Univerzitet u Beogradu
Elektrotehnički fakultet

Predmet:
Mašinska vizija

Projektni zadatak:

Procena deformacija praćenjem speckle obrazca - metoda praćenja trouglova

Mihajlo Karličić 3131/18
km183131m@student.etf.bg.ac.rs

15. juli 2019

Sadržaj

1 Opis projektnog zadatka

Potrebno kamerom je pratiti deformacije na nekom objektu. Te deformacije su sporopromenljive i nalaze se u ravni koju kamera snima.

Površina objekta čija se deformacija posmatra je ofarbana belom bojom i delimično isprskana (našarana) crnim tačkicama. Kamera snima sukcesivne frejmove dok se vrši defomacija. Na osnovu frejmova je potrebno rekonstruisati kretanje pojedinačnih tačaka na površini pa samim tim i formirati model deformacije.

2 Ideja realizacije

Preko slike se prevuče zamišljeni grid horizontalnih i vertikalnih linija koje se za nedeformisan objekat seku pod pravim uglovima. Za svaki presek tih linija, nazovimo ga `Q`, potrebno je naći 3 tačke u okolini i odrediti pondersianu funkciju kojom će se obezbediti da je taj presek u težištu trougla koji čine te tri tačke (koliko je god to moguće obezbediti). Dakle, za svaki presek u gridu je potrebno pamtiti 3 tačke `P1`, `P2` i `P3` kao i dodatne koeficijente ili parametre ponderisane funkcije. Tu funkiju je potrebno osmisliti i definisati.

Kad se završi inicijalizacija kreće praćenje deformacija:

1. Nakon male deformacije snimi se novi frejm slike - mala deformacija znači da se svaka tačka samo malo pomerila (par piksela).

2. Uzme se prethodna slika (za drugi frejm to je prvi frejm tj. onaj na kom je izračunat originalni grid) na kojoj su prisutne sve tačke kao i na trenutnoj i one se porede. Za svaku tačku preseka na gridu `Q` traži se gde su se na novom frejmu pomerile tri tačke `P1`, `P2` i `P3` koje je definišu preko ponderisane funkcije. Za svaku od tih tačaka traži se tačka na novom frejmu koja je najbliža njenim koordinatama na starom frejmu - lokacija te tačke na novom frejmu je njena lokacija nakon deformacije.

3. Kada se nađu sve pomerene koordinate tačaka, „iskrivljeni“ grid se dobija iz novih tačaka `P1`, `P2` i `P3` i parametara funkcije.

4. Na kraju se iscrta „iskrivljeni“ grid i ide se na isti postupak za naredni frejm.

U narednom poglavlju će biti opisana praktična realizacija ove ideje koristeći `OpenCV` biblioteku za `C++` programski jezik. Crvenom bojom će biti naznačena neka mesta u realizaciji koja bi se mogla unaprediti i doraditi.

3 Praktična realizacija

3.1 Učitavanje video snimka i predobrada

Na početku koda se je inicijalizuje `cv::VideoCapture` objekat koji u suštini predstavlja video fajl u kom se nalazi snimak deformacije koja se prati. Takođe, incijalizuje se i `cv::VideoWriter` objekat koji će služiti za upis obrađenih frejmova u izlazni video fajl.

Kad se učita prvi frejm potrebno je nekako razlikovati objekat koji je od interesa od okoline. Postoje razni načini kako je ovo moguće izvesti. Moguće je pretpostaviti da je objekat pravougaonog oblika pa da će ga duge i jasne ivice odvojiti od okoline. Međutim, takve pretpostavke možda nisu opravdane pa je zato ovde odlučeno da se korisniku omogući da na prvom frejmu odabere region od interesa za posmatranje deformacija. To se radi tako što korisnik klikne u gornji levi ugao regiona koji ga zanima, stisne SPACE dugme, klikne na donji desni ugao regiona i opet stisne SPACE.

Osim izbora regiona koji je od interesa, moguće je izabrati i gustinu grida koji se posmatra. To se radi tako što se u kodu promeni vrednost makroa koji definiše kolika će biti jedna stranica kvadrata u gridu u pikselima.

Učitani frejm se iz BGR pretvara u HSV kolor sistem. Razlog za ovo je taj što je objekat od interesa ofarban nekom svetlom bojom a tačkice su neke tamne boje (ili obrnuto). Dakle, dovoljno je posmatrati samo V kanal HSV kolor sistema i mogu se razlikovati tačkice od pozadine.

NAPOMENA: Ovaj deo koda je testiran u uslovima dobrog osvetljenja i lepog kontrasta između pozadine i tačaka. Za robusniju aplikaciju bi trebalo dodatno obraditi sliku (binarizacija, drugi kolor sistem i sl.).

3.2 Računanje početnog grida i crtanje

Za posmatrani region od interesa je potrebno odrediti preseke u gridu. Preko regiona od interesa se prevlači zamišljeni grid i računaju se tačke preseka u tom gridu. U kodu to predstavlja `for` petlju u `for` petlji koje postavljaju jednu tačku preseka na odgovarajućim mestima u zavisnosti od makroa koji definiše gustinu grida.

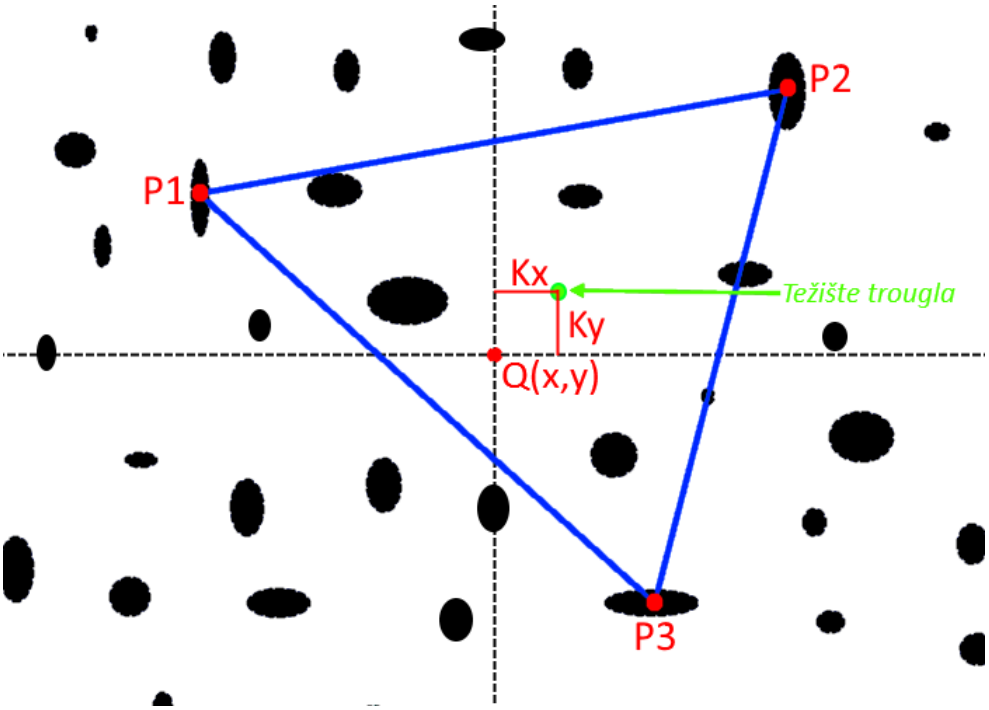
Svaka tačka preseka `Q` u gridu je potrebno da se opiše preko tri tačke iz njene okoline i određenih parametara koji je jednoznačno određuju preko ponderisane funkcije. Ovaj problem nije jednostavno rešiti jer se sve tri tačke mogu proizvoljno pomerati u proizvoljnim smerovima u zavisnosti od vrste deformacije. Način na koji je u ovom slučaju aproksimirano rešenje problema je sledeći:

- pronađu se tri tačke `P1`, `P2` i `P3`
- izračunaju se koordinate težišta trougla čija su temena te tri tačke
- izračuna se udaljenost tačke preseka `Q` od težišta i sačuva se (parametri `Kx` i `Ky`)

U suštini, ta udaljenost koja se čuva predstavlja glavni izvor greške u ovoj aproksimaciji. Sa pomeranjem temena trougla (rotacija i promena površine) se ovi parametri menjaju, međutim to trenutni algoritam ne uračunava i smatra da su parametri konstantni i kao na početnom frejmu.

Očigledno je da se ovaj deo algoritma može unaprediti. Treba prevazići aproksimaciju rešenja i naći ekzaktno rešenje. Zato se može reći da trenutni algoritam dobro radi samo za male deformacije

Da bi ideja bila jasnija, na slici 1 je nacrtan jedan fiktivni presek u gridu sa naznačenim temenima i parametrima. Oni su naznačeni crvenom bojom. Kako se tačka preseka dobija iz temena i parametara opisano je jednačinama 3.1 i 3.2. Za potrebe opisivanja tačaka preseka u gridu, u kodu je definisana klasa `GridPoint` (slika 2).



Slika 1: Tačka preseka sa naznačenim temenima trougla i koeficijentima koji je opisuju

$$Q.x = \frac{P1.x + P2.x + P3.x}{3} + Kx \tag{3.1}$$

$$Q.y = \frac{P1.y + P2.y + P3.y}{3} + Ky \tag{3.2}$$

```
class GridPoint
{
public:
    Point2i Q;                          // location of grid intersection
    vector<KeyPoint> triangle;          // three points of triangle in whose 'centroid' the intersection lies
    int kx;                             // X axis offset from triangle centroid
    int ky;                             // Y axis offset from triangle centroid

    GridPoint(Point2i O = Point2i(0, 0), KeyPoint A = KeyPoint(), KeyPoint B = KeyPoint(), KeyPoint C = KeyPoint(), int Kx = 1, int Ky = 1);
    GridPoint(const GridPoint&);
    ~GridPoint();
};
```

Slika 2: Polja i metode u klasi `GridPoint`

Egzaktan način kako se obavlja ovaj korak je sledeći:

- definiše se objekat klase `cv::SimpleBlobDetector` i podesi se koliki blobovi su od interesa

- u neposrednoj okolini preseka u gridu (30ak piksela) se detektuju blobovi i kao rezultat dobijaju se `cv::KeyPoint`-i
- (opciono) ukoliko u okolini nema dovoljno `KeyPoint`-a (bar 10) povećava se posmatrana okolina pa se opet vrši detekcija, sve dok ne bude dovoljno `KeyPoint`-a
- iterira se kroz sve `KeyPoint`-e u grupama od po 3 i traži se trougao čije težište najmanje odstupa od tačke preseka u gridu
- kad se nađe takav trougao čuvaju se njegova temena i udaljenost tačke preseka `Q` od njegovog težišta kroz parametre `Kx` i `Ky`.

Kad su sve tačke preseka u gridu opisane na ovaj način tj. kad se popune sva polja u klasi, moguće je iscrtati prvi, nedeformisani grid. Pozivom funkcije `gridDraw()` se povežu tačke preseka crnim linijama.

3.3 Praćenje deformacija

Učitava se naredni frejm, pretvara se u HSV kolor sistem i posmatra se samo V kanal. Potrebnu obradu u ovom trenutku obavlja funkcija `findNewGrid()`. Ova funckija kao argumente prima grid u prethodnom frejmu a kao rezultat daje poziciju tačaka preseka u gridu u novom frejmu.

Za svaku tačku preseka `Q` u gridu iterira se kroz tri temena trougla koji ga opisuju i traži se njihova nova lokacija na novom frejmu. Za svako teme se polazi od njegove neposredne okoline (3px × 3px) i traži se blob tj. `KeyPoint`. Ukoliko se ne pronađe, povećava se okolina i traži se opet. Ideja je da će najbliži `KeyPoint` u novom frejmu biti upravo pomereno teme trougla iz prethodnog frejma.

Kad se nađu temena, iz 3.1 i 3.2 se nalazi nova pozicija preseka u gridu. Kao što je ranije naglašeno, parametri `Kx` i `Ky` su izračunati za početni frejm i dalje se tako koriste.

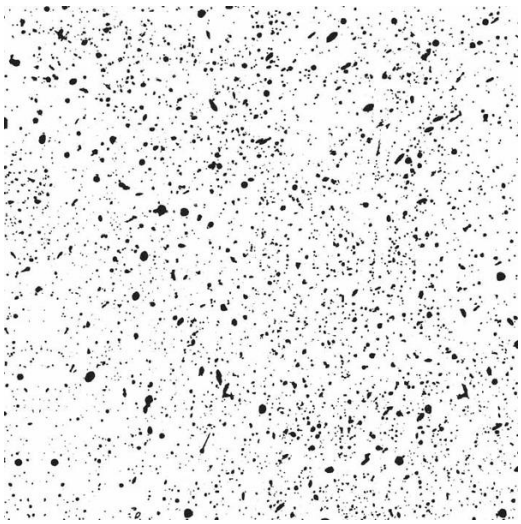
Sve što je sad potrebno je da se spoje nove lokacije preseka u gridu linijama i frejm upiše u izlazni video fajl.

4 Rezultati

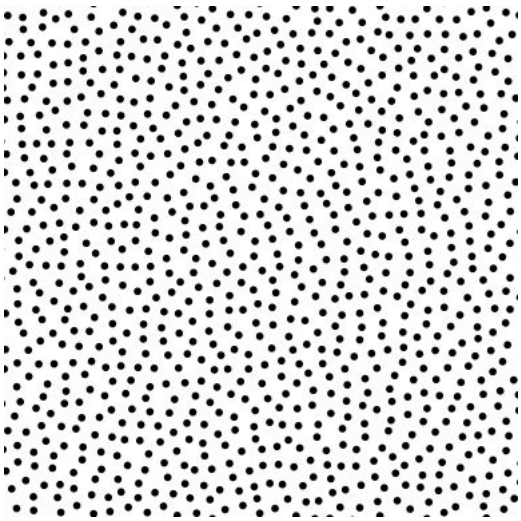
Napisani program je testiran u nekoliko različitih slućajeva:

- Ako se koristi *speckle pattern* koji se dobija prskanjem tamne farbe na svetlu pozadinu (slika 3), rezultati su na granici prihvatljivog. Dešava se da grid ,,poskakuje“ jer praćenje novih lokacija temena između frejmova ne radi na idealan način. To se moglo i unapred pretpostaviti. Dešava se da kad se posmatra okolina nekog temena na novom frejmu, u istoj okolini oko temena bude više `KeyPoint`-a i program ne zna koji `KeyPoint` je onaj od interesa. Uprkos tome, iz izlaznog snimka se može prepoznati o kakvoj se deformaciji otprilike radi. Greška koja nastaje zbog aproksimacije ponderisane funkcije su prisutne.

- Ukoliko se koristi *pattern* koji se sastoji od manje većih tačaka približnih veličina (slika 4), rezultati su dosta bolji. Nema poskakivanja kao u prethodnom slučaju ali greška usled aproksimacije ponderisane funkcije i dalje postoji.



Slika 3: *Speckle pattern* nastao prskanjem



Slika 4: *Speckle pattern* nastao tačkanjem

Plan je u da se u narednom periodu snimi video snimak sa detaljnijom demonstracijom rezultata.

5 Zaključak

Ovaj projekat predstavlja prostu demonstraciju ideje koja bi mogla biti od koristi kao jednostavnija alternativa već postojećim *DIC* algoritmima. Ideja ima svoje nedostatke koji bi se možda mogli prevazići kvalitetnijim kamerama koje mogu da snimaju ulazni video sa dosta većim vrednostima *FPS*-a. Takođe, u praktičnoj realizaciji ima mesta za unapređenja, ne samo na mestima koja su pomenuta u izveštaju. Realizacija radi u idealnim slučajevima i u kodu je *error checking* minimalan.