SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 51

Kalibracija vanjskih parametara kamera numeričkom optimizacijom

Vladimir Perković

Zagreb, lipanj 2010.

Sadržaj

Uvod

Upravljanje vozilom ma koliko uobičajen postupak i dalje uzrokuje velike probleme kao što su ljudske i materijalne štete. Sa ciljem olakšavanja vožnje sve veći broj automobila se opskrbljuje sa pomoćnim sustavima koji čine kontrolu vozila jednostavnijom i ugodnijom. Takvi sustavi uključuju zvučna ili vizualna upozorenja kod približavanja objektu pri vožnji, automatsku kontrola brzine, senzore za parkiranje i sl.

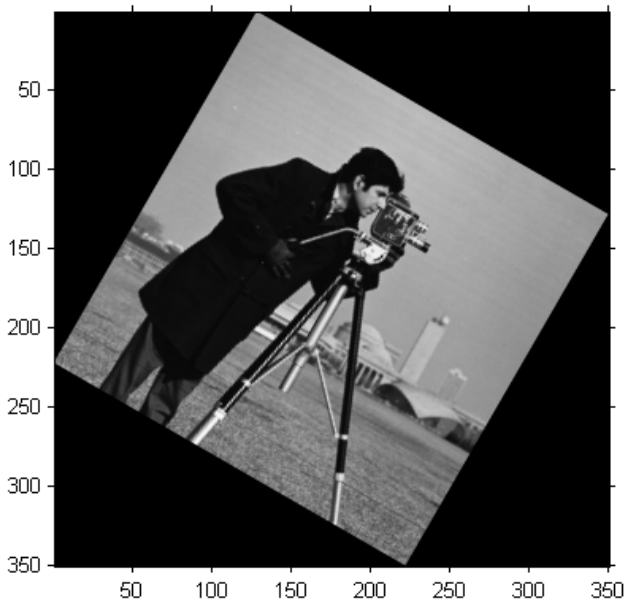
Ovaj rad se bavi jednim takvim sustavom koji bi omogućio vozaču da tijekom vožnje ima dojam pogleda odozgo. Time bi dobio pregledniju vožnju i lakše parkiranje. Kako bi se postigao takav pogled, potrebne su četiri kamere sa svake strane automobila. Svaku od slika potrebno je projicirati te “zašiti” kako bi se dobila slika koja daje pogled od 360°. Kamere su širokokutne kako bi osiguravale međusobni preklop slika koji se koriste za šivanje. Svaka kameru potrebno je kalibrirati kako bi preklop slika bio što bolji. U slučaju pomaka kamere potrebno je ponovno softverski pronaći najbolji preklop.

Geometrijske transformacije

Izometrija

Izometrija se sastoji od rotacije i translacije te čuva dužine stranica, kuteve i površine između njih.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & t_x \\ \sin(\phi) & \cos(\phi) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} x$$

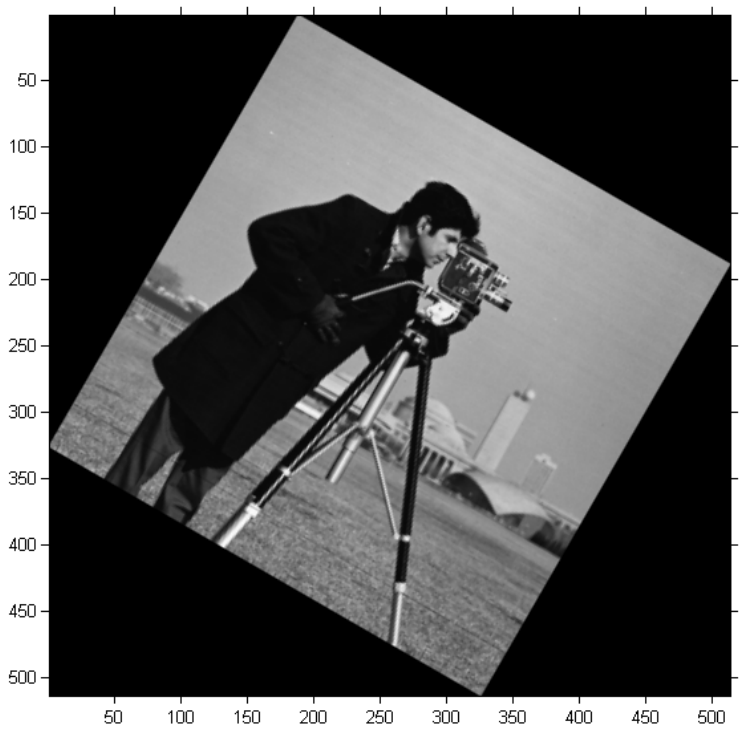


Slika 1 – primjer izometrije

Sličnost

Sličnost se sastoji od skaliranja, rotacije i translacije, a zadržava omjere dužina i kuteve.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{s} \cdot \cos(\phi) & -\mathrm{s} \cdot \sin(\phi) & t_x \\ \mathrm{s} \cdot \sin(\phi) & \mathrm{s} \cdot \cos(\phi) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{s}R & t \\ 0 & 1 \end{bmatrix} x$$

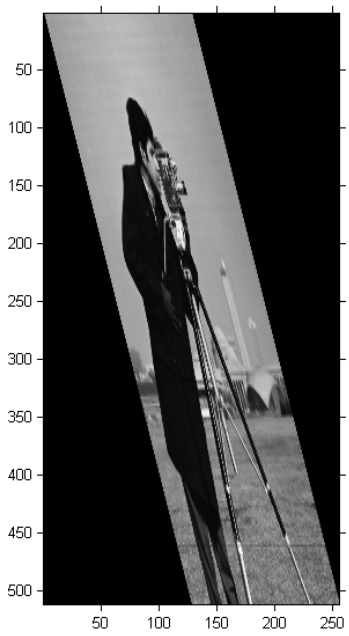


Slika 2 - Primjer sličnosti

Afina transformacija

Afina transformacija se sastoji od rotacije, skaliranja, translacije te zakošenja. Čuva paralelnost, omjere površina te omjeri dužina u paralelnim smjerovima.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & t_x \\ \mathrm{a}_{21} & \mathrm{a}_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} x$$



Slika 3 - Primjer afine transformacije

Projekcijska transformacija

Projekcijska transformacija (*eng. homography, collineation, projective transformation*) je inverzna transformacija koja opisuje što se događa sa promatranim objektima kada se promijeni točka gledanja. Projekcijska transformacija čuva kolinearnost i križne omjere četvorke točaka.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \approx \begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & t_x \\ \mathrm{a}_{21} & \mathrm{a}_{22} & t_y \\ v_1 & v_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \approx \begin{bmatrix} A & t \\ v^T & 1 \end{bmatrix} x \approx Hx$$

Određivanje 2D homografske matrice

Pretpostavimo da gledamo ravnu površinu i promatramo je sa kamerom kojoj pripada projekcijska matrica P. Ravna površina je 2D tako da joj se može pridodati koordinatni sustav (s,t). Točke na površini su u 3D svijetu te se njihove 3D pozicije mogu izraziti u XYZ koordinama kamere pomoću jednadžbe:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & b_x & X_0 \\ a_y & b_y & Y_0 \\ a_z & b_z & Z_0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}$$

Prva dva stupca se mogu protumačiti kao smjerovi vektora koji odgovaraju koordinatama jediničnih vektora sustava, odnosno preslikavanju (1,0,0) i (0,1,0). Treći stupac odgovara 3D poziciji središta ravne površine, (s,t)=(0,0). Tako se središte (s,t)=(0,0) preslikava u (X_0,Y_0,Z_0) , kut (s,t)=(0,1) u $(X_0+b_x,Y_0+b_y,Z_0+b_z)$, (s,t)=(1,0) u $(X_0+a_x,Y_0+a_y,Z_0+a_z)$ itd.

Piksel (x,y) koji odgovara točkama (s,t) računa se:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Ove dvije transformacije zajedno definiraju 3x3 transformaciju matrice H iz (s,t,1) u (wx,wy,w).

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = H \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}$$

Matrica H se naziva matrica homografije.

Inverz je definiran idućom jednadžbom:

$$\begin{bmatrix} w's \\ w't \\ w' \end{bmatrix} = H^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Parametri kamere

Svaka kamera je određenja unutarnjim i vanjskim parametrima. Unutarnji parametri predstavljeni su matricom K, gdje α_x i α_y označavanju žarišnu udaljenost u pikselima, u_0 i v_0 središnju točku slike, također u pikselima. Zakošenje je označeno sa s (eng. *skew*) koje je različito od nule samo u slučaju da u_0 i v_0 nisu okomiti što je iznimno rijetko u modernim kamerama.

$$K = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Vanjski parametri određeni su matricom M veličine, sastavljenom od matrice rotacije R te matrice C. Matrica R određuje rotaciju u stvarnom svijetu, a matrica C položaje koordinata centra u stvarnom svijetu.

$$M = [R \quad C]$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Matrica M odgovara euklidskoj transformaciji iz koordinata svijeta u kamerin koordinatni sustav.[4] 3D točka predstavljena sa vektorom M_w u koordinatama svijeta bit će predstavljena sa vetktorom

$$M_C = RM_w + C$$

u koordinatnom sustavu kamere. Iz ovog odnosa lako se dobiva izraz za centar kamere. Potretbno je zadovoljiti jednadžbu

$$0 = RC + T$$

što daje

$$T = -R * C = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Konačno, matrica H se dobiva

$$H = K * \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

Pretvaranje matrice rotacije u Eulerove kutove

Za potrebe optimizacije, matrice rotacije, radi jednostavnosti treba pretvoriti u kuteve. Matrica rotacije sastavljena je od tri matrice koje predstavljaju rotaciju u x, y i z osi.

Rotacija za ψ radijana oko osi x je definirana kao:

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}$$

Rotacija za θ oko osi y:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Rotacija za φ oko osi z:

$$R_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Metoda najbržeg spusta

Metoda najbržeg spusta je najjednostavnije od svih metoda optimizacija. Kretanje je smjeru tamo gdje se funkcija F(x) najbrže smanjuje, što je suprotan smjer od $\nabla f(x_i)$, odnosno u smjeru negativnog gradijenta. Pretraga kreće od točke x_0 koja se odabire nasumično ili ako postoji neko prethodno znanje o problemu tako da smanjimo trajanje pretrage.

Jednadžba sljedeće promatrane vrijednosti iznosi:

$$x_{k+1} = x_k - \lambda_k \cdot \nabla f(x_k) = x_k - \lambda_k \cdot g(x_k)$$

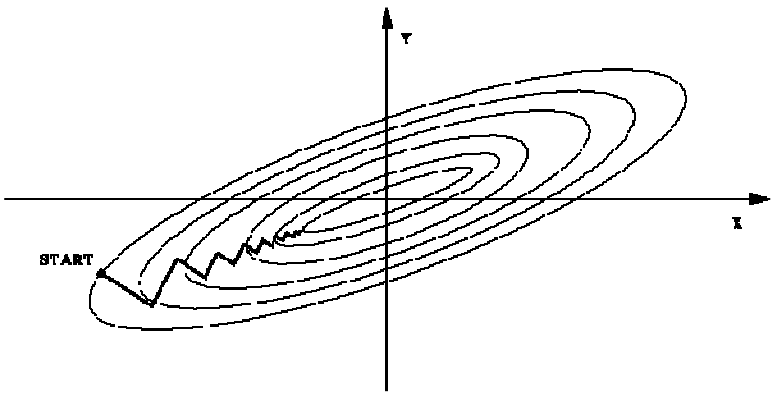
gdje je $g(x_k)$ gradijent u zadanoj točci.

Potrebno je odrediti vrijednost koraka λ_k. Najbolji smjer kretanja je očito tamo gdje funkcija f poprima najmanju vrijednost, dakle gdje je usmjerena derivacija jednaka nuli.

Jednadžba usmjerene derivacije iznosi

$$\frac{d}{d\lambda_k} f(x_{k+1}) = \nabla f(x_{k+1})^T \cdot \frac{d}{d\lambda_k} x_{k+1} = -\nabla f(x_{k+1})^T \cdot g(x_k)$$

Izjednačavajući jednadžbu s nulom, odabiremo λ_k takav da su $\nabla f(x_{k+1})$ i $g(x_k)$ međusobno ortogonalni. Rezultat toga je cik – cak putanja prikazana na slici 4.



Slika 4 - Konvergencija najbržeg spusta [5]

Iteracija se nastavlja sve dok minimum nije pronađen sa definiranom preciznošću.

Najbrži spust je jednostavna, ali prilično spora metoda. Problemi sa najbržim spustom se javljaju kada postoji puno lokalnih minimuma gdje može konvergirati te tako zaobići globalni minimum.

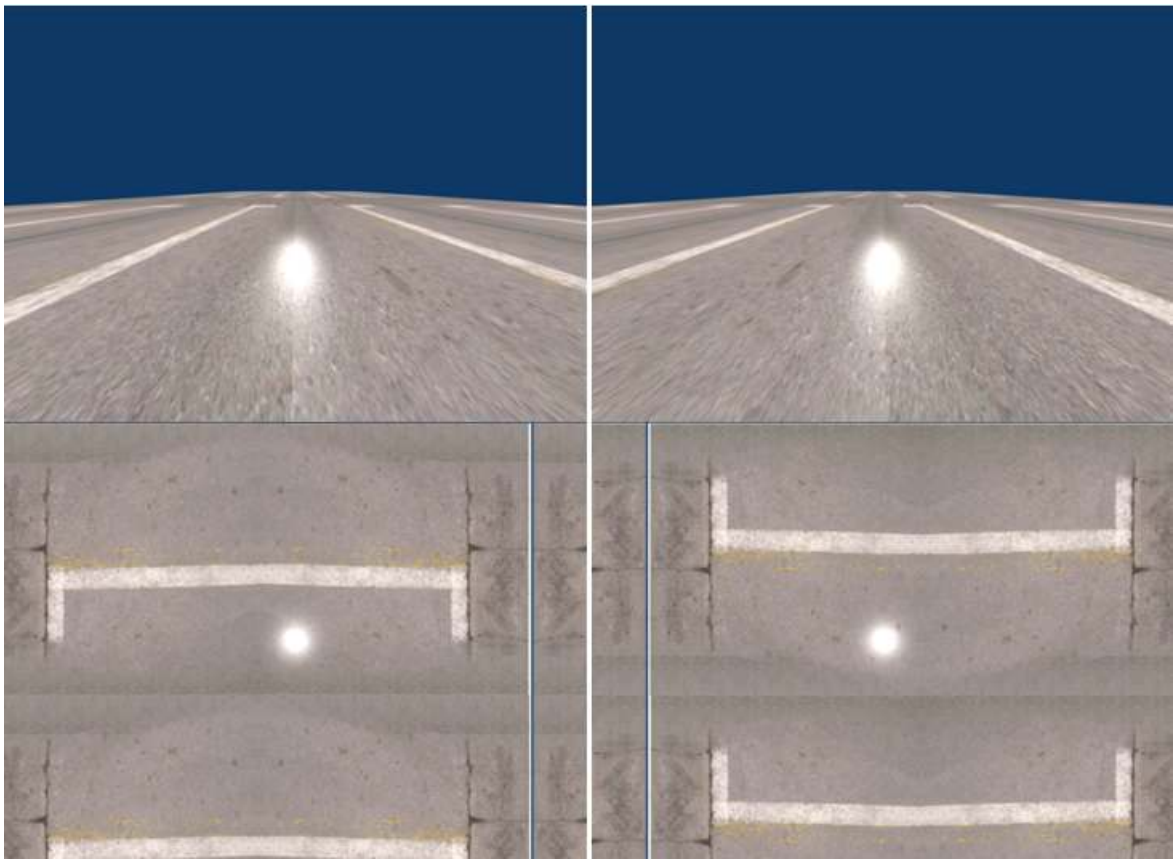
Praktični rad

Programska podrška

Programska podrška napisana je u C/C++ programskom jeziku koristeći biblioteku OpenCV. OpenCV je biblioteka orijentirana na računalni vid originalno razvijena od strane Intela. Korištenje je dozvoljeno pod open source BSD licencom.

Uzorci

Testiranje se vrši sa četiri sintetske slike koje predstavljaju parkiralište. Na slici 5 su testni uzorci generirani sa kalibriranim kamerama.

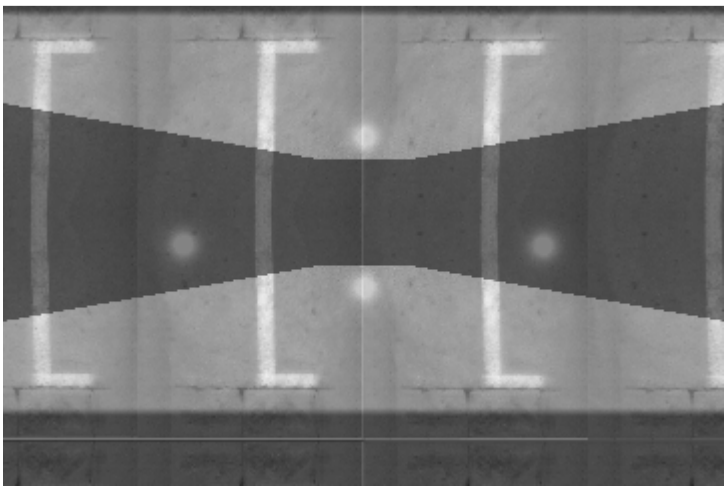


Slika 5 - Testni uzorci, redom: prednji, stražnji, lijevi, desni

Na slici 5 su prikazane projicirane slike iz sve četiri kamere. Slike su međusobno zbrojene te je zato intenzitet kod preklopa veći nego na ostatku slike.

Početna ideja je bila pretraživati najbolji preklop pomoću najbržeg spusta koristeći samo kuteve, a zanemarujući pomake. Metoda najbržeg spusta može vrlo lako završiti u lokalnom minimumu te nije pogodna kada bi se koristila sa 12 varijabli (4 para slika puta 3 kuta). Zbog toga je optimizacija obavljena pomoću iscrpnog pretraživanja za svaka dva para slika te traženja najboljeg globalnog preklopa tj. najbolje sume preklopa što će biti objašnjeno kasnije.

Testiranje će se vršiti na nekoliko načina. U prvom slučaju najbolji kut će se pretraživati samo jednom te će se gledati najbolji globalni preklop koristeći ispravne pomake. Drugi slučaj je identičan prvome, ali će pomaci biti zanemareni, odnosno koristit će se originalni pomaci kalibriranih kamera. Treći puta će se koristiti ispravni pomaci, ali će se više puta pretraživati za točnijim kutovima. Točnije, nakon prve pretrage tražit će se globalno najbolji preklop, a nakon toga će se tražiti kutevi koji će biti bliže rješenju.



Slika 6 - Projicirane i zbrojene slike iz sve četiri kamere

Iscrpno pretraživanje

Pretraživanje se vrši uspoređivanjem dvije susjedne slike (prednja i lijeva, lijeva i zadnja itd.). Svaka matrica sastoji se od matrice rotacije R koja se može prebaciti u tri kuta rotacije to x, y i z osi (*eng. jaw, roll, pitch*). Pomak kamere se može dogoditi u bilo kojem kutu što znači da za dvije slike postoji šest varijabli po kojima treba tražiti preklop. Trajanje jednog projiciranja dvije slike te međusobno oduzimanje traje oko 25 ms, dakle ako se uzme pretraživanje pet kuteva svake osi to iznosi 5^6=15625*25ms, oko 400 sekundi za jedno pretraživanje.

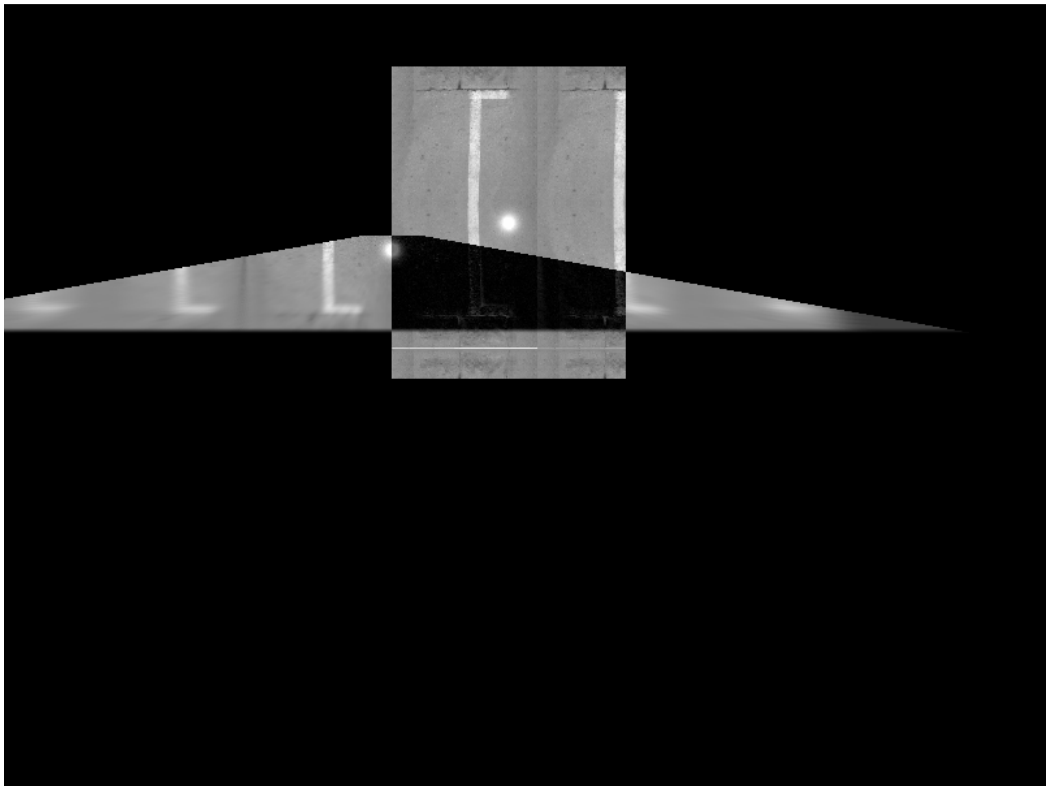
Kriteriji najboljeg preklopa

Suma apsolutne razlike preklopa

Kao prvi kriterij uzeta je suma apsolutne razlike dviju slika. Sumira se samo područje preklapanja slika.

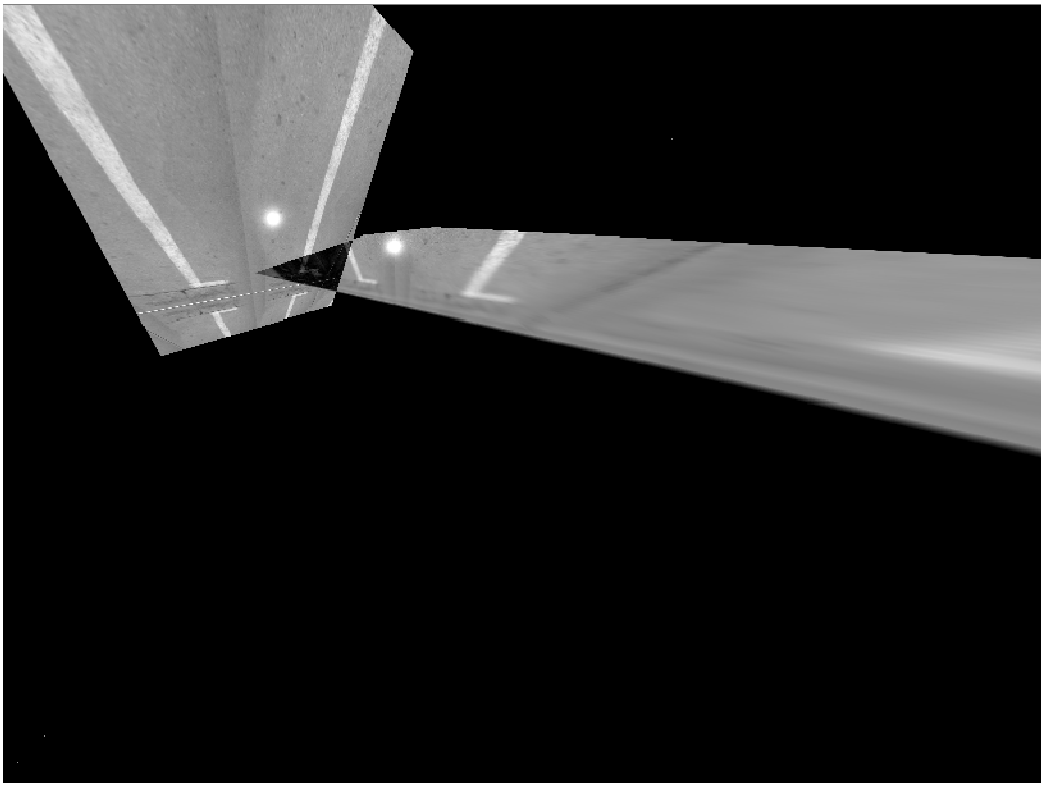
$$\sum_{i,j=0}^{w,h} |A - B|$$

Primjer dobrog preklapanja prikazan je na slici 6.



Slika 7 - Primjer dobrog preklopa

Problem sa ovakvim kriterijem je što su testni uzorci (slika parkirališta) najvećim dijelom jednake sive boje sa malo detalja osim bijelih linija prikazanih na slici. Slika 7 pokazuje najbolji slučaj preklapanja dobiven iscrpnim pretraživanjem. Vidljivo je da je ovaj slučaj dobiven kao najbolji isključivo radi male površine preklapanja. Zbog toga se postojeća formula normira sa površinom.



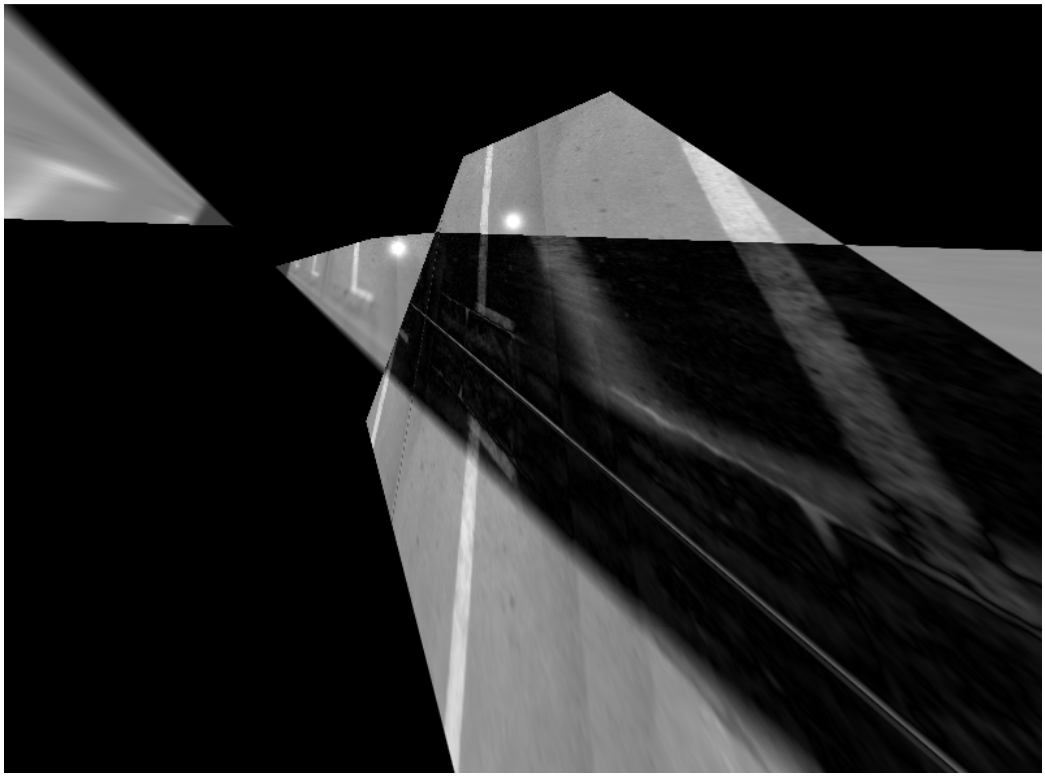
Slika 8 - Primjer lošeg preklopa

Omjer površine preklapanja i sume apsolutne razlike

$$\frac{}{}$$

P – površina preklapanja

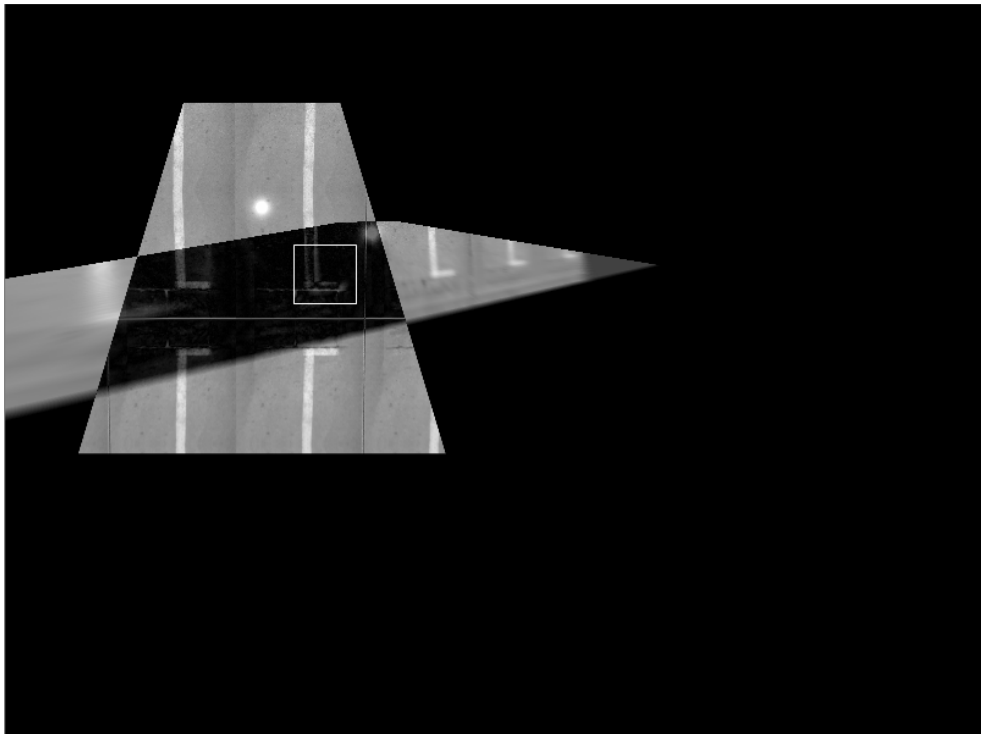
Nažalost, dijeljenjem sa površinom rješava se jedan problem (preklapanje samo crnih dijelova), ali se pojavljuje drugi problem prikazan na slici 9. Vidljivo je da su slike slabo preklopljene, ali je to najbolji koeficijent zbog velike površine koju zauzima preklop.



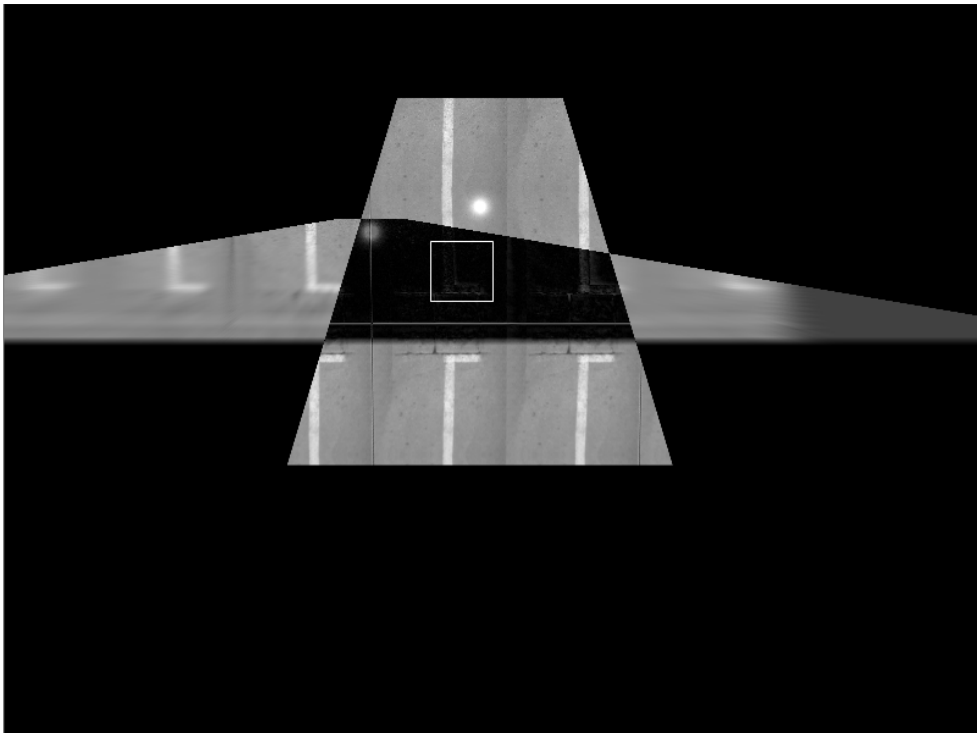
Slika 9 - Primjer lažno dobrog preklopa

Suma apsolutne razlike fiksne veličine

Posljednji kriterij korišten se pokazao dovoljno dobrim za pronalaženje odgovarajuće projekcije slika. Koristi se suma apsolutne razlike slika u fiksnom prozoru. Najbolji rezultati obično rezultiraju dobrim preklopom (malom sumom), ali pod nekim kutem ako se koriste originalni pomaci (slika 10) ili odmah rezultiraju željenom projekcijom (slika 11) kada se koriste točni pomaci. Problem slika pod kutem se rješava pretragom sve četiri slike detaljno objašnjenom u idućem poglavlju.



Slika 10 - Slika dobrog preklopa pod kutem dobivena koristeći fiksni prozor



Slika 11 - Slika dobro preklopa korištenjem fiksnog prozora i točnih pomaka

Pretraga pomoću sve četiri slike

Budući da traženjem preklopa dvije slike često nije uspješno pronađena željena projekcija, sljedeći postupak se sastoji od traženja apsolutne razlike korištenjem fiksog prozora za sve četiri slike.

Algoritam:

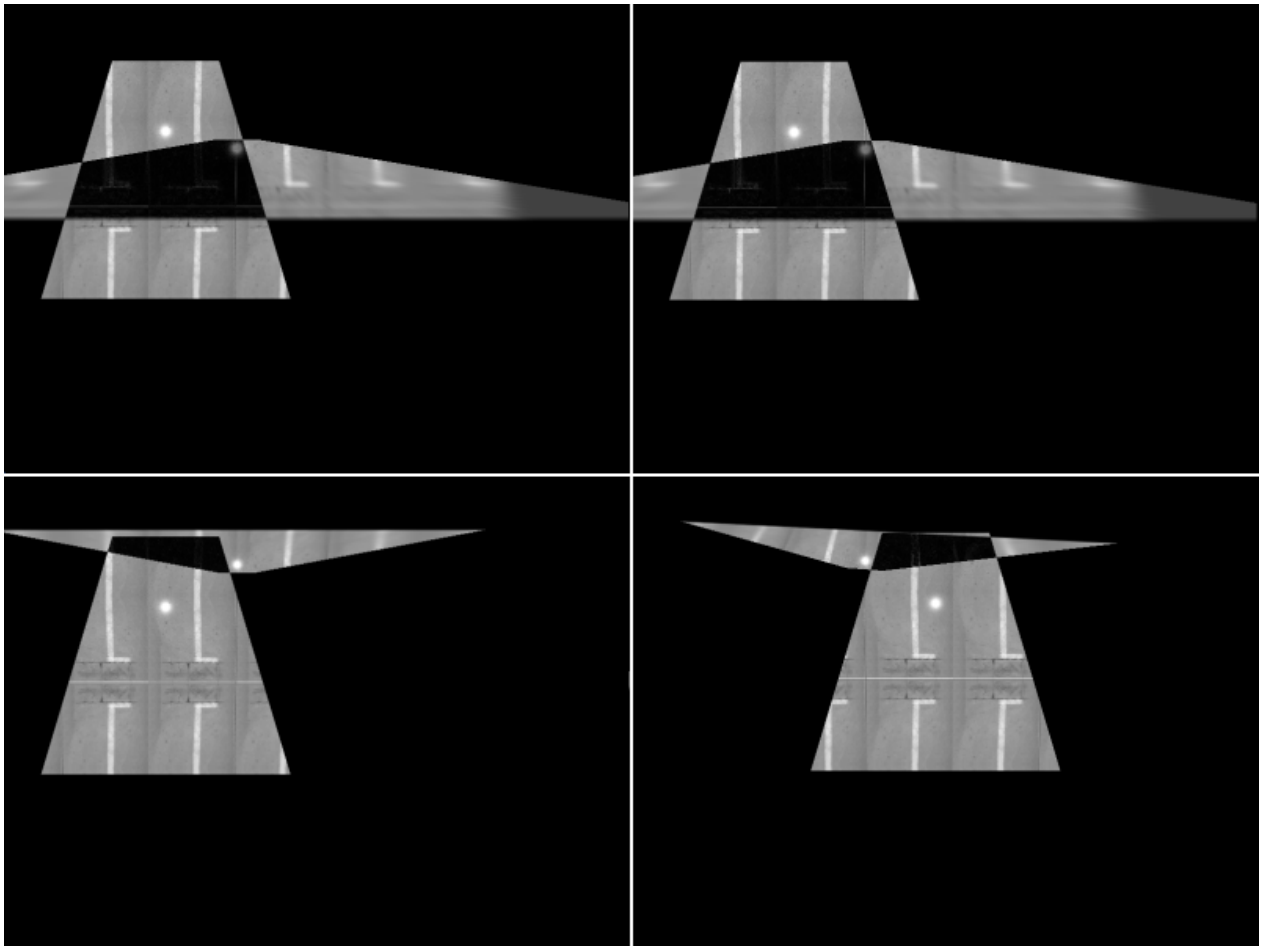
1. Pronađi razlike za sva četiri para slika (prednja – lijeva, lijeva – zadnja, zadnja – desna, desna – prednja) te sačuvaj sve vrijednosti u četiri polja
2. Kreni od najbolje (najmanje) razlike za jedan par npr. prednja – lijeva (nije potrebno pretraživanje svih kombinacija četiri para slika, jer susjedni parovi preklopa moraju imati jednake kutove)
3. Pronađi odgovarajuće parove (sa istim kutevima) susjednih slika (npr. lijeva slika mora biti pod istim kutevima kod parova prednja – lijeva i lijeva – zadnja)
4. Pronađi posljednji odgovarajući par (zadnja – desna)
5. Odredi sumu sva četiri para te zabilježi rezultat u novo polje
6. Ako su prođene sve kombinacije zaustavi algoritam
7. Inače trenutno promatrani član para prednja – lijeva postavi na nulu i idi na korak 3

Pretpostavka koja stoji iza ovog postupka je da će se ostvariti bolje željeno preklapanje jer se lažno pozitivni rezultati mogu dogoditi za tri slike, ali se onda četvrta “raspada” kao na slici 12. Dakle, omjer četvrte stranice je nizak u odnosu na omjere preostale tri.

Slika 12 – Slika “raspada”

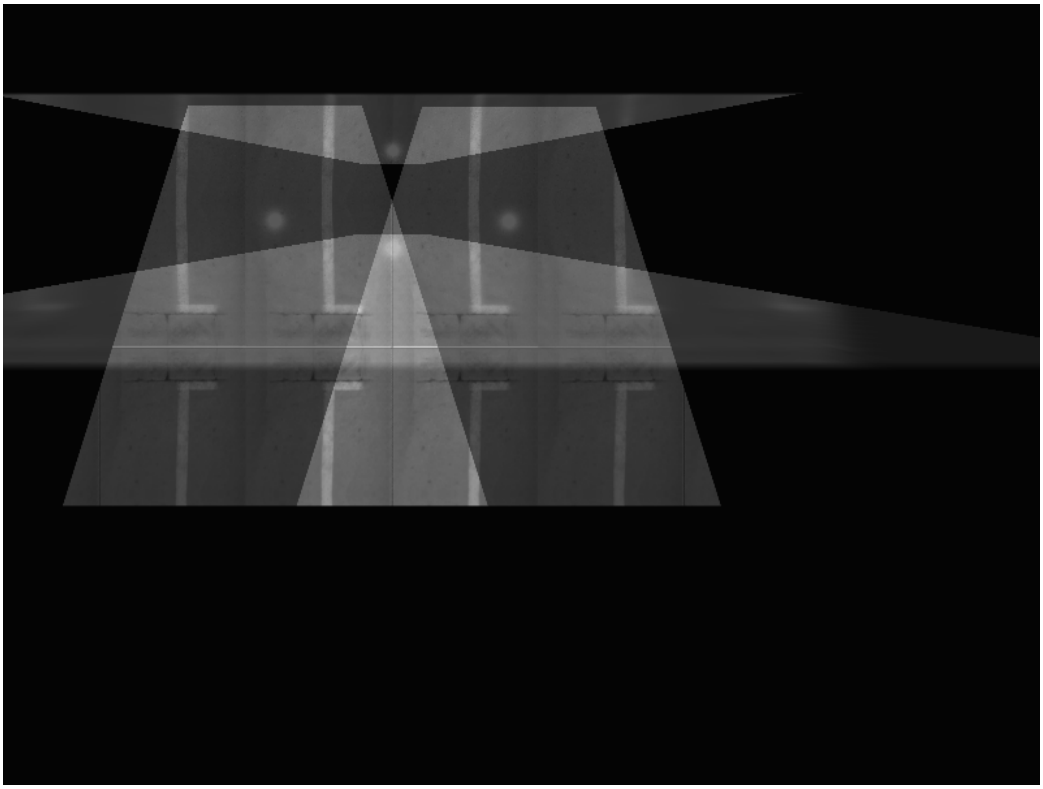
Testiranje

Korištenje točnih pomaka



Slika 13 - Rezultati korištenjem točnih pomaka (preklopi redom: prednja – lijeva, prednja – desna, zadnja – desna, zadnja – lijeva)

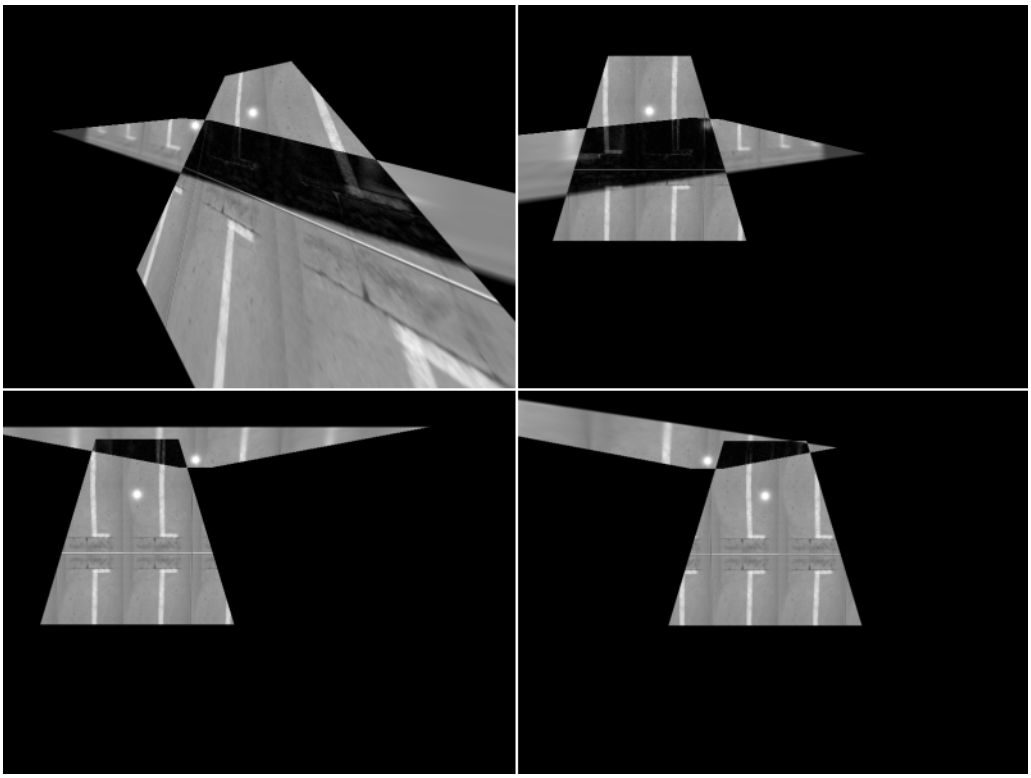
Na slici 13 vidljivi su rezultati korišenja točnih pomaka za skupinu uzoraka 1 koji su imali najmanju apsolutnu razliku u fiksnom prozoru. Nisu svi preklopi točni, jer se na zadnjoj slici, projicirana slika stražnje kamere (gornja) nalazi pod kutem. Taj problem se u potpunosti rješava usporedbom sve četiri slike te je rezultat prikazan na slici 14.



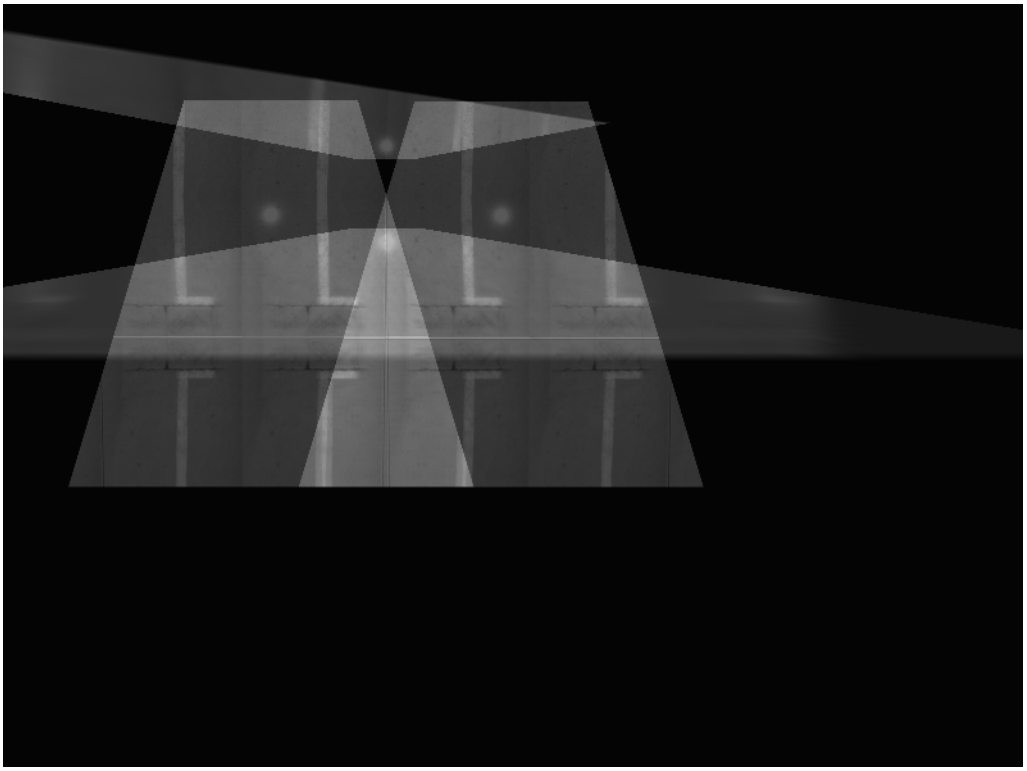
Slika 14 - Sumirani rezultati dobiveni pretragom sva četiri para preklopa

Korištenje originalnih pomaka

Korištenjem originalnih pomaka dobivaju se malo lošiji rezultati (slika 15), odnosno zbog netočnih pomaka najbolje rješenje se nalazi pod nekim kutem. Također, taj problem se relativno dobro rješava korištenjem sve četiri slike te je na slici 15 vidljivo da je samo jedna stranica netočno postavljena.



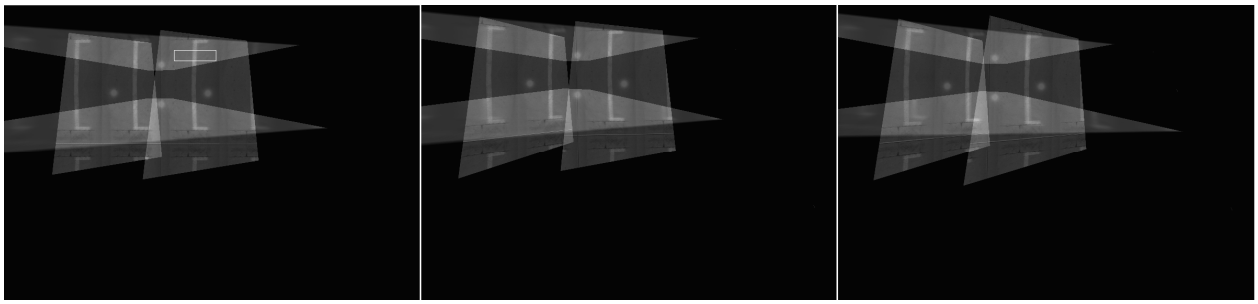
Slika 15 - Preklopi parova slika korištenjem originalnih pomaka kamere



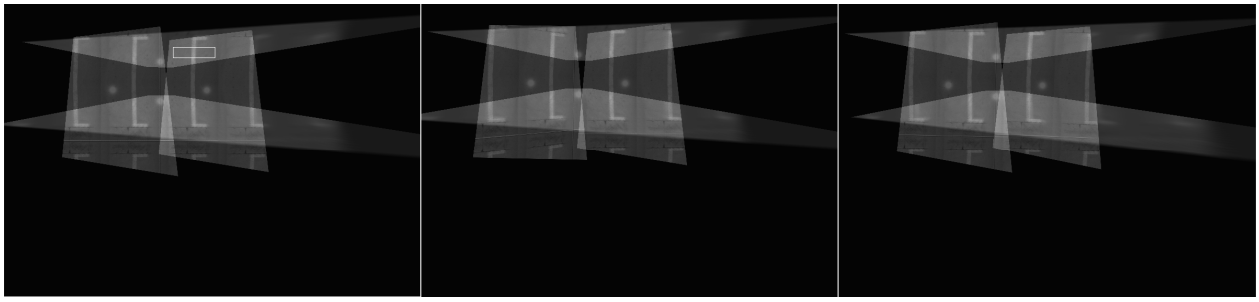
Slika 16 - Rezultat kombiniranja sve četiri slike

Tablica 1 – Prikaz rezultata

Skupina 1	Prednja - lijeva	Prednja - desna	Zadnja - Lijeva	Zadnja - Desna	Nakon korištenja sve 4
Točni pomaci	Točno	Točno	Pod kutem	Točno	4/4
Originalni (netočni) pomaci	Pod kutem	Pod kutem	Pod kutem	Pod kutem	3/4
Skupina 2					
Točni pomaci	Pod kutem	Pod kutem	Pod kutem	Pod kutem	3/4
Originalni (netočni) pomaci	Pod kutem	Pod kutem	Pod kutem	Pod kutem	0/4
Skupina 3					
Točni pomaci	Pod kutem	Pod kutem	Pod kutem	Pod kutem	3/4
Originalni (netočni) pomaci	Pod kutem	Pod kutem	Pod kutem	Pod kutem	1/4



Slika 17 - skupina 2, redom: točno rješenje, točni pomaci, originalni pomaci



Slika 18 - skupina 3, redom: točno rješenje, točni pomaci, originalni pomaci

Zaključak

Ideja aplikacije je dati vozaču što bolji pregled kao pomoću u vožnji. Dakle, zahtjev je da rezultati budu vizualno prihvatljivi čovjeku. Sa slika 17 i 18 je vidljivo da se čak i kada slike nisu idealno preklopljene dobiva slika jako slična točnom rješenju.

Vrijeme realizacije je oko 27 minuta za pregled sva četiri para stranica te dodatnih 60 sekundi za međusobni pregled svih preklopa. Za svaki dodatni stupanj preciznosti treba dodati još 27 minuta pa se sa sigurnošću može reći da ova aplikacija ne može raditi u realnom vremenu. Praktično bi se mogla koristiti kod kalibracije u proizvodnom procesu, ali ne u slučaju da kamere promijene položaj ili rotaciju u vožnji. Očigledno, rezultati su bili bolji kada su korišteni točni pomaci što bi zahtjevalo dodatno vrijeme pretrage te se taj dio ne bi mogao ostvariti iscrpnim pretraživanjem. U nastavku bi se trebalo razmotriti korištenje drugih, bržih metoda optimizacije.

Literatura

[1] Michael Langer, Homography, 5. studeni 2009., *Fundamentals of Computer Vision – Lecture 19, Homography,* http://www.cim.mcgill.ca/~langer/558/lecture19.pdf , travanj 2010.

[2] Gregory G. Slabaugh, Computing Euler angles from a rotation matrix, http://www.gregslabaugh.name/publications/euler.pdf , lipanj 2010.

[3] Intel Corporation, Open Source Computer Vision Library - Reference Manual, Intel Corporation 2001. (digitalna verzija)

[4] Lepetit V., Pascal F., Monocular Model-Based 3D Tracking of Rigid Objects: A Survey, Foundations and Trends in Computer Graphics and Vision, Vol. 1, No 1 (2005) 1–89

[5] Trond Hjorteland, Method of Steepest Descent, 5. lipnja 1999., *Method of Steepest Descent,* http://trond.hjorteland.com/thesis/node26.html , svibanj 2010.

[6] Shil Z.J., Shen J., Step-size Estimation for Unconstrained Optimization Methods, Computational & Applied Mathematics, vol.24 studeni/prosinac 2005.

Naslov, sažetak i ključne riječi

Naslov: Kalibracija vanjskih parametara kamera numeričkom optimizacijom

Sažetak: Tema proučavanja ovog rada je dobivanje najbolje moguće slike šivanjem četiri slike dobivene kamerama smještenim na svakoj strani automobila. Proučeni su kriteriji najboljeg preklopa te metode optimizacije slika iz susjednih kamera te sve četiri slike. Kao kriterij najboljeg preklopa je uzeta suma apsolutne razlike na fiksnom otvoru, a za metodu optimizacije iscrpno pretraživanje. Razvijena aplikacija s obzirom na vrijeme izvođenja nije pogodna za upotrebu u stvarnom vremenu, ali bi se mogla koristiti za tvorničku kalibraciju kamera.

Ključne riječi: kalibracija kamere, vanjski parametri kamere, metoda najbržeg spusta, optimizacija, pogled iz više kamera

Title, Abstract, Key words

Title: Calibration of extrinsic camera parameters using numerical optimization

Abstract: The theme of this research work is getting the best possible picture by stiching four images obtained by cameras positioned on each side of the car. It studies criteria for best image overlap (stitching) and optimization of images from adjacent cameras, and all four images. As a criterion for the best overlap, the sum of absolute differences on a fixed aperture is chosen. And exhaustive search is used as metod for optimization. Developed applications with respect to run time is not suitable for use in real time, but could be used for factory calibration of cameras.

Key Words: camera calibration, extrinsic camera parameters, steepest descent, optimization, multiple cameras view

Privitak

Funkcija za projiciranje matrica

```
IplImage *projMat2(char naziv[], IplImage **src, CvMat *ulazna, int w, int
h,double kutevi[3],double pomaci[2],char t_parametri[]){
      char *inname = naziv;
      char *inname2 = "H:\\zavrsni\\zavrsni\\images\\K.txt";
      char *inname_param = t_parametri;


      ifstream infile(inname);
      ifstream infile2(inname2);
      ifstream infile3(inname_param);

    if ((!infile) || (!infile2)) {
        cout << "There was a problem opening file "
             << inname <<" or "<<inname2
             << " for reading."
             << endl;
        return 0;
    }

      double temp_ul[2]={cvmGet(ulazna,0,0),cvmGet(ulazna,1,0)};

      /*******************spremi sve u matrice******************/
      CvMat *R=cvCreateMat(3,3, CV_64FC1);
      CvMat *T=cvCreateMat(3,3, CV_64FC1);
      CvMat *F = cvCreateMat(4,3,CV_64FC1);
      CvMat *CF=cvCreateMat(3,1,CV_64FC1);
      CvMat *C=cvCreateMat(3,1,CV_64FC1);
      CvMat *K=cvCreateMat(3,3,CV_64FC1);
      CvMat *MG=cvCreateMat(3,3,CV_64FC1);

      string str ("txt");
      int pok=str.find(t_parametri);

      for (int i=0; i<=11; i++){
            double temp,temp2;

            infile >> temp;
            infile3 >> temp2;
            cvmSet (F, i/3, i%3, (double) temp);
            if (i<9){
                  cvmSet(R,i/3,i%3, (double) temp);
            }
            else{

                  if (pok==NULL){
                        cvmSet(CF,i-9,0, (double) temp);
                  }
                  else {
                        cvmSet(CF,i-9,0, (double) temp2);
```

```
                }
            }
        }

        for (int i=0; i<=8; i++){
            double temp;
            infile2 >> temp;

            cvmSet(K, i/3, i%3,(double) temp);
        }

        /*****************izracunaj matrice transformacija******/
        //pretvaranje matrice R u kuteve
        //dodavanje pomaka kuteva kod optimizacije

        CvMat *MM=cvCreateMat(3,3,CV_64FC1);

        //računanje kuteva

        double th1,th2,psi1,psi2,fi1,fi2;

        th1=-asin(cvmGet(R,2,0)); th2=pi-th1;
        psi1=atan2(cvmGet(R,2,1)/cos(th1),cvmGet(R,2,2)/cos(th1));
psi2=atan2(cvmGet(R,2,1)/cos(th2),cvmGet(R,2,2)/cos(th2));
        fi1=atan2(cvmGet(R,1,0)/cos(th1),cvmGet(R,0,0)/cos(th1));
fi2=atan2(cvmGet(R,1,0)/cos(th2),cvmGet(R,0,0)/cos(th2));


        psi1+=kutevi[0];
        th1+=kutevi[1];
        fi1+=kutevi[2];


        CvMat
*Ax=cvCreateMat(3,3,CV_64FC1),*Ay=cvCreateMat(3,3,CV_64FC1),*Az=cvCreateMat(3,
3,CV_64FC1);

        cvmSet(Ax,0,0,1); cvmSet(Ax,0,1,0);         cvmSet(Ax,0,2,0);
        cvmSet(Ax,1,0,0); cvmSet(Ax,1,1,cos(psi1)); cvmSet(Ax,1,2,-sin(psi1));
        cvmSet(Ax,2,0,0); cvmSet(Ax,2,1,sin(psi1)); cvmSet(Ax,2,2,cos(psi1));


        cvmSet(Ay,0,0,cos(th1));  cvmSet(Ay,0,1,0); cvmSet(Ay,0,2,sin(th1));
        cvmSet(Ay,1,0,0);     cvmSet(Ay,1,1,1); cvmSet(Ay,1,2,0);
        cvmSet(Ay,2,0,-sin(th1)); cvmSet(Ay,2,1,0); cvmSet(Ay,2,2,cos(th1));


        cvmSet(Az,0,0,cos(fi1));  cvmSet(Az,0,1,-sin(fi1)); cvmSet(Az,0,2,0);
        cvmSet(Az,1,0,sin(fi1));      cvmSet(Az,1,1,cos(fi1));
cvmSet(Az,1,2,0);
        cvmSet(Az,2,0,0);       cvmSet(Az,2,1,0); cvmSet(Az,2,2,1);


        cvMatMul(Az,Ay,MM);
        cvMatMul(MM,Ax,MM);
        cvCopy(MM,R);
```

```
	cvMatMul(R,CF,CF);
	for (int i=0; i<=2; i++){
		cvmSet(CF,i,0, -cvmGet(CF,i,0));
	}

	for (int i=0;i<=8;i++){
		if (((i%3)==0) || ((i%3)==1)){
			cvmSet(T,i/3,i%3,cvmGet(R,i/3,i%3));
		}
		else cvmSet(T,i/3,i%3,cvmGet(CF,i/3,0));
	}



	//matrica rotacije i skaliranja
	double kut=0;

	cvmSet(MG,0,0,cos(kut));cvmSet(MG,0,1,-sin(kut));cvmSet(MG,0,2,0);
	cvmSet(MG,1,0,sin(kut));cvmSet(MG,1,1, cos(kut));cvmSet(MG,1,2,0);
	cvmSet(MG,2,0,0);cvmSet(MG,2,1,0);cvmSet(MG,2,2,15);

	cvTranspose(MG,MG);

	CvMat *TEMP=cvCreateMat(3,3,CV_64FC1);

	cvMatMul(K,T,TEMP);
	cvMatMul(TEMP,MG,TEMP);
	cvInvert(TEMP,T);

	//računanje izlaznih točaka
	CvMat *izlaz=cvCreateMat(2,2,CV_64FC1);
	izlaz=izlazna(ulazna,w,h,T);

	//matrica translacije
	CvMat *MT=cvCreateMat(3,3,CV_64FC1);

	cvmSet(MT,0,0,1);cvmSet(MT,0,1,0);cvmSet(MT,0,2,0*cvmGet(izlaz,0,0)+poma
ci[0]);
	cvmSet(MT,1,0,0);cvmSet(MT,1,1,1);cvmSet(MT,1,2,0*cvmGet(izlaz,0,1)+poma
ci[1]);
	cvmSet(MT,2,0,0);cvmSet(MT,2,1,0);cvmSet(MT,2,2,1);

	cvMatMul(TEMP,MT,T);
	cvInvert(T,T);


	IplImage *dst = cvCreateImage(cvSize((*src)->width,(*src)->height
),(*src)->depth,(*src)->nChannels);
	cvZero(dst);


	cvWarpPerspective((*src), dst, T);


	cvReleaseMat(&R);
	cvReleaseMat(&T);
```

```
	cvReleaseMat(&F);
	cvReleaseMat(&CF);
	cvReleaseMat(&C);
	cvReleaseMat(&K);
	cvReleaseMat(&MG);
	cvReleaseMat(&TEMP);
	cvReleaseMat(&MT);
	cvReleaseMat(&izlaz);
	cvReleaseMat(&MM);

	return dst;

}
```

Iscrpno pretraživanje kuteva

```
void trazi_min_maska(int opcija,char datoteka[],char nazivF[],char nazivL[]){

	ofstream myfile;

	myfile.open (datoteka);

	IplImage *imgF,*imgL;
	double pomaciL[2]={0,0};

	if (opcija==1){
		// front / rear
		imgF=cvLoadImage("H:\\Projekt\\tmp4\\Front.png");


		//left / right
		imgL=cvLoadImage("H:\\Projekt\\tmp4\\Left.png");
		pomaciL[0]=-412; pomaciL[1]=-152;
	}

	if (opcija==2){
		// front / rear
		imgF=cvLoadImage("H:\\Projekt\\tmp4\\Front.png");


		//left / right
		imgL=cvLoadImage("H:\\Projekt\\tmp4\\Right.png");

		pomaciL[0]=-187; pomaciL[1]=-151;
	}
	if (opcija==3){

		// front / rear
		imgF=cvLoadImage("H:\\Projekt\\tmp4\\Rear.png");

		//left / right
		imgL=cvLoadImage("H:\\Projekt\\tmp4\\Left.png");
		pomaciL[0]=-412; pomaciL[1]=-152;
	}
	if (opcija==4){

		// front / rear
```

```
            imgF=cvLoadImage("H:\\Projekt\\tmp4\\Rear.png");
            //nazivF="H:\\Projekt\\tmp1\\Front.txt";

            //left / right
            imgL=cvLoadImage("H:\\Projekt\\tmp4\\Right.png");
            //nazivL="H:\\Projekt\\tmp1\\Right.txt";
            pomaciL[0]=-187; pomaciL[1]=-151;
      }



      double start, finish;

      // front / rear
      double kuteviF[3]={0,0,0}, pomaciF[2]={-300,-150};


      // left / right

      double kuteviL[3]={0,0,0};
      IplImage *imgLG=cvCreateImage(cvSize(imgL->width,imgL->height),imgL-
>depth,1);
      cvCvtColor(imgL,imgLG,CV_RGB2GRAY);

      IplImage *imgFG=cvCreateImage(cvSize(imgF->width,imgF->height),imgF-
>depth,1);
      cvCvtColor(imgF,imgFG,CV_RGB2GRAY);


      CvMat *ulaznaL=cvCreateMat(3,1,CV_64FC1);
      cvmSet(ulaznaL,0,0,800); cvmSet(ulaznaL,1,0,0); cvmSet(ulaznaL,2,0,1);

      CvMat *ulaznaF=cvCreateMat(3,1,CV_64FC1);
      cvmSet(ulaznaF,0,0,800); cvmSet(ulaznaF,1,0,270); cvmSet(ulaznaF,2,0,1);


      CvScalar s;
      s.val[0]=0;
      for (int i=0;i<cvmGet(ulaznaL,1,0);i++){
            for (int j=0;j<cvmGet(ulaznaL,0,0);j++){

                  cvSet2D(imgLG,i,j,s);

            }
      }

      for (int i=0;i<cvmGet(ulaznaF,1,0);i++){
            for (int j=0;j<cvmGet(ulaznaF,0,0);j++){

                  cvSet2D(imgFG,i,j,s);

            }
      }
      IplImage *cropL,*cropF,*cropG,*cropMaska;

      if (opcija<=2){
            cropL=cvCreateImage(cvSize(50,50),imgL->depth,1);
```

```
            cropF=cvCreateImage(cvSize(50,50),imgL->depth,1);
            cropG=cvCreateImage(cvSize(50,50),imgL->depth,1);
            cropMaska=cvCreateImage(cvSize(50,50),imgL->depth,1);
      }
      else if (opcija>2){
            cropL=cvCreateImage(cvSize(80,20),imgL->depth,1);
            cropF=cvCreateImage(cvSize(80,20),imgL->depth,1);
            cropG=cvCreateImage(cvSize(80,20),imgL->depth,1);
            cropMaska=cvCreateImage(cvSize(80,20),imgL->depth,1);
      }

      //parametri T
      char parametriF[]="H:\\Projekt\\tmp4\\Front.txt";
      char parametriL[]="H:\\Projekt\\tmp4\\Left.txt";
      char parametriR[]="H:\\Projekt\\tmp4\\Right.txt";
      char parametriB[]="H:\\Projekt\\tmp4\\Rear.txt";


      double delta=0.0175, kraj=0.0349;

      IplImage *tempL;

      start=clock();

      for (int i=0; i<15625; i++){

            if ((i%1000)==0) cout<<i<<endl;

            IplImage *imgG=cvCreateImage(cvSize(imgL->width,imgL-
>height),imgL->depth,1);

            IplImage *maskaF=cvCreateImage(cvSize(imgG->width,imgG-
>height),imgG->depth,1),*maskaL=cvCreateImage(cvSize(imgG->width,imgG-
>height),imgG->depth,1);
            IplImage *maskaG=cvCreateImage(cvSize(imgG->width,imgG-
>height),imgG->depth,1),*maskaT=cvCreateImage(cvSize(imgG->width,imgG-
>height),imgG->depth,1);

            IplImage *imgTF, *imgTL=cvCreateImage(cvSize(imgL->width,imgL-
>height),imgL->depth,1);


            kuteviL[0]=(i/3125) * delta - kraj; kuteviL[1]=((i/625)%5) * delta
- kraj; kuteviL[2]=((i/125)%5) * delta - kraj;
            kuteviF[0]=((i/25)%5) * delta - kraj; kuteviF[1]=((i/5)%5) * delta
- kraj; kuteviF[2]=(i%5) * delta - kraj;


            if (opcija==1){

      imgTF=projMat2(nazivF,&imgFG,ulaznaF,800,600,kuteviF,pomaciF,parametriF)
;

                  if ((kuteviL[2]!=((((i-1)/125)%5) * delta - kraj)) || i==0){

      tempL=projMat2(nazivL,&imgLG,ulaznaL,800,600,kuteviL,pomaciL,parametriL)
;
```

```
                }
            }
            if (opcija==2){

        imgTF=projMat2(nazivF,&imgFG,ulaznaF,800,600,kuteviF,pomaciF,parametriF)
;
                if ((kuteviL[2]!=((((i-1)/125)%5) * delta - kraj)) || i==0){

        tempL=projMat2(nazivL,&imgLG,ulaznaL,800,600,kuteviL,pomaciL,parametriR)
;
                }
            }
            if (opcija==3){

        imgTF=projMat2(nazivF,&imgFG,ulaznaF,800,600,kuteviF,pomaciF,parametriB)
;
                if ((kuteviL[2]!=((((i-1)/125)%5) * delta - kraj)) || i==0){

        tempL=projMat2(nazivL,&imgLG,ulaznaL,800,600,kuteviL,pomaciL,parametriL)
;
                }
            }
            if (opcija==4){

        imgTF=projMat2(nazivF,&imgFG,ulaznaF,800,600,kuteviF,pomaciF,parametriB)
;
                if ((kuteviL[2]!=((((i-1)/125)%5) * delta - kraj)) || i==0){

        tempL=projMat2(nazivL,&imgLG,ulaznaL,800,600,kuteviL,pomaciL,parametriR)
;
                }
            }

            cvCopy(tempL,imgTL);



            //oduzimanje


            cvZero(imgG);

            //maske

            cvThreshold(imgTF,maskaF,0,255,CV_THRESH_BINARY);
            cvThreshold(imgTL,maskaL,0,255,CV_THRESH_BINARY);

            //front & left
            cvZero(maskaT); cvZero(maskaG);
            cvAnd(maskaF,maskaL,maskaT);

            if (opcija==1){ //FL

                cvSetImageROI(imgTF,cvRect(350,200,50,50));
```

```
      cvSetImageROI(imgTL,cvRect(350,200,50,50));
      cvSetImageROI(maskaT,cvRect(350,200,50,50));
}

else if (opcija==2){ //FR
      cvSetImageROI(imgTF,cvRect(235,200,50,50));
      cvSetImageROI(imgTL,cvRect(235,200,50,50));
      cvSetImageROI(maskaT,cvRect(235,200,50,50));
}
else if (opcija==3){ //BL
      cvSetImageROI(imgTF,cvRect(325,85,80,20));
      cvSetImageROI(imgTL,cvRect(325,85,80,20));
      cvSetImageROI(maskaT,cvRect(325,85,80,20));
}
else if (opcija==4){ //BR
      cvSetImageROI(imgTF,cvRect(195,85,80,20));
      cvSetImageROI(imgTL,cvRect(195,85,80,20));
      cvSetImageROI(maskaT,cvRect(195,85,80,20));
}



cvZero(cropF); cvZero(cropL); cvZero(cropG);

cvCopy(imgTF,cropF);
cvCopy(imgTL,cropL);
cvCopy(maskaT,cropMaska);



double suma_bef, suma_aft;

suma_bef=cvSum(cropF).val[0]+cvSum(cropL).val[0];

cvAbsDiff(cropF,cropL,cropG);

suma_aft=cvSum(cropG).val[0];


long nzFL=0;

nzFL=cvCountNonZero(cropMaska);



if (i==-1){
      cvResetImageROI(imgTF);
      cvResetImageROI(imgTL);
      cvShowImage("imgL",imgTL);
      cvWaitKey(0);
      cvShowImage("imgF",imgTF);
      cvWaitKey(0);
      cvShowImage("img2",cropMaska);
      cvWaitKey(0);
      cvShowImage("img3",cropG);
      cvWaitKey(0);
```

```
                    cvAbsDiff(imgTF,imgTL,imgG);

                    CvPoint kor1=cvPoint(325,85),kor2=cvPoint(405,105);
                    CvScalar boja=cvScalar(255);
                    cvRectangle(imgG,kor1,kor2,boja,1,8);

                    cvShowImage("imgG",imgG);
                    cvWaitKey(0);

                    cout<<suma_aft<<endl;
                    cout<<nzFL<<endl;
                    cout<<(nzFL!=((cropMaska->width)*(cropMaska-
>height)))<<endl;
                    getchar();

              }


              if (nzFL<((cropMaska->width)*(cropMaska->height))){
                    myfile<<1e9;
                    myfile << " ";
              }
              else {
                    myfile << (suma_aft);
                    myfile << " ";
              }


              cvReleaseImage(&imgTF);
              cvReleaseImage(&imgTL);


              cvReleaseImage(&imgG);
              cvReleaseImage(&maskaF);
              cvReleaseImage(&maskaG);
              cvReleaseImage(&maskaL);
              cvReleaseImage(&maskaT);

        }



        finish=clock();

        myfile.close();


        cout<<(finish-start)/CLOCKS_PER_SEC<<endl;

        cvReleaseImage(&tempL);
        cvReleaseImage(&imgF);

        cvReleaseImage(&imgL);
```

```
        cvReleaseImage(&imgLG);
        cvReleaseImage(&imgFG);

        cvReleaseMat(&ulaznaL);
        cvReleaseMat(&ulaznaF);

        cvReleaseImage(&cropF);
        cvReleaseImage(&cropL);
        cvReleaseImage(&cropMaska);

}
```