Izgradnja 3D modela u obliku teksturirane mreže trokuta na temelju dvije slike

Brkić, Valent

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IZGRADNJA 3D MODELA U OBLIKU TEKSTURIRANEMREŽE TROKUTA NA TEMELJU DVIJE SLIKE

Završni rad

Valent Brkić

Osijek, 2016.

SADRŽAJ

1.Uvod

U modernom dobu, gdje je težnja da se računalnim sustavima omogući snalaženje u novim situacijama, tzv. umjetna inteligencija, računalni vid ima veliku ulogu. Jedan od temeljnih problema računalnog vida je trodimenzionalna rekonstrukcija slike koju nije moguće izvesti informacijama dobivenim iz samo jedne slike budući da se unutar jedne slike može nalaziti više trodimenzionalnih scena. Međutim, ako se ista scena snimi iz dvije točke gledišta moguće je odrediti informacije o trodimenzionalnoj geometriji scene. Uz pomoć dviju kamera međusobno paralelno postavljenih dobiju se dva potrebna pogleda na scenu iz različitih kuteva na temelju kojih odgovarajućim metodama određujemo položaj i oblik predmeta u prostoru.

Zadatak ovog završnog rada je unaprijediti postojeći program, koji na temelju dvije slike neke scene određuje trodimenzionalne pozicije karakterističnih točaka scene[1]. Od dobivenog skupa točaka potrebno je izgraditi model u obliku mreže trokuta (engl. *triangular mesh*).Pod mrežom trokuta podrazumijeva se Delaunayeva triangulacija, metoda koju je 1934. razvio ruski matematičar Boris Delaunay, a pomaže nam pri izgradnji modela u obliku trokuta na temelju skupa točaka. Program također treba omogućiti vizualizaciju dobivenog modela te modifikaciju istog od strane korisnika. Zadatak je potrebno obaviti korištenjem razvojnog okruženja MS Visual C++, a za realizaciju zadatka također će se koristiti OpenCV biblioteku.

U drugom poglavlju detaljnije je objašnjen princip stereo vizije te na koji se način pomoću dviju slika snimljenih stereo parom kamera dobiju pozicije trodimenzionalnih karakterističnih točaka.

U trećem poglavlju opisana je Delaunayeva triangulacija, objašnjeno je o kakvoj se zapravo triangulaciji radi, njena glavna svojstva te su dani primjeri po čemu je ona prikladan način prikaza modela u usporedbi s drugim triangulacijama.

U četvrtom poglavlju opisano je grafičko sučelje za uređivanje triangulacije s opisanim metodama realizacije i rješavanjem problema.

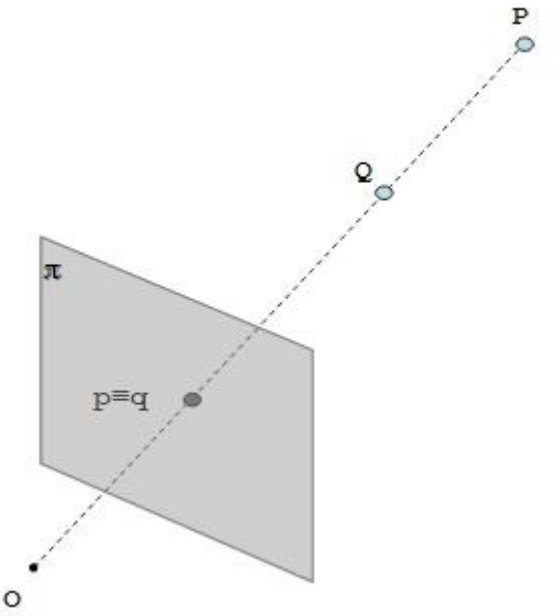
U petom poglavlju dan je uvid u rezultate dobivene primjenom Delaunayeve triangulacije na temelju dvodimenzionalnog skupa točaka dobivenog iz inicijalnog programa za određivanje trodimenzionalnih pozicija karakterističnih točaka scene.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je unaprijediti postojeći program koji određuje karakteristične točke scene na temelju dvije slike snimljene stereo kamerama. Na temelju dobivenog skupa točaka potrebno je odrediti 3D model u obliku teksturirane mreže trokuta pod nazivom Delaunay triangulacija. Također je potrebno omogućiti modifikaciju mreže trokuta putem grafičkog sučelja.

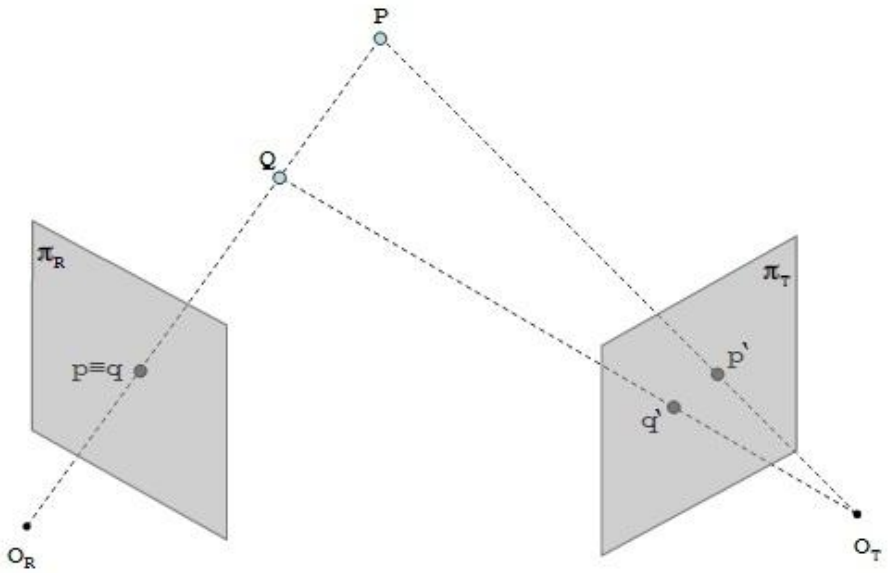
2. Trodimenzionalna rekonstrukcija scene pomoću stereo vizije

Stereo vizija [2] je metoda pomoću koje se može izvesti zaključak o dubini, poziciji i obliku objekta na temelju dvije kalibrirane kamere. U slučaju korištenja jedne kamere sve točke koje se nalaze na istom pravcu projicirati će se u istu točku (Sl. 2.1.)



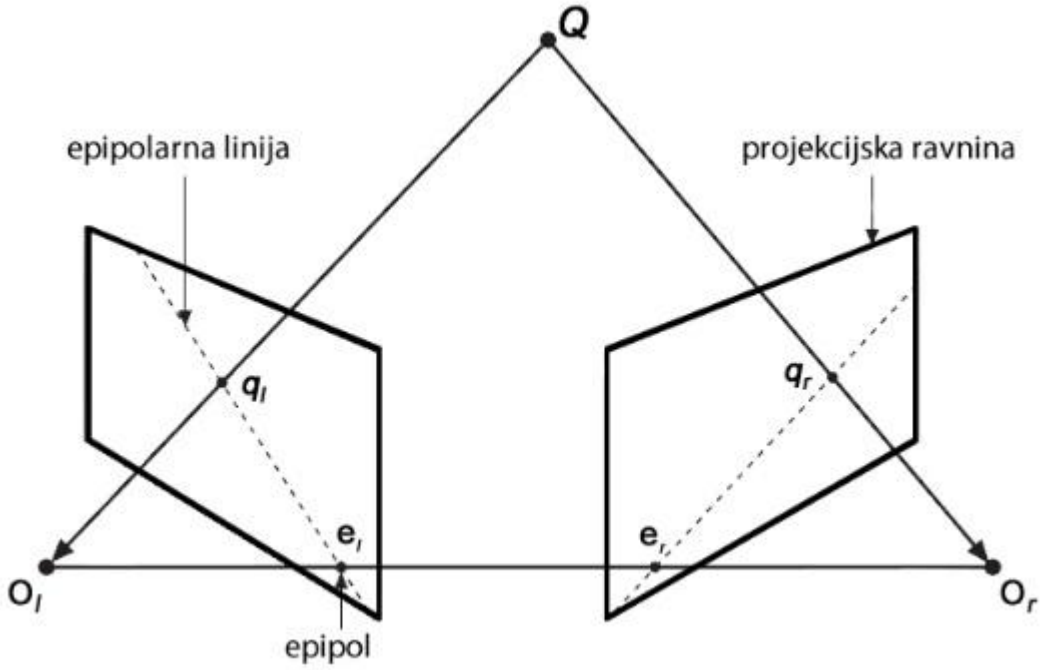
Slika 2.1. Dvije različite točke projiciraju se u jednu točku na slici kamere (Slika preuzeta iz [3])

U slučaju da se koriste dvije kamere, ukoliko su poznate međusobno odgovarajuće točke pomoću triangulacije je moguće izvući pretpostavke o dubini prema slici 2.2.



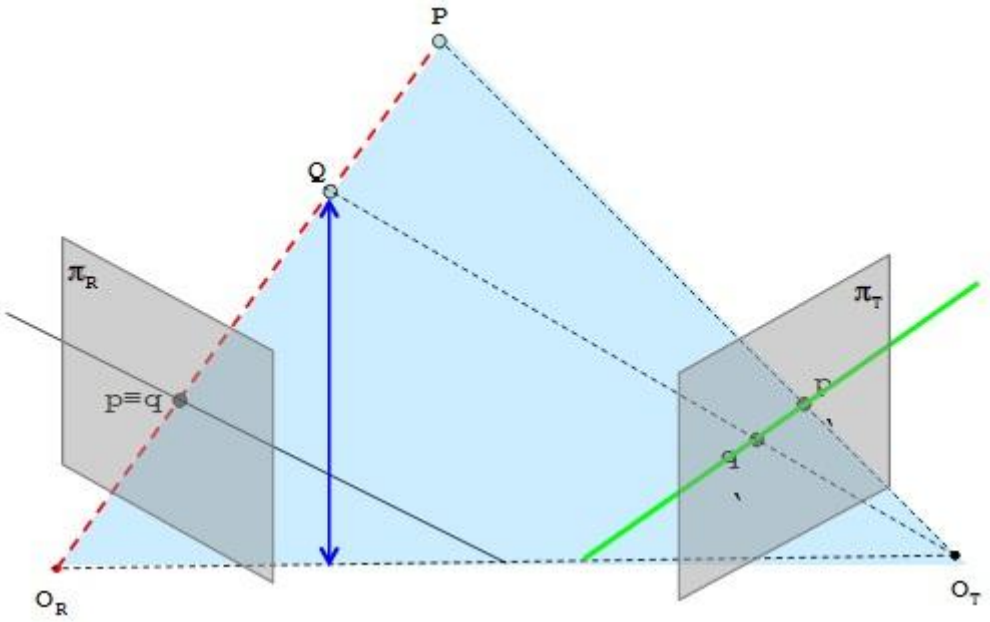
Slika 2.2. Par stereo kamera (Slika preuzeta iz [3])

Da bi se riješio problem stereo korespodencije koristi se epipolarna geometrija (Sl. 2.3.), točnije epipolarno ograničenje.



Slika 2.3. Epipolarna geometrija (Slika preuzeta iz [4])

Za svaku točku na prvoj slici moguće je odrediti odgovarajuću točku na drugoj slici. Optička zraka na prvoj slici s pripadajuće dvije točke koje se projiciraju u istu točku na ravnini kamere imaju zasebno svoj odgovarajući par na drugoj slicigdje se projiciraju u odgovarajuće dvije točke na ravnini druge kamere. Budući da se svaka točka može nalaziti samo na jednoj optičkoj zraci njena projekcija se može nalaziti samo na odgovarajućoj epipolarnoj liniji što se naziva epipolarno ograničenje koje ograničava vjerojatnost pogrešnog sparivanja. Epipolarno ograničenje i određivanje odgovarajuće točke prikazano je prema slici 2.4.

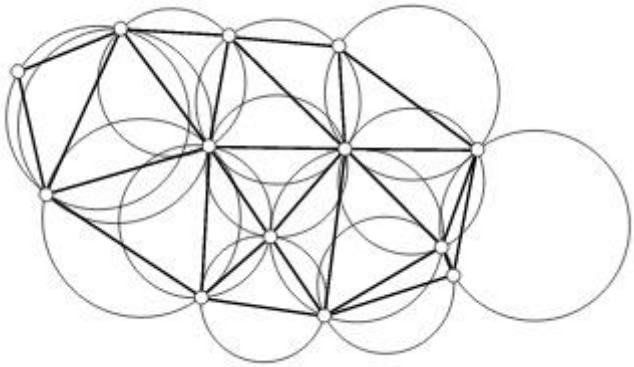


Slika 2.4. Epipolarno ograničenje (Slika preuzeta iz [3])

Za algebarski prikaz epipolarne geometrije koristi se fundamentalna matrica koja sadrži podatke o intrinsičnim parametrima kamera i njihovim međusobnim položajima. Također, poput fundamentalne matrice, esencijalna matrica opisuje epipolarno ograničenje, no ne ovisi o intrinsičnim parametrima kamera, nego samo o njihovom međusobnom položaju[5].

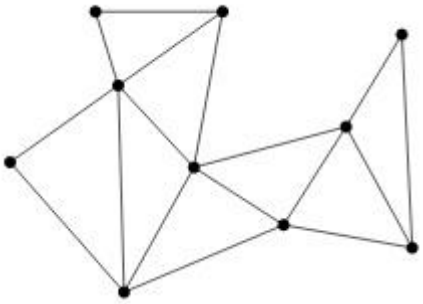
3. Delaunay Triangulacija

Delaunay triangulacija [6] je geometrijska struktura koju inženjeri koriste još od začetaka stvaranja mreže trokuta. U dvodimenzionalnom prostoru ima veliku prednost u usporedbi s drugim triangulacijama koje koriste određeni skup točaka jer maksimalizira minimalni kut. Također, predstavljena od strane Borisa Nikolayevicha Delaunaya 1934. karakteristična je i po tome što u skupu točaka *S* nijedna točka ne leži unutar opisane kružnice trokuta što se može vidjeti prema slici 3.1.

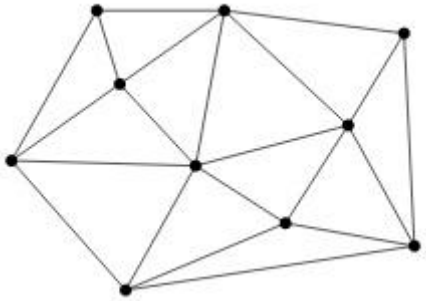


Slika 3.1. Svaki trokut u Delaunayevoj triangulaciji ima praznu opisanu kružnicu (Slika preuzeta iz [6])

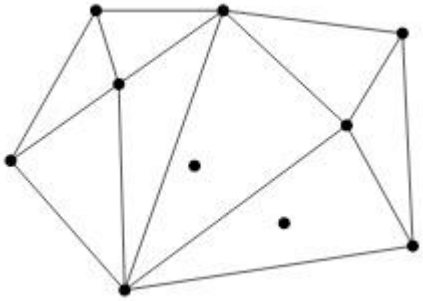
Skup točaka nema određenu unutrašnjost kao što imaju mnogokuti no kada se krajnje točke skupa točaka spoje sliče mnogokutu koji ima točke u svojoj unutrašnjosti. Triangulacija služi da se te točke povežu u trokute. To se može napraviti na više načina, kao što se vidi na slikama 3.2. i 3.3. Sve točke moraju biti spojene pa je triangulacija prema slici 3.4. neispravna.



Slika 3.2. Jednostavna triangulacija (Slika preuzeta iz [6])



Slika 3.3.Zatvorena triangulacija (Slika preuzeta iz [6])

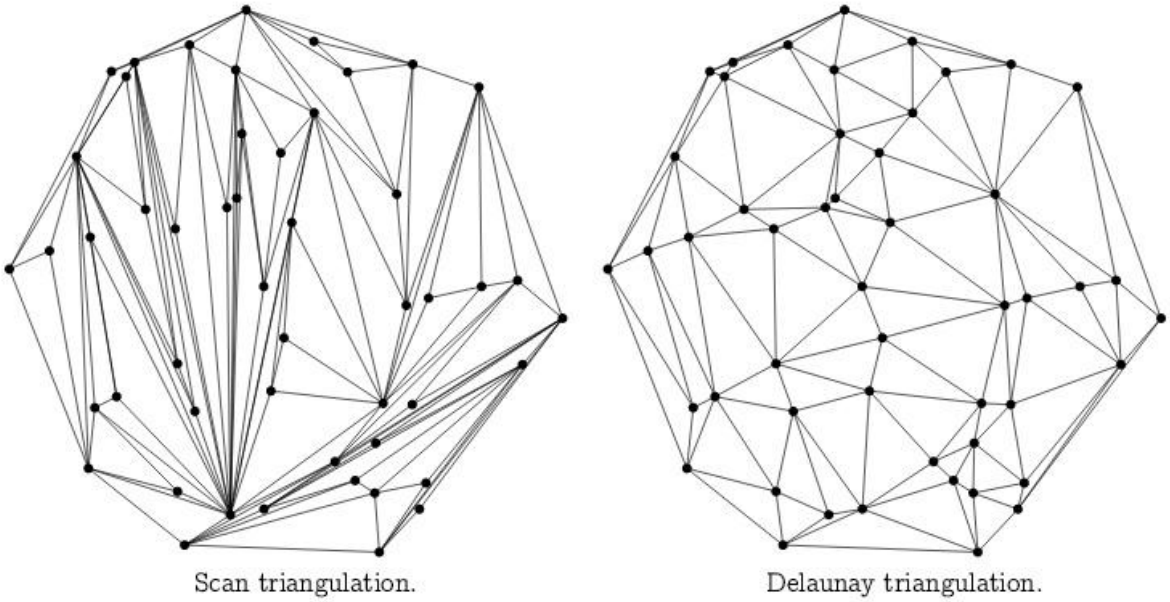


Slika 3.4. Neispravna triangulacija (Slika preuzeta iz [6])

Za konstrukciju triangulacije skupa točaka *S*,gdje su $p_1,...,p_n$točke skupa točaka *S*, a *m* minimalan broj potrebnih točaka gdje $p_1,...,p_m$ nisu kolinearni spajamo $p_1,...,p_{m-1}$. Zatim dodajemo $p_{m+1},...,pn$ prema slici 3.5. Na taj način se konstruira *scan triangulation*[6].

Slika 3.5. Konstrukcija triangulacije (Slika preuzeta iz [6])

Problem kod takve triangulacije nije samo neprivlačnost oku zbog dugačkih krakova već nije praktična kod izgradnje modela scene. Upravo zato je Delaunayeva triangulacija stekla takvu popularnost. Usporedba ovih dvaju triangulacija nalazi se na slici 3.6.

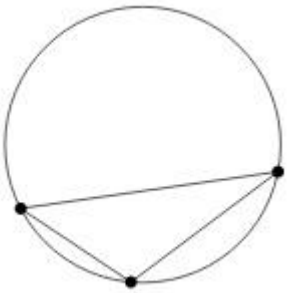


Slika 3.6. Usporedba Scan i Delaunay triangulacija

(Slika preuzeta iz [6])

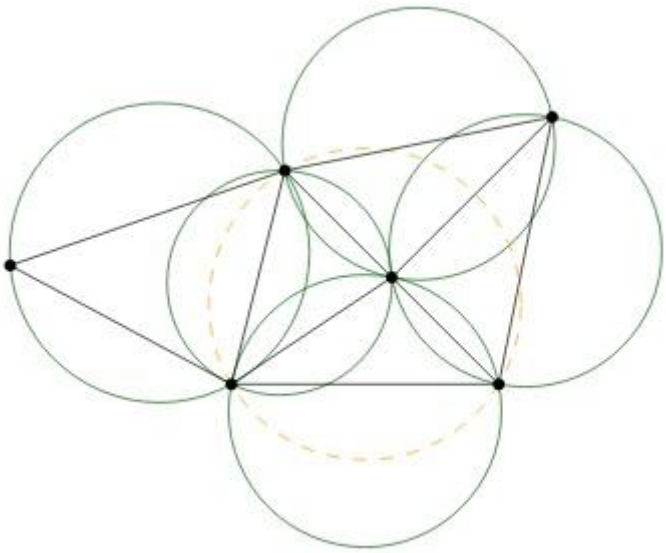
3.1. Svojstvo prazne opisane kružnice

Opisana kružnica je jedinstvena kružnica koja prolazi kroz sve tri točke trokuta (Sl. 3.7.)



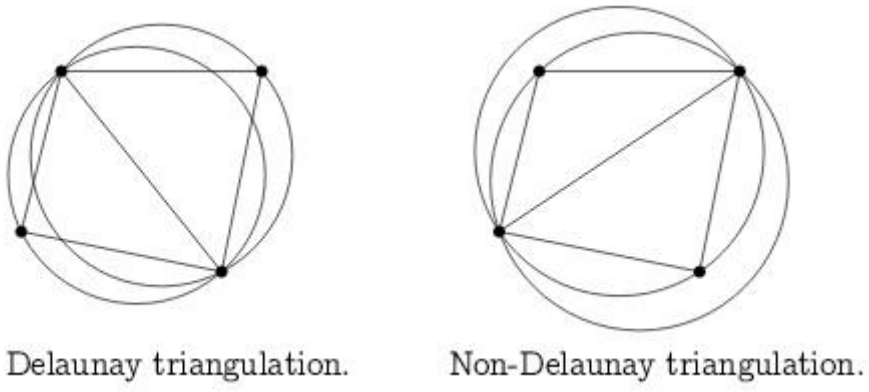
Slika 3.7. Opisana kružnica (Slika preuzeta iz [6])

Triangulacija konačnog skupa točaka *S* naziva se Delaunayeva triangulacija ukoliko su opisane kružnice svakog trokuta prazne, to jest, ne postoji točka *p* iz skupa točaka *S* koja se nalazi u njihovoj unutrašnosti. Na slici 3.8. nalazi se primjer sa skupom od šest točaka gdje opisane kružnice svakog trokuta zadovoljavaju svojstvo prazne opisane kružnice.

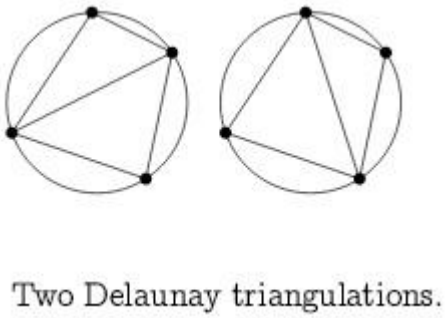


Slika 3.8. Svojstvo prazne opisane kružnice (Slika preuzeta iz [6])

Na primjeru sa skupom od četiri točke gdje su moguće dvije triangulacije generalno postoji samo jedna Delaunayeva triangulacija (Sl. 3.9.), no ako se točke nalaze na zajedničkoj kružnici koja je ujedno njihova opisana kružnica tada su obje triangulacije Delaunayeve (Sl. 3.10.)



Slika 3.9. Slučaj sa samo jednom Delaunayevom triangulacijom (Slika preuzeta iz [6])

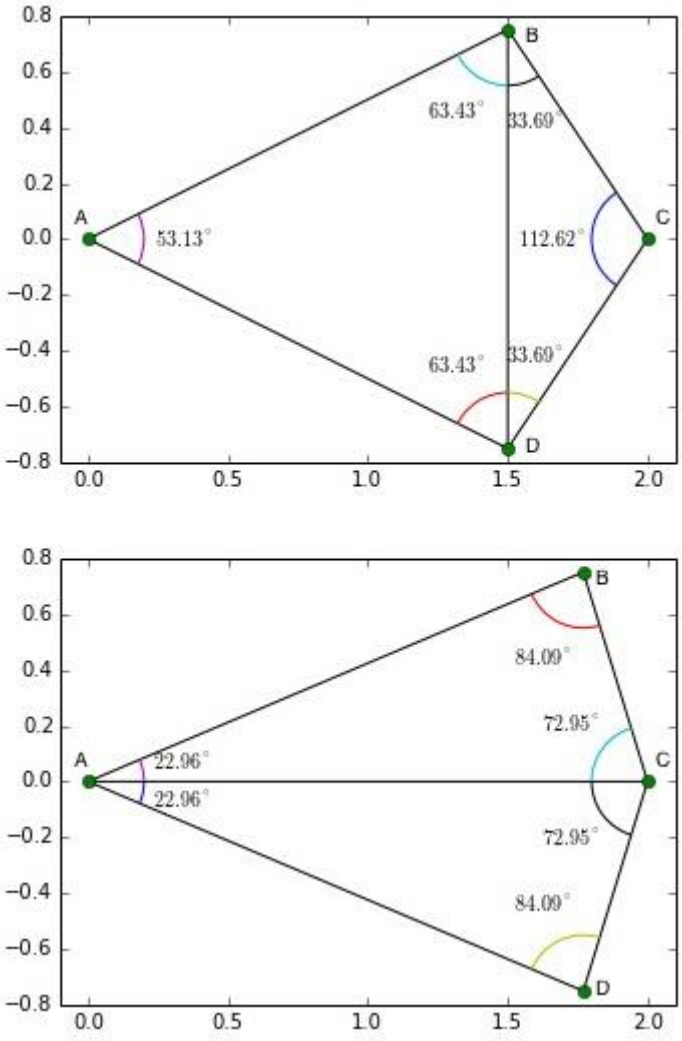


Slika3.10. Slučaj s obje Delaunayeve triangulacije (Slika preuzeta iz [6])

3.2. Maksimalizacija malih kuteva

Osim svojstva prazne opisane kružnice, Delaunayeva triangulacija ima svojstvo da maksimalizira manje kuteve. Na slici 3.11. nalaze se dva primjera. Na gornjem primjeru točke B i D imaju koordinate 1.5 na osi apscisa dok su na donjem primjeru pomaknuti na 1.75. Na gornjem primjeru primjećujemo velike kuteve ∠ABC i ∠ADC koje Delaunayeva triangulacija spajanjem točaka B i D dijeli na manje kuteve ∠ABD, ∠ADB, ∠CDB i ∠CBD.

S druge strane, na donjem primjeru maksimalizacija malih kuteva ostvaruje se spajanjem točaka A i C te se najveći kut ∠BCD dijeli na manje kuteve ∠BCA i ∠ACD.



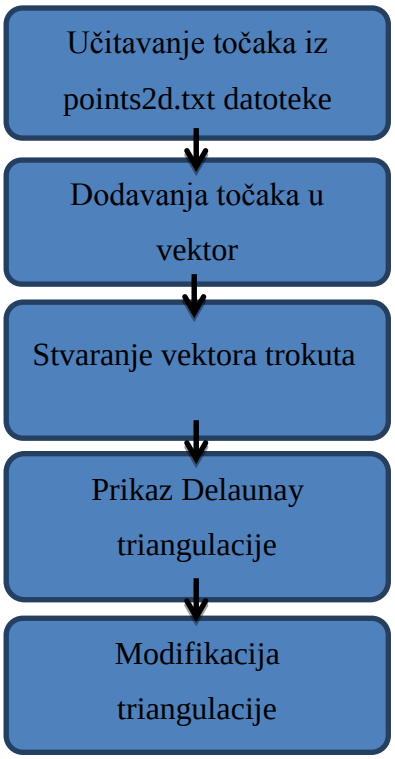
Slika 3.11. Maksimalizacija manjih kuteva (Slika preuzeta iz [7])

4. Grafičko sučelje za uređivanje triangulacije

Pomoću razvojnog okruženja Microsoft Visual Studio 2012, biblioteke OpenCV 3.0 (Open Computer Vision, verzija 3.0), aplikacije za trodimenzionalnu rekonstrukcije scene iz dvije slike, te objektnog programskog jezika C++ izrađen je program za izgradnjuobliku mreže trokuta na temelju dvodimenzionalnih karakterističnih točaka scene.OpenCV je biblioteka poglavito namijenjena za razvoj programa i pomoć pri problematici iz područja računalnog vida. Biblioteka je multiplatformska, što znači da ju je moguće koristiti na više različitih operacijskih i mobilnih sustava te različitih računalnih arhitektura. OpenCV je pod BSD (Berkeley Software Distribution) licencom otvorenog koda te jebesplatan za korištenje.

Glavna bit programa je na temelju dvodimenzionalnog polja točaka izgraditi mrežu trokuta koja daje grafički uvid u izgled scene te omogućiti modifikaciju od strane korisnika ukoliko je potrebno ispraviti bridove trokuta koji ne odražavaju stvarne bridova objekata na sceni.Program je napisan za korištenje pod operativnim sustavom Microsoft Windows.

Dijagram toka programa prikazan je u nastavku



Program kao ulaznu informaciju koristi tekstualnu datoteku points2d.txt u kojoj se nalazi dvodimenzialno polje točaka dobivenih pomoću programa za trodimenzionalnu rekonstrukciju. Navedeni program učitava dvije slike snimljene kalibriranom kamerom sa poznatim projekcijskim matricama. Nakon učitavanja druge slike pokreće se SIFT algoritam koji detektira značajke slike koje zapravo predstavljaju ključne točke kojima su pridruženi deskriptori, tj. vektori od 128 komponenti s informacijama o okolini neposredno od točke. Ovaj postupak radi se s obje slike te se time sparuju pojedine značajke. Pogrešno sparene značajke nakon toga se eliminiraju epipolarnom geometrijom opisanom u drugom poglavlju. Nakon toga se pomoću RANSAC metode računa fundamentalna matrica iz koje proizlazi matrica na temelju koje se izdvajaju dobro uparene značajke. Cjelokupni proces detaljnije je objašnjen u [1]. Uspješno sparene koordinate točaka spremaju se u points2d.txt koji program za Delaunay triangulaciju koristi za daljni rad.

Unutar programa za Delaunay triangulaciju se prvobitno definira vektor točaka u koji se potom spremaju sve točke iz dane tekstualne datoteke. Zatim se odredi pravokutnik pomoću kojeg se definira prostor za korištenje. Budući da sve korištene točke pripadaju slici, pravokutnik se ograničava na dimenzije slike na sljedeći način:

```
Size size = img.size();
Rect rect(0, 0, size.width, size.height);
```

Pomoću dobivenog pravokutnika kreira se instanca klase subDiv2d.

```
Subdiv2D subdiv(rect);
```

U subdiv se dodaju točke koristeći funkciju insert. Funkcija getTriangleList vraća popis svih trokuta. Svaki trokut predstavljen je kao vektor od šest komponenata gdje svaka komponenta predstavlja koordinatu apscise ili ordinate, a uzastopne dvije komponente predstavljaju jednu točku trokuta. Klasa subDiv2d ne vraća indekse točaka, već samo njihove koordinate stoga ne bi bilo praktično raditi izmjene na podatkovnoj strukturi subDiv2d. Rješenje ovog problema je napraviti vlastitu strukturu koja bi se sastojala od popisa točaka i trokuta. Popis točaka nalazi se u vektoru točaka dobivenom iz tekstualne datoteke koja je inicijalno učitana. Potrebno je napraviti vektor trokuta gdje će se svaki trokut sastojati od tri komponente. Svaka komponenta predstavlja indeks jedne točke trokuta. Trokute je na ovaj način potrebno kopirati iz subDiv2d-a. Budući da indeksima točaka nije omogućen pristup, o indeksima točaka morati će se zaključiti na temelju njihovih koordinata. Trokutima klase subDiv2d pristupa se pomoću već spomenute funkcije getTriangleList. Za svaki trokut biti će potrebno dodati jedan vektor od tri komponente u vektor trokuta. S obzirom da subDiv2d sadrži trokute koji se nalaze izvan slike, potrebno je

odrediti uvjet koji će izbacivati neodgovarajuće trokute. Uvjet koji trokut unutar subDiv2d-a mora zadovoljiti je da se sve tri koordinate njegovih točaka nalaze unutar pravokutnika koji je definiran na početku postupka. Uvjet je definiran na sljedeći način:

if (rect.contains(pt[0]) && rect.contains(pt[1]) && rect.contains(pt[2])),gdje pripadna funkcija contains klase Rect vraća vrijednost "true" ako se točka, koja joj je proslijeđena kao argument nalazi unutar pravokutnika.

Kada je ovaj uvjet zadovoljen, na temelju koordinata točaka traži se najbliža točka svakom od vrhova trokuta. Pomoću formule za udaljenost dviju točaka (4-1) uspoređuju se koordinate točaka trokuta iz subDiv2d s vektorom točaka. Pamti se indeks najbliže točke koji se upisuje u odgovarajuće mjesto u vektoru trokuta.Neka su A_1 i A_2dvije točke s pripadajućim koordinatama (x_1,y_1) i (x_2,y_2). Tada se njihova udaljenost može izračunati sljedećim izrazom:

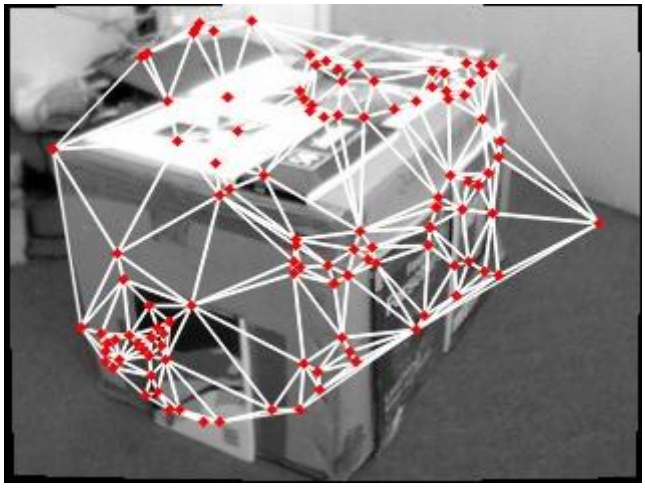
$$d(A_1, A_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4-1}$$

Na ovaj način u vektoru trokuta dobije se kompletna mreža trokuta na kojoj se mogu vršiti jednostavne izmjene.

Funkcija draw_delaunay služi za prikaz Delaunay triangulacije. Određena je na sljedeći način:

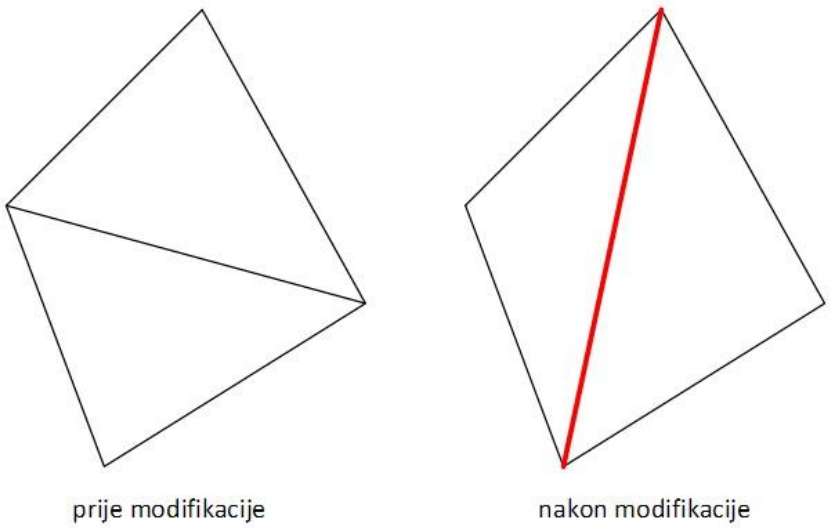
staticvoid draw_delaunay (Mat&img, vector<Vec3i>&triangles, vector<Point2f>&points, Scalardelaunay_color)

Pomoću vektora *triangles* pristupa odgovarajućim indeksima točaka vektora points. Budući da unutar vektora trokuta svaka dva uzastopna indeksa točaka čine jedan brid trokuta vektora *triangles*, pomoću njega je moguće prikazati sve postojeće bridove. Za to služi funkcija line koja crta liniju između dvije točke.Ovaj postupak omogućuje uspješno prikazivanje Delaunay triangulacije kao što je vidljivo na slici 4.1.



Slika 4.1. Delaunay triangulacija

Delaunay triangulacija ponekad daje bridove koji ne odražavaju stvarne bridove objekata na sceni kao što je vidljivo prema slici 4.1. U takvom slučaju potrebno je modificirati Delaunay triangulaciju tako da se neodgovarajući brid između dva trokuta izbriše i umjesto njega doda novi brid koji spaja nasuprotne točke ta dva trokuta kao što je ilustrirano na slici 4.2.



Slika 4.2. Ilustracija modifikacije bridova

Program omogućuje selekciju bridova preko grafičkog sučelja na način da se klikne na brid koji ne odgovara karakteristikama scene. OpenCV omogućava poziv korisnički definirane callback funkcije kada se mišem klikne na sliku. OpenCV toj funkciji proslijeđuje koordinate vrha strelice na temelju kojih se može zaključiti koji brid je odabran za modifikaciju. Pridruživanjecallback funkcije prozoru u kojemu je prikazana slika radi se na sljedeći način:

```
setMouseCallback("Delaunay triangulation", CallBackFunc, &data);
```

Poziva sefunkcija CallBackFunc u kojoj se pomoću pokazivača podatkovne strukture MeshEditData proslijeđuju potrebne vrijednosti funkciji edge_flip koja je objašnjena u nastavku poglavlja.

Struktura MeshEditData potrebna je za prosljeđivanje informacija callback funkciji. Sastoji se od pokazivača na podatke koji su potrebni za realizaciju modificiranja triangulacije. Omogućeno je da se svaka izmjena vektora *triangles* manifestira promjenom izgleda triangulacije. Iz tog razloga se sa korisničke strane promjene na interaktivan način događaju u stvarnom vremenu. Ova podatkovna struktura definirana je na sljedeći način:

```
structMeshEditData
{
        Mat *img_orig;
        Mat *img;
        vector<Vec3i> *triangles;
        vector<Point2f> *points;
        Scalar edges_color;
};
```

Za selekciju i okretanje bridova koristi se funkcija edge_flip koja ažurira vektor trokuta *triangles* novonastalim bridovima. Odabiranje brida omogućeno je na način da se za svaki brid provjerava koliko je udaljen od točke koja je proslijeđena od strane callback funkcije. Izabire se onaj brid koji je točki najbliži. Da bi ispravan brid bio odabran moraju se ispuniti dva uvjeta.

Jedan uvjet provjerava udaljenost točke od brida i odabire onaj brid za koji je ta udaljenost najmanja. Neka su koordinate vrha strelice miša koje callback funkcija vraća (x, y). Neka su koordinate krajnjih točaka dužine (brida) (x_1, y_1) i (x_2, y_2). Tada je udaljenost točke (x,y) od navedene dužine određena izrazom (4-3).

$$d = \frac{|-(x-x_1)(y_2-y_1)+(y-y_1)(x_2-x_1)|}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \tag{4-2}$$

Drugi uvjet izračunava poziciju ortogonalne projekcija točke (x, y) na pravac na kojemu leži razmatrani brid te provjerava pada li ona između točaka (x_1, y_1) i (x_2, y_2). Ako se uzme da oznake imaju isto značenje kao izraz (4-2), pozicija ortogonalne projekcije točke (x, y) na navedeni pravac se može izračunati na sljedeći način:

$$p = \frac{(x-x_1)(x_2-x_1)+(y-y_1)(y_2-y_1)}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \tag{4-3}$$

Uvjet koji mora biti zadovoljen za odabir brida prikazan je sljedećom linijom koda:

if (p >= 0.0f && p <= l), gdje je

$$l = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \tag{4-4}$$

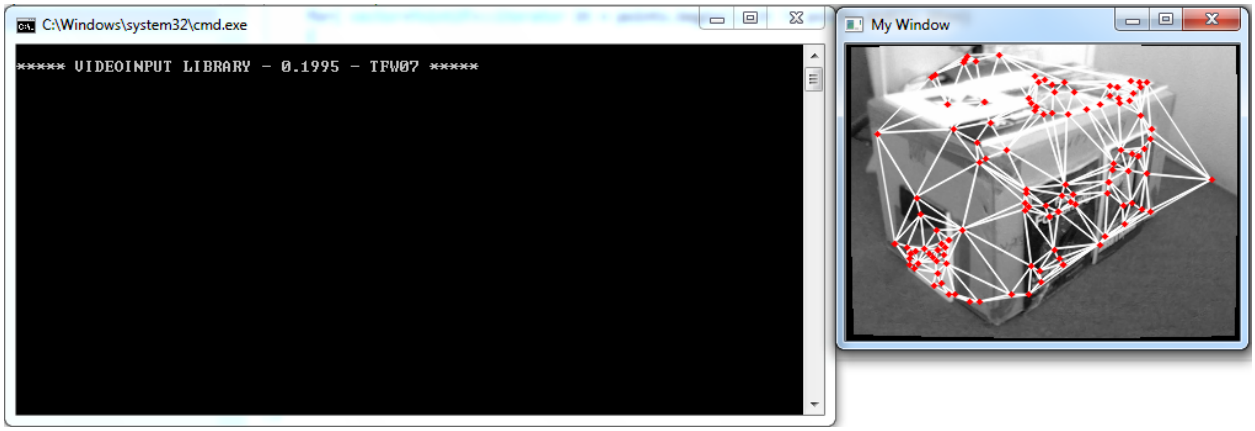
Kada su ova dva uvjeta određena izrazima (4-2), (4-3) i (4-4) zadovoljena pamte se indeksi točaka unutar vektora *triangles* za koje su uvjeti zadovoljeni. Te dvije točke čine odabrani brid.

Da bi se odabrani brid mogao okrenuti, potrebno je pronaći trokute unutar vektora *triangles* kojima je taj brid zajednički. To se postiže tako da se prolazi kroz vektor *triangles* i provjerava koji od trokuta unutar vektora *triangles* sadrže dva indeksa točaka kojima je određen odabrani brid. Moguće je da najviše dva trokuta zadovolje taj uvjet. Da bi se odabrani brid zamijenio bridom određenim nasuprotnim točkama koriste se dva pomoćna vektora. U jedan od njih su spremljena dva indeksa trokuta koji sadrže odabrani brid, a u drugi su spremljene 4 različita indeksa točaka odabranih trokuta.

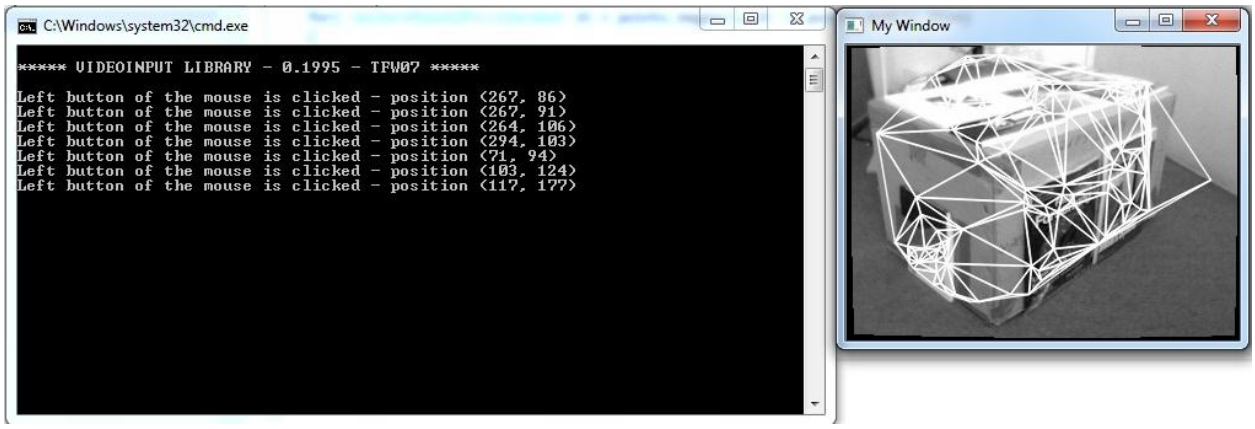
Potrebno je preraspodijeliti točke dva trokuta koji dijele odabrani brid. Neka su dva trokuta koji dijele odabrani brid T_1 i T_2i njihove pripadajuće točke (pt_1, pt_2, pt_3) i (pt_1, pt_4, pt_2). Nakon raspodjele točaka trokut T_1 treba se sastojati od točaka (pt_3, pt_1, pt_4), a trokut T2 od točaka (pt_3, pt_4, pt_2).

Na ovaj način dobije se modificirani vektor *triangles* kojemu je mreža trokuta izmijenjena. Takva izmijenjena mreža trokuta ažurira se na slici prilikom svakog okretanja bridova.

Rad i izgled grafičkog sučelja za uređivanje triangulacije prikazan je prema slikama 4.3. i 4.4.

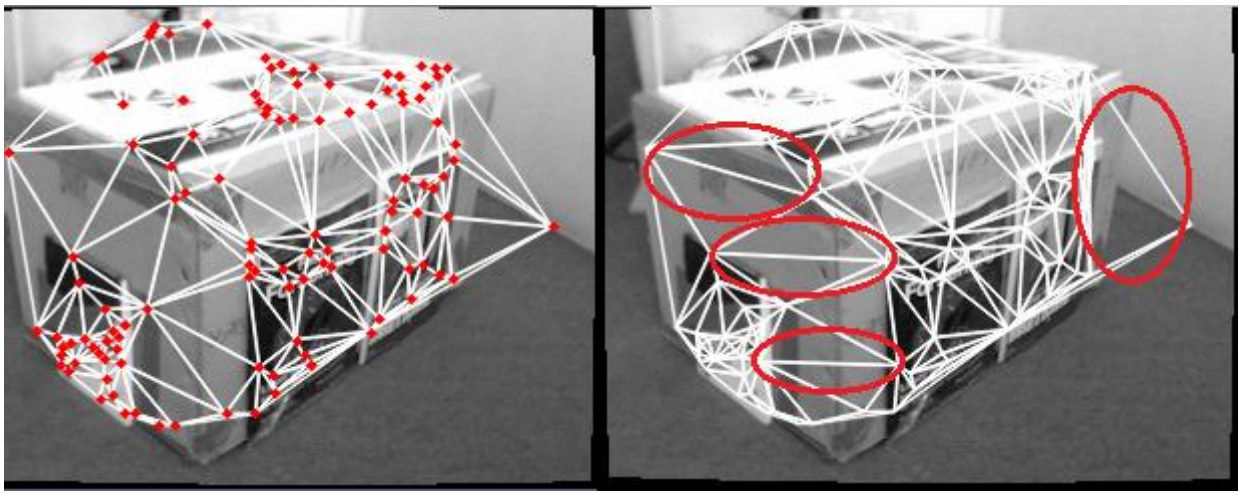


Slika 4.3. Početni izgled Delaunay triangulacije



Slika 4.4. Izgled Delaunay triangulacije nakon modifikacije

Prema slici 4.4. vidljivo je da su ispisane koordinate gdje su se nalazili bridovi koji su modificirani. Također su vidljive izmjene na mreži trokuta. Uvećana usporedba mreže trokuta prije i poslije modifikacije prikazana je na slici 4.5. Razlike su naglašene crvenim elipsama.

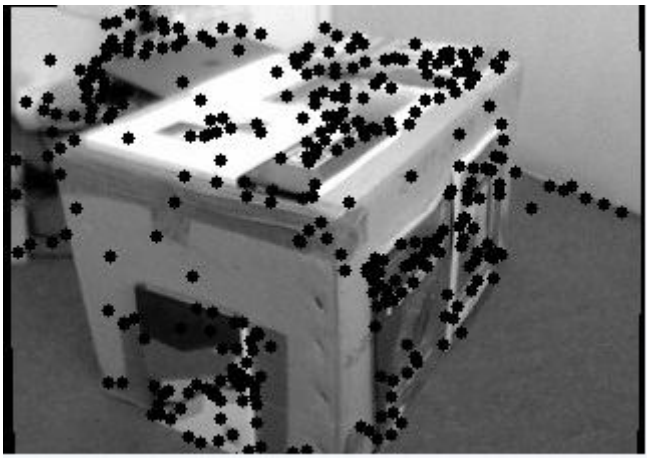


Slika 4.5. Uvećani izgled triangulacije prije i poslije modifikacije

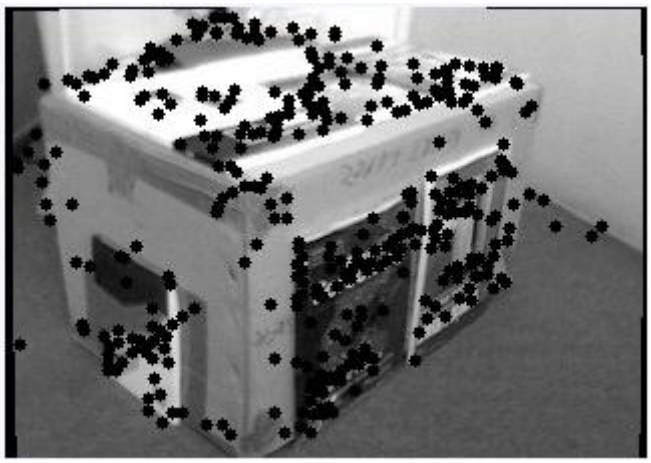
5. Rezultati pokusa

U ovom poglavlju dan je prikaz rezultata dobivenih primjenom programa opisanog u četvrtom poglavlju. Prvo se određuju karakteristične točke na temelju dvije slike na koje se primjenjuje program za trodimenzionalnu rekonstrukciju scene. Izlaz navedenog pograma je tekstualna datoteka sa koordinatama točaka koje se koriste kao ulazni podatak za program Delaunay triangulacije.

Na slici 5.1. i 5.2. su prikazane značajke detektirane SIFT-metodom.

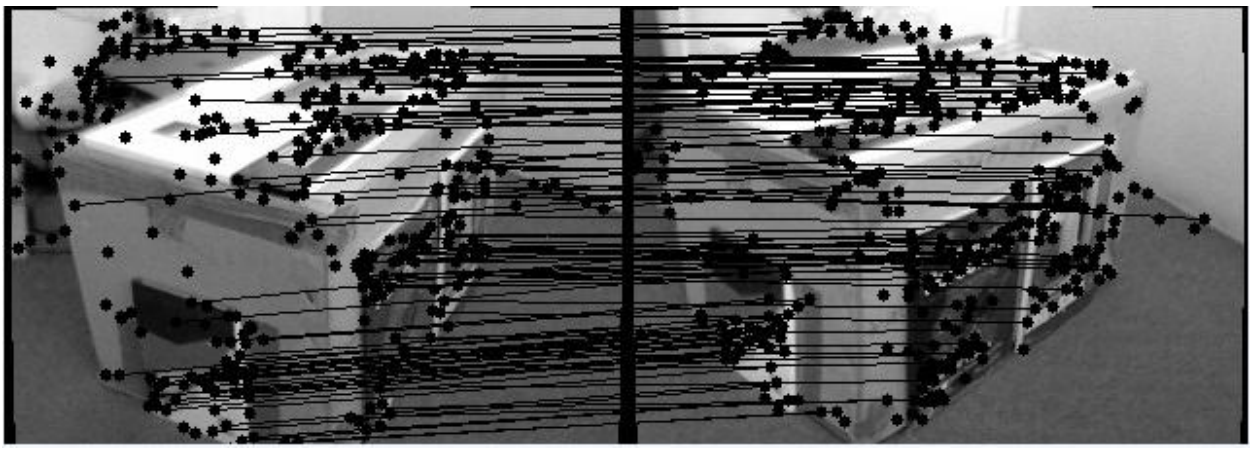


Slika 5.1. Detektirane značajke prve slike



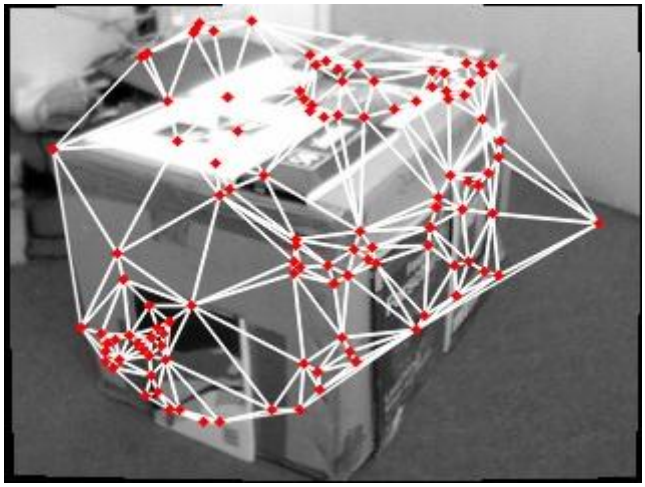
Slika 5.2. Detektirane značajke druge slike

Prikaz parova značajki nakon izbacivanja onih koji ne zadovoljavaju epipolarno ograničenje prikazan je na slici 5.3.



Slika 5.3. Prikaz parova značajki

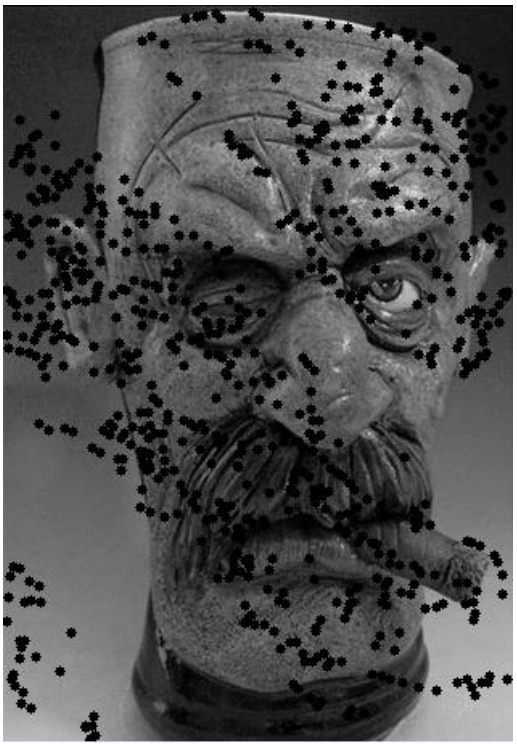
Kada se dobiju ispravno spojeni parovi značajki izvlače se dvodimenzionalne koordinate točaka s druge slike i na njima se provodi Delaunayeva triangulacija čije rezultate vidimo na slici 5.4.



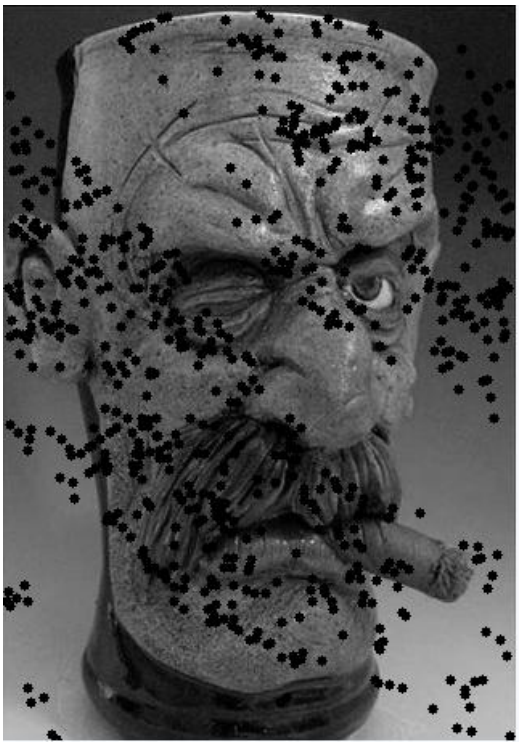
Slika 5.4. Prikaz Delaunayeve triangulacija

Prema slici je evidentno da postoje odstupanja koja treba korigirati. Za rješenje tog nedostatka će služiti modifikacija koju će korisnik moći koristiti u svrhu ispravljanja odstupanja na modelu scene.

Na slikama 5.5. – 5.12. prikazana su dodatna dva primjera stereo slika na kojima je proveden postupak generiranja karakterističnih točaka, sparivanja značajki i Delaunay triangulacije.

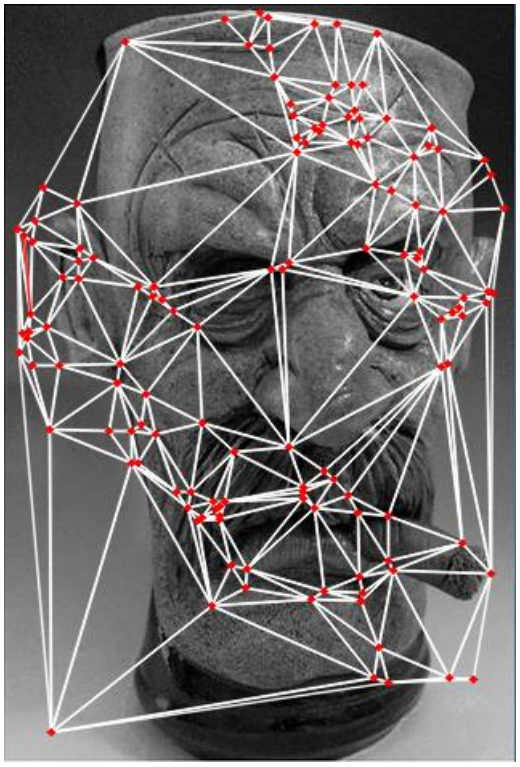


Slika 5.5. Detektirane značajke prve slike drugog primjera



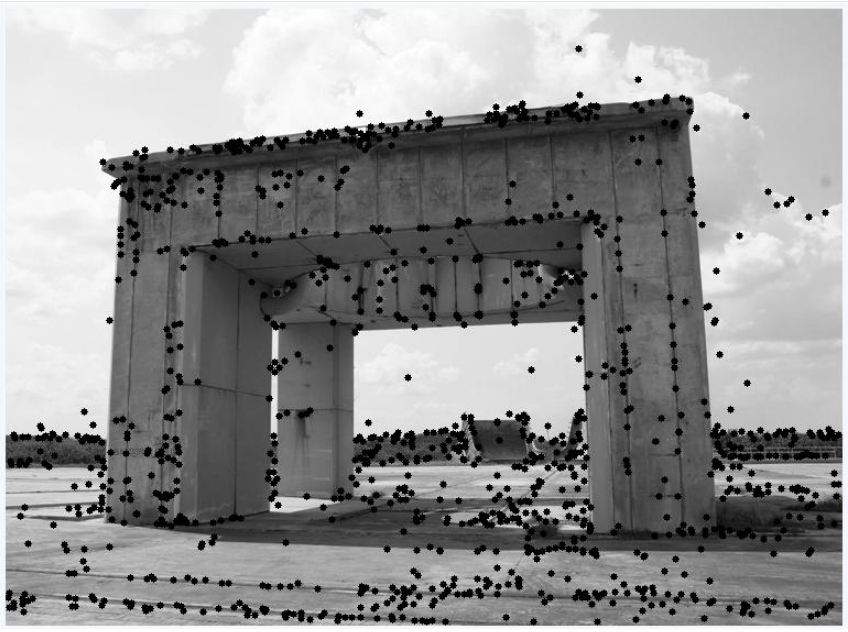
Slika 5.6. Detektirane značajke druge slike drugog primjera

Slika 5.7. Prikaz parova značajki drugog primjera koji zadovoljavaju epipolarno ograničenje

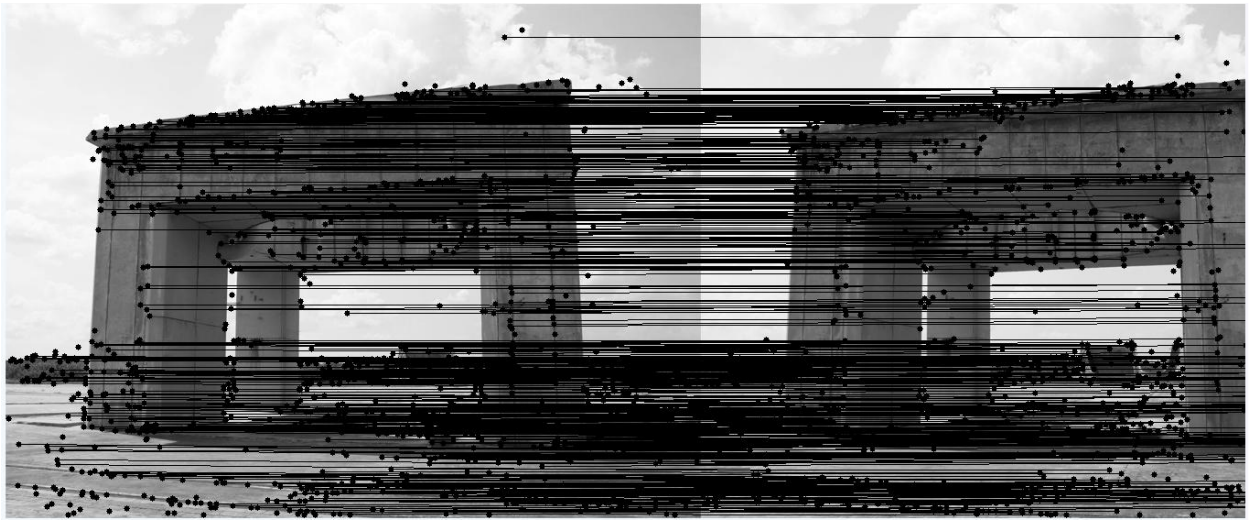


Slika 5.8. Prikaz Delaunayeve triangulacije drugog primjera

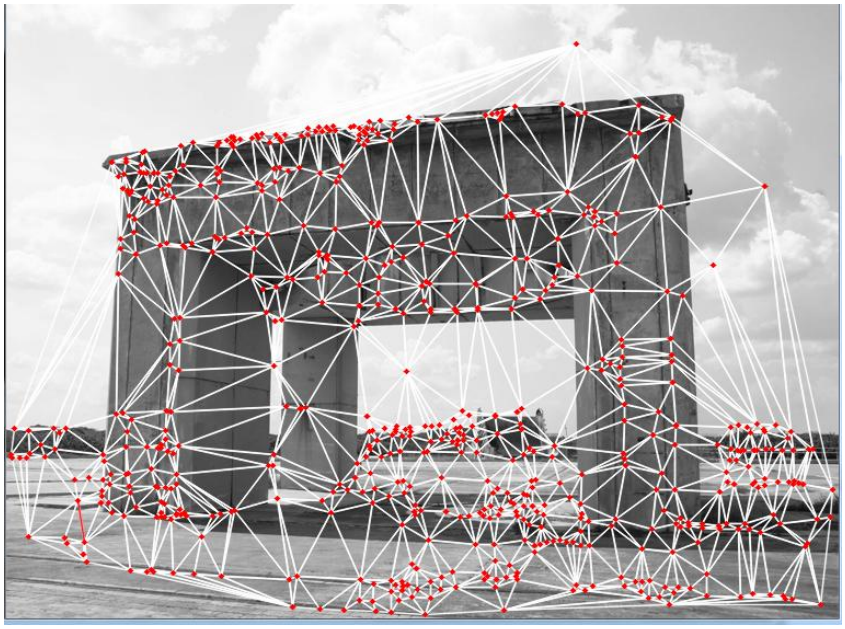
Slika 5.9. Detektirane značajke prve slike trećeg primjera



Slika 5.10. Detektirane značajke druge slike trećeg primjera



Slika 5.11. Prikaz parova značajki trećeg primjera koji zadovoljavaju epipolarno ograničenje



Slika 5.12. Prikaz Delaunayeve triangulacije trećeg primjera

6. Zaključak

U ovom radu opisano je rješenje za izgradnju trodimenzionalnog modela u obliku teksturirane mreže trokuta na temelju dvije slike, tj. na temelju dvodimenzionalnog skupa točaka sparenih i filtriranih pomoću fundamentalne matrice da bi se odbacile pogrešno sparene značajke. Rad je realiziran pomoću programskog jezika C++, OpenCV biblioteke te razvojnog okruženja Microsoft Visual studio 2012. Program učitava dvodimenzionalni skup točaka izgeneriran pomoću programa za trodimenzionalnu rekonstrukciju scene. Točke se spremaju u vektor točaka koji potom dodajemo u instancu klase subDiv2d. Budući da klasa subDiv2d ne omogućuje pristup indeksima točaka, pomoću funkcije getTriangleList se pristupa trokutima te ih se kopira u vektor trokuta na način da za svaki trokut zapišemo indekse točaka od kojih se sastoji. Pomoću vektora *triangles* i funkcija OpenCV-a dobije se željeni prikaz triangulacije. Modifikaciju triangulacije putem grafičkog sučelja omogućuju korisnički definirane callback funkcije kojima se pozivanjem odgovarajućih metoda može promijeniti izgled konstruirane triangulacije.Izrađeni program bi se mogao nadograditi trodimenzionalnim prikazom scene koristeći VTK biblioteku. VTK je *open-source* biblioteka koja se koristi za trodimenzionalnu vizualizaciju i procesiranje slika.

7. Literatura

[1] D. Kurtagić, *Trodimenzionalna rekonstrukcija scene iz dvije slike* (završni rad – preddiplomski studij). Elektrotehnički fakultet, Osijek, September 9, 2010.

[2] S. Mattoccia, *Stereo Vision: Algorithms and applications,* University of Bologna, January 12, 2013.

[3] http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf

[4] http://www.zemris.fer.hr/~ssegvic/project/pubs/sambol12ms.pdf

[5] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2003.

[6] http://www.ti.inf.ethz.ch/ew/Lehre/CG13/lecture/Chapter%206.pdf

[7] http://www.learnopencv.com/delaunay-triangulation-and-voronoi-diagram-using-opencv-c-python/

8. Sažetak

U ovom radu opisano je rješenje za izgradnju trodimenzionalnog modela u obliku teksturirane mreže trokuta na temelju dvije slike, tj. na temelju dvodimenzionalnog skupa točaka sparenih i filtriranih pomoću fundamentalne matrice da bi se odbacile pogrešno sparene značajke.Program učitava dvodimenzionalni skup točaka izgeneriranih pomoću programa za trodimenzionalnu rekonstrukciju scene. Na temelju točaka detektiranih na jednoj slici generira se mreža trokuta pomoću Delaunay triangulacije. Grafičko sučelje omogućava ručnu korekciju navedene mreže trokuta.Promjene putem grafičkog sučelja omogućuje OpenCV callback funkcija koja koristi koordinate točke izabrane klikom miša.Program je izrađen u C++ programskom jeziku korištenjem OpenCV biblioteke. Izlazna informacija programa je mreža trokuta dobivena korekcijom Delaunayeve triangulacije.

Ključne riječi: stereo vizija, mreža trokuta, Delaunayeva triangulacija, OpenCV, C++

Abstract

Construction of 3D triangular mesh model based on two images

This work describes a solution for constructing a three-dimensional model of the scene using triangular mesh grid based on a vector of corresponding 2D points.The vector of corresponding points is generated using aprogram for 3D scene reconstruction. The presented program creates a triangular mesh using Delaunay triangulation, where the points obtained by 3D scene reconstruction are triangle vertices.A graphical interface which enables a user to change the aformentioned triangular mesh is created. Editing of the triangular mesh is implemented by a callback function which uses the coordinates of the point chosen by a mouse click.The presented program is created using C++ programming language using OpenCV library. The output of the program is a triangular mesh obtained by correction of the Delaunay triangulation.

Keywords: stereo vision, triangular mesh, Delaunay triangulation, OpenCV, C++

9. Životopis

Valent Brkić je rođen 13.08.1993. u Osijeku. Završio je Osnovnu školu „Retfala“ u Osijeku. Srednješkolsko obrazovanje stekao je u 3. Gimnaziji u Osijeku. Pohađao je i Tamburašku školu Batorek gdje je svirao u glavnom orkestru. Obrazovanje na Fakultetu Elektrotehnike, Računarstva i Informacijskih Tehnologija započeo je 2012. gdje je trenutno student 3. godine preddiplomskog studija računarstva.

Kompetitivno se natječe u šahu u 3. hrvatskoj šahovskoj ligi za ŠK „Mladost“ te obavlja dužnost potpredsjednika u ŠK „Dragovoljac Osijek“ , također igra za šahovsku ekipu fakulteta i za ekipu Sveučilišta u Osijeku.

_____

Vlastoručni potpis

10. Prilozi