# Mjera simetrije 2D objekta

**Antunović, Mato**





**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni preddiplomski studij računarstva**

# MJERA SIMETRIJE 2D OBJEKTA

**Završni rad**

**Mato Antunović**

**Osijek, 2018.**

# SADRŽAJ

# 1. UVOD

Simetrične objekte možemo pronaći svugdje oko nas. Određeni objekti se mogu rastaviti u više manjih cjelina kako bi jednostavnije promatrali simetriju između njih ili biti sastavljeni od više simetričnih gradivnih blokova koji se ponavljaju. Ljudi se u svakodnevnom životu susreću sa simetrijom bilo u graditeljstvu, arhitekturi, matematici... U različitim područjima simetrija ima drugačije definicije, ali možemo pronaći mnoge zajedničke točke. Inspiracija i temelj za ovaj završni bio je rad izdan 2006 godine pod nazivom „Partial and Approximate Symmetry Detection for 3D Geometry“ [1]. Kako bi naše računalo prepoznalo mjeru simetrije moramo se poslužiti računalnim vidom, za razliku od ljudi koji jednostavno prepoznaju simetriju u predmetima oko njih. Mjeru simetrije u našem slučaju pronalazimo obradom poligona koji opisuje konturu 2D objekta i koristimo se translacijom, rotacijom, refleksijom i skaliranjem kako bi opisali i izračunali simetriju to jest uparivanjem točaka i uparivanjem transformacija. Simetrični objekti poput trokuta, kruga imaju visoku mjeru simetrije približno ili jednako 100%, a asimetrični objekti trebaju imati ocjenu simetrije približno ili jednako 0%. U MATLAB programskom jeziku smo razvili metodu računanja mjera simetrije. Računanje simetrije može postati jako zahtjevno s povećanjem broja točaka koje uzimamo kako bi povećali točnost. Stoga koristimo CUDA C++ paralelizaciju za izvođenje na grafičkoj kartici kako bi ubrzali izvođenje samog programa. Iako simetriju možemo pronaći svugdje oko nas, literature ima manje nego što je za očekivati.

## 1.1 Zadatak završnog rada

U završnom radu potrebno je obraditi metodu za računanje mjeru simetrije 2D objekta. Dobivenu metodu je potrebno optimizirati pomoću CUDA C++ programskog jezika za izvođenje na grafičkim karticama.

# 2. PREGLED POSTOJEĆIH RJEŠENJE

U ovom poglavlju bit će spomenuta kratka povijest obrade slike i najpoznatije metode koje su utjecale na nastanak [1] rada. Problematika vezana uz prepoznavanje simetrije i prepoznavanja objekata na slikama ili videu proučavana u različitim područjima od računalnog vida, robotike, te vizualne percepcije . Tom problematikom znanstvenici se ne bave samo u zadnjem desetljeću nego možemo pronaći radove koji govore o prepoznavanju 2D i 3D simetrije već iz 80-tih godina 20 stoljeća. Prvobitno se pokušavala pronaći perfektna simetrija za 2D i 3D planarne skupove točaka [2,3]. Zbog ograničenja tih metoda i nemogućnosti prilagodbe za korištenje u stvarnosti. Zabrodsky [4] formira metoda za računanje približne simetrije izražavajući simetriju kao kontinuiranu značajku. Iako izlaze mnogi radovi na temu računanje približne 3D simetrije kompleksnost algoritama i količina potrebne snage za obradu podataka ograničavaju primjenu. Pojavljuju se različite metode za računanje simetrije koje se primjenjuju za računanje 2D i 3D modela, računanje preko brzih Fourierovih transformacija ili računanje simetrije upotrijebim gradijenata.

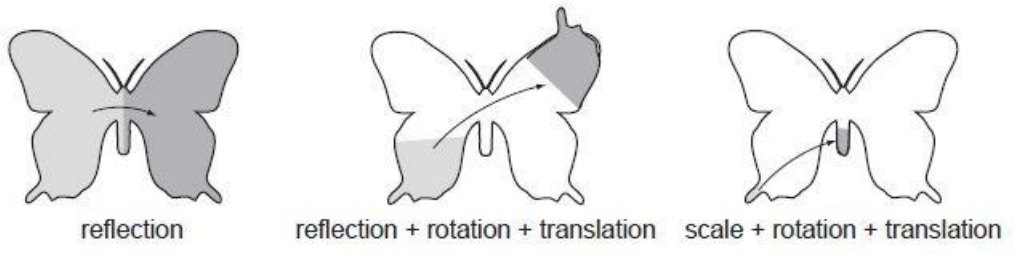
Metoda koju obrađujemo u ovom završnom radu ima određene sličnosti s Houghovom transformacijom [5], koja se najviše koristi kao metoda ekstrakcije u obradi slika. Započinje rad pronalaženjem početnog seta testnih točaka koje su dobivene korištenjem različitih metoda za detekcije rubova. Nakon dobivanja početnog seta točaka ova metoda uzima podskupove početnog seta točaka kako bi odredila parametre istaknute krivulje. Pri provjeri naših podskupova pamtimo koji se oblici najčešće pojavljuju te nam to utječe kako će nam izgledati istaknuta krivulja. Iako je Houghova transformacija[5] izašla još 1959 godine i u današnje vrijeme izlaze radovi i nastaju ideje koji proučavaju simetriju i temelje njemu. Primjer možemo pronaći u radu [6] koji je proučavao rotacijsku i refleksijsku simetriju. „Random Sample Consensus“ [7] je algoritam koji se koristi pri računanju velikih modela koje sadrži dosta vrijednosti koje ne prate krivulju nego su nepravilno udaljene od promatranih točaka. Radi na principu odabira slučajnih parova reprezentativnih podataka na objektu. Zatim primjenjuje transformacije i provjerava pogreške između objekata. Istraživši dovoljan broj transformacija, možemo otkriti relativnu simetriju. Problem kod ove metoda je potreba za velikom računalnom snagom pri obradi prostornih testiranja, te računanju geometrijskih ključeva. Metoda koju u ovom radu obrađujemo [1] dijeli određene sličnosti s gore spomenutim radovima i metodama, ali izbjegava zahtjevne pretrage koje prethodne metode izvršavaju. Prvo tražimo potvrdu simetrije, a zatim pronalazimo greške u transformacijama. Što također dovodi do znatne uštede prostora, jer nije potrebno spremati tablice ključeva.

# 3. UVOD U SIMETRIJU I KORIŠTENE ALATE

U ovom poglavlju će biti opisani osnovni pojmovi potrebi kako bi mogli razumjeti i provjeriti simetriju određene 2D slike. Također će ukratko biti spomenuti MATLAB koji ćemo koristi pri realizaciji programskog problema završnog rada. CUDA C++ nećemo objašnjavati i pokazivati u ovom završnom radu zbog kompleksnosti i opsežnosti same teme.

## 3.1 Općenito o simetrija

Sa simetrijom smo bili upoznati još u osnovnoj školi no kroz godine se mijenja naše shvaćanje i znanje o njoj. Simetriju shvaćamo kao invarijancu pod skupom transformacija[1]. Skup transformacija koje u ovom slučaju promatramo obuhvaća: rotacije(vrtnja), translacije(pomak), refleksiju, iako bi se i skalirane trebalo nalaziti među promatranim transformacijama zbog kompliciranosti i potrebe za puno više provjera nismo u obzir uzimali skaliranje. Simetrija u širem pojmu može obuhvaćati globalne ili lokalne. U fizici globalna simetrija obuhvaća sve točke prostor-vremena i njene transformacije su jednake u svim točkama prostor-vremena. Kod lokalne simetrije ili djelomične[1] imamo drugačije transformacije u drugačijim točkama prostor-vremena. Globalne i djelomične simetrije važne su nam u ovom radu, jer za prepoznavanje i pronalazak velikih globalnih simetrija ne trebamo puno točaka, dok za pronalazak malih djelomičnih simetrija trebamo puno gušći uzorak što dovodi do većeg broja točaka[1]. Iako o simetriji možemo još puno pisati i dijeliti ju u različite kategorije za potrebe ovoga rada i za rad na 2D slikam-a gore spomenuti pojmovi će nam biti dostatni.




*Slika 3.1 Prikaz transformacija za sliku leptira [1 str. 560]*

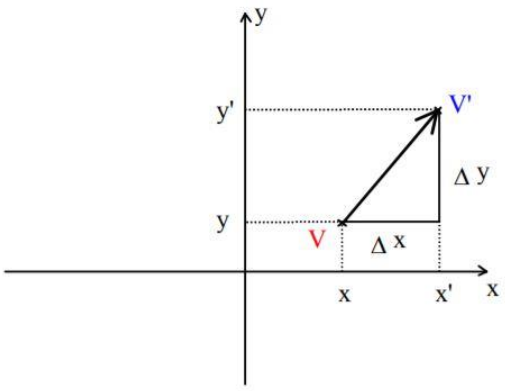
## 3.2 Teorijski dio

U ovom dijelu upoznajemo se s osnovnim pojmovima koji nam trebaju za daljnji rad i razumijevanje literature [1] na kojoj je ovaj završni rad temeljen. Od 2D transformacija obradit ćemo translaciju i rotaciju.

Translacija ili pomak je preslikavanje koju svaku točku ili vektor pomakne za određeni vektor. Možemo definirati translaciju u prostoru i ravnini pošto trenutno radimo za 2D koristimo translaciju u ravnini. Označimo našu trenutnu poziciju s točkama x i y. Pomak koji želimo napraviti označit ćemo s $\Delta x$ i $\Delta y$, a T je matrica translacija. Iz toga slijedi da će translatirane koordinate naših točaka biti x' i y'.(3-1)

$$x' = x + \Delta x \qquad y' = y + \Delta y$$

$$[x' \quad y' \quad h'] = [x \quad y \quad h] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \Delta x & \Delta y & 1 \end{bmatrix} \qquad \mathbf{V'} = \mathbf{V} \cdot \mathbf{T}$$




*Slika 3.2 Prikaz translacije u ravnini [8]*

Rotaciju prikazujemo oko ishodišta za kut φ. (3-2)

x'=x*cosφ-y*sinφ $\qquad R(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$

y'=x*sinφ+ y*cosφ

Zbog zadane orijentacije vektora pri radu ovog završnog rada moramo koristiti rotaciju u suprotnom smjeru kazaljke na satu. Iz toga možemo vidjeti da bi rotacija za φ= -90° stupnjeva bila. (3-3)

$$R(-\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$$

Moramo paziti na veličine naših matrica pošto u *Poglavlju 4* moramo dobiti združenu matricu **H.** Kako bi to bilo moguće moramo proširiti matricu rotacija, pa iz toga slijedi:

$$R(-\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3\text{-}4)$$

## 3.3 Korišteni programi

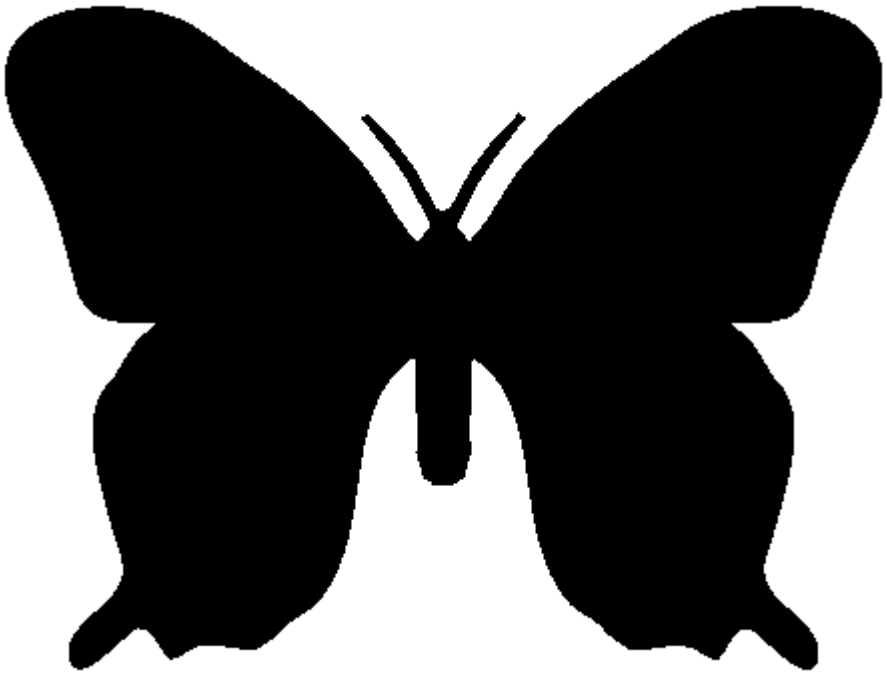
U izradi završnog rada korištena je MATLAB programska platforma koja je dizajnira za inženjere i znanstvenike. MATLAB nam omogućava lagano rukovanje s matricama, a uz to i razvitak algoritama i aplikacija, analizu podataka i kreiranje različitih modela. Ima puno gotovih sučelja koji omogućavaju pretvorbu iz MATLAB koda u C/C++ kod za obradu na CUDA karticama. Za određene naredbe u obradi slika poput naredbe *imread* koju koristimo na samom početku nema našeg programa ipak je potrebno ručno prevoditi i pisati kod. Jedna od najvećih prednosti CUDA je što nam omogućava izvedbu petlji u paraleli.

# 4. REALIZACIJA METODE OCJENE SIMETRIJE

U ovom poglavlju objašnjeno je kako naš algoritam radi i prikazana su objašnjenja uz dobivene rezultate. U računanju mjere simetrije ovog modela kao glavna literatura će nam poslužiti [1] i težit ćemo da rezultati i put do tih rezultate bude što sličniji.

## 4.1 Crno bijela slika u točke

U prvom dijelu našeg programa za računanje mjere simetrije učitavamo prvo sliku za koju želimo mjeru simetrije. Za odabranu crno bijelu sliku prvo ju učitavamo u MATLAB u ovom radu koristit ćemo se slikom leptira koja je najsličnija slici korištena u radu [1]. Zatim tu sliku moramo pretvoriti u točke kako bi mogla s njom manipulirati i kako bi dobili obris tijela za koje gledamo.



*Slika 4.1 Crno bijela slika koju učitavamo u MATLAB*

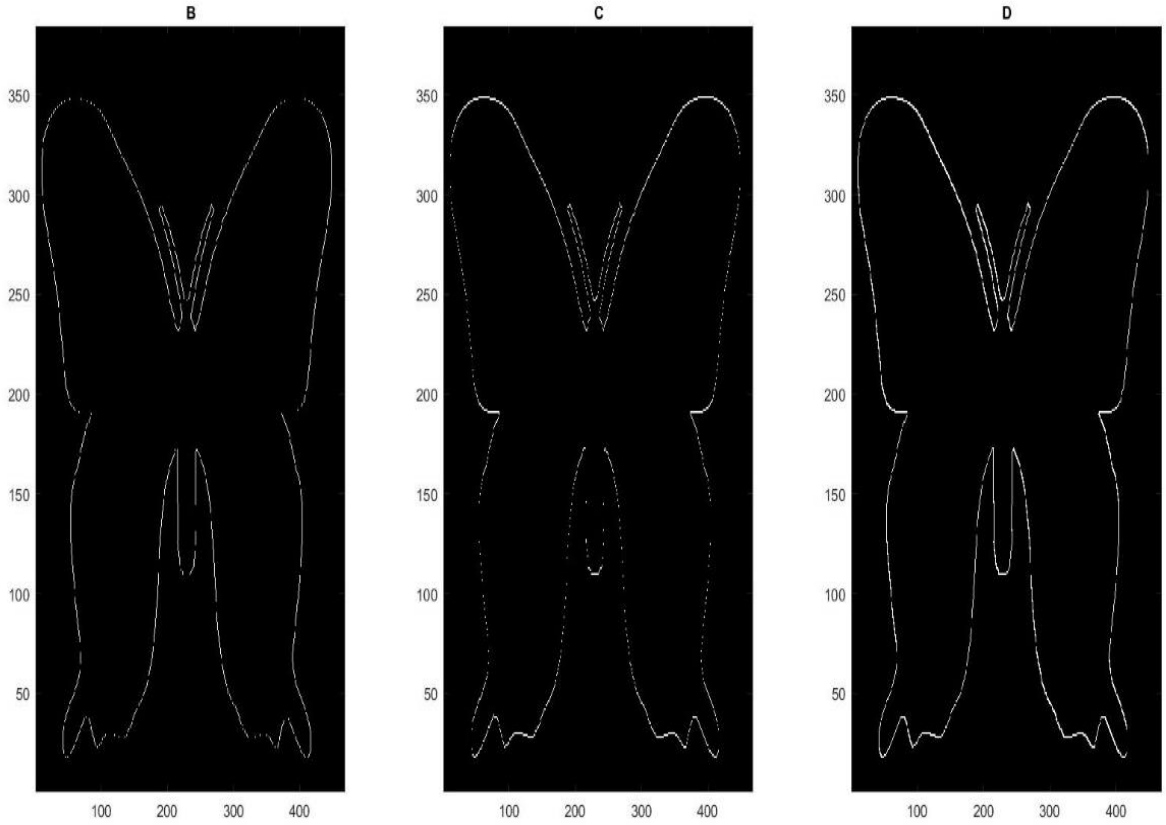
MATLAB preko naredbe *imread* omogućavam nam da pročitamo sliku i spremimo ju kao matricu. Pošto učitavamo crno bijelu sliku u ovom slučaju 2 vrijednosti koje imamo su bijela i crna boja. Crna boja poprima vrijednost 1 što znači da učitavši sliku na svim pozicijama gdje je slika oboja to jest nalazi se crna boja vrijednost će postati 1, naravno isto tako di je slika bijela vrijednost će postati 0.

Matrica nam postaje veličine 468x384 u slučaju *Slike 4.1*, klase logical jer imamo samo 0 i 1. Iako nam *imread* omogućava čitanje i upravljanje i slikama u boji zbog jednostavnosti i zauzeća manje memorije smo koristili crno bijele slike.

```
sA = size(A);
B = [abs(diff(A)); zeros(1,sA(2))];
C = [zeros(sA(1),1), abs(diff(A'))'];
D = or(B,C);
[x,y] = ind2sub(sA,find(D(:)>0));
```

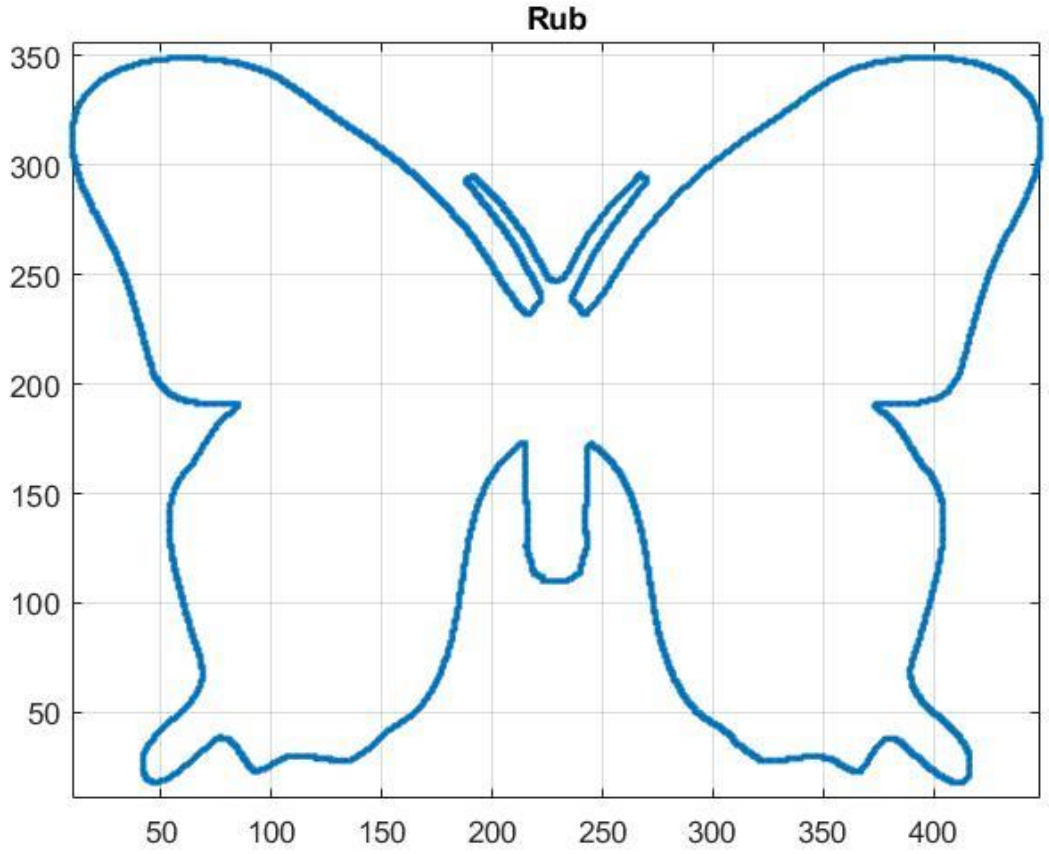
*Slika 4.2 Pronalazak inverznih matrica*

U naš spremnik sA spremamo dimenzije naše slike iz varijable A. Kreiramo varijable B i C. Pomoću naredbe *diff(A)* računamo razliku između susjednih elemenata duž A matrice, dok pomoću *zeros* kreiramo u našem slučaju matricu ispunjenu nulama.




*Slika 4.3 Prikaz razlike između koraka detekcije obruba*

Pošto je naša *Slika 4.1* ispunjena bojom unutar promatranog oblika, a trebamo dobiti samo konturu moramo pronaći razliku između njih. Mapa boja koju koristimo pri manipuliranju s varijablama B,C,D je siva. Za računanje D-a koristimo logički operator ili kako bi dobili što točniji i popunjeniji rub prikazano na primjeru *Slika 4.3*.




*Slika 4.4 Prikaz plota za varijable P*

Naredba i*nd2sub* nam omogućava određivanje vrijednosti ekvivalentnih indeksa koji odgovaraju elementima u nizu i spremamo ih u x i y u našem kodu. Za daljnji rad s koordinatama zbog jednostavnijeg korištenja i lakšeg manipuliranja s jednom matricom nego 2 odvojena niza spremamo x i y u varijablu P koja postaje matrica rubnih točaka *Slika 4.4.*

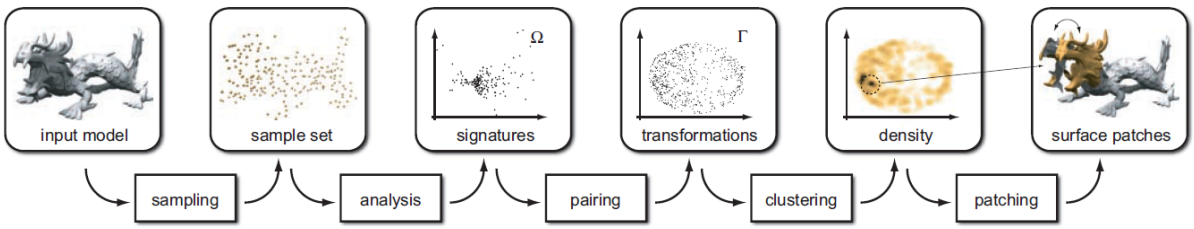
## 4.2 Ostale korištene funkcije

Za rješavanje problema simetrije i zbog čestog korištenja određenih naredbi kreirali smo funkcije kako bi se lakše snašli i omogućili si veću preglednost koda. Slike velike kvalitete dale bi nam puno ljepše rezultate i puno više točaka za obradu, no zbog količine podataka koje je onda potrebno

obraditi pri radu s velikim slikama za ovaj rad koristimo slike s manje elemenata što odmah utječe na njenu kvalitetu. Za obradu većih slika koristili bi onda obradu na grafičkim karticama koju bi ostvarili pretvorbom i prilagodbom MATLAB koda za C++ koji podržava rad s CUDA grafičkim karticama. Koristeći slike manje s puno manje elemenata opada nam i kvaliteta slike. Smanjenje kvalitete slike nam utječe na prikaz naših rubova pri uvećanom gledanju primijeti ćemo neravne rubove zbog toga koristimo funkciju s*moothPolygon* . Jedini problem je što sada imamo samo točke, a ne poligon. Prije upotrebe funkcije *smoothPolygon* moramo naše točke pretvoriti u poligon. Za tu pretvorbu koristimo funkciju *poligonFromPoints*.

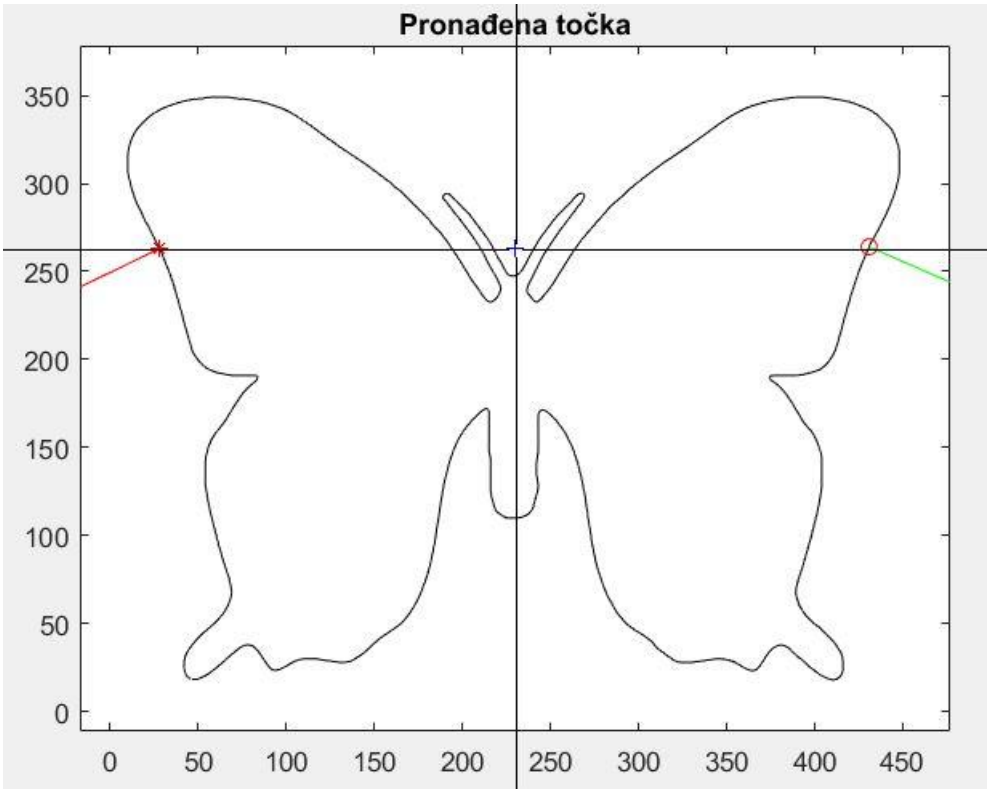
Pri pozivu te funkcije šaljemo joj točke u našem slučaju je to varijabla P i maksimalnu udaljenost za koju će se odabrati manji skup točaka, ako je udaljenost između naše dvije točke veća od maksimalne udaljenosti. Kao izlaz iz te funkcije dobijemo varijablu poli u koju je zapisan naš poligon i varijablu indices u kojoj se nalaze pozicije naših točaka u x matrici. Zatim možemo pozvati funkciju s*moothPolygon* ona nam omogućava da izgladimo naš poligon računajući srednju vrijednost između dviju točaka. Za svaki par naših točaka moramo pronaći normalu na, ako je razlika u kutu i normalama velika tu ćemo točku odbaciti. Normalizaciju točaka odradit ćemo u funkciji, dok proračun u glavnom dijelom programa. Za računanje normale ne koristimo gotovu MATLAB funkciju *normalize* već smo kreirali našu funkciju *normalizeVector*.Također kreirali smo dvije funkcije za računanje kuta između vektora u radijanima *vec2rad* i *vec2deg* za računanje kuta između vektora u stupnjevima. U slučaju da je kut veći od 180 stupnjeva ili π onda ga oduzimamo od punog kruga u slučaju stupnjeva 360 stupnjeva , a u slučaju radijana od 2π.

## 4.3 Glavni dio programa



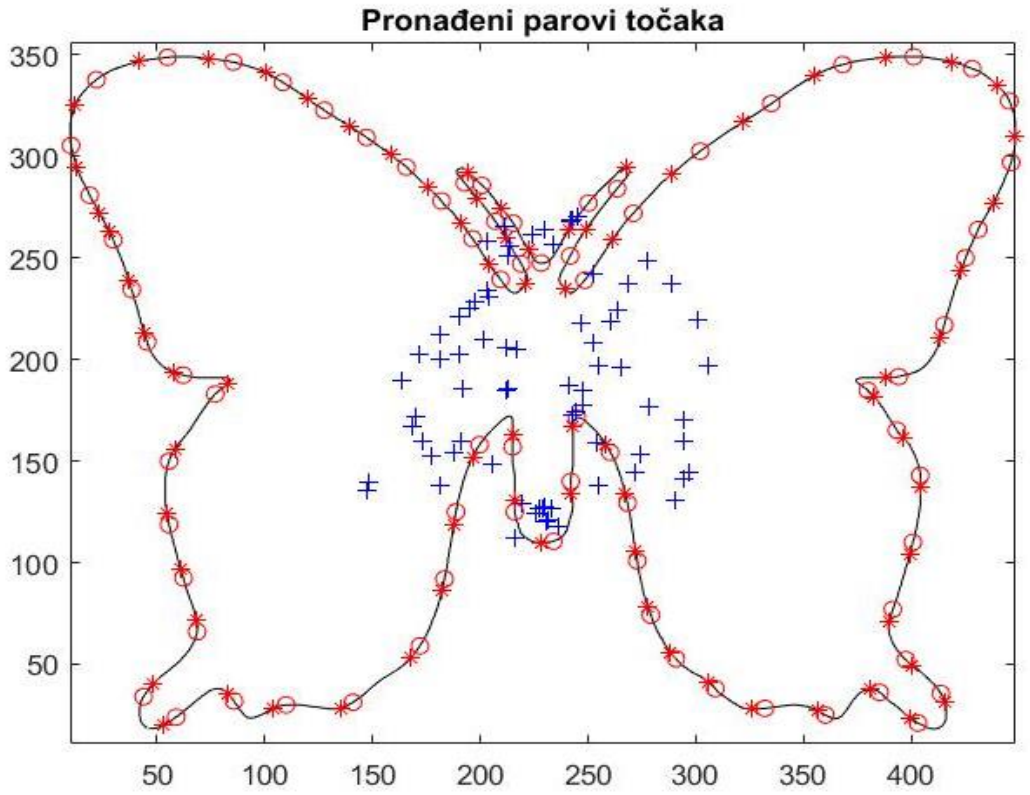

*Slika 4.5 Iz literature [1 str. 562] prikazuje tijek rada našeg programa*

Iz gore prikazane *Slike 4.5* vidimo korake potrebne kako bi došli do konačnog rezultata, jedina razlika kod nas će biti što radimo i računanje postotka simetrije koji nije rađen u gore prikazanom pregledu rada programa. Svaki par točaka (p, q) na rubu našeg modela definira jedinstvenu refleksiju u odnosu na polovište modela (p+q)/2, s normalom smjera p-q.

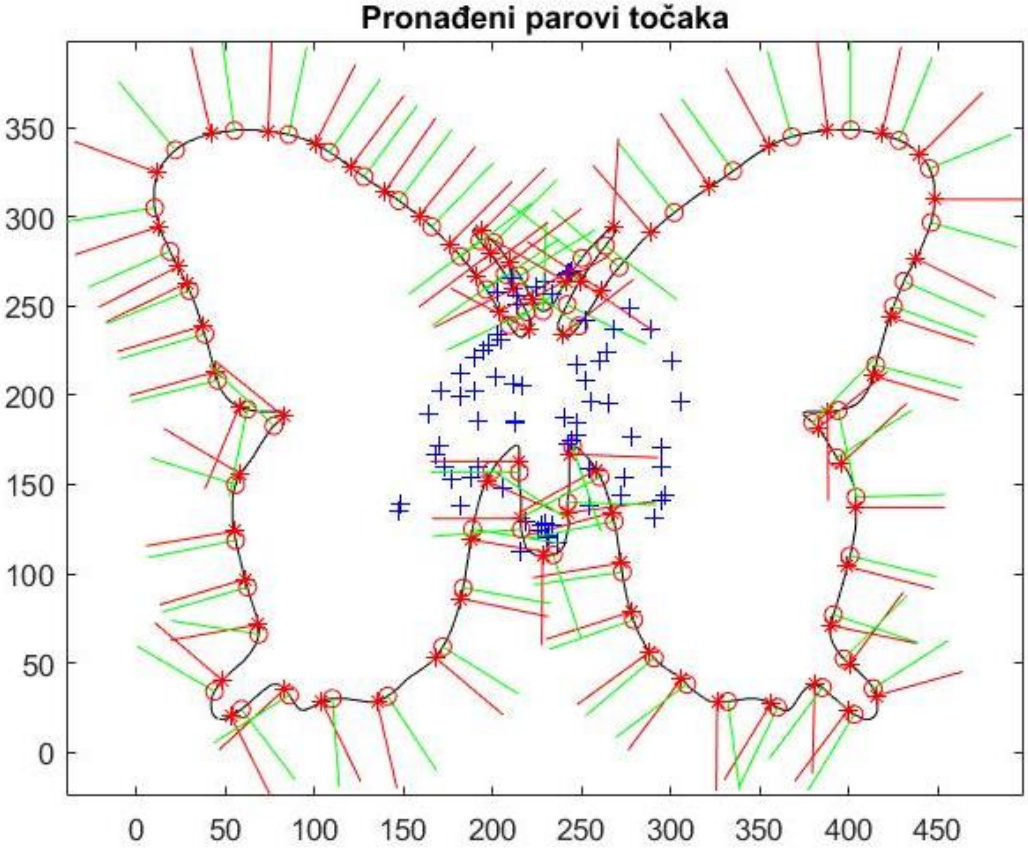



*Slika 4.6 Prikaz točke p i q sa pripadajućim normalama i polovištem*

Na primjeru *Slike 4.6* vidimo kako bi to izgledalo za jedan par točaka (p, q) s pripadajućim normalama, dok na *Slici 4.7* smo ispisali sve promatrane točke, ali bez pripadajućih normala. Točka p je označena s crvenom zvjezdicom s pripadajućom normalom u crvenoj boji, točka q je označena kružićem u crvenoj boji s pripadajućom normalom u zelenoj boji radi lakše uočljivosti. Za naš rad moramo pronaći više parove točaka (p ,q) za koje radimo provjeru simetrije. Iz tih točaka možemo doći do zaključka o simetrije. Uspoređujući više točaka na približno istoj refleksiji, tek tada možemo reći da postoji vjerojatnost pojave simetrije u našem modelu *Slika 4.8.* Možemo pronaći simetriju promatrajući grupacije naših točaka u transformacijskoj ravnini gdje svaka točka odgovara određenom pravcu refleksije *Slika 4.9.* Pošto pri mapiranju naših točaka ne zapisujemo prostorne koordinate , više točaka može se nalaziti na istom mjestu. Zato u sljedećoj fazi moramo izvući povezane komponente koje su nepromjenjive pod izvučenom transformacijom. Iz gustoća grupacija točaka u ravnini možemo odrediti koji dijelovi su simetrični, a koji ne.

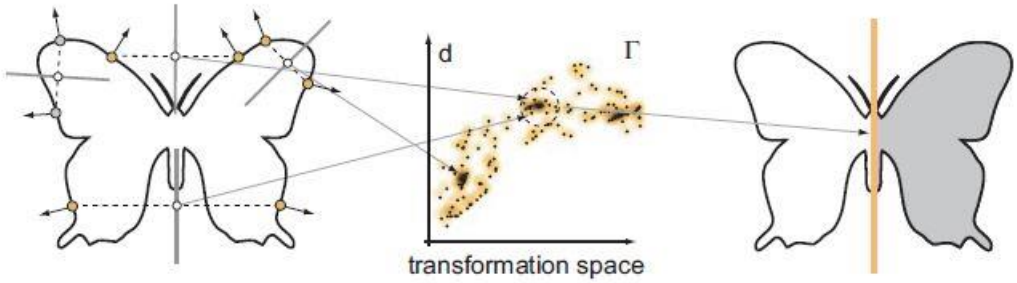



*Slika 4.7 Pronađeni parovi točaka bez normala*




*Slika 4.8 Prikaz pronađenih parova točaka za kojih promatramo simetriju*

Problem koji može nastati u ovom dijelu programa je što prethodno spominjani prostor transformacija Γ moramo nekako definirati da obuhvati u jednom mjerilu: rotaciju, translaciju i skaliranje. Rješenje pronalazimo u radovima [1, 11] u kojem se definira norma transformacije $\mathbf{T} = (s,R_x,R_y,R_z,t_x,t_y,t_z) \in \Gamma$ (4-1)kao težinska suma $||\mathbf{T}||^2 = \beta_1 s^2+\beta_2(R_x^2 + R_y^2 + R_z^2) + \beta_3(t_x^2 + t_y^2 + t_z^2)$ (4-2). U našim primjerima i proračunima postavili smo β kao težinsku konstantu tako da rotacija od 180 stupnjeva odgovara pomaku pola granične dijagonale i faktoru skaliranja 10.[1]




*Slika 4.9 Prepoznavanja simetrije na osnovu refleksije[1]*

Pomak od pola granične dijagonale uvijek se poklapa sa rubom slike, tako da će sjecište naših dijagonala uvijek biti srednja vrijednost p i q koordinata. Malo slovo s je skaliranje koje u ovom primjeru završnog rada nismo koristili zbog kompliciranosti računanja i samog programa pa smo ga zanemarili. $R_x$,$R_y$,$R_z$ predstavljaju rotacije naših vektora x,y,z. Pošto imamo samo 2D, a ne 3D kao što je pokazano u radu [1] koristimo samo rotacije za x i y u našem slučaju $R_q$,$R_p$. Isto to nam vrijedi za tx,ty,tz pa tako u našem radu i programu koristimo za translaciju samo $t_x$ i $t_y$. Kako bi iz T-a dobili vrijednost za naš Γ moramo primijeniti formulu iz [1] $d(\mathbf{T},\mathbf{T}') = \| \mathbf{T} - \mathbf{T}' \|$ (4-3). Gdje nam $\mathbf{T}'$ predstavlja konjugiranu transponiranu matricu, a oduzimanje se vrši za svaku komponentu. Za naš slučaj u 2D imali bi $\mathbf{T} = (R_q,R_p,t_q,t_p) \in \Gamma$ (4-4) ,a kao težinska suma $||\mathbf{T}||^2 = \beta_1(R_q^2 + R_p^2) + \beta_2(t_q^2 + t_p^2)$ (4-5). Zbog oznaka u programu i radu zamijenjene su oznaka za rotaciju $R_x$,$R_y$,$R_z$ za $R_q$,$R_p$ i za translaciju $t_x$, $t_y$ za $t_q$ i $t_p$ *Slika 4.10* .

```
%delta p(x1,y2)
x1=(poli(m,2)-poli(m,1));
y1=(NN(m,2)-NN(m,1));
kutq=atan2(y1,x1)*(180/pi);
%rotacija
Rq = [cos( kutq  ) sin( kutq  ) 0; -sin( kutq  ) cos( kutq  ) -sin( kutq  ); 0 0 1];
%translacija
tq=[poli(m,1) ; NN(m,1) ; 0];
%združena
H1=[Rq tq];
```

*Slika 4.10 Računanje potrebnih podataka za d(**T**,**T'**) u MATLABU*

Transformacijski prostor Γ predstavit ćemo histogramom gledajući učestalost pojavljivanja za odabrane parove točaka u kvantiziranom prostoru. Osi histograma moramo kvantizirati radi lakšeg i preglednijeg prikazivanja. Transformacijski prostora Γ prikazujemo postavljajući na apscisi dobiveni kut ϕ, dok na osi ordinata udaljenost d. Kut ϕ dobijemo tako da gledamo normalni smjer p-q što znači da će naš kut biti u suprotnom smjeru kazaljke na satu i ići od normale točke q prema normali točke p. Pošto nam je točka p udaljena od točke q prvo ju moramo translatirati do točke q, a zatim i njenu normalu tek tada možemo gledati kut. Zbog mogućnosti malih kutova najsigurnije je koristiti atan2, ali atan2 je puno robusnija i teže obradiva funkcija. Pri računanju kutova moramo odlučiti da li nam je važnija točnost ili si možemo dozvoliti rad s atan2 koji će nam uzeti puno više memorije. Mala broj elemenata i točaka na slici nam neće stvarati problem, ali ako bi radili s većim slikama koje imaju puno više elemenata preporučeno bi bilo da se napravi prilagodba za C++ CUDA obradu.

U radu [1] primijetimo da je promijenit mapa boja pri prikazu gustoće točaka u bakrenu ili u MATLABU *colormap(copper).*

Gledamo gdje se najviše točaka pojavilo u našem histogramu i oko toga dijela iscrtavamo kružnicu i uzimamo točke koje ćemo gledati za naš rezultat. Zatim te točke povezujemo ponovno s njihovim koordinatama i izračunati mjeru simetrije kao odnos podudarnih točaka kroz broj točaka promatranog uzorka. Na kraju ispišemo rezultat i vratimo simetričnu sliku.

# 5. ZAKLJUČAK

Obrađena metoda za prepoznavanje simetrije ima široki spektar primjene. U usporedbi s drugim metodama vidimo da je za računala puno manje zahtjeva u smislu potreba za resursima, iako sama obrada slika uzima dosta memorije. Demonstrirano je kako uparivanje lokalnih oblika i grupiranje u prostoru transformacija dovodi do efikasnijeg pronalaska simetrije u 2D i 3D geometrijskim modelima. Ovaj program bi se trebao optimizirati za rad na CUDA grafičkim karticama pomoću C/C++ programskog jezika, te bi nam onda omogućio da simetriju računamo za puno veće slike, te da možemo raditi računanje i prepoznavanje simetrije za 3D geometrijske objekte.

Zbog kompleksnosti ove teme[1] i potrebnih znanja koje ona zahtjeva iz područja računalnog vida i određenih osnova robotike planirano je završni rad nastaviti i završiti ga za diplomski rad.

# SAŽETAK

**Naslov:** Mjera simetrije 2D objekta

Cilj ovog završnog rada je upoznavanje sa obradom slike i računalnim vidom. Naučeno je baratati puno bolje s transformacija prostora i služiti se MATLAB programskim paketom. Samostalno istraživati i upoznavati nove pojmove te primijeniti naučeno znanje bilo je ključno za ovaj završni rad. Prateći [1] pokušano je realizirati što sličnije rješenje i u isto vrijeme sačuvati najbolje karakteristike te metode.

**Ključne riječi:** C/C++, MATLAB, prepoznavanje simetrije, obrada slike

# LITERATURA


[1] Mitra, *Partial and Approximate Symmetry Detection for 3D Geometry* ,2006

[2]Alliez, *Anisotropic Polygonal Remeshing*, 2003

[3] Cohen-Steiner, *Restricted Delaunay Triangulations and Normal Cycle*, 2003

[4] Karpathy, *Object Discovery in 3D scenes via Shape Analysis*, 2013

[5] Richtsfeld, *Segmentation of Unknown Objects in Indoor Environments*, 2012

[6] Richtsfeld, *Learning of perceptual grouping for object segmentation on RGB-D dana*, 2014

[7] Stein, *Convexity based object partitioning for robot applications*,2014

[8] http://www.zemris.fer.hr/predmeti/rg/predavanja/3_primitive.pdf , 20.9.2018

[9] http://www2.geof.unizg.hr/~zeljkat/p14.pdf , 20.9.2018 godine

[10] http://matematika.fkit.hr/novo/matematika%201/predavanja/Mat1_Lekcija3.pdf , 20.9.2018

[11] Niloy J. Mitra, Symmetry Detection and Symmetrization,2007

[12] Niloy J. Mitra,Symmetry in Shapes Theory and Practice,University College London,2013

[13] Tamosiunaite, *Perceptual influence of elementary three dimenzional geometry. (2) fundamental object parts*, 2015

[14] Amato, N. M., Bayazit, O. B., Ddale, L. K., Jones, C., AND Vallejo, D. 2000. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *IEEE Trans. On Robotics and Automation*, 442–447.

[15] Z. Kovačić, S. Bogdan, V. Krajči, Osnove Robotike,2002

## ABSTRACT

**Title**: Symmetry measure of 2D object

In this paper, we are calculating a measure of symmetry. Our program is showing a result in degree while objects that have a high measure of symmetry will have around 100%, but the asymmetric object will have a measure of symmetry around 0%. Because of a large number of points that are needed to get precise result our method is optimized for CUDA C++ for running on graphics cards.



**Keyword**: C/C++, MATLAB,recognition of symmetry,image processing

# ŽIVOTOPIS

Mato Antunović rođen 27.07.1994 godine u Bochumu,Njemačka. Odrastao u Čepinu i završava matematičko prirodoslovnu gimnaziju u Osijeku. Upisao je Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku 2014. godine odabravši smjer računarstvo.