija se može dobiti kao funkcija:

$$u = \Psi(\delta, \theta, p), \tag{3.3}$$

gdje se

$$\delta = (f_x, f_y, c_x, c_y, k_1, \dots, k_n), \tag{3.4}$$

odnosi na unutarnje parametre kamere, koji se sastoje od parametara žarišta (f_x, f_y), optičkog centra (c_x, c_y) i parametara izobličenja $(k_1, \dots, k_n)$ [14].

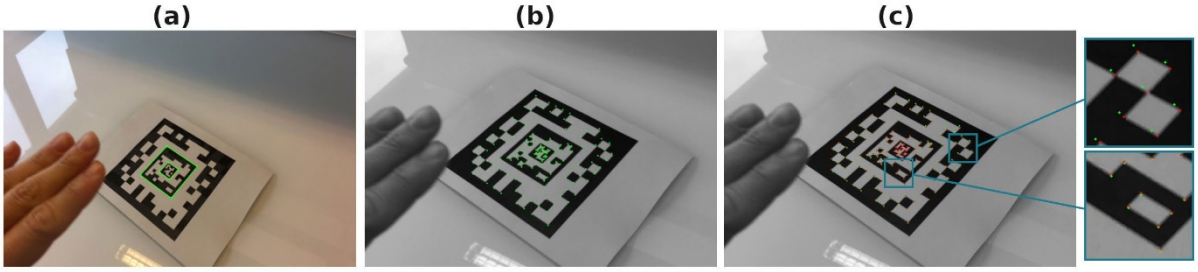
Procjena položaja markera je sada problem minimiziranja pogreške reprojekcije promatranih uglova markera:

$$\hat{\theta} = \underset{\theta}{argmin} \sum_{p \in D} [\Psi(\delta, \theta, p) - O(p)]^2 \quad (3.5)$$

gdje je $O(p) \in \mathbb{R}^2$ promatrani položaj ugla $p \in \mathcal{D}$ na slici kamere. Skup uglova $\mathcal{D}$ može imati bilo koju vrstu uglova (tj. vanjske i unutarnje uglove). Poseban slučaj je kada sve točke leže u istoj ravnini, a može se riješiti specifičnim metodama poput Infinitezimalne procjene položaja na temelju ravnine (IPPE) [19].

3.1.5. Projekcija ugla i izoštravanje

Nakon početne procjene položaja markera iz smanjenog skupa uglova C', moguće je pronaći sve vidljive uglove i pomoću njih još preciznije odrediti položaj. Za to se najprije na slici kamere projiciraju svi markeri u $C(F)$ (jed. 3.3). Zatim se njihova lokacija dorađuje do preciznosti podpiksela. Dobivene vrijednosti se tada koriste za određivanje još preciznijeg položaja ponovo koristeći jed. 3.5.



Slika 8: (a) Detekcija markera i vanjskih uglova na izvornoj slici. (b) Početna procjena položaja pomoću vanjskih uglova detektiranih markera. (c) Doradivanje procjene položaja: zelene točke predstavljaju procjenu prethodnog koraka (b), crvenom bojom novu procjenu. [12]

Izoštravanje uglova podpiksela sastoji se u analizi malog kvadratnog područja duljine s_{min} oko mjesta ugla kako bi se pronašli maksimumi derivacije unutar područja. Na manjim slikama područje analize je također manje zbog čega se znatno smanjuje vrijeme računanja. Slijedom toga, postupak izoštravanja ugla izvodi se kao postupak kroz više nivoa veličine slike iz piramide izvorne slike. Proces započinje pronalaženjem piramide za svaki ugao manje slike na kojoj se procjena ugla može inicijalno doraditi. Nakon početnog izoštravanja, procjena njegovog mjesta se ponovno dorađuje na sljedećoj (i većoj) slici piramide. Postupak se ponavlja sve dok se položaj ugla konačno ne izoštri na izvornoj slici.

Neka je $I = (I^0, I^2, \dots, I^p)$ oznaka za piramidu slika, gdje je I^0 izvorna slika, koja se skalira faktorom dva. Za svaki marker se odabire početna sliku u piramidi $I^j \in I$ za izoštravanje kao:

$$I^j = \underset{I^i \in I}{argmin} |P(f) - \tau(f)^2| \quad (3.6)$$

gdje je $P(f)$ projicirano područje markera f na slici I^i, a $\tau(f)$ optimalna duljina markera za izoštravanje, s tim da između uglova za izoštravanje mora postojati minimalan broj piksela. Definira se $\tau(f) = s_{min} \times s(f)$ pa tako npr. za $s_{min} = 10$, za marker f takav da je $s(f) = 12$,

se dobiva $\tau(f) = 120$. Ako marker izgleda vrlo malo na izvornoj slici I^0 (tj. $P(f) < \tau(f)$), njegovi uglovi nisu izoštreni niti se koriste za procjenu položaja.

Na slici 8a je prikazana ulazna slika gdje su dva unutarnja markera (prikazana zelenom bojom) detektirana metodom opisanom u potpoglavlju 3.1. Slika 8b prikazuje projicirane unutarnje uglove nakon prve procjene položaja. Konačno, slika 8c crvenom bojom prikazuje izoštrena mjesta ugla. Kao što se može primijetiti, početno projicirani uglovi (zeleni) nisu toliko precizno smješteni kao izoštreni. Izoštreni uglovi koriste se kasnije kako bi se dobila preciznija procjena položaja markera.

Postupak obrade ugla mora također uzeti u obzir mogućnost zaklanjanja, tj. postupak izoštravanja se ne može provesti za markere koji su na slici zaklonjeni. Da bi se uzela u obzir ta situacija, tijekom postupka izoštravanja analizira se nekoliko uvjeta za svaki ugao. Prvo se analizira ima li područje oko ugla slab kontrast. Budući da se koriste crno-bijeli markeri, za očekivati je da će ugao biti u području visokog kontrasta, pa ako je razlika između najsvjetlijih i najmračnijih piksela unutar područja ugla manja od τ_c, ugao se smatra zaklonjenim i odbacuje se iz postupka. Drugo, odbacuju se uglovi koji podliježu velikim pomacima tijekom postupka izoštravanja.

3.2. Detekcija markera prema ključnoj točki

U slučaju da nakon koraka detekcije markera (potpoglavlje 3.1.) nije detektiran nijedan marker, metoda ima cilj pronaći fraktalni marker koristeći prethodno dostupnu detekciju. Da bi to bilo izvedivo, provodi se pretraživanje uglova markera oko njihove posljednje promatrane lokacije koristeći pristup temeljen na ključnoj točki koji će biti obrađen kroz iduće odjeljke.

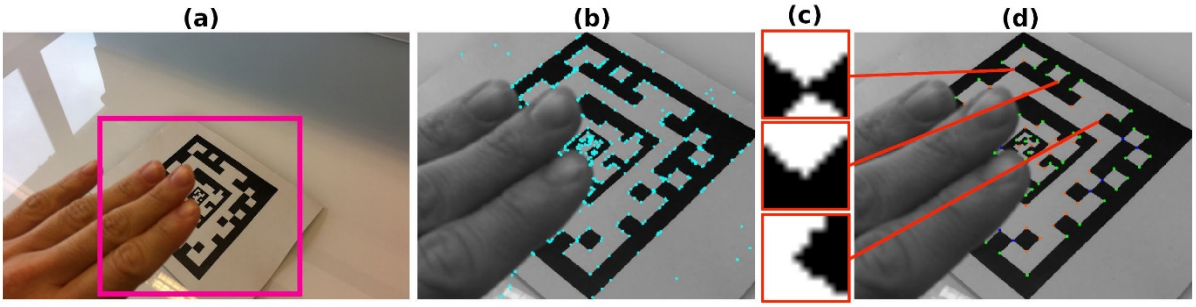
3.2.1. Procjena područja od interesa

Ako pomicanje markera (ili kamere) nije jako brzo, marker bi se trebao pojaviti u sljedećem kadru blizu lokacije u prethodnom. Kako bi se ubrzao postupak, definirano je područje kojim se ograničavaju moguća mjesta za detekciju ugla (sljedeći korak). Područje je definirano oko središta prethodne detekcije markera, s površinom nešto većom od prethodno promatranog područja markera (slika 9a). Doista, u slučaju velikog pomicanja fotoaparata između kadrova, područje od interesa možda ne prekriva novi položaj markera i stoga marker možda neće biti pronađen. U tom slučaju je potrebno pričekati dok se marker ne detektira prethodno objašnjenom metodom (potpoglavlje 3.1.).

3.2.2. Detekcija i klasifikacija ugla

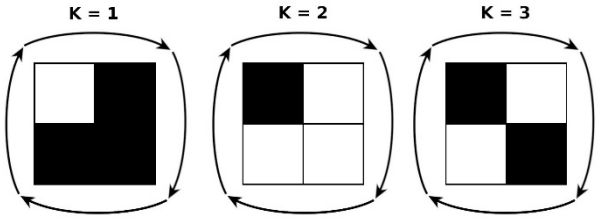
Algoritam detekcije ključnih točaka FAST [15] primjenjuje se u području od interesa (slika 9b) i provodi se nekoliko ispitivanja za svaku detektiranu ključnu točku kako bi se uklonile one za koje je manja vjerojatnost da pripadaju uglovima markera. Prvo, uklanjaju se ključne točke sa slabim odzivom detektora FAST, zadržavajući samo ove iznad 20. postotka. Drugo, uklanja se ključna točka ako je kontrast u četvrtastom kvadratu veličine $l \times l$ piksela manji od τ_c.

Eksperimentalno je primijećeno da vrijednost $l = 10$ daje dobre rezultate. Za preostale ključne točke se primjenjuje algoritam koji analizira pripada li jednoj od tri moguće kategorije $K \in 1,2,3$, odnosno tri vrste uglova koje marker može imati (slika 10). Može se promatrati kao vrlo jednostavan deskriptor ključne točke sa samo tri različite vrijednosti.



Slika 9: (a) Izvorna slika koja prikazuje područje interesa. (b) Rezultati primjene FAST detektora (plave točke). (c) Primjeri klasifikacije uglova (d) Filtrirane i klasificirane ključne točke. Svaka boja (plava, zelena i crvena točka) predstavljaju različitu klasu ključnih točaka.[12]

Metoda za klasifikaciju ključnih točaka objašnjena je u algoritmu 1. Prvo, područje oko ključne točke se binarizira koristeći prosječni intenzitet piksela kao prag. Zatim se izračunavaju povezane komponente i za klasifikaciju se primjenjuju jednostavna pravila prikazana u linijama 5-13. Rezultat klasifikacije ključnih točaka na slici 9b prikazan je na slici 9 (c-d), pri čemu je ključna točka $K = 1$ prikazana zelenom bojom, $K = 2$ crvenom bojom i $K = 3$ plavom bojom.



Slika 10: Tri kategorije kojima ključna točka može pripadati. Svaka ključna točka dodijelit će se jednoj od ove tri kategorije ili će se odbaciti. [12]

Algoritam 1 klasifikacija ključnih točaka

1: $R \leftarrow roi(I, k, l)$ # Područje od interesa na slici I, centrirano u ključnoj točki k područja $l \times l$
2: $R^b \leftarrow thresholdAvrg(R)$ # Binariziranje R korištenjem sred. intenziteta piksela za granicu
3: $C \leftarrow connectedComponents(R^b)$ # Određivanje broja povezanih komponenata R^b
4: $K \leftarrow 0$ # Inicijalizacija klase
5: **if** $C = 2$ **then**
6: **if** $countNonZero(R^b) > countZero(R^b)$ **then**
7: $K \leftarrow 1$; # Postavi k kao klasu 1
8: **else**
9: $K \leftarrow 2$; # Postavi k kao klasu 2
10: **end if**
11: **else if** $C > 2$ **then**
12: $K \leftarrow 3$; # Postavi k kao klasu 3
13: **end if**
14: **return** K

3.2.3. Povezivanje ključnih točka RANSAC-om

Nakon što su ključne točke klasificirane, sljedećim korakom se određuje kojem unutarnjem uglu markera (W^i) odgovara svaka ključna točka. Iako klasifikacija značajno pomaže smanjiti broj kandidata, nije dovoljna za jedinstveno podudaranje s ključnom točkom. Korištenjem prethodne detekcije fraktalnih markera se dodatno smanjuju moguća podudaranja uz računanje radijusa za pretraživanje r. Radijus se automatski izračunava prema iznosu vidljive površine markera. Pod pretpostavkom da pomicanje kamere/markera nije jako veliko, detektirana ključna točka mora odgovarati bilo kojem unutarnjem uglu uočenom unutar područja pretraživanja na prethodnoj slici. Bez obzira na to, svakoj ključnoj točki se može dodijeliti više od jednog unutarnjeg ugla iste klase. Stoga se koristi metoda s RANSAC pristupom za robusno podudaranje svake ključne točke sa pripadajućim unutarnjim uglom.

Osnovna zamisao je da postoji homografija koja povezuje unutarnje uglove W^i s promatranim ključnim točkama na slici kamere. Minimalni broj korespondencija za izračunavanje takve homografije je četiri, a ako su korespondencije točne, tada će homografija projicirati unutarnji ugao vrlo blizu detektirane ključne točke odgovarajuće klase. U tom slučaju nastaje podudaranje, i ako je homografija izračunata pomoću ove četiri točke dobra, tada mora proizvesti puno podudarnih točaka. Na tragu ovih ideja se koristi algoritam RANSAC za izračunavanje korespondencija. Algoritam će se zaustaviti kad se dosegne najveći broj iteracija (n_{it}) ili ako je postotak podudaranja iznad postotka ukupnog broja unutarnjih uglova α. Ako je dostignut maksimalni broj ponavljanja, dobiven rezultat se smatra valjanim ako je broj podudaranja veći od postotka β.

Kao rezultat prethodnih koraka, dobiva se početni skup unutarnjih uglova markera koji se koriste za dobivanje početnog položaja kamere.

4. KARAKTERISTIKE FRAKTALNIH MARKERA

Karakteristike su većinom navedene još u uvodnom poglavlju. Na ovom mjestu će biti predstavljeni konkretni odnosi između tradicionalnih kvadratnih markera i fraktalnog markera. Učinci će biti prikazani kroz procjenu dometa detekcije, otpornost na djelomično zaklanjanje, preciznost u procjeni položaja i brzinu predložene metode u odnosu na ArUco kvadratni marker iz kojega je fraktalni marker izveden.

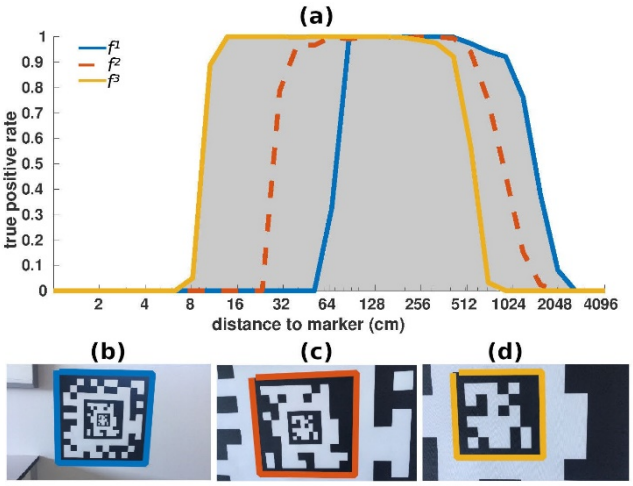
Tablica 1: Vrijednosti parametara [12]

Parameter	Value	Description
m	3	Number of internal markers of Fractal Marker F
$s(f^1)$	14	Length of black region of internal marker f^1 of Fractal Marker F
$n(f^1)$	12	Length of identification region of internal marker f^1 of Fractal Marker F
$k(f^1)$	6	Length of white region of internal marker f^1 of Fractal Marker F
$s(f^2)$	12	Length of black region of internal marker f^2 of Fractal Marker F
$n(f^2)$	10	Length of identification region of internal marker f^2 of Fractal Marker F
$k(f^2)$	4	Length of white region of internal marker f^2 of Fractal Marker F
$s(f^3)$	8	Length of black region of internal marker f^3 of Fractal Marker F
$n(f^3)$	6	Length of identification region of internal marker f^3 of Fractal Marker F
$k(f^3)$	0	Length of white region of internal marker f^3 of Fractal Marker F
s_{min}	10	Optimal spacing between bits used in the refinement process
l	10	Region size used to classify corners according to the three possible categories
n_{it}	500	Maximum number of iterations used by Ransac
α	0.7	Percentage of matches needed to consider a generated model as good
β	0.1	Percentage of minimum matches necessary to consider the model as a candidate
τ_c	25	Contrast threshold for corners

4.1. Analiza dometa detekcije

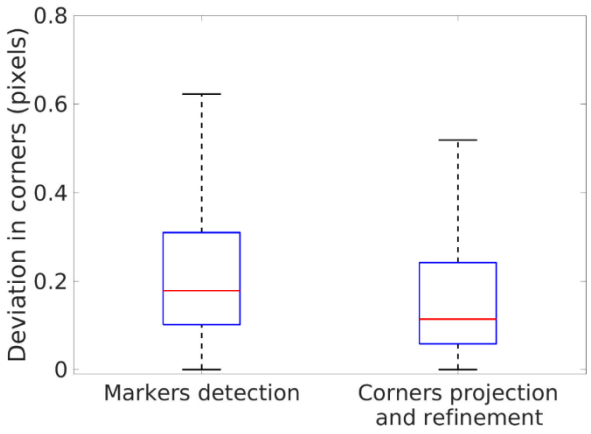
Za analizu je dan prikaz dometa fraktalnog markera koji se sastoji od tri unutarnja obilježja f^1, f^2, f^3 s duljinama bočnih stranica 41,3 cm, 17,5 cm i 5,9 cm za veće udaljenosti početka detekcije do gubitka autofokusa uslijed malih udaljenosti. Sl. 11 (b-d) prikazuje slike iz jednog od video sekvenci na različitim udaljenostima. Crte u boji koje okružuju markere (plava, crvena i žuta) su dodane iz prezentacijskih razloga.

Tradicionalni marker iz biblioteke ArUco je prilagođen za detekciju unutarnjih markera fraktalnog markera ignoriranjem bitova u središnjem području bočne duljine $k(f^i)$. Na taj način se mogu usporediti rezultati ArUco-a i fraktalnog markera u istim uvjetima jer se detekcija provodi na istoj sekvenci.



Slika 11: (a) Stope stvarno pozitivnih detekcija u ovisnosti o udaljenosti od markera. Svaka obojena crta odgovara jednom od unutarnjih markera koji čine fraktalni marker. Sivo područje odgovara opsegu detekcije kompletnog fraktalnog markera. (b-d) Različiti pogledi na fraktalni marker korišten za eksperimente. [12]

Slika 11a prikazuje stopu stvarno pozitivnih (TPR) obje metode u ovisnosti o udaljenosti od markera u logaritamskoj skali. Iako obojene linije pokazuju TPR za svaki pojedini marker koji koristi ArUco, sivo područje (pod svim krivuljama) odgovara fraktalnom markeru. Kao što se može primijetiti, predloženi fraktalni marker se može detektirati unutar velikog raspona udaljenosti, tj. [7, 2000] cm, dok svaki pojedini marker ima znatno smanjen domet detekcije.



Slika 12: Podrhtavanje vrhova prije i nakon predložene dorade. Predložena metoda poboljšava točnost. [12]

4.2. Analiza podrhtavanja vrhova

Podrhtavanje vrhova podrazumijeva standardno odstupanje u procjeni uglova kad se ni marker ni kamera ne pomiču za vrijeme detekcije. Standardno odstupanje od središnjeg položaja indikacija je preciznosti procjene položaja markera (jednadžba 3.5). Prikazana je analiza utjecaja podrhtavanja vrhova temeljem sedam video sekvenci statičke scene s fraktalnim markerom (s tri unutarnja markera širina 15 cm, 6,4 cm i 2,1 cm) za različite udaljenosti između 49 cm i 2,74 m.

Slika 12 prikazuje podrhtavanje vrhova kod izvorne metode detektiranja markera ArUco (tj. izlaz detekcije markera kao na slici 6) i nakon primjene cijelog predloženog tijeka rada (tj. nakon projekcije i izoštravanja uglova). Sukladno primjetnom smanjenju podrhtavanja vrhova fraktalnog markera se može očekivati stabilnija i preciznija procjena položaja kamere.

4.3. Vrijeme računanja

Fraktalni markeri zahtijevaju više vremena za računanje položaja nego sustav koji detektira samo tradicionalne markere, jer se provodi niz dodatnih koraka. Tablica 2 prikazuje prosječno vrijeme računanja Intel Core i7-7500U procesorom za različite korake prikazane na slici 6, koristeći ukupno 1037 slika razlučivosti 3840 × 2160. Za ove testove, ArUco [4] biblioteka je korištena za detektiranje markera u načinu DM_NORMAL.

Tablica 2: Prosječno vrijeme računanja (u milisekundama) različitih koraka uključenih u detekciju i praćenje fraktalnih markera. [12]

Process	Avrg (ms)
Markers detection (ArUco)	29.6
Keypoint-based marker detection	6.9
Camera pose estimation (x2)	0.2
Corners projection and refinement	1.0

S obzirom na odnose između vremena računanja, metoda fraktalnog markera dodaje malo opterećenja ukupnom vremenu računanja. Početni korak „detekcija markera“, koji je isti kao i kod tradicionalne detekcije markera, najdugotrajniji je postupak, s tim da broj unutarnjih markera fraktalnog markera nema značajan utjecaj na vrijeme računanja ovog koraka. Osim toga, postupak detekcije markera pomoću ključne točke se provodi samo ako nijedan unutarnji marker nije prethodno uspješno detektiran. Stoga se u većini slučajeva radi samo o zanemarivom dodatku na vrijeme ukupnog izračunavanja.

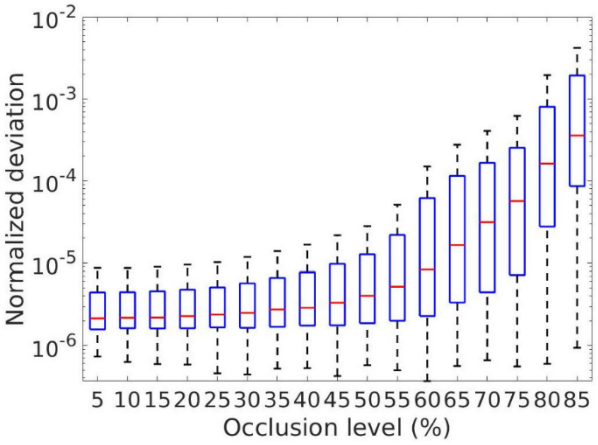
Slika 13: Neke su slike korištene za testiranje detekcije uz zaklanjanje. Sintetički se na slike primjenjuju različite razine zaklanjanja:(a) 11.29%, (b) 33.19%, (c) 53.92%, (d) 73.37%. [12]

4.4. Detekcija fraktalnog markera uz zaklanjanje

Analiza otpornosti i preciznosti detekcije fraktalnih markera započinje sa referntnih 60 slika pod kontroliranim unutarnjim osvjetljenjem. Analiziraju se tri različita fraktalna markera s različitih gledišta i udaljenosti (u rasponu od 10 cm do 1,5 m). Prvi fraktalni marker ima dva unutarnja markera širina 29,0 cm i 7,2 cm, drugi ima tri unutarnja markera širina 29,0 cm, 11,5 cm i 2,9 cm, a treći ima četiri unutarnja markera širina 29,0 cm, 14,5 cm, 3,6 cm i 0,9 cm.

Da bi se proizvelo sustavno zaklanjanje, [20] predlaže upotrebu predloška od bijelog papira na markeru koji se nalazi u donjem kutu markera tako da se površina markera postupno zaklanja papirom. Krugovi nasumičnih polumjera prekrivaju marker na slučajnim mjestima, kao što je prikazano na slici 13. Boja kruga je nasumično odabrana kao bijela ili crna. Budući da je riječ o sintetičkom zaklanjanju, poznat je postotak zaklanjanja. Analiza uslijed zaklanjanja se temelji na 1000 sintetičkih slika (ukupno 3000), tako da su razine zaklanjanja jednako raspoređene u rasponu [1, 85]%, dok detekcije iznad 85% postaju gotovo nemoguće.

Referentni elementi slike su lokacije četiriju vanjskih uglova f^1 dobivenih bez zaklanjanja. Potom se za svaku sliku sa zaklanjanjem pogreška mjeri kao prosječna udaljenost između referentnog i procijenjenog položaja. S obzirom da se udaljenost mjeri u pikselima, pogreška je obrnuto proporcionalna udaljenosti markera (ili području koje marker zauzima na slici). Kako bi se taj efekt izbjegao i omogućila usporedba rezultata slika snimljenih na različitim udaljenostima, pogreška se normalizira dijeljenjem s površinom markera na slici.

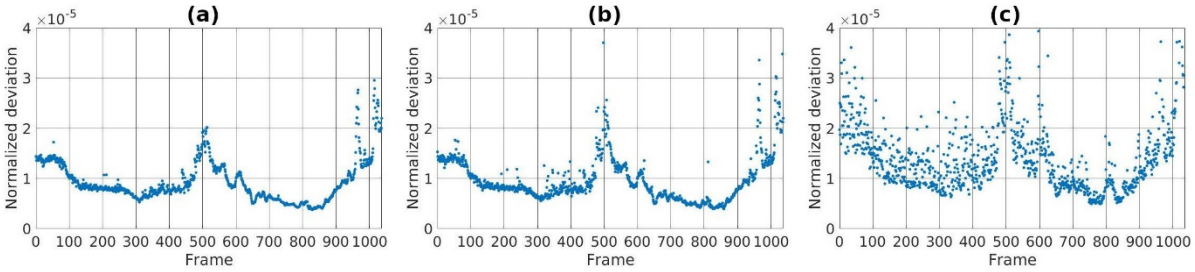


Slika 14: Prosjek (crvena) i standardna devijacija (plava) normalizirane pogreške za različite razine zaklanjanja. [12]

Na slici 14 pravokutnicima je prikazano prosječno i standardno odstupanje prema kojoj razina zaklanjanja ispod 50% ima zanemariv utjecaj na grešku, a tek veće vrijednosti zaklanjanja počinju utjecati na preciznost. Fraktalni markeri pokazuju otpornost nasuprot tradicionalnim markerima kao što su ArUco ili AprilTag koji nisu otporni na zaklanjanje.

4.5. Analiza detekcije markera prema ključnim točkama

Detekcija fraktalnog markera je moguća čak i kada nisu detektirani unutarnji markeri. Za detekciju se u tim situacijama koristi deskriptor ključnih točaka u kombinaciji s RANSAC algoritmom. Ovaj je odjeljak usmjeren na analizu preciznosti i robusnosti detekcije markera temeljene na ključnim točkama. U tu svrhu je korišten video od 1037 sličica u kojem je fraktalni marker sastavljen od tri unutarnja markera širina 15 cm, 6,4 cm i 2,1 cm snimljen na različitim udaljenostima (u rasponu od 28 cm do 1,44 m) i pod kontroliranim osvjetljenjem u zatvorenom prostoru.



Slika 15: Normalizirana pogreška piksela detekcije fraktalnog markera temeljene na ključnim točkama za jednu video sekvencu koja koristi različite razine sintetskog zaklanjanja: a) 0%, (b) 30%, (c) 60%. [12]

Ako se video sekvenca obrađuje korištenjem predloženog tijeka rada (slika 6), detektor markera pomoću ključnih točaka se nikada ne bi primijenio jer je u svakom okviru detektiran barem jedan marker. Da bi se mogla analizirati detekcija markera temeljena na ključnim točkama, sustav se prisiljava pretpostavkom da nisu detektirani markeri osim za prvu sliku.

Referentni dio svake sličice su četiri najudaljenija vanjska ugla najšireg markera fraktalnog markera, koji su izračunati predloženom metodom kroz redoviti tijek izvođenja. Zatim se rezultat uspoređuje s procjenom lokacije slijedeći detekciju markera temeljem ključnih točaka, a greška se normalizira podjelom promatranog područja markera u sličici. Na slici 15a. se primjećuju najveće vrijednosti oko sličice 800 jer je kamera tada bliža markeru, ali su razlike kod standardne metode zanemarive.

Za analizu utjecaja zaklanjanja na grešku, sličice su 30% i 60% umjetno zaklonjene nasumičnim krugovima. Na slici 15(b-c) su prikazane prosječne pogreške dobivene iz 20 sintetičkih slika za svaki okvir i postotak zaklanjanja. Detekcija fraktalnih markera je zamjetno otporna na zaklanjanje jer su pogreške pri 30% zaklanjanja slične onima kada nema zaklanjanja, dok je za zaklanjanje od 60% primjetan porast pogreške.

5. IZAZOVI U PRIMJENI I PREDVIĐENI DOPRINOS RADA

U literaturi se pronalaze primjeri primjene fiducijalnih markera u laboratorijskim uvjetima za dronove manjih dimenzija, gdje su udaljenosti kamere do markera nekoliko metara. Takva rješenja nisu adekvatna za primjene u realnim uvjetima leta u aeronautici, gdje će i markeri i kontrola letjelice biti podvrgnuti puno zahtjevnijim izazovima.

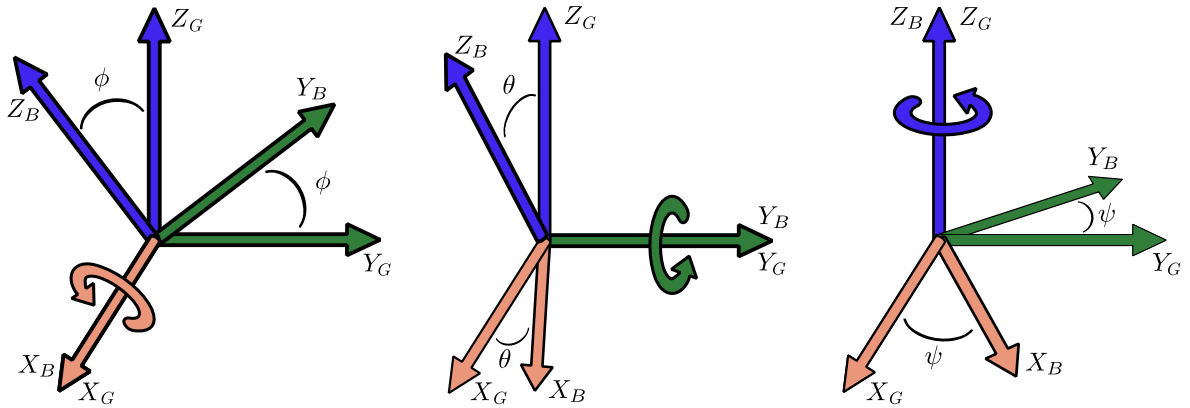
Slika 16: Kvadrotor AR.Drone 2.0 koji se kreće prema Aruco markeru u laboratorijskim uvjetima. [21]

Cilj mog istraživanja je razviti nove tipove markera i povratnih sprega između prepoznavanja markera i upravljanja zračnim brodom. Takav sustav još ne postoji, a posebna specifičnost je primjena na zračne brodove, koji su još uvijek rijetkost u aeronautici. Iako su zračni brodovi dominirali nebom prije stotinjak godina, brzi razvoj zrakoplova potpuno ih je potisnuo iz upotrebe nakon završetka 2. svjetskog rata. Međutim, zadnja dva desetljeća porastao je interes za njihov razvoj za potrebe transporta, te sigurnosti i nadzora. Primjena na koji se direktno veže moje istraživanje je Europski H2020 projekt „BorderUAS: Semi-autonomous border surveillance platform combining next generation unmanned aerial vehicles with ultra-high-resolution multi-sensor surveillance payload“ koji uključuje i niz graničnih policija na istočnim granicama EU [22].

Slika 17: Računalna ilustracija zračnog broda [hipersfera.hr]

Najveći izazov za autonomne letjelice je faza uzlijetanja i slijetanja. Zračni brodovi su u tim fazama dodatno zahtjevni jer imaju neutralnu plovnost (poput podmornica u moru). Letjelice kod uzlijetanja i slijetanja moraju imati nezavisne redundantne sustave za određivanje

orijentacije i položaja letjelice u prostoru. Svaki od tih sustava mora moći odrediti udaljenost od tla, te tri rotacijska kuta orijentacije: valjanje, poniranje i skretanje (*eng. yaw, pitch, roll*). U nominalnom radu letjelice greške se minimiziraju kroz usporedbu dobivenih vrijednosti iz svakog od sustava, dok u ekstremnim situacijama letjelica mora moći uzletjeti ili sletjeti koristeći makar samo jedan sustav.



(a) Roll Rotation **(b)** Pitch Rotation **(c)** Yaw Rotation

Slika 18: Rotacije oko tri osi [23]

Letjelice od interesa u mojem istraživanju su pritom znatno veće od dronova, i uz to moraju funkcionirati u režimu leta izvan idealnih laboratorijskih uvjeta. Plan je primijeniti sustav fraktalnih markera na zračnom brodu dimenzija od 20-ak metara u promjeru i s 5 (ili više) propulzijskih jedinica (motora s propelerima). Svaka propulzijska jedinica imala bi vlastitu kameru za praćenje markera na tlu. Optimalno, samo jedna takva kamera morala bi moći iz položaja i oblika markera u vidnom polju kamere odrediti visinu i orijentaciju letjelice u prostoru. U praksi se mogu očekivati problemi poput naglih trzaja letjelice zbog naleta vjetra, gubitka markera iz vidnog polja jedne kamere, ulaska u vidno polje druge kamere, otežane vidljivosti zbog loših uvjeta atmosfere, šum [24] itd. Uz to, govorimo o markerima koji moraju imati dovoljno velike dimenzije i dovoljno velik broj unutarnjih markera da ih letjelica može koristiti od samog tla (nekoliko metara udaljenost) do 100 ili više metara iznad tla. Takav problem do sada nitko nije adresirao, niti primijenio u realnim uvjetima leta.

U biti, čitavo područje gdje zračni brod polijeće ili slijeće se prekriva velikim markerom, te je otvoreno pitanje i kako bi takav marker morao izgledati i kako bi sustav s jednom ili više kamera trebao analizirati u realnom vremenu geometriju takvog markera da osigura pouzdan let u rizičnim fazama, kao što su uzlijetanje i slijetanje letjelice, gdje se očekuju snažne perturbacije i na položaje kamera, i na vidljivost markera.

6. ZAKLJUČAK

Fraktalni marker je vrsta fiducijalnog kvadratnog markera koja se može detektirati u većem rasponu udaljenosti od tradicionalnih markera. Sastoje se od rekruzivnog skupa pravokutnih markera i može se formirati s proizvoljnim brojem unutarnjih markera, prema čemu je moguće postizanje željenog raspona detektiranja.

Za razliku od tradicionalnih markera koji su vrlo osjetljivi na zaklanjanje, metoda detekcije fraktalnog markera može detektirati jako zaklonjene markere uz zanemarivo dodatno računanje. Čak i ako se markeri ne mogu detektirati u prvoj fazi postupka, marker se može detektirati temeljem ključnih točaka. Ključne točke razlikuju tri vrste uglova koje marker može imati, a algoritam temeljen na RANSAC-u detektira fraktalni marker na temelju tih ključnih točaka.

Prikazane analize ukazuju na pouzdanost i točnost metode, uz malo dodatnog vremena računanja u odnosu na tradicionalne procese detekcije markera. Metoda detekcije fraktalnog markera je integrirana u biblioteci ArUco i javno dostupna za korištenje i istraživanje, a sažeti pregled spomenutih algoritama se nalazi u dodatnom poglavlju na kraju ovog rada.

Fraktalni marker je pogodan izbor za procjenu položaja objekta na otvorenom prostoru gdje se može očekivati djelomični prikaz markera. Jedan od primjera su slijetanja bespilotnih letjelica pri kojima se marker mora detektirati na vrlo velikom rasponu udaljenosti. Takve letjelice, naročito one većih dimenzija s poznatim mjestom slijetanja su prikladni kandidati za primjenu, između ostalog zbog pouzdanosti sustava detekcije. Od posebnog interesa za istraživanje je mogućnost generiranja višestrukih fraktalnih markera za spomenute aplikacije uzlijetanja, a naročito slijetanja UAV letjelice.

Literatura

[1] Shah Alam, Md & Oluoch, Jared, „A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (UAVs)“, Expert Systems with Applications, 179, 115091, 2021.

[2] Šuligoj, Filip & Jerbic, Bojan & Svaco, Marko & Sekoranja, Bojan, „Fully Automated Point-Based Robotic Neurosurgical Patient Registration Procedure,“ International Journal of Simulation Modelling, 17. 458-471. 2018.

[3] Scher, N., Bollet, M., Bouilhol, G. et al., „Safety and efficacy of fiducial marker implantation for robotic stereotactic body radiation therapy with fiducial tracking“, Radiat Oncol 14, 167, Sep. 2019.

[4] Hartman, Leo & Jagersand, Martin & Jasiobedzki, Piotr & Rekleitis, Ioannis, „Workshop on Robot Vision for Space Applications“, IEEE/RSJ, IROS, 2005.

[5] Verhulst NO, Loonen JA, Takken W., „Advances in methods for colour marking of mosquitoes“, Parasit Vectors, 2013.

[6] Krajnik T., Nitsche M., Faigl, Vanek, Saska, Preučil, Duckett, Mejail, „A practical multirobot localization systemJournal of Intelligent and Robotic Systems“, Heidelberg, Springer 2014.

[7] Lange,S.,Sünderhauf,N.,&Protzel,P., „Autonomous Landing for a Multirotor UAV Using Vision“, In Workshop Proceedings of SIMPAR 2008 Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots, Venice, Italy, 3-4, pp.482-491., Nov. 2008.

[8] Acuna, R., & Willert, V. „Dynamic Markers: UAV Landing Proof of Concept“, 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), 2018.

[9] Wubben Jamie, Francisco Fabra, Carlos T. Calafate, Tomasz Krzeszowski, Johann M. Marquez-Barja, Juan-Carlos Cano, and Pietro Manzoni, „Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition" Electronics 8, no. 12: 1532., 2019.

[10] S. Garrido-Jurado, R. Muñoz Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, „Automatic generation and detection of highly reliable fiducial markers under occlusion“, Pattern Recognit., vol. 47, no. 6, pp. 2280–2292, 2014.

[11] Sagitov, A., Shabalina, K., Lavrenov, R., & Magid, E., „Comparing fiducial marker systems in the presence of occlusion“, 2017 International Conference on Mechanical, System and Control Engineering (ICMSC), 2017.

[12] Romero-Ramirez, Francisco & Muñoz-Salinas, Rafael & Medina-Carnicer, Rafael., „Fractal Markers: A New Approach for Long-Range Marker Pose Estimation Under Occlusion“, IEEE Access, PP. 1-1, 2019.

[13] V. Lepetit, F. Moreno-Noguer, and P. Fua, „EPnP: An accurate O(n) solution to the PnP problem“, Int. J. Comput. Vis., vol. 81, no. 2, pp. 155–166, 2009.

[14] G. Bradski and A. Kaehler, „Learning OpenCV 3: Computer Vision in C++ With the OpenCV Library“, 2nd ed. Newton, MA, USA: O'Reilly Media, 2013.

[15] E. Rosten, R. Porter, and T. Drummond, „Faster and better: A machine learning approach to corner detection“, 2008, arXiv:0810.2434. [Online]. Available: https://arxiv.org/abs/0810.2434

[16] S. Suzuki, K. Abe, „Topological structural analysis of digitized binary images by border following“, Comput. Vis., Graph., Image Process., vol. 30, no. 1, pp. 32–46, 1985.

[17] D. H. Douglas and T. K. Peucker, „Algorithms for the reduction of the number of points required to represent a digitized line or its caricature“, Cartographica, Int. J. Geograph. Inf. Geovis., vol. 2, no. 10, pp. 112–122, 1973.

[18] N. Otsu, „A threshold selection method from gray-level histograms“, IEEE Trans. Syst., Man, Cybern., vol. 9, no. 1, pp. 62–66, Jan. 1979.

[19] T. Collins and A. Bartoli, „Infinitesimal plane-based pose estimation“, Int. J. Comput. Vis., vol. 109, no. 3, pp. 252–286, Sep. 2014.

[20] K. Shabalina, A. Sagitov, M. Svinin, and E. Magid, „Comparing fiducial markers performance for a task of a humanoid robot self-calibration of manipulators: A pilot experimental study“, in Proc. 3rd Int. Conf. ICR, Leipzig, Germany, pp. 249–258., Sep. 2018.

[21] M. F. Sani and G. Karimian, „Automatic navigation and landing of an indoor AR. Drone quadrotor using ArUco marker and inertial sensors“, in Proc. Int. Conf. Comput. Drone Appl. (IConDA), pp. 102–107, Nov. 2017

[22] BorderUAS consortium, *https://borderuas.eu/consortium/* , Horizon Europe 2020 Project webpage

[23] Tiago Gomes Carreira, „Quadcopter Automatic Landing on a Docking Station“, Thesis, Tecnico Lisboa, 2013.

[24] Zakiev, A., Tsoy, T., Shabalina, K., Magid, E., & Saha, S. K. (2020). „Virtual Experiments on ArUco and AprilTag Systems Comparison for Fiducial Marker Rotation Resistance under Noisy Sensory Dana“, 2020 International Joint Conference on Neural Networks (IJCNN), 2020.

[25] Chris Solomon, Toby Breckon, „Fundamentals of Digital Image Processing“, John Wiley & Sons, Ltd, 2011

[26] Emanuela Rešetar, „Machine vision system for automated quality control or rotary switch inserts“, Master's thesis, Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, 2020

[27] Goran Jerneić, „Promjena površine objekata pri pojednostavljenju linija“, diplomski rad, Geodetski fakultet, Sveučilište u Zagrebu, 2017

[28] Alasdair McAndrew, „A computational introduction to digital image processing“, 2nd edition, CRC Press, 2016

[29] Ugurcan Budak, „Accuracy Evaluation of Methods for Pose Estimation from Fiducial Markers“, 2021 Master of Science Thesis, Faculty of Engineering and Natural Sciences, Tampere University

[30] OpenCV, „FAST Algorithm for Corner Detection“, Documentation webpage https://docs.opencv.org/4.x/df/d0c/tutorial_py_fast.html, 2022

[31] M. A. Fischler and R. C. Bolles, „Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography“, in Comm. ACM 24, 381-395, 1981.

[32] R. Hartley and A. Zisserman, „Multiple View Geometry in Computer Vision“, University Press, Cambridge, 2003

Dodatak A – Algoritmi

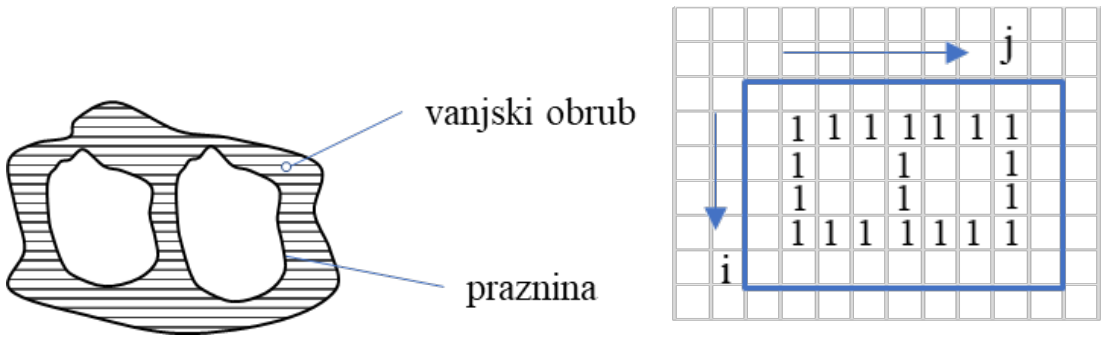
6.1. Lokalni adaptivni prag

Prilagodljivo postavljanje praga osmišljeno je za prevladavanje ograničenja konvencionalnog, globalnog postavljanja praga, računanjem praga na svakom mjestu piksela na slici. Ovaj lokalni prag općenito je određen vrijednostima piksela u susjedstvu piksela. Dakle, adaptivno određivanje praga funkcionira na temelju pretpostavke da se osvjetljenje može razlikovati na slici, ali se može pretpostaviti da je otprilike ujednačeno u dovoljno malom, lokalnom susjedstvu.

Vrijednost lokalnog praga t se može odrediti na više načina. Ipak, prag se najčešće bira takav da bude ili $t = ar.sred. + C$, $t = medijan + C$ ili $\left\lceil \frac{(max-min)}{2} \right\rceil$ lokalnog susjedstva $N \times N$ piksela koji okružuje svaki piksel. Odabir vrijednosti N je važan i mora formirati prozor kojim obuhvaća dovoljno piksela pozadine i prednjeg plana u svakoj točki, ali ne prevelik da globalni nedostaci osvjetljenja utječu na prag. Kada nema dovoljno piksela u prednjem planu, tada odabrani prag postaje loš (tj. srednja/medijalna ili prosječna maksimalna/minimalna razlika u susjedstvu uvelike varira s šumom slike). Iz tog razloga se konstantni pomak C dodaje ukupnom pragu kako bi se postavio prag iznad opće varijance šuma slike u područjima jednolikih piksela [25].

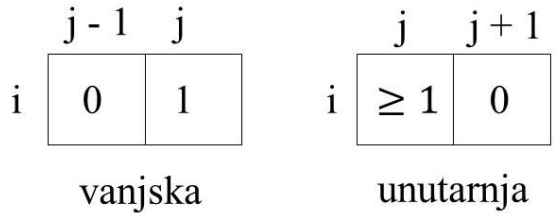
6.2. Suzuki-Abeov algoritam

Ovaj algoritam definira hijerarhijsku vezu između granica. Slika 19 prikazuje binarnu sliku koju algoritam koristi kao ulazni parametar pri određivanju konture.



Slika 19: Binarni zapis granice objekta [26]

Funkcija $f(i,j)$ se sastoji od vrijednosti piksela na lokacijama (i,j). Pritom se svakoj pronađenoj granici na putu dodjeljuje jedinstveni NBD broj (*newest border*) i pohranjuje LNBD (*last newest border*) kako bi se dobio odnos okruženja između granica. Na mjestu gdje se nalazi granica pretpostavljena vrijednost odgovara vrijednosti 1. Ostatak granice je numeriran sekvencijalno. Skeniranje slike izvršava se s lijeva na desno pri čemu se detektiranim pikselima dodjeljuju vrijednosti prema slici. Pritom se odlučuje je li pronađeni piksel pripada vanjskom rubu ili granici praznine. U nastavku su koraci daljnje analize:



Slika 20: Određivanje vanjske i unutarnje granice [26]

1. *Korak :*

   Za pronađene vrijednosti $f_{i,j} = 1$ i $f_{i,j-1} = 0$ se na lokaciji (i_2, j_2) dodjeljuju vrijednosti $f_{i,j-1}$

   U suprotnom je pronađena unutarnja kontura pa se vrijednosti (i_2, j_2) postavljaju na vrijednosti $f_{i,j+1}$, a $f_{i,j}$ postaje roditelj ako vrijedi $f_{i,j} > 1$, inače se prelazi na korak 3

2. *Korak :*

   2.1. *Započinje od (i_2, j_2) i obilaze se sva susjedstva od (i, j) u smjeru kazaljke na satu. Ako naiđe vrijednost $\neq 0$ i označava ih se kao (i_1, j_1). Inače se postavlja $f_{i,j} = -NBD$ i ide na korak 3.*

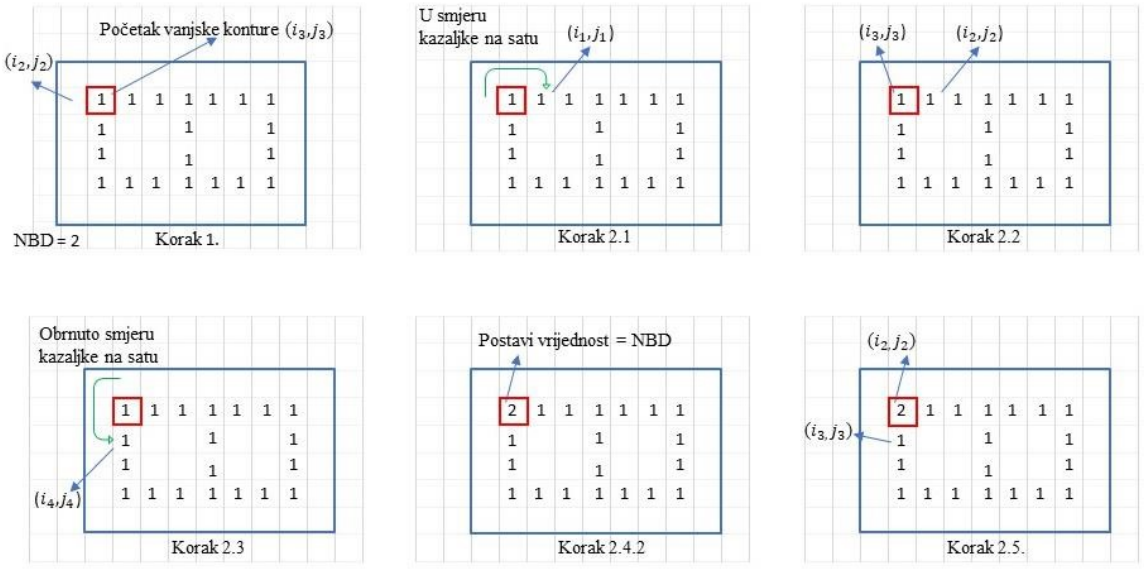
   2.2. *Postavlja $(i_2, j_2) = (i_1, j_1)$ i $(i_3, j_3) = (i, j)$*

   2.3. *Promatra sljedeći element piksela (i_2, j_2) u smjeru obrnutom od smjera kazaljke na satu. Ponovno prelazi u susjedni element (i_3,j_3) i obilazi susjedstva u smjeru suprotnom od smjera kazaljke na satu kako bi našao prvu vrijednost različitu od 0 i postavlja je na (i_4, j_4)*

   2.4. *Mijenja vrijednost trenutnog piksela (i_3, j_3) prema uvjetima:*

      2.4.1. *Ako je vrijednost piksela (i_3, j_{3+1}) jednaka 0, ovaj piksel pripada vanjskom području te njegovu vrijednost postavlja na $-NBD$*

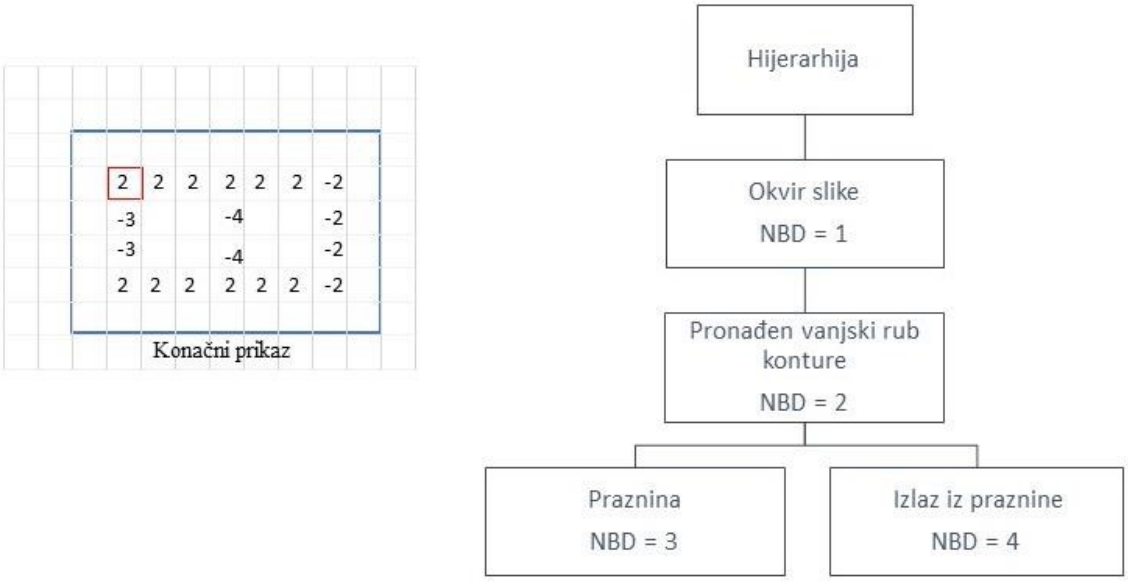
      2.4.2. *Ako je vrijednost piksela $(i_3, j_{3+1}) \neq 0$, postavlja trenutni piksel na vrijednost NBD. Inače ne mijenja vrijednost piksela*



Slika 21: Koraci detekcije konture [26]

2.4.3. *Ako se u koraku 2.3 ponovno vrati u početnu točku odnosno ako vrijedi* $(i_4, j_4) = (i, j)$ *i* $(i_3, j_3) = (i_1, j_1)$ *prelazi na korak 3. Inače postavlja* $(i_2, j_2) = (i_3, j_3)$ *i* $(i_3, j_3) = (i_4, j_4)$ *i ide natrag na korak 2.3*

3. *Korak :*
Ako $f_{i,j}! = 1$ *postavlja* $LNBD = |f_{i,j}|$ *te započinje skenirati od sljedećeg piksela* $(i, j + 1)$. *Kada dođe u donji desni kut označava kraj [16][26].*

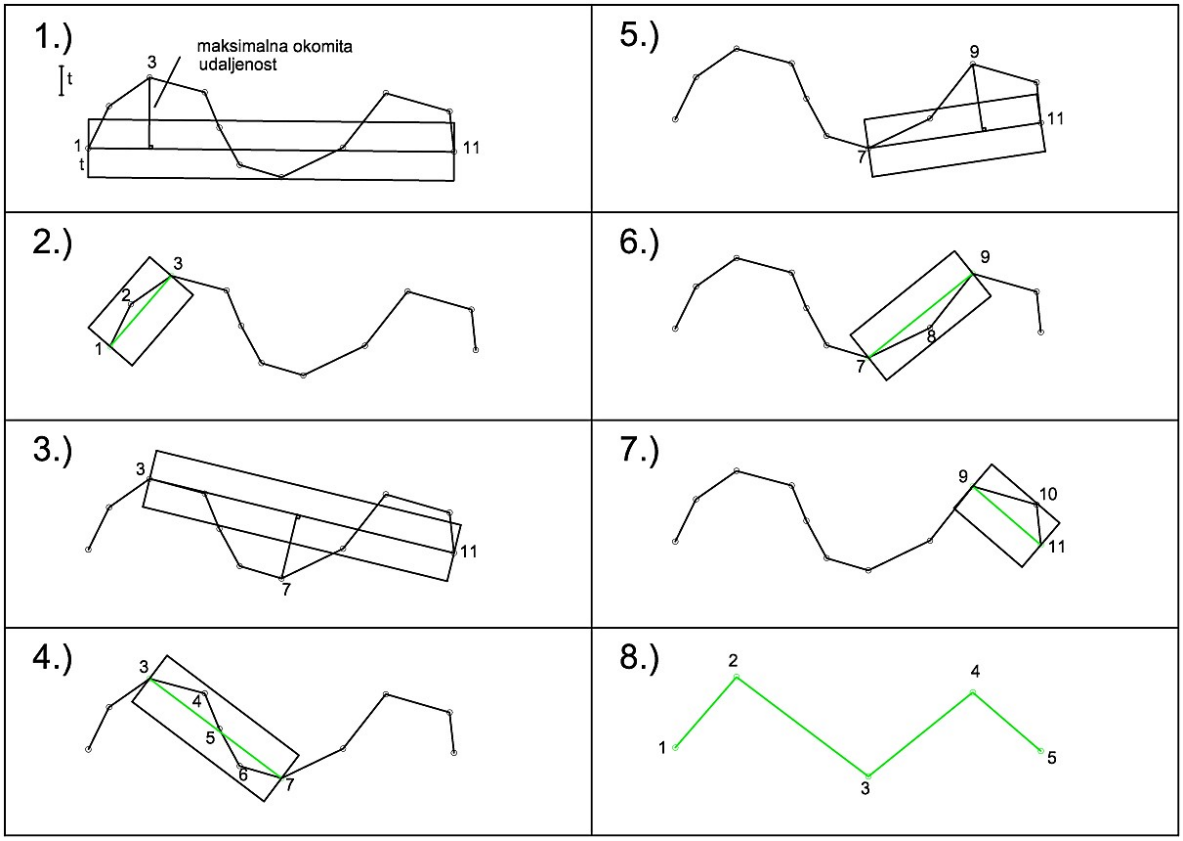


Slika 22: Konačni rezultat detekcije konture [26]

6.3. Douglas-Peuckerov algoritam

Na slici 23 je prikazan princip pojednostavljenja linije Douglas-Peuckerovim algoritmom. Prvi korak je definiranje granice tolerancije ili koridora t koju definira korisnik. Koridor je definiran kao udaljenost sa svake strane ravne linije koja povezuje točku sidrišta i plutajuću točku (točke 1 i 11). Nakon toga se računaju okomite udaljenosti između linije koja povezuje točke 1-11 i svih točaka između njih (točke 2-10) kako bi se odredila ona koja je najudaljenija. U ovom slučaju to je točka 3 koja se nalazi izvan koridora pa se u tom slučaju sprema na početnu poziciju u stog. Kako je algoritam došao do početka linije posljednja točka ubačena u stog (točka 3) se vraća i postaje nova plutajuća točka. Nadalje, računa se novi koridor, između točaka 1 i 3. Točka 2 se nalazi unutar koridora te se ravna linija koja povezuje točke 1 i 3 smatra odgovarajućom za predstavljanje linijskog segmenta između točaka 1 i 3, točka 2 se stoga odbacuje, briše. Nova točka sidrišta postaje točka 3, a plutajuća točka je posljednja točka linije (točka 11). Računa se koridor između tih točaka i pronalazi najudaljenija točka koja se nalazi izvan koridora (točka 7). Ta točka se sprema na posljednje mjesto stoga. Kako je algoritam ponovno došao do početka linijskog segmenta vraća se točka 7 iz stoga te se računa koridor između točaka 3 i 7. Sve točke se nalaze unutar koridora te se ravna linija smatra odgovarajućom za prikaz tog dijela linije u pojednostavljenom obliku. Točka sidrišta sada je

točka 7. Ovaj postupak se ponavlja sve dok cijela linija nije pojednostavljena, odnosno dok se sve točke ne nalaze unutar izračunatih koridora [17] [27].



Slika 23: Douglas-Peuckerov algoritam [27]

6.4. Otusov algoritam

Otsuov algoritam [18][28] pronalazi najbolje mjesto za razdvajanje slike u komponente "prednjeg plana" i "pozadinske" komponente tako da se varijanca među klasama maksimizira. Pretpostavi se da je slika podijeljena na piksele prednjeg plana i pozadinske piksele na pragu t, gdje je udio u slici svakog od njih w_f odnosno w_b. Također, pretpostavi se da su srednje vrijednosti μ_f i μ_b . Tada je varijanca između klasa definirana kao

$$w_f w_b \left(\mu_f - \mu_b\right)^2 \qquad \text{(a.1)}$$

Ako slika ima n sivih piksela vrijednosti i, tada se definira $p_i = \frac{n_i}{N}$ gdje je N ukupan broj piksela. Dakle, p_i je vjerojatnost da će piksel imati sivu vrijednost i. Tada je zadana vrijednost praga t

$$w_b = \sum_{k=0}^{t-1} p_k \quad , \qquad w_f = \sum_{k=t}^{L-1} p_k \qquad \text{(a.2)}$$

gdje je L broj sivih nijansi na slici. Budući da po definiciji mora biti

$$\sum_{k=0}^{L-1} p_k = 1 \qquad \text{(a.3)}$$

slijedi da je $w_k + w_f = 1$. Pozadinske težine mogu se izračunati kumulativnim zbrojem svih p_k vrijednosti. Srednje vrijednosti se mogu definirati kao

$$\mu_b = \frac{1}{w_b}\sum_{k=0}^{t-1} kp_k \ , \qquad \mu_f = \frac{1}{w_f}\sum_{k=t}^{L-1} kp_k \ . \qquad \text{(a.4)}$$

6.5. IPPE algoritam

Ovaj algoritam pronalazi procjenu položaja korištenjem redundantnosti matrice homografije. U prisutnosti šuma, nakon što se homografija procijeni kroz točke u prostoru, na ravnini modela, točnost položaja bolja je u nekim regijama nego u drugima. Metoda koja se naziva *Infinitesimal Plane-based Pose Estimation* (IPPE) pronalazi točku u kojoj se homografija najbolje procjenjuje na ravnini modela. Zatim se na temelju ove optimalne točke izračunava položaj kamere. Rješenje se izvodi rješavanjem problema položaja kroz parcijalnu diferencijalnu jednadžbu nultog i/ili prvog reda koja je lokalno neredundantna.

Neka je **K** matrica intrinzičnih parametara kamere:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(a.5)}$$

u kojoj $\mathbf{c} = \left[c_x, c_y\right]^\intercal$ predstavlja centar slike, s iskošenje senzora, a $\left[f_x, f_y\right]$ fokalnu duljinu kamere. Ploha modela je postavljena na $z = 0$, a distorzija zanemarena. Za pretvorbu homogenih 3D koordinata u nehomogene 2D se koristi funkcija $\pi([x, y, z])^\intercal = z^{-1}[x, y]^\intercal$. S pretpostavkom da je **q** promatrana točka slike generirana temeljem perspektivne projekcije, normalizirane koordinate se mogu dobiti izrazom:

$$\tilde{\mathbf{q}} = \mathbf{K}_{22}^{-1}(\mathbf{q} - \mathbf{c}) \qquad \text{(a.6)}$$

Neka je $\mathbf{u}_i \in \mathbb{R}^2, i \in 1, 2, \dots, n$ točka na ravnini modela, odnosno na sceni u svijetu, i $\mathbf{q}_i \in \mathbb{R}^2, i \in 1, 2, \dots, n$ točka na ravnini slike. Transformacija točke u na ravninu slike je definirana kao:

$$\begin{bmatrix} \tilde{\mathbf{q}} \\ 1 \end{bmatrix} \propto [\mathbf{I}_3 \quad \mathbf{0}] \propto \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\intercal & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 0 \\ 1 \end{bmatrix} \propto \mathbf{H} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \qquad \text{(a.7)}$$

Simbol ~ iznad naziva označava normaliziranu komponentu, a simbol ^ komponentu koja sadrži šum. Pravo preslikavanje između koordinata svijeta i slike se definira kao $s(\mathbf{u}) =$

$\mathbf{R}[\mathbf{u}^\intercal, 0]^\intercal + \mathbf{t}$, tako da $\mathbb{R}^2 \to \mathbb{R}^3$. Za dobivanje transformacije w između svijeta i slike, s se kombinira s gore spomenutom funkcijom π. Funkcija transformacije je definirana kao

$$w(u) = \pi\big(\mathbf{H}([\mathbf{u}^\intercal, 1]^\intercal)\big) = (\pi \circ s)(\mathbf{u}) \tag{a.8}$$

Znajući da je H homografska matrica bez šuma, pretpostavlja se da je jedna točka $\mathbf{u}_0$ u svijetu, element podprostora $\Omega_{\mathbf{H}}$, koji je podprostor od $\mathbb{R}^2$ koji se ne preslikava putem homografske matrice na pravac u beskonačnost. Dakle, umetanje $\mathbf{u}_0$ u jednadžbu a.8 umjesto $\mathbf{u}$ daje dva ograničenja za parcijalnu diferencijalnu jednadžbu nultog reda. Diferencijacija jednadžbe a.8 dovodi do dobivanja četiri ograničenja parcijalnih diferencijalnih jednadžbi prvog reda na s

$$J_w(\mathbf{u}_0) \;=\; (J_\pi \circ s)(\mathbf{u}_0) J_s(\mathbf{u}_0). \tag{a.9}$$

Jacobianova funkcija J_f izračunava Jacobianovu matricu od f. Jacobianova funkcija koja izračunava s je $J_s(u_0) \;=\; \mathbf{R}_{32}$, gdje je s kruta transformacija, te vrijedi

$$J_s(\mathbf{u}_0)^\intercal J_s(\mathbf{u}_0) \;=\; \mathbf{I}_2, \tag{a.10}$$

gdje je $\mathbf{I}_2$ matrica identiteta 2×2, a $\mathbf{R}_{32}$ je podmatrica 3×2 od $\mathbf{R}$. Za procjenu položaja kamere, $s(\mathbf{u}_0)$ i $J_s(\mathbf{u}_0)$ treba pronaći izračunavanjem kroz parcijalnu diferencijalnu jednadžbu prvog reda (s jednadžbama a.8 i a.9). Ukupno je 6 uvjeta dobiveno iz ove dvije jednadžbe, što je ujedno i minimalni broj za određivanje položaja.

Uvođenjem x kao nepoznatog vektora koji sadrži 3D koordinate $\mathbf{u}_0$ u koordinatnom sustavu kamere, to jest $s(\mathbf{u}_0) \in \mathbb{R}^3$; i nepoznata matrica $\mathbf{R}_{32} = J_s(\mathbf{u}_0)$, jednadžbe a.8, a.9 i a.10 se rješavaju zamjenom x i $\mathbf{R}_{32}$. Budući da je cilj pronaći $\mathbf{t}$, potrebni su x i $\mathbf{R}$, a relacija je definirana kao x: $x = s(\mathbf{u}_0) = \mathbf{R}[\mathbf{u}_0, 0]^\intercal + \mathbf{t}$. Matrica rotacije $\mathbf{R}$ i vektor translacije $\mathbf{t}$ dobivaju se na sljedeći način

$$\mathbf{R} = \left[\mathbf{R}_{32} \middle| \mathbf{R}_{32}\begin{bmatrix}1\\0\end{bmatrix} \times \mathbf{R}_{32}\begin{bmatrix}0\\1\end{bmatrix}\right], \qquad \mathbf{t} = x - \mathbf{R}\begin{bmatrix}\mathbf{u}_0\\0\end{bmatrix}. \tag{a.11}$$

Kako homografska matrica sadrži šum, tako $\mathbf{v} \in \mathbb{R}^2$ i $\mathbf{J} \in \mathbb{R}^{2\times 2}$ jesu procjene sa šumom koje predstavljaju $w(\mathbf{u}_0)$ i $J_w(\mathbf{u}_0)$. Pod pretpostavkom da je $\widehat{H}_{33} = 1$, $\mathbf{v}$ i $\mathbf{J}$ se izračunavaju na sljedeći način

$$\begin{aligned} \mathbf{v} \;&=\; \pi\big(\widehat{\mathbf{H}}[\mathbf{u}_0^\intercal\ 1]^\intercal\big) \approx w(\mathbf{u}_0), \\ \mathbf{J} &= \big(1 + u_x\widehat{H}_{31} + u_y\widehat{H}_{32}\big)^{-2}\begin{bmatrix}J_{11} & J_{12}\\ J_{21} & J_{22}\end{bmatrix} \approx J_w(\mathbf{u}_0) \end{aligned} \tag{a.12}$$

Uvrštavanjem izraza $[x_1, x_2]^\intercal = x_3\mathbf{v}$ u $J_\pi(\mathbf{x})$ slijedi

$$J_\pi(x) \approx J_\pi(x_3[\mathbf{v}^\intercal\ 1]^\intercal) = \; x_3^{-1}\,[\mathbf{I}_2|-\mathbf{v}] \tag{a.13}$$

Definiranje $\gamma = x_3^{-1}$ dovodi do skraćivanja jednadžbi s ciljem pronalaska γ i $\mathbf{R}$, gdje je

$$\gamma[\mathbf{I}_2 - \mathbf{v}]\mathbf{R}_{32} = J \ , \ \mathbf{R}_{32}^{\mathsf{T}}\mathbf{R}_{32} = \mathbf{I}_2 \ , \ \gamma > 0 \qquad \text{(a.14)}$$

a položaj ravnine može se izračunati iz jednadžbe a.11 primjenom $\mathbf{x} = \gamma^{-1}\,[\mathbf{v}^{\mathsf{T}}, 1]^{\mathsf{T}}$.

Jednadžba a.14 se može napisati kao

$$\gamma[\mathbf{I}_2 - \mathbf{v}]\mathbf{R}_{32} \ = \ \gamma[\mathbf{I}_2| - \mathbf{v}]\mathbf{R}_v\mathbf{R}_v^{\mathsf{T}}\mathbf{R}_{32} \qquad \text{(a.15)}$$

U jednadžbi a.15, $\mathbf{R}_v$ je definiran kao rotacija oko $[\mathbf{I}_2 - \mathbf{v}]$ na takav način da postoje $\mathbf{B} \in \mathbb{R}_{2\times 2}$ i $[\mathbf{I}_2|{-}\mathbf{v}]\mathbf{R}_v = [\mathbf{B}|0]$. Oznaka za normaliziranu matricu rotacije $\widetilde{\mathbf{R}} \ = \ \mathbf{R}_v^{\mathsf{T}}\mathbf{R}$. Nenormalizirani oblik matrice rotacije postaje $\mathbf{R} = \mathbf{R}_v\widetilde{\mathbf{R}}$ [4]. Prvi redak jednadžbe a.14 poprima oblik

$$\gamma[\mathbf{I}_2|{-}\mathbf{v}]\mathbf{R}_{32} = \mathbf{J} \Leftrightarrow \gamma[\mathbf{B}|0]\widetilde{\mathbf{R}}_{32} = \mathbf{J} \Leftrightarrow \gamma\widetilde{\mathbf{R}}_{22} = \mathbf{A}\,, \ \mathbf{A} = \mathbf{B}^{-1}\mathbf{J} \qquad \text{(a.16)}$$

Pretpostavi se da je σ_1^A najveća singularna vrijednost matrice **A**, te izrazi zadnji stupac $\mathbf{R}_v$ kao vektor $\mathbf{r}_{v3}$; $[\mathbf{I}_2|{-}[\mathbf{v}^{\mathsf{T}}, 1]^{\mathsf{T}}]\mathbf{R}_v = [\mathbf{B}|0]$ i $[\mathbf{I}_2|{-}\mathbf{v}]\mathbf{r}_{v3} = 0$. Rješenje γ izražava se kao

$$\gamma = \sigma_1^A = \sqrt{0.5(a_u + a_w + \sqrt{(a_u - a_w)^2 \ + \ 4a_v^2})}\,, \qquad \begin{bmatrix} a_u & a_v \\ a_v & a_w \end{bmatrix} = \mathbf{A}\mathbf{A}^{\mathsf{T}} \qquad \text{(a.17)}$$

$\mathbf{R}_v$ se definira kao najmanja rotacija koja osigurava poravnanje između z-osi i $[\mathbf{v}^{\mathsf{T}}, 1]^{\mathsf{T}}$. S obzirom na ortogonalno svojstvo **R** s ovom definicijom, poprima oblik

$$\mathbf{R}_v = \mathbf{I}_3 + sin\theta[\mathbf{k}]_x + (1 - cos\theta)[\mathbf{k}]_x^2\,, \qquad s \ = \|[\mathbf{v}^{\mathsf{T}}\ 1]^{\mathsf{T}}\|_2,$$
$$cos\theta = \frac{1}{s}, \qquad sin\theta = \sqrt{1 \ - \frac{1}{s^2}}\,, \qquad \text{(a.18)}$$
$$t = \| \ v \ \|_2\,, \qquad [\mathbf{k}]_x = 1/t \begin{bmatrix} \mathbf{0} & \mathbf{v} \\ -\mathbf{v}^{\mathsf{T}} & 0 \end{bmatrix}.$$

$\mathbf{R}_1$ i $\mathbf{R}_2$ su dva rješenja rotacijske matrice, kako slijedi

$$\mathbf{R}_1 = \mathbf{R}_v\widetilde{\mathbf{R}}_1\,, \qquad \mathbf{R}_2 = \mathbf{R}_v\widetilde{\mathbf{R}}_2\,, \qquad \widetilde{\mathbf{R}}_1 = \begin{bmatrix} \widetilde{\mathbf{R}}_{22} & \mathbf{c} \\ \mathbf{b}^{\mathsf{T}} & a \end{bmatrix},$$
$$\widetilde{\mathbf{R}}_2 = \begin{bmatrix} \widetilde{\mathbf{R}}_{22} & -\mathbf{c} \\ -\mathbf{b}^{\mathsf{T}} & a \end{bmatrix}, \qquad \widetilde{\mathbf{R}}_{22} = \gamma^{-1}\mathbf{A}\,,$$
$$\mathbf{b} = rank_1\big(\mathbf{I}_2 - \widetilde{\mathbf{R}}_{22}^{\mathsf{T}}\widetilde{\mathbf{R}}_{22}\big) = \big[\sqrt{r_u}sign(r_v)\sqrt{r_w}\big]^{\mathsf{T}}, \qquad \text{(a.19)}$$
$$\begin{bmatrix} r_u & r_v \\ r_v & r_w \end{bmatrix} = \mathbf{I}_2 - \widetilde{\mathbf{R}}_{22}^{\mathsf{T}}\widetilde{\mathbf{R}}_{22}\,,$$
$$\begin{bmatrix} \mathbf{c} \\ a \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{R}}_{22} \\ \mathbf{b}^{\mathsf{T}} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} \widetilde{\mathbf{R}}_{22} \\ \mathbf{b}^{\mathsf{T}} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Algoritam je podijeljen u dva dijela, a prvi dio rješava jednadžbu a.14:

1: Inicijalizira **v** i **J** tako da $\mathbf{v} \in \mathbb{R}^2, \mathbf{J} \in \mathbb{R}^{2\times 2}$ i $\mathbf{J} \neq \mathbf{0}$

2: **Funkcija** IPPE($\mathbf{v}, \mathbf{J}$)

3: Računa $\mathbf{R}_v$ preko $\mathbf{v}$

4: $[\mathbf{B}|0] \leftarrow [\mathbf{I}_2|-\mathbf{v}]\mathbf{R}_v$

5: $\mathbf{A} \leftarrow \mathbf{B}^{-1}\mathbf{J}$

6: Traži najveću svojstvenu vrijednost od $\mathbf{A}$ tako da $\gamma \leftarrow \sigma_1^A$

7: $\widetilde{\mathbf{R}}_{22} \leftarrow \gamma^{-1}\mathbf{A}$

8: $\mathbf{b} \leftarrow rank_1\left(\mathbf{I}_2 - \widetilde{\mathbf{R}}_{22}^{\intercal}\widetilde{\mathbf{R}}_{22}\right)$

9: $\begin{bmatrix}\mathbf{c}\\ a\end{bmatrix} \leftarrow \begin{bmatrix}\widetilde{\mathbf{R}}_{22}\\ \mathbf{b}^{\intercal}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} \times \begin{bmatrix}\widetilde{\mathbf{R}}_{22}\\ \mathbf{b}^{\intercal}\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}$

10: $\mathbf{R}_1 = \mathbf{R}_v\begin{bmatrix}\widetilde{\mathbf{R}}_{22} & \mathbf{c}\\ \mathbf{b}^{\intercal} & a\end{bmatrix}, \mathbf{R}_2 = \mathbf{R}_v\begin{bmatrix}\widetilde{\mathbf{R}}_{22} & -\mathbf{c}\\ -\mathbf{b}^{\intercal} & a\end{bmatrix}$

11: Vraća $\gamma, \mathbf{R}_1$ i $\mathbf{R}_2$

Drugi dio IPPE algoritma je namijenjen perspektivnoj projekciji. Dohvaća korespondencije točaka i unutarnje parametre kamere i formira **v**, **J**, zatim procjenjuje dva položaja. Algoritam odabire položaj s najmanjom pogreškom reprojekcije. Troškovna funkcija korištena u IPPE algoritmu je izražena ovim izrazom:

$$c(\mathbf{t};\ [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3], \{\mathbf{u}_i\}, \{\mathbf{q}_i\}) = \sum_{i=1}^{n} \left\| \begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}\mathbf{u}_i + \begin{bmatrix}t_1\\ t_2\end{bmatrix} - (\mathbf{r}_3[\mathbf{u}_i^{\intercal}\ 0]^{\intercal} + t_3)\widetilde{\mathbf{q}}_i \right\|_2^2 \quad \text{(a.20)}$$

koji rješava linearni sustav najmanjih kvadrata u formi $\| \mathbf{W}_j\mathbf{t}_j - \mathbf{b}_j \|_2^2$ gdje je $\mathbf{W}_j$ matrica dimenzija $2n \times 3$, a $\mathbf{b}_j$ vektor $2n \times 1$. Rješenje jednadžbe a.20 dobiva se označavanjem $\mathbf{t}_j = \left(\mathbf{W}_j^{\intercal}\mathbf{W}_j\right)^{-1}\mathbf{W}_j\mathbf{b}_j$ gdje je matrica **W** ranka 3. Slijedi drugi dio algoritma:

0: Inicijalizira n točaka $\sum_i \mathbf{u}_i = 0$ na sceni u svijetu $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}$ takvih da je $\mathbf{u}_i \in \mathbb{R}^2$
Inicijalizira n točaka $\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$ takvih da je $\mathbf{q}_i \in \mathbb{R}^2$
Inicijalizira intrinzičnu matricu **K**

1: **Funkcija** IPPE($\{\mathbf{u}_i\}, \{\mathbf{q}_i\}, \mathbf{K}$)

2: $[\widetilde{\mathbf{q}}_i^{\intercal}\ \ 1]^{\intercal} \leftarrow\ \mathbf{K}^{-1}[\mathbf{q}_i^{\intercal}\ \ 1]^{\intercal}$

3: Traži optimalnu homografiju između $\{\mathbf{u}_i\}$ i $\{\widetilde{\mathbf{q}}_i^{\intercal}\}$, gdje je $H_{33} = 1$ tako da
$H \leftarrow homog(\{\mathbf{u}_i\}, \{\widetilde{\mathbf{q}}_i\})$

4: Računa Jakobijan od $\pi(\mathbf{H}[\mathbf{u}_0^{\intercal}\ \ 1]^{\intercal})$ za $\mathbf{u}_0 = \mathbf{0}$ tako da $\mathbf{J} \leftarrow \begin{bmatrix}J_{11} & J_{12}\\ J_{21} & J_{22}\end{bmatrix}$

5: $J_{11} \leftarrow H_{11} - H_{31}H_{13}$

6: $J_{12} \leftarrow H_{12} - H_{32}H_{13}$

7: $J_{21} \leftarrow H_{21} - H_{31}H_{23}$

8: $J_{22} \leftarrow H_{22} - H_{32}H_{23}$

9: $\mathbf{v} \leftarrow [H_{13}, H_{23}]^{\intercal}$

10: $(\gamma, \mathbf{R}_1, \mathbf{R}_2) \leftarrow IPPE(\mathbf{v}, \mathbf{J})$

11: $\mathbf{t}_1 \leftarrow (\mathbf{W}_1^{\intercal}\mathbf{W}_1)^{-1}\mathbf{W}_1\mathbf{b}_1$

$\mathbf{t}_2 \leftarrow (\mathbf{W}_2^{\mathrm{T}}\mathbf{W}_2)^{-1}\mathbf{W}_1\mathbf{b}_2$

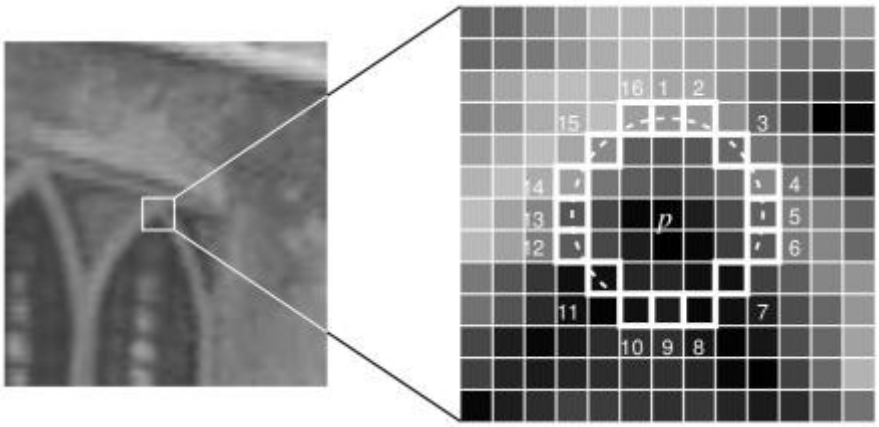
12: Vraća $\{\mathbf{R}_1, \mathbf{t}_1\}, \{\mathbf{R}_2, \mathbf{t}_2\}$

Ovaj algoritam može izračunati višestruke procjene položaja, od kojih je ona s najmanjom pogreškom reprojekcije optimalno rješenje [19] [29].

6.6. FAST algoritam

Detekcija značajki iz ubrzanog segmentnog testa (FAST) je metoda otkrivanja uglova, koja se može koristiti za izdvajanje ključnih točaka i kasnije koristiti za praćenje i mapiranje objekata u mnogim zadacima računalnog vida. Prednost FAST detektora uglova koja najviše obećava je njegova računalna učinkovitost. Štoviše, kada se primjenjuju tehnike strojnog učenja, mogu se ostvariti vrhunske performanse u smislu vremena računanja i resursa. Detektor uglova FAST vrlo je prikladan za primjenu obrade videa u stvarnom vremenu zbog velike brzine izvođenja [15][30].

1. Odabire piksel p na slici koji treba identificirati kao interesnu točku ili ne, s intenzitetom I_p.
2. Postavlja odgovarajuću vrijednost praga t.
3. Analizira krug od 16 piksela oko piksela koji se testira.
4. Piksel p je ugao ako postoji skup od n uzastopnih piksela u krugu (od 16 piksela) koji su svi svjetliji od $I_p + t$, ili svi tamniji od $I_p - t$. (Prikazano kao bijele crtice na slici 15). n je izabran da bude 12.



Slika 24: Prikaz analiziranog područja oko odabranog piksela p [29]

5. Predložen je test velike brzine kako bi se isključio veliki broj točaka koji ne predstavljaju ugao. Ovaj test ispituje samo četiri piksela na 1, 9, 5 i 13 (prvi 1 i 9 se testiraju jesu li previše svjetliji ili tamniji. Ako jesu, tada se provjeravaju 5 i 13). Ako je p ugao, tada najmanje tri od njih moraju biti svjetlije od $I_p + t$ ili tamnije od $I_p - t$. Ako nijedno od ovoga nije slučaj, tada p ne može biti kut. Kriterij potpunog testa segmenta se tada može primijeniti na kandidate koji su prošli ispitivanjem svih piksela u krugu.

Ovaj detektor sam po sebi pokazuje visoke performanse, ali ima nekoliko slabosti:

a. Ne odbija toliko kandidata za $n < 12$.
b. Izbor piksela nije optimalan jer njegova učinkovitost ovisi o redoslijedu ispitivanja i raspodjeli pojavljivanja uglova.
c. Rezultati brzih testova se odbacuju.
d. Otkriva se više značajki koje stoje jedna uz drugu.

Zato se primjenjuje oblik algoritma koji koristi strojno učenje:

1. Odabire se skup slika za treniranje (po mogućnosti iz željene domene aplikacije)
2. Pokreće FAST algoritam na svakoj slici za pronalazak značajki.
3. Za svaku ključnu točku pohranjuje 16 piksela oko nje kao vektor, i to za sve slike kako bi dobili vektor značajki P.
4. Svaki piksel (x) u ovih 16 piksela može imati jedno od sljedeća tri stanja:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t & (tamnije) \\ s, & I_p - t < I_{p\to x} < I_p + t & (slično) \\ b, & I_p + t \leq I_{p\to x} & (svjetlije) \end{cases}$$

5. Ovisno o tim stanjima, vektor obilježja P podijeljen je u 3 podskupa, P_d, P_s, P_b.
6. Definira novu booleovu varijablu, K_p, koja je istinita ako je p ugao, a lažna u suprotnom.
7. Koristi ID3 algoritam (klasifikator stabla odlučivanja) za ispitivanje svakog podskupa pomoću varijable K_p za saznanje o istinitoj klasi. Odabire x koji daje najviše informacija o tome je li piksel kandidat za ugao, mjereno entropijom K_p.
8. Ovo se rekurzivno primjenjuje na sve podskupove dok njegova entropija ne bude nula.
9. Tako stvoreno stablo odlučivanja koristi se za brzo otkrivanje u drugim slikama.

Otkrivanje više interesnih točaka na susjednim lokacijama još je jedan problem. Rješava se korištenjem supresije ne-maksimuma:

1. Izračuna funkciju reultata, V za sve otkrivene značajke. V je zbroj apsolutne razlike između vrijednosti p i 16 okolnih piksela.
2. Odabire dvije susjedne ključne točke i izračuna njihove V vrijednosti.
3. Odbacuje onaj s nižom V vrijednošću.

6.7. RANSAC algoritam

RANSAC (*RANdom SAmple Consensus*) je tehnika ponavljanog uzorkovanja koja generira rješenja kandidata korištenjem minimalnog broja promatranja (podatkovnih točaka) potrebnih za procjenu osnovnih parametara modela. RANSAC koristi najmanji mogući skup i nastavlja povećavati ovaj skup s konzistentnim podatkovnim točkama. Osnovni algoritam je sažet na sljedeći način [31][32]:

1. Odabire nasumično minimalni broj točaka potrebnih za određivanje parametara modela.
2. Traži rješenje za parametre modela
3. Određuje koliko točaka iz skupa svih točaka odgovara modelu s unaprijed definiranom tolerancijom ϵ.

4. Ako udio broja točaka koji odgovaraju modelu u ukupnom broju točaka u skupu premašuje unaprijed definirani prag τ, ponovno procjenjuje parametre modela koristeći sve identificirane odgovarajuće točke i algoritam završava.
5. U suprotnom, ponavlja korake od 1 do 4 (maksimalno N puta).

Broj ponavljanja, N, odabire se dovoljno visok da osigura da vjerojatnost p da barem jedan od skupova nasumičnih uzoraka ne uključuje točku koja ne odgovara modelu. Neka u predstavlja vjerojatnost da je bilo koja odabrana podatkovna točka odgovarajuća, a $v = 1 - u$ vjerojatnost odabira točke koja ne odgovara modelu. Tada je:

$$1 - p = (1 - u^m)^N \tag{a.21}$$

odakle slijedi N kao broj potrebnih iteracija nad minimalnim brojem točaka m:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)} \tag{a.22}$$