SVEUČILIŠTE J. J. STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Petar Taler

Statistička procjena površine objekta iz snimke s aditivnom greškom

Doktorski rad

Osijek, 2019.

Doktorski rad je izrađen na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Sveučilišta Josipa Jurja Strossmayera u Osijeku.

Mentor:	doc. dr. sc. Emmanuel Karlo Nyarko
Sumentor:	prof. dr. sc. Mirta Benšić
Doktorski rad ima:	74 stranice
Doktorski rad broj:	71

Sadržaj

1. Uvod

Problem procjene površine objekta na slici javlja se u mnogim ljudskim djelatnostima. Primjerice, u medicini je potrebno procijeniti veličinu tumorskog tkiva zabilježenog uređajem za magnetsku rezonanciju (MRI) ili izmjeriti opseg trbuha fetusa ultrazvukom [1]; u mikrobiologiji često se procjenjuje površina mikroorganizma snimljenog fluorescentnim mikroskopom [2]; u geodeziji se vrše procjene objekata ispod površine zemlje koristeći snimke GPR (engl. *Ground Penetrating Radar*) uređaja [3]. Pri tom postupku uobičajeno je aproksimirati objekt čija se površina želi procijeniti kružnicom ili elipsom. Iz parametarski zadane kružnice ili elipse trivijalno je potom izračunati njezinu površinu.

Razvijeno je mnogo algoritama za detekciju elipsoidnog ili kružnog objekta na slici kada je taj objekt jasno prikazan. Ti se algoritmi uglavnom oslanjaju na detekciju rubova objekta. Međutim, u praksi je čest slučaj da je snimka objekta loše kvalitete, odnosno da je na njoj prisutna velika količina šuma. Kod takvih slika postojeći algoritmi često imaju problema s detekcijom rubova i uslijed toga daju nekvalitetne rezultate ili uopće ne detektiraju objekt.

Pojam *šuma* se, u općem slučaju, definira kao neželjeni signal. On se, osim na slikama, javlja i u drugim medijima. Primjerice, neželjene električne fluktuacije u signalu kojeg prima AM radio uređaj uzrokuju čujne akustičke smetnje, tzv. s*tatički šum*.

Šum na slikama definira se prema [4] kao slučajna varijacija svjetline ili intenziteta boje. Dva glavna uzroka pojave šuma na slikama su prisutnost neželjenih struktura uz objekt kojeg se pokušava snimiti i nesavršenost same mjerne opreme kojom se snimanje vrši. Neke od osnovnih vrsta šumova na slikama koje se sreće u praksi su:

- bijeli šum: aditivan šum, najčešće iz Gaussove distribucije, neovisan za različite piksele slike i neovisan o intenzitetu signala. Nastaje primjerice na fotografijama prilikom slabog osvjetljenja scene.
- šiljasti šum (engl. *spike noise, salt-and-pepper noise*): „impulsni“ šum koji se očituje tamnim pikselima na svijetlim područjima slike i svijetlim pikselima na tamnim područjima

slike. Do ovakvog šuma dolazi primjerice zbog grešaka u analogno-digitalnim pretvaračima ili zbog grešaka u prijenosu signala.

- šum snimanja (engl. *shot noise*): nastaje zbog varijacija u broju fotona koje zabilježi fotografski senzor. Proporcionalan je svjetlini slike.
- šum kvantizacije: uzrokovan je uzorkovanjem (kvantizacijom) zabilježene svjetline piksela na najbliži diskretni iznos kojeg je moguće pohraniti.

U ovoj disertaciji bit će opisana metodologija i predstavljeno programsko rješenje koje učinkovito procjenjuje površinu objekta snimljenog na slici s aditivnom greškom, odnosno šumom. Razvijena metodologija temeljena je na prirodnom parametarskom modelu za procjenu nosača uniformne distribucije mjerene s greškom.

Ovo istraživanje rezultiralo je trima izvornim znanstvenim doprinosima.

1. Izrada algoritma i softvera za procjenu duljine presjeka proizvoljnog pravca i objekta na slici snimljenoj s aditivnom greškom
2. Izrada algoritma i softvera za procjenu površine kružnog ili elipsoidnog objekta na slici snimljenoj s aditivnom greškom
3. Objektivna ocjena predloženog algoritma za procjenu površine usporedbom s drugim algoritmima na referentnim podacima

Disertacija je organizirana na slijedeći način: 2. poglavlje daje pregled postojećih metoda za detekciju kružnih i elipsoidnih objekata na slikama s posebnim naglaskom na metode s kojima će se u kasnijim poglavljima usporediti ovdje razvijena metoda. U 3. poglavlju ukratko su predstavljeni parametarski modeli kojima se, na temelju jednostavnog slučajnog uzorka, procjenjuje duljina uniformne distribucije u prisustvu aditivne greške. Nadalje, 4. poglavlje daje detaljan opis metodologije kojom se, koristeći rezultate predstavljene u prethodnom poglavlju, učinkovito procjenjuje duljina presjeka proizvoljnog pravca i objekta na slici snimljenoj s aditivnom greškom. U 5. poglavlju koriste se rezultati iz prethodnih dvaju poglavlja u svrhu procjene površine kružnog ili elipsoidnog objekta snimljenog na slici ili zadanog numerički. U 6. poglavlju predstavljen je programski paket LeArEst u kojem su implementirane razvijene metode. Poglavlje 7 bavi se usporedbom predloženog algoritma za procjenu površine s drugim, iz literature poznatim, algoritmima. Zaključak rada prikazuje rekapitulaciju dobivenih rezultata i osvrt na predložene znanstvene doprinose.

2. Pregled postojećih algoritama za detekciju kružnih i elipsoidnih objekata na slikama

U ovom poglavlju bit će predstavljene dvije popularne metode za detekciju kružnih i elipsoidnih objekata sa slika. To su *Houghova transformacija* [5, 6, 7], koja sa svojim brojnim varijacijama predstavlja standard za detekciju krivulja različitih porodica i relativno novi algoritam *EDCircles* [8] koji se u posljednje vrijeme često spominje u literaturi. Također će biti opisana i metoda koju su razvili M. Fornaciari i dr. [9] koja kombinira neke elemente prethodnih dviju metoda.

2.1. Houghova transformacija

Houghovu transformaciju (HT) predstavio je Paul Hough u svom patentu iz 1962. godine [5], a popularizirana je u području računalnog vida 1969. godine u radu A. Rosenfelda [7] i 1972. godine radom A. Dude i P. Harta [6]. U ovom trenutku se vjerojatno najviše implementacija metoda za detekciju kružnica temelji na nekoj vrsti HT algoritma.

HT je inicijalno dizajnirana za detekciju ravnih linija na slici, no razvijene su brojne modifikacije čiji je cilj detekcija proizvoljnih oblika, posebice kružnica [10]. U nastavku će biti predstavljeni HT algoritmi za detekciju ravnih linija i kružnica.

2.1.1. Detekcija ravnih linija iz zadanih točaka

U ovom poglavlju predstavit će se HT za detekciju ravnih linija iz zadanog skupa dvodimenzionalnih točaka P_i. Dobro poznata jednadžba pravca glasi

$$y = mx + b \tag{2.1}$$

gdje m označava nagib koji pravac zatvara s pozitivnim dijelom osi apscise, a b odsječak pravca na ordinati.

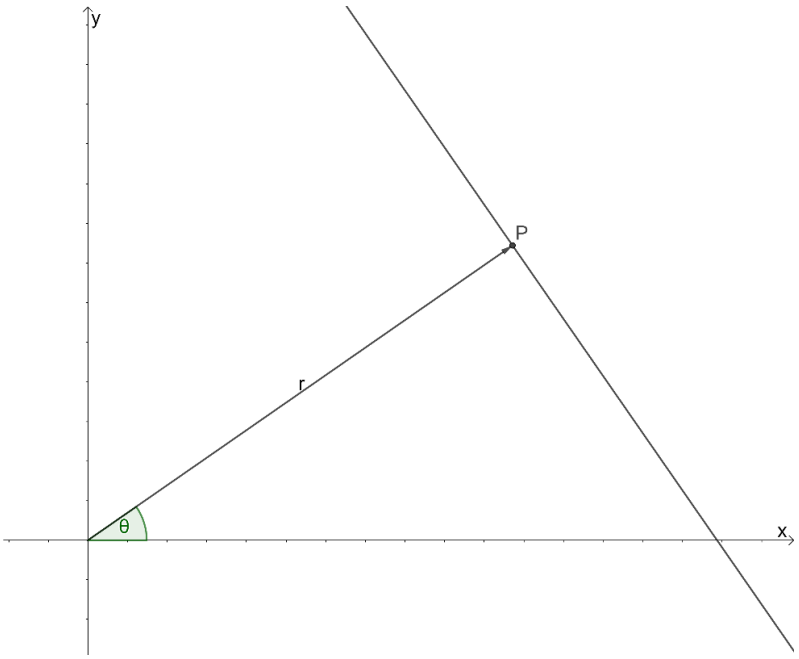
Međutim, budući da se pravci paralelni s ordinatom ne mogu predstaviti ovom jednadžbom, koristit ćemo jednadžbu pravca u polarnim koordinatama

$$y = -\frac{\cos(\theta)}{\sin(\theta)}x + \frac{r}{\sin(\theta)} \tag{2.2}$$

koja se može transformirati u

$$r = x\cos(\theta) + y\sin(\theta)\,. \tag{2.3}$$

U izrazima (2.2) i (2.3) r predstavlja udaljenost točke (x, y) od ishodišta, a θ kut radijus-vektora do točke na pravcu najbliže ishodištu (Slika 2.1). Bitno je primijetiti da se na ovaj način svaki pravac može predstaviti uređenim parom (θ, r).



Slika 2.1 Prikaz dvodimenzionalne točke u polarnim koordinatama

U nastavku će se koristiti ideja da se svaka točka u ravnini može predstaviti skupom pravaca koji se u njoj sijeku. Kako bi se primijenila ta ideja, skup pravaca koje prolaze točkom P_i u *prostoru točaka* (x, y) promatra se kao skup sinusoida u *prostoru parametara* (θ, r). Primjerice, za točku $P_1 = (x_1, y_1)$ skup svih pravaca koje kroz nju prolaze definiran je sinusoidom u prostoru parametara:

$$r = x_1\cos\theta + y_1\sin\theta \tag{2.4}$$

Uz pretpostavku da su sve točke P_i kolinearne, sinusoide svih zadanih točaka definirane izrazom (2.4) će se sjeći u prostoru parametara u točki (θ_0, r_0). Kada se ta točka u prostoru parametara pretvori u pravac u prostoru točaka, dobit će se pravac koji prolazi kroz zadane točke.

U realnim primjenama sve zadane točke obično nisu kolinearne, stoga su razvijene metode za pronalazak pravaca koji prolaze kroz najviše točaka. Uobičajeni postupak je na zadovoljavajuću preciznost kvantizirati parametre θ i r te napraviti dvodimenzionalnu mrežu kojoj su koordinate kvantizirane vrijednosti θ i r. Takva se mreža nadalje promatra kao dvodimenzionalno polje - *akumulator*. Koordinate akumulatora predstavljaju odgovarajuće vrijednosti uređenog para (θ, r), dok će pojedine vrijednosti u akumulatoru predstavljati brojač za taj par parametara. Za svaku točku (x_i, y_i) u prostoru točaka odgovarajuća sinusoida dobivena iz (2.4) unosi se u akumulator na način da se inkrementira brojač za sva polja akumulatora koja odgovaraju toj sinusoidi. Tim će postupkom vrijednost pojedinog polja dvodimenzionalnog akumulatora odgovarati broju sinusoida u prostoru parametara koje kroz njega prolaze. Kada se na ovaj način obrade sve točke, potrebno je još pronaći mjesta lokalnih maksimuma u akumulatorskom polju. Koordinate (θ_i, r_i) pronađenih maksimuma definirat će pravce koji prolaze kroz najviše početnih točaka.

2.1.2. Detekcija kružnica sa slike

Prvi korak koji treba izvršiti ako se HT algoritmima želi obraditi slika jest detekcija rubova na slici kako bi se izolirao snimljeni objekt. U literaturi se često susreće algoritam kojeg je predstavio J. Canny 1987. godine [11]. Ovaj i njemu srodni algoritmi za detekciju rubova oslanjaju se na računanje gradijenata intenziteta pojedinih piksela slike. Pikseli koji imaju veliki gradijent intenziteta, odnosno koji su znatno tamniji ili svjetliji od njima susjednih piksela, postaju kandidati za rubne točke. Ti se kandidati filtriraju koristeći algoritam za stanjivanje detektiranih rubova. Nadalje, preostale se rubne točke dijele u tri grupe: *jake rubne točke*, čiji je gradijent intenziteta veći od zadanog gornjeg praga; *slabe rubne točke*, čiji je gradijent intenziteta manji od donjeg praga i *srednje rubne točke*. Dalje se pretpostavlja da jake i srednje rubne točke dolaze od pravih rubova objekta i one automatski ulaze u završni rezultat. No, ostaje dilema dolaze li slabe rubne točke zaista od rubova objekta ili su rezultat normalnih varijacija svjetline na slici. Uobičajeni je postupak promatrati osam susjednih točaka za svaku slabu rubnu točku. Ukoliko među tim susjednim točkama postoji barem jedna jaka rubna točka, promatrana slaba rubna točka ulazi u

završni rezultat, a u suprotnom se ignorira. Opisanim algoritmom procijenjene rubne točke predstavljat će skup zadanih točaka P za HT algoritam.

U prethodnom poglavlju opisana metoda za detekciju pravaca može se proširiti i na detekciju drugih porodica krivulja. U tom slučaju računalna kompleksnost algoritma raste ovisno o broju parametara koje je potrebno detektirati. U nastavku će biti opisana modifikacija HT za detekciju kružnica.

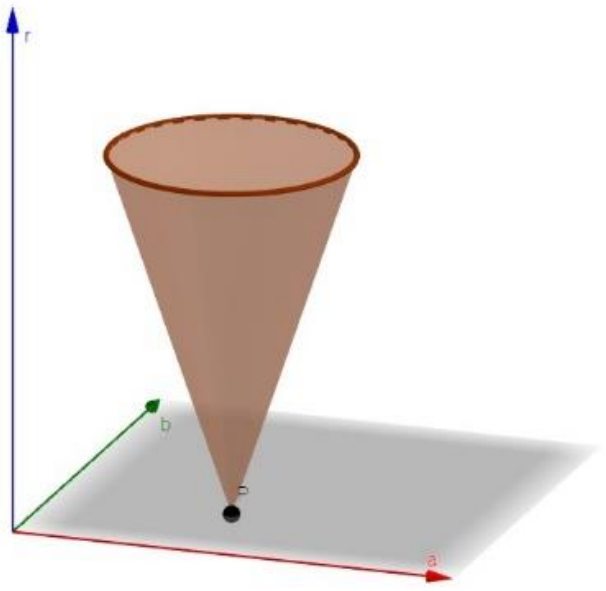
Kružnica se, u prostoru zadanih točaka, može opisati jednadžbom

$$(x-a)^2+(y-b)^2=r^2 \tag{2.5}$$

gdje točka (a,b) označava središte kružnice, a r njezin polumjer. U ovom slučaju, za zadanu točku $P_i=(x_i,y_i)$ skup svih kružnica koje prolaze kroz tu točku definiran je izrazom

$$(x_i-a)^2+(y_i-b)^2=r^2 \tag{2.6}$$

Ovdje je potrebno procijeniti tri parametra, a, b i r, stoga je prostor parametara trodimenzionalan. Prema jednadžbi (2.6) svaka se točka u prostoru točaka može preslikati u obrnuti stožac u prostoru parametara (Slika 2.2).



Slika 2.2 Preslikavanje točke u prostor parametara (a, b, r)

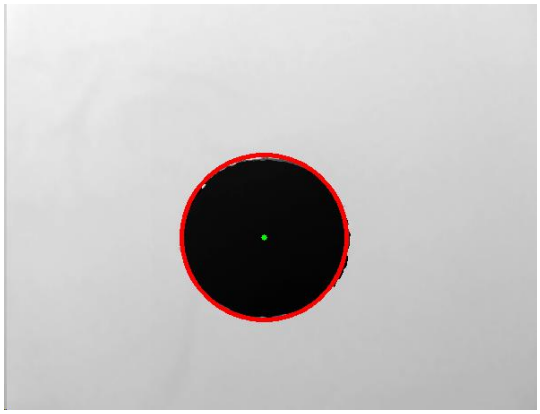
Analogno metodi opisanoj u prethodnom poglavlju gdje su se promatrala sjecišta sinusoida, ovdje je zadatak pronaći sjecišta stožaca za sve zadane točke. Također se kreira akumulatorska

matrica s time da će ona za ovaj slučaj biti trodimenzionalna. Pri tome je potrebno prethodno definirati maksimalni polumjer kružnica koje se pokušava detektirati te na zadovoljavajuću preciznost kvantizirati parametre a, b i r. Nakon inkrementalnog popunjavanja akumulatorske matrice obradom svih zadanih točaka, potrebno je pronaći njezine lokalne maksimume. Koordinate lokalnih maksimuma akumulatorske matrice odgovaraju parametrima detektiranih kružnica.

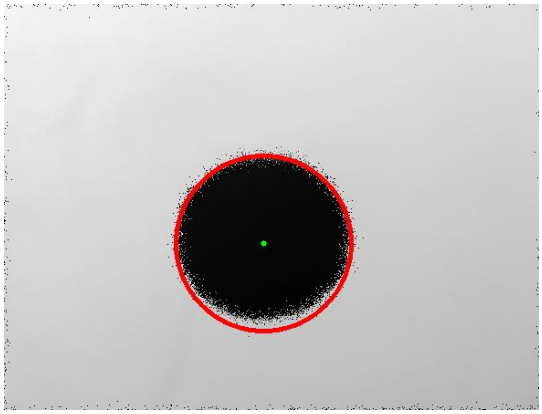
U literaturi se može naći i nekoliko varijacija klasične HT za detekciju kružnica. Najpoznatije su:

- probabilistička HT [12]: predlaže se nasumično uzorkovanje rubnih točaka umjesto promatranja svih rubnih točaka. Ovim se postupkom umanjuje veličina akumulatorske matrice i ubrzava izvršavanje algoritma, ali i umanjuje preciznost detekcije.
- nasumična (engl. *randomized*) HT [13]: umjesto preslikavanja svake rubne točke u trodimenzionalni parametarski prostor, autori ove metode predlažu preslikavanje nasumično odabranih *n* rubnih točaka u jednu točku u parametarskom prostoru. Na taj se način smanjuje akumulatorska matrica, skraćuje vrijeme izvršavanja i omogućuje korištenje parametarskih prostora viših dimenzija.
- neizrazita (engl. *fuzzy*) HT [14]: koriste se elementi neizrazite (engl. *fuzzy*) logike kako bi se detektirale kružnice čije rubne točke donekle odstupaju od točaka savršene kružnice.

Na Slici 2.3 prikazana je jasna slika objekta čiju je površinu potrebno procijeniti i detektirana kružnica dobivena klasičnom HT metodom. Vidljivo je da je algoritam precizno detektirao objekt na slici. Međutim, ako se isti algoritam izvrši na slici s aditivnim šumom iz Gaussove distribucije (Slika 2.4) može se primijetiti da kvaliteta detekcije opada, odnosno da detektirana kružnica ne odgovara u potpunosti objektu na slici.



Slika 2.3 Rezultat HT metode za detekciju kružnice na jasnoj slici



Slika 2.4 Rezultat HT metode za detekciju kružnice na slici s aditivnim šumom

2.2. EDCircles

C. Akinlar i C. Topal u svom radu iz 2012. godine predstavljaju EDCircles [8], algoritam za detekciju kružnica i elipsi sa slike. Algoritam se često spominje u literaturi, jer učinkovito detektira jasno prikazane kružne i eliptične objekte, a za izvršavanje ne zahtjeva podešavanje niti jednog parametra, preciznije, koristi skup internih parametara čija je vrijednost fiksna. Predloženi algoritam može se raščlaniti na 6 dijelova:

i. Detektiranje segmenata rubova

U ovom dijelu autori koriste vlastiti ED algoritam [15] koji detektira rubove objekta drukčije nego tradicionalni algoritmi kao što je primjerice Canny [11]. ED prvo identificira skup točaka na slici i zatim ih spaja koristeći algoritam koji autori nazivaju *smart routing procedure*. ED kao rezultat ne daje samo skup detektiranih točaka koje predstavljaju rubove, već i skup rubnih segmenata – ulančanih rubnih točaka. Budući da ED algoritam zahtjeva podešavanje mnogo ulaznih parametara, autori su ga modificirali i stvorili EDPF algoritam [16] u kojem su ulazni parametri fiksirani na ekstremnim vrijednostima. U tom slučaju algoritam daje mnogo lažno pozitivnih rubova, no oni se nadalje evaluiraju korištenjem Helmholtzovog principa [17] kako bi se eliminirali lažno pozitivni rezultati i ostavili samo perceptualno smisleni rubni segmenti. Nadalje se promatraju samo zatvoreni rubni segmenti, tj. oni segmenti čija je posljednja točka susjedna početnoj. Svaki zatvoreni rubni segment opiše se kružnicom koristeći metodu najmanjih kvadrata [18]. Ako je odstupanje te kružnice od točaka rubnog segmenta zadovoljavajuće malo, taj se segment dodaje u popis kandidata za kružnice. Preostali zatvoreni rubni segmenti pokušavaju se

istom metodom opisati elipsom. Oni za koje je to, uz zadovoljavajuće malu grešku, moguće dodaju se u popis kandidata za elipse. Kandidati za kružnice i elipse uklanjaju se iz popisa segmenata i dalje se ne procesuiraju. Preostali zatvoreni i svi otvoreni rubni segmenti ulaze u daljnju obradu.

ii. Pretvorba rubnih segmenata u linije

Ideja ovog dijela algoritma je promatrati svaki preostali rubni segment od njegovog početka kao ravnu liniju i produljivati tu liniju sve dok ona bude na zadovoljavajuće maloj udaljenosti od rubnog segmenta. Kada ta udaljenost postane prevelika, linija se završava i istim postupkom započinje nova, dok se ne dođe do kraja promatranog rubnog segmenta. Postupak se detaljnije opisuje u radu [19] istih autora.

iii. Detektiranje kružnih lukova

Autori definiraju kružni luk kao skup od barem tri susjedne linije koje su rotirane u istom smjeru. Koristeći tu definiciju kružni luk se detektira na način da se promatraju sve linije koje čine jedan rubni segment i računaju kutovi među susjednim linijama te smjer njihove rotacije. Ako se barem tri susjedne linije rotiraju u istom smjeru i ako je kut među njima unutar definiranih granica (6° - 60°), te linije postaju kandidati za kružni luk. Svaki kandidat za kružni luk pokušava se opisati kružnicom i, ako je to moguće uz zadovoljavajuće malu grešku, dodaje se u popis kružnih lukova. Ako to nije moguće, uzima se kratki luk od samo tri početne točke promatrane linije koji se potom postupno proširuje kako bi se formirala kružnica. Postupak se ponavlja dok god se tako dobivena kružnica nalazi zadovoljavajuće blizu detektiranom rubnom segmentu. Kada to više nije slučaj, započinje se s novim kratkim lukom od sljedeće tri točke i postupak se ponavlja do kraja segmenta. Opisani algoritam na kompleksnim slikama može generirati stotine kružnih lukova koji ulaze u daljnju obradu.

iv. Detektiranje kandidata za kružnice spajanjem kružnih lukova

Nakon izračunavanja kružnih lukova, idući je zadatak spojiti detektirane kružne lukove u kandidate za kružnice. Prvi korak pri tom zadatku je sortirati sve lukove po duljini silaznim redoslijedom i pokušati ih produljivati krenuvši od najduljeg. Motivacija za ovaj postupak dolazi od činjenice da je najdulji luk najbliži po svom obliku punoj kružnici, pa je logično prvo njega pokušati produljiti u kružnicu. U samom postupku produljivanja traže se kružni lukovi koji imaju

bliske polumjere i središta. Konkretno, polumjeri dvaju kružnih lukova smiju se razlikovati za najviše 25%, a udaljenost njihovih središta smije biti najviše 25% duljine polumjera. Poštujući te uvjete formiraju se skupovi lukova – kandidata za produljenje početnog dugačkog luka. Sljedeći je zadatak za svaki takav skup kandidata odlučiti treba li ga dodati u skup kandidata za kružnice. Kao kriterij za tu odluku prvo se, poštujući kriterij najmanjih kvadrata, u skup kandidata smješta kružnica. Potom se provjerava je li zbroj opsega lukova-kandidata od kojih je formirana kružnica veći od 50% opsega te kružnice. Ako je taj uvjet ispunjen, taj se skup kružnih lukova dodaje u popis kandidata za kružnice, a u suprotnom u popis kandidata za elipse.

v. Detektiranje kandidata za elipse spajanjem kružnih lukova

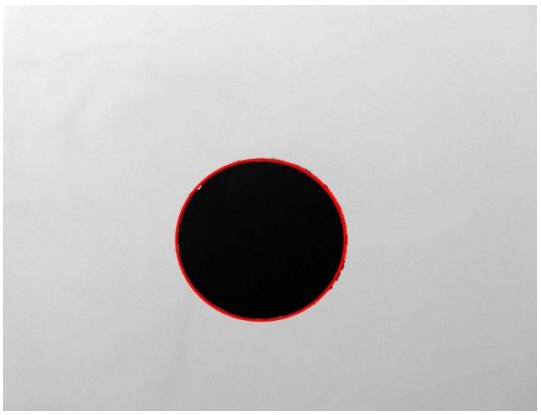
Postupak detektiranja kružnih lukova - kandidata za elipse vrlo je sličan ranije opisanom postupku detektiranja kandidata za kružnice. Preostali kružni lukovi ponovo se sortiraju po duljini i nadalje grupiraju po kriteriju bliskosti polumjera i središta, no ovdje su ti uvjeti relaksirani: sada se polumjeri dvaju kružnih lukova smiju razlikovati za najviše 50%, a udaljenost njihovih središta smije biti najviše 50% duljine polumjera. Dok se u prošlom koraku u svaki skup kandidata opisivao kružnicom, ovdje se on opisuje elipsom [20]. Ako je odstupanje te elipse od kružnih lukova zadovoljavajuće maleno, taj se skup lukova nastavlja analizirati, a u suprotnom se odbacuje. Dalje se, slično kao u prethodnom koraku algoritma, provjerava čini li zbroj opsega promatranih lukova barem 50% opsega cijele elipse. Skupovi kružnih lukova koji prođu i tu provjeru ulaze u posljednji korak algoritma.

vi. Validacija kružnica i elipsi korištenjem Helmholtzovog principa

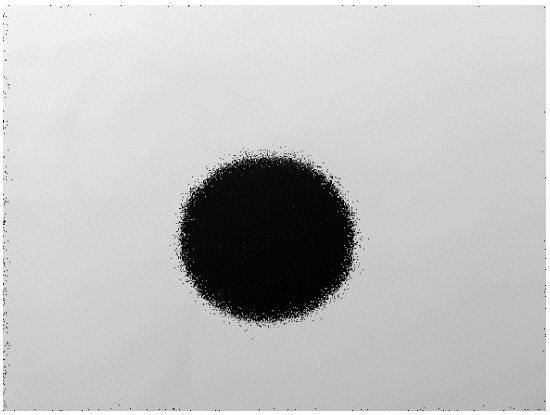
Činjenica da se nekoliko kružnih lukova može spojiti u kružnicu ili elipsu ne znači nužno da ta kružnica ili elipsa predstavlja objekt na početnoj slici. Zato se, kao posljednji korak EDCircles algoritma, provodi validacija koristeći modifikaciju Helmholtzovog principa [17, 21, 22]. U osnovi, Helmholtzov princip tvrdi da kako bi geometrijska struktura bila perceptualno smislena, vjerojatnost pojavljivanja te strukture u slučajnoj situaciji mora biti niska. Ovdje je riječ o tzv. *a contrario* pristupu u kojem se objekti promatraju kao stršeće vrijednosti u modelu pozadine. U modifikaciji ovog principa koja se koristi u EDCircles algoritmu autori, po idejama iznesenim u [21, 22], promatraju *usmjerenost točaka* na potencijalnoj elipsi ili kružnici u odnosu na okolne točke. Računa se broj lažnih detekcija (engl. *number of false alarms* – NFA) za svaku potencijalnu

elipsu i kružnicu. Ukoliko je taj broj manji od prethodno definirane granične vrijednosti, promatrana elipsa ili kružnica prihvaća se kao detektirani objekt na slici.

Pokusi pokazuju da EDCircles algoritam precizno i brzo detektira kružne i eliptične objekte kada su oni jasno prikazani na slici (Slika 2.5). Međutim, pokazat će se da algoritam za detekciju rubova ne radi zadovoljavajuće dobro kada je na slici prisutan šum. Slika 2.6 prikazuje objekt s aditivnim šumom iz Gaussove distribucije na kojoj EDCircles metoda uopće ne detektira objekt.



Slika 2.5 Rezultat EDCircles metode za detekciju elipse na jasnoj slici



Slika 2.6 Rezultat EDCircles metode za detekciju kružnice na slici s aditivnim šumom

2.3. Fornaciari

Michele Fornaciari i dr. u radu [9] iz 2014. god. predstavljaju algoritam za detekciju elipsi koji koristi neke ideje iz algoritma EDCircles, kao i Houghovu transformaciju za procjenu nekih parametara elipsi.

Algoritam prvo detektira rubove objekta korištenjem Canny detektora [11]. Slično kao u EDCircles algoritmu, detektirane rubne točke se, koristeći informacije o njihovom gradijentu, grupiraju u eliptične lukove. Sobelovim operatorom [23] određuje se *usmjerenost* svakog luka kao usmjerenost rubnih točaka od kojih je sačinjen. Lukovi se potom klasificiraju po svojoj konveksnosti u dvije grupe: otvoreni *prema gore* i otvoreni *prema dolje*. Uz pomoć podataka o usmjerenosti i konveksnosti, svaki se eliptični luk može grupirati u jedan od četiri kvadranta koordinatnog sustava sa ishodištem u središtu zamišljene elipse koja proširuje promatrani luk.

U drugom dijelu algoritma grupiraju se *trojke* eliptičnih lukova u koje će biti odabrana tri luka iz različitih kvadranata. Pri tome se vodi računa o kriterijima vezanim uz konveksnost, međusobni

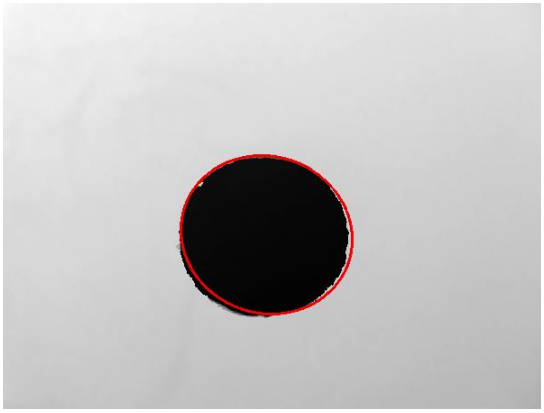
položaj i bliskost središta potencijalnih lukova. Ovako formirane trojke predstavljat će kandidate za elipse. Budući je njihovo središte već izračunato, preostaje izračunati još tri parametra (dvije poluosi i kut rotacije) kako bi se parametarski potpuno definirala elipsa – za tu se svrhu koristi Houghova transformacija.

U završnoj fazi kandidati za elipse se verificiraju računanjem udaljenosti između detektiranih rubnih točaka i točaka na dobivenoj elipsi. Ako je ta udaljenost prevelika, elipsa se odbacuje.

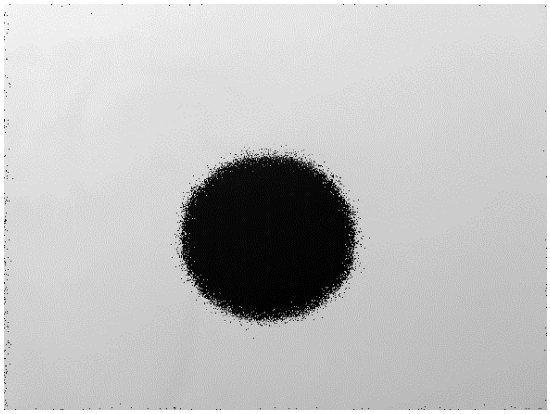
Moguće je da se od različitih trojki formira ista elipsa. Iz tog se razloga sve dobivene elipse segmentiraju. Ukoliko se utvrdi da dvije ili više elipsi pripadaju istom segmentu, one se spajaju i generiraju jednu detektiranu elipsu.

Naglasak pri razvijanju ovog algoritma je bio na njegovoj brzini: autori su imali u vidu njegovo izvršavanje na pametnim telefonima koji su, u pravilu, slabije računske snage od osobnih računala. U slučaju jasne slike s elipsoidnim objektima on daje kvalitetne rezultate, no kvaliteta detekcije objekata je lošija kada su prikazani objekti vrlo izduženi, imaju fragmentirane rubove, te u prisustvu šuma na slici.

Algoritam daje prihvatljive rezultate u slučaju jasnih slika (Slika 2.7), dok kod nejasnih slika s aditivnim šumom ne uspijeva detektirati objekt (Slika 2.8).



Slika 2.7 Rezultat Fornaciari metode za detekciju elipse na jasnoj slici



Slika 2.8 Rezultat Fornaciari metode za detekciju elipse na slici s aditivnim šumom

3. Statistički model za procjenu širine uniformne distribucije za podatke zagađene aditivnom greškom

U ovom poglavlju bit će opisan model jednostavnog slučajnog uzorka iz distribucije slučajne varijable X koja u sebi sadrži aditivnu grešku, a dana je izrazom

$$X = U + \varepsilon. \tag{3.1}$$

Ovdje je slučajna varijabla *U* uniformno distribuirana na intervalu $[-a, a]$, $a > 0$ i ima funkciju gustoće

$$f_U(x; a) = \begin{cases} \frac{1}{2a}, & x \in [-a, a] \\ 0, & \text{inače.} \end{cases} \tag{3.2}$$

ε je slučajna varijabla kojom je modelirana pogreška mjerenja, ima očekivanje 0 i nezavisna je od U. Njezina funkcija gustoće ovisi o pretpostavljenoj distribuciji – primjerice, za normalnu distribuciju slučajne varijable ε funkcija gustoće glasi

$$f_\varepsilon^N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}, \tag{3.3}$$

a za Laplaceovu distribuciju

$$f_\varepsilon^L(x) = \frac{1}{2\lambda} e^{-|x|/\lambda}. \tag{3.4}$$

Ovaj model bit će korišten kao temelj za procjenu duljine presjeka objekta s pravcem i površine kružnog ili elipsoidnog objekta, što je tema ovog istraživanja. Model je parametarski te sadrži najviše dva nepoznata parametra: polovinu duljine nosača uniformne distribucije a i varijancu greške σ^2. Za njihovu se procjenu mogu primijeniti standardne tehnike za procjenu parametara za koje se zna da daju kvalitetne procjenitelje.

Statistički model za procjenu širine uniformne distribucije za podatke zagađene aditivnom greškom

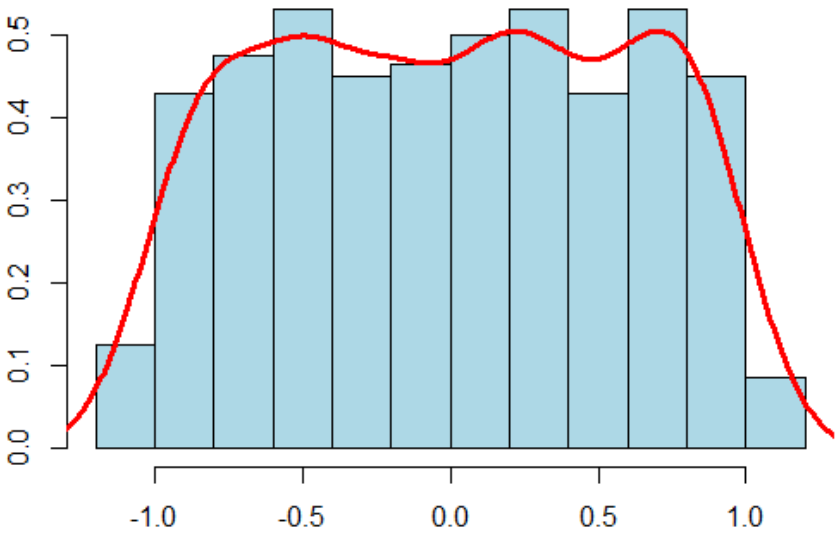
Postupak uklanjanja grešaka i procjene gustoće slučajne varijable U poznat je u literaturi kao dekonvolucijski problem. Razvijeno je nekoliko neparametarskih metoda za rješavanje ovog problema [24], no pokazalo se da one, iako daju dobre rezultate, u praksi vrlo sporo konvergiraju.

U radovima [25, 26, 27, 28, 29] pokazano je da funkcija gustoće slučajne varijable X ima vrlo prikladan oblik za analizu procjenitelja za a i σ^2 metodom maksimalne vjerodostojnosti (engl. *maximum likelihood* – ML) i metodom momenata (engl. *method of moments* – MM).

Ako je F funkcija distribucije slučajne varijable ε, funkcija gustoće slučajne varijable X dana je izrazom

$$f_X(x;a) = \frac{1}{2a}(F(x+a) - F(x-a)). \tag{3.5}$$

Histogram 1000 simuliranih podataka iz uniformne distribucije na intervalu $[-1,1]$ s dodanom normalnom greškom $\sigma^2 = 0.1$ prikazan je na Slici 3.1.



Slika 3.1 Histogram simuliranih podataka i neparametarski procijenjena gustoća

U ranije spomenutim radovima razvijena je parametarska metoda za rješavanje ovoga problema, odnosno za procjenu poluširine uniformne distribucije (parametar a u izrazu (3.2)) kada su

Statistički model za procjenu širine uniformne distribucije za podatke zagađene aditivnom greškom

podaci mjereni s normalnom aditivnom greškom. Varijanca σ^2 slučajne varijable ε u praksi obično nije poznata, pa se ona procjenjuje ML ili MM metodom.

Pokazalo se da model s normalnom aditivnom greškom ne daje dovoljno kvalitetne rezultate procjene kada je u podacima prisutno mnogo stršećih vrijednosti (engl. *outliers*). Iz tog je razloga u radu [30] razvijena inačica modela s aditivnom greškom iz Laplaceove distribucije, a u posljednje se vrijeme analiziraju i modeli s općenitim tipom simetrične greške. Na temelju tih rezultata može se izabrati tip greške koji daje dobre rezultate u prisustvu stršećih vrijednosti u slučajnom uzorku varijable X.

U članku [28] pokazano je da je ML procjenitelj za oba parametra koja se procjenjuju asimptotski normalan i superioran u odnosu na MM procjenitelja u modelu s normalnom greškom. Međutim, taj članak ujedno sugerira kako neki skupovi podataka mogu dovesti do problema prilikom rješavanja optimizacijskog problema, u smislu da rješenje ne postoji. Taj problem je najviše izražen prilikom istovremene procjene a i σ^2 ML metodom. Iz tog su razloga analizirani i procjenitelji za σ^2 MM metodom kao alternativa.

Kada se koristi ML metoda, oblik maksimizacijske funkcije $\ell(a,\sigma)$ za procjenu parametara ovisi o tipu distribucije greške. Ako su podaci $x_1, x_2 \ldots, x_n$ realizacija jednostavnog slučajnog uzorka iz distribucije slučajne varijable X, maksimizacijska funkcija za normalnu grešku $\varepsilon \sim \mathcal{N}\left(0,\sigma^2\right)$ je oblika

$$\ell(a,\sigma) = -n\log(2a) + \sum_{i=1}^{n} \log\left(\Phi\left(\frac{a - x_i}{\sigma} \right) - \Phi\left(\frac{-a - x_i}{\sigma} \right) \right), \tag{3.6}$$

gdje je Φ funkcija distribucije standardne normalne slučajne varijable

$$\Phi(x) = \int_{-\infty}^{x} \varphi(t)\mathrm{d}t, \qquad \varphi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}. \tag{3.7}$$

Za grešku iz Laplaceove distribucije $\varepsilon \sim \mathcal{L}\left(0,\lambda\right)$ maksimizacijska funkcija je oblika

$$\ell(a,\lambda) = \begin{cases} -n\log(2a) + \sum_{i=1}^{n} -\frac{|x_i|}{\lambda} + \log\sinh\frac{a}{\lambda}, & |x_i| \geq a, \\ -n\log(2a) + \sum_{i=1}^{n} \log\left(1 - e^{-\frac{a}{\lambda}} \cosh\frac{x_i}{\lambda}\right), & |x_i| < a, \end{cases} \tag{3.8}$$

gdje je parametar λ vezan uz varijancu σ^2.

Obzirom na oblik logaritma funkcije vjerodostojnosti $\ell(a,\sigma)$, za procjenu parametara, tj. određivanje vrijednosti u kojima $\ell(a,\sigma)$ postiže maksimum, bit će potrebno koristiti numeričke metode za optimizaciju.

Za općenitiji slučaj greške korisno je parametrizirati model korištenjem oblika

$$X = U + \sigma\varepsilon, \tag{3.9}$$

gdje je ε slučajna varijabla greške koja je centrirana, simetrična i standardizirana ili prikladno parametrizirana.

Ako se za procjenu a i σ^2 koristi MM, procjenitelje za parametre može se izračunati koristeći drugi i četvrti uzorački i teorijski moment uz prethodni odabir distribucije. Tako su primjerice za normalnu grešku $\varepsilon \sim \mathcal{N}\left(0,\sigma^2\right)$ MM procjenitelji dani kao

$$\hat{a}_{MM} = \sqrt[4]{\frac{15}{2}\left(3m_2^2 - m_4\right)}\ , \tag{3.10}$$

$$\hat{\sigma}_{MM} = \sqrt{m_2 - \sqrt{\frac{5}{6}\left(5m_4 - 9m_2^2\right)}}\ , \tag{3.11}$$

gdje su m_2 i m_4 drugi i četvrti uzorački moment:

$$m_2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2, \qquad m_4 = \frac{1}{n}\sum_{i=1}^{n} x_i^4.$$

U slučaju Laplaceove greške $\varepsilon \sim \mathcal{L}\left(0,\lambda\right)$ MM procjenitelji su definirani kao

$$\hat{a}_{MM} = \sqrt{5m_2 - \sqrt{5}\sqrt{m_4 - m_2^2}}\ , \tag{3.12}$$

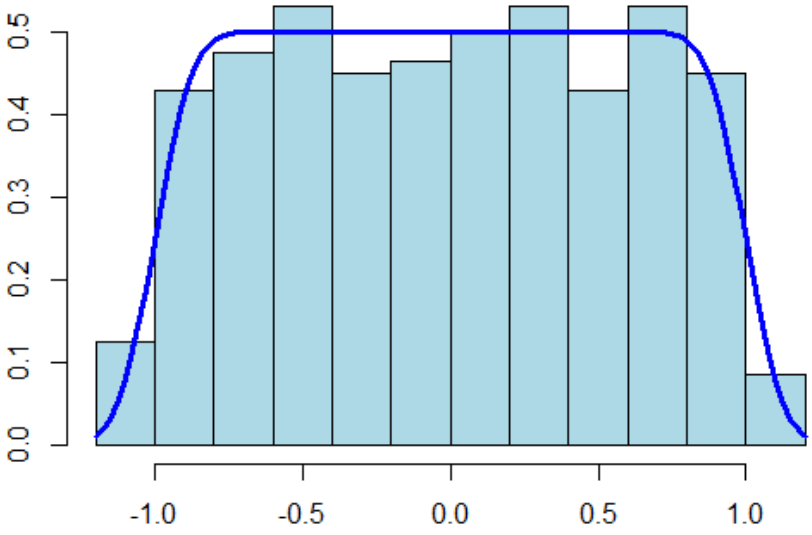
$$\hat{\lambda}_{MM} = \frac{1}{\sqrt{6}}\sqrt{-2m_2 + \sqrt{5}\sqrt{m_4 - m_2^2}} \tag{3.13}$$

gdje su m_2 i m_4 također drugi i četvrti uzorački moment.

Budući da su MM procjenitelji u izrazima (3.10) - (3.13) eksplicitno definirani, uobičajeno ih je koristiti kao početnu aproksimaciju za ML metodu.

Statistički model za procjenu širine uniformne distribucije za podatke zagađene aditivnom greškom

Slika 3.2 prikazuje histogram podataka sa Slike 3.1 i funkciju gustoće koja je dobivena procjenom parametra a ML metodom.



Slika 3.2 Histogram simuliranih podataka i funkcija gustoće dobivena procjenom parametra a ML metodom

Parametar a je ovdje procijenjen na $a = 1.009$. Kako bi se ilustrirala kvaliteta procijene može se upotrijebiti studija iz [29] po kojoj za 1000 simuliranih podataka i standardnu devijaciju $\sigma = 0.1$ standardna greška procjenitelja iznosi $\sqrt{0.0001} = 0.01$. Ako je standardna devijacija aditivne greške jednaka 10% vrijednosti polovine duljine nosača uniformne distribucije, pouzdani interval parametra a od 95% iznosi $1.009 \pm 2 \cdot 0.01 = (0.989, 1.029)$. Uz pretpostavku da se vrijednosti procjenitelja zaokružuju na jednu decimalu, može se zaključiti da metoda daje realne vrijednosti procijenjenog parametra u ovom slučaju.

4. Procjena duljine presjeka objekta s pravcem

Koristeći rezultate spomenute u prethodnom poglavlju razvijeni su algoritmi za procjenu duljine presjeka proizvoljnog pravca i objekta na slici snimljenoj s aditivnom greškom. Nadalje, ti su algoritmi upotrjebljeni za izradu algoritma za procjenu površine kružnog ili elipsoidnog objekta, također snimljenog s aditivnom greškom. Bit će pokazano kako uz pomoć tih algoritama procijeniti duljinu presjeka proizvoljnog pravca i objekta na slici, što će biti prvi korak u algoritmu za procjenu površine kružnog ili elipsoidnog objekta.

Problem procjene duljine presjeka objekta i pravca ilustriran je na Slici 4.1. Na toj je slici prikazana rendgenska snimka šake iz koje je potrebno procijeniti širinu kosti zapešća, odnosno duljinu presjeka pravca zelene boje i kosti.

Slika 4.1 Problem procijene duljine presjeka pravca i objekta

Može se primijetiti da navedeni problem nije trivijalan jer su na slici, osim bijelih područja koja predstavljaju kost i crnih područja koja predstavljaju pozadinu, zabilježena i rubna siva područja različitih nijansi koja označavaju hrskavicu, mišiće, potkožno masno tkivo i sl. Ako se problem promatra iz aspekta slike snimljene s aditivnom greškom (šumom), može se reći da spomenuta

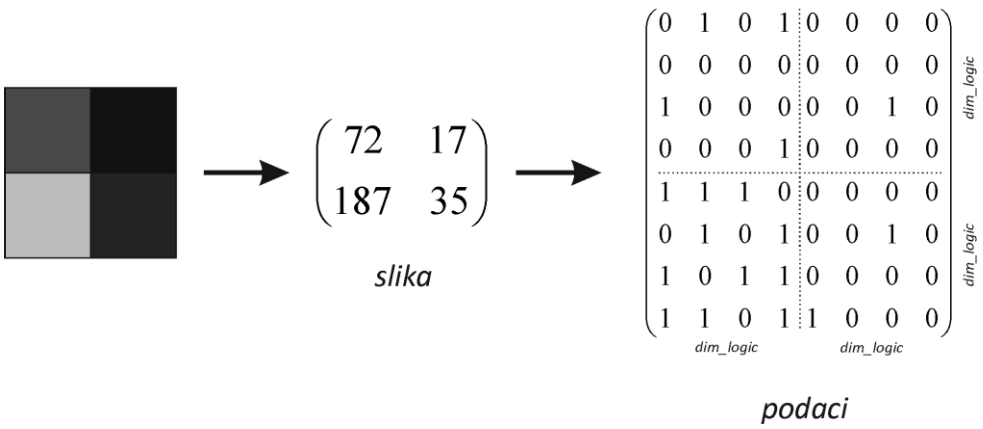
siva područja predstavljaju šum na slici i potrebno ih je eliminirati prilikom procjene duljine presjeka.

Prvi problem kojeg treba riješiti je transformacija podataka očitanih sa slike u oblik pogodan za računalnu obradu [31]. U tu svrhu kreirat će se matrica *slika* dimenzije *visina_slike* $\times$ *širina_slike*, gdje te veličine označavaju dimenzije početne slike izražene u pikselima. Ta se matrica ispunjava cjelobrojnim vrijednostima 0 do 255 na način da je svaki piksel slike predstavljen jednom vrijednošću u matrici *slika* – potpuno crnim pikselima pridružuje se vrijednost 0, a potpuno bijelim 255.

Sljedeći je zadatak kreiranje matrice logičkih vrijednosti *podaci* na temelju matrice *slika*. Ovdje se koristi ideja predstavljanja svakog piksela slike kvadratnom matricom logičkih vrijednosti dimenzije *dim_logic* $\times$ *dim_logic*. U takvu se logičku matricu potom uniformno rasporedi *n* logičkih stanja 1, gdje je *n* proporcionalan svjetlini promatranog piksela, tj. pripadajućoj vrijednosti u matrici *slika*. Za potpuno crni piksel logička matrica koja ga predstavlja imat će sve vrijednosti 0, a potpuno bijeli piksel predstavit će se logičkom matricom u potpunosti ispunjenom vrijednostima 1.

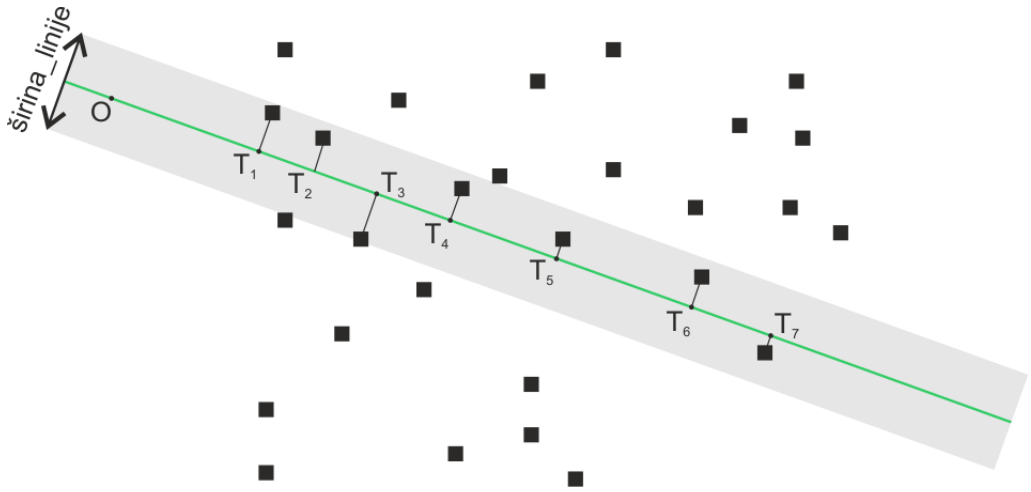
Matrica *podaci* se na kraju dobije spajanjem logičkih matrica za sve piksele na slici, pa je dimenzija matrice *podaci* jednaka *visina_slike* $\cdot$ *dim_logic* $\times$ *širina_slike* $\cdot$ *dim_logic*.

Opisani je algoritam predstavljen za minijaturnu sliku 2×2 piksela na Slici 4.2. Na toj je slici parametar *dim_logic* postavljen na vrijednost 4, pa je pripadajuća matrica *podaci* dimenzije 8 $\times$ 8.



Slika 4.2 Transformacija početne slike u matricu logičkih podataka

Na Slici 4.3 matrica *podaci,* generirana prethodno opisanim postupkom, predstavljena je bijelim i crnim kvadratima: bijeli, na slici nevidljivi, kvadrati označavaju logička stanja 0, a crni logička stanja 1. Zelena linija predstavlja pravac koji siječe objekt i čiju duljinu presjeka s objektom je potrebno procijeniti. Sivo područje obuhvaća sve točke (logičke varijable) koje će se pri tom postupku uzimati u obzir, a određeno je proizvoljnim parametrom *širina_linije*. Ovdje se koristi ideja ortogonalnog projiciranja svih točaka u interesnom području na pravac. Na taj se način stvara skup projiciranih točaka T sačinjen od točaka $T_1, T_2, T_3, \ldots, T_n$



Slika 4.3 Projekcija matrice podaci *na pravac koji presijeca objekt*

Iz ovako dobivenog skupa točaka T sada treba izračunati vektor euklidskih udaljenosti točaka od neke proizvoljne fiksne točke na pravcu, na Slici 4.3 označene s O

$$\mathbf{d} = \{d_2(O, T_i), \quad i = 1, \ldots, n\}. \tag{4.1}$$

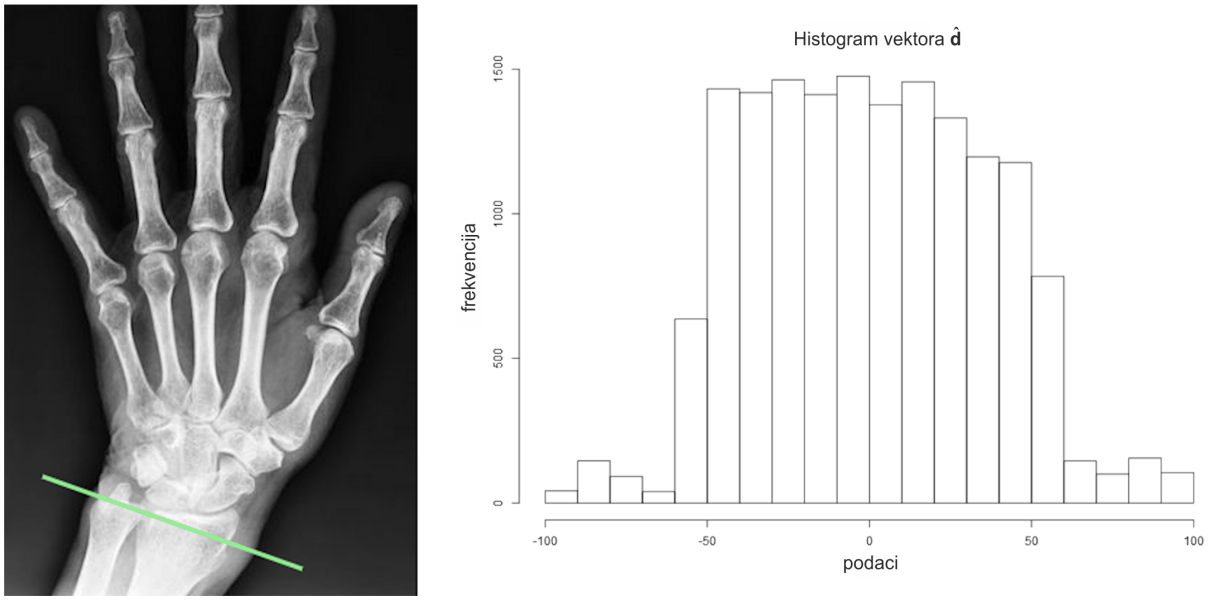
Nakon izračuna srednje vrijednosti vektora udaljenosti

$$c = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{4.2}$$

potrebno je izračunati centrirani vektor udaljenosti

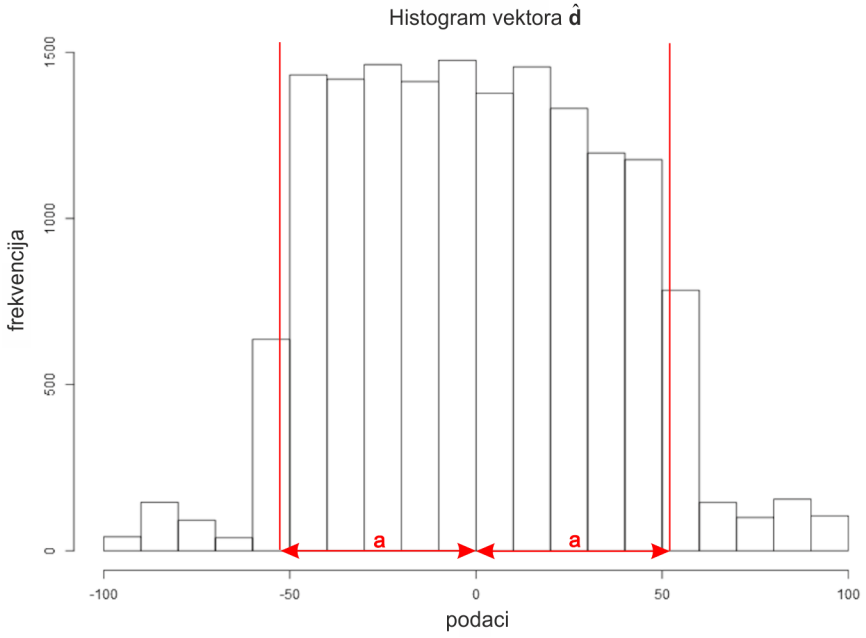
$$\hat{\mathbf{d}} = \{d_i - c, \quad i = 1, \ldots, n\}. \tag{4.3}$$

Histogram vektora $\hat{\boldsymbol{d}}$ za primjer sa Slike 4.1 prikazan je na Slici 4.4.



Slika 4.4 Histogram centriranog vektora udaljenosti $\hat{\boldsymbol{d}}$

Može se primijetiti da distribucija podataka u ovom histogramu nalikuje grafu procijenjene gustoće vjerojatnosti slučajne varijable X prikazanom na Slici 3.1. Konačno, vektor $\hat{\boldsymbol{d}}$ se koristi kao skup ulaznih podataka za parametarsku metodu za procjenu širine uniformne distribucije, opisanu u poglavlju 3. Navedena metoda kao rezultat daje procjenu širine uniformne distribucije (parametar a u izrazu (3.2)), kao što je prikazano na Slici 4.5.



Slika 4.5 Procjena širine uniformne distribucije za rendgensku sliku šake

Ovako dobiveni parametar a izračunat je u *prostoru logičkih varijabli* matrice *podaci*. Kako bi se u konačnici dobila tražena informacija, tj. duljina presjeka objekta s pravcem, potrebno je parametar a translatirati u *prostor slike,* odnosno izraziti ga u pikselima. Budući je svaki piksel predstavljen u matrici *podaci* s *dim_logic* $\times$ *dim_logic* logičkih vrijednosti, procijenjena duljina presjeka objekta s pravcem izražena u pikselima je:

$$a_{px} = \frac{2a}{dim\_logic}. \tag{4.4}$$

U svrhu dobivanja predodžbe procijenjene duljine na primjeru sa Slike 4.1, izračunata vrijednost a_{px} je na Slici 4.6 predstavljena dužinom crvene boje. Pri pozicioniranju navedene dužine na početni pravac, njezino polovište smješta se na točku koja odgovara srednjoj vrijednosti vektora udaljenosti c iz izraza (4.2).

Slika 4.6 Procijenjena duljina presjeka objekta s pravcem

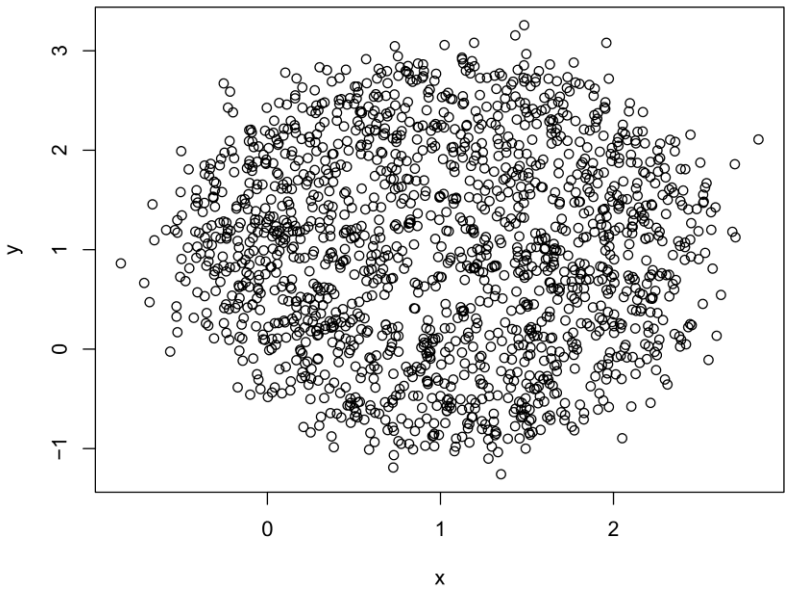
5. Problem procjene površine kružnog ili elipsoidnog objekta

Problem koji se nadovezuje na procjenu duljine presjeka objekta s pravcem jest problem procjene površine objekta. U poglavlju 2 opisane su neke od u literaturi najzastupljenijih metoda za rješavanje tog problema koje detektiraju i procjenjuju površine kružnih i elipsoidnih objekata na jasnim slikama. Međutim, opisanim metodama u pravilu opada kvaliteta detekcije za slike na kojima je prisutno puno šuma. U tim je slučajevima površina objekta nepreciznije procijenjena, ili opisane metode uopće ne detektiraju objekt na slici.

5.1. Procjena površine numerički zadanog objekta

Pretpostavimo da je kružni ili elipsoidni objekt čiju je površinu potrebno procijeniti zadan numerički, skupom dvodimenzionalnih točaka (Slika 5.1)

$$D = \{(x_i, y_i), \quad i = 1, \ldots, n\}. \tag{5.1}$$



Slika 5.1 Objekt zadan skupom točaka

Algoritam za procjenu površine na ovaj način zadanog objekta sastoji se od dva koraka: procjena skupa rubnih točaka i optimalno smještanje elipse ili kružnice u procijenjene rubne točke.

5.1.1. Procjena skupa rubnih točaka

U ovom koraku skup točaka D bit će podijeljen u više horizontalnih i vertikalnih pruga [27, 31]. *ALGORITAM 1* definira grupiranje točaka na horizontalne pruge, tj. pruge paralelne s x-osi koordinatnog sustava.

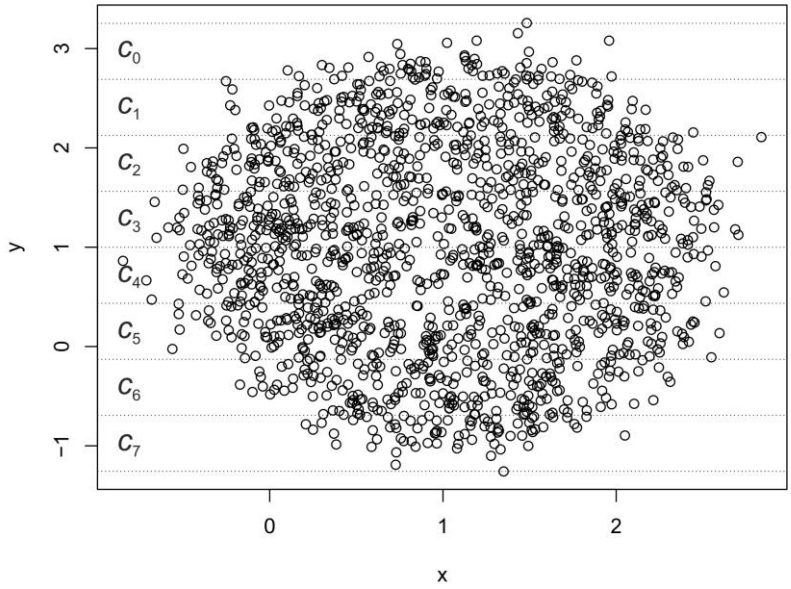
ALGORITAM 1 *(podjela točaka na horizontalne i vertikalne pruge)*

1. Odabrati cijeli broj $m < n$ i realne brojeve $\eta_1 < \eta_2 < \ldots < \eta_m$ takve da
   i. $\eta_1 \le \min\{y_i : i = 1,\ldots,n\},\ \max\{y_i : i = 1,\ldots,n\} \le \eta_m$ i
   ii. $C_k := \left\{(x_i, y_i) \in D : y_i \in \left[\eta_k, \eta_{k+1}\right]\right\},\ k = 1,\ldots m-1$ bude neprazan skup (Slika 5.2).
2. Odrediti srednje vrijednosti skupova C_k s obzirom na x-os

$$c_k = \frac{1}{|C_k|} \sum_{(x_i, y_i) \in C_k} x_i,\ k = 1,\ldots,m-1.$$

3. Odrediti skupove $\overline{C_k}$ koji se sastoje od x komponente udaljenosti svake točke iz skupa C_k od središnje točke toga skupa c_k

$$\overline{C_k} := \left\{x_i - c_k : (x_i, y_i) \in C_k\right\},\ k = 1,\ldots,m-1$$



Slika 5.2 Podjela točaka kojima je zadan objekt na horizontalne pruge

Opisanim algoritmom generira se $m-1$ skupova jednodimenzionalnih podataka – $\overline{C_k}$. Za polazišne podatke, odnosno skup dvodimenzionalnih točaka koji opisuje objekt, histogram svakog od skupova $\overline{C_k}$ bit će nalik onome prikazanom na Slici 4.4. Može se primijetiti da su takvi podaci pogodni za primjenu parametarske metode za procjenu širine uniformne distribucije (poglavlje 3) koja će kao rezultat dati po dvije procijenjene rubne točke objekta za svaku od horizontalnih pruga.

U svrhu procjene više rubnih točaka, *ALGORITAM 1* je poželjno ponoviti analogno i po y-osi, podijeliti točke u vertikalne pruge i prethodno navedenim postupkom procijeniti rubne točke i za te pruge.

Kod ovoga algoritma potencijalni problem predstavljaju rubne pruge – skupovi C_0 i C_7 na Slici 5.2. Rubne pruge će u pravilu sadržavati mali broj točaka kojima je opisan objekt – uslijed toga moguće je da metoda za procjenu rubnih točaka daje neprecizne rezultate. Iz navedenog je razloga konstruiran alternativni algoritam za podjelu točaka (*ALGORITAM 2*) koji podatke ne dijeli na horizontalne i vertikalne, već na *zvjezdaste* skupove (pruge).

ALGORITAM 2 *(podjela točaka na zvjezdaste pruge)*

1. Odabrati parametre $m>1$ (broj pruga) i $w>0$ (širina pruge)
2. Odrediti središnju točku objekta

$$D_c=(x_c,y_c):=\frac{1}{n}\left(\sum_{i=1}^{n}x_i,\ \sum_{i=1}^{n}y_i\right)$$

3. Odrediti skupove C_k (Slika 5.3)

$$C_k:=\left\{(x_i,y_i)\in D: x_i\in\left[x_c-\frac{w}{2\sin\varphi_k},x_c+\frac{w}{2\sin\varphi_k}\right],y_i\in\left[y_c-\frac{w}{2\cos\varphi_k},x_c+\frac{w}{2\cos\varphi_k}\right]\right\},$$

gdje je $\varphi_k=\frac{\pi}{m-1}\cdot k,\ \ k=1,\dots m-1$, a $w>0$ proizvoljni parametar koji označava širinu pruge (Slika 5.3).

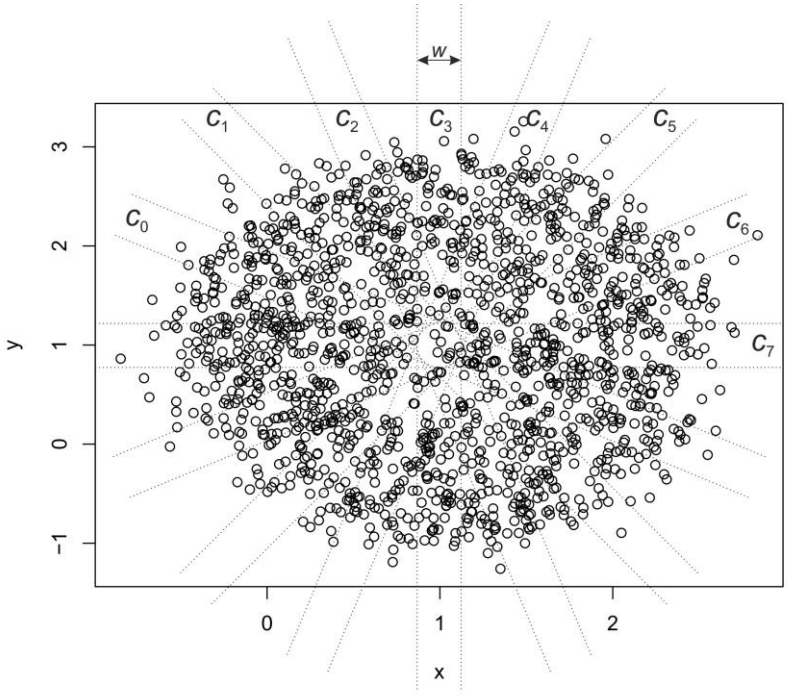
4. Odrediti skupove C_k^r na način da se točke iz skupa C_k zarotiraju za kut $-\varphi_k$ oko točke D_c

$$C_k^r=\left\{\left(x_c+(x_i-x_c)\cos(-\varphi)-(y_i-y_c)\sin(-\varphi)\right),\ \left(y_c+(x_i-x_c)\sin(-\varphi)+(y_i-y_c)\cos(-\varphi)\right)\right\}$$

za svaku točku $(x_i,y_i)\in C_k,\ \ k=1,\dots,m$.

5. Odrediti skupove $\overline{C_k}$ koji se sastoje od x komponente udaljenosti svake točke iz skupa C_k^r od središnje točke objekta D_c

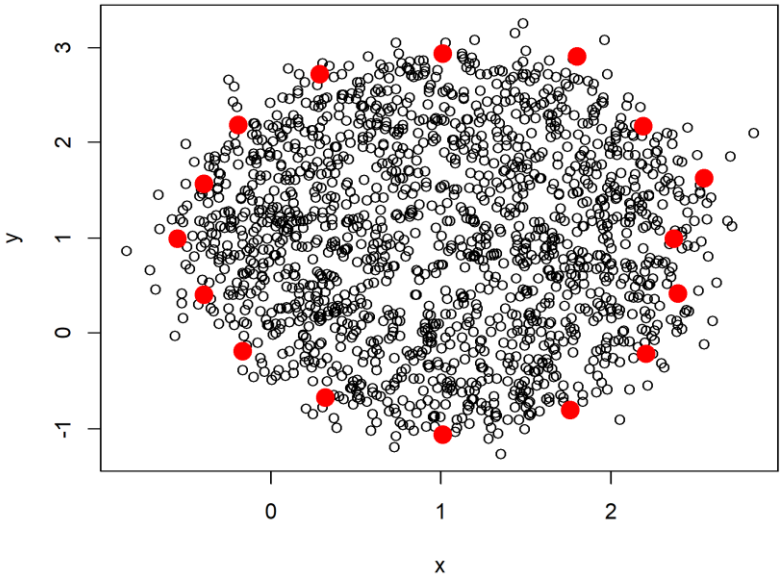
$$\overline{C_k}:=\left\{x_i-x_c:(x_i,y_i)\in C_k^r\right\},\ \ k=1,\dots,m-1.$$



Slika 5.3 Zvjezdasta podjela točaka kojima je zadan objekt

Istovjetno *ALGORITAM 1*, i ovdje se generira $m-1$ skupova jednodimenzionalnih podataka $\overline{C_k}$ na kojima se primjenjuje parametarska metoda za procjenu širine uniformne distribucije u svrhu procijene para rubnih točaka. Međutim, dok je prilikom korištenja *ALGORITAM 1* broj elemenata u skupovima $\overline{C_k}$ znatno varirao i bio vrlo malen za rubne pruge, ovdje to nije slučaj zbog drukčijeg načina generiranja pruga. Sve pruge prolaze kroz središte objekta pa će, za slučaj kružnih objekata, veličine skupova $\overline{C_k}$ biti približno konstantne, što uvelike doprinosi kvaliteti procjene rubnih točaka.

Procijenjene rubne točke dobivene jednim od predstavljenih algoritama označene su crvenom bojom na Slici 5.4 i predstavljaju polazišnu točku za idući korak.



Slika 5.4 Procijenjene rubne točke numerički zadanog objekta

5.1.2. Optimalno smještanje kružnice ili elipse u skup rubnih točaka

Sljedeći je zadatak optimalno smjestiti kružnicu, za kružne objekte, odnosno elipsu, za eliptične objekte, u koordinatni sustav na način da ona što je moguće manje odstupa od skupa procijenjenih rubnih točaka (x_i, y_i) dobivenih u prethodnom koraku. Taj se problem u literaturi naziva *circle fit*, odnosno *ellipse fit*. Razvijeno je mnogo algoritama koji rješavaju taj problem.

Za problem optimalnog smještanja kružnice, jedan od najpoznatijih algoritama predstavio je I. Kåsa [32]. U toj se metodi kružnica predstavlja algebarskom jednadžbom

$$A\left(x^2+y^2\right)+Bx+Cy+D=0 \tag{5.2}$$

gdje je $A\neq 0$ i $B^2+C^2-4AD>0$. Ovdje je zadatak optimalno procijeniti parametre A, B, C i D te nakon toga izrazima

$$a=-\frac{B}{2A},\quad b=-\frac{C}{2A},\quad R^2=\frac{B^2+C^2-4AD}{4A^2} \tag{5.3}$$

prikazati kružnicu parametarski

$$(x-a)^2+(y-b)^2=R^2\ . \tag{5.4}$$

Ideja predstavljena u radu [32] oslanja se na minimizaciju funkcije

$$F_K = \sum \left(r_i^2 - R^2 \right)^2 = \sum \left(x_i^2 + y_i^2 - 2ax_i - 2by_i + a^2 + b^2 - R^2 \right)^2 \ . \tag{5.5}$$

Drugim riječima, potrebno je minimizirati $F_k = \sum f_i^{\,2}$, gdje $f_i = r_i^2 - R^2$ predstavljaju algebarske udaljenosti točke (x_i, y_i) od kružnice. Normalizacijom izraza (5.2) ($A = 1$) parametri B, C i D postaju $B = -2a$, $C = -2b$, $D = a^2 + b^2 - R^2$ te se izraz (5.5) transformira u problem minimizacije funkcije

$$F_K = \sum \left(z_i + Bx_i + Cy_i + D \right)^2, \tag{5.6}$$

gdje je $z_i = x_i^2 + y_i^2$. Ovime je problem sveden na sustav linearnih jednadžbi koje je potrebno riješiti po parametrima B, C i D te napokon izrazima (5.3) kružnicu izraziti parametarski (5.4).

Opisanu metodu dodatno je optimizirao V. Pratt [33] koji predlaže minimizaciju funkcije $F = \dfrac{1}{4R^2} F_K$. Kada se uvedu parametri A, B, C i D funkcija koju je potrebno minimizirati postaje

$$F_P = \sum \frac{\left| Az_i + Bx_i + Cy_i + D \right|^2}{B^2 + C^2 - 4AD} \ . \tag{5.7}$$

Pokazalo se da Prattova metoda, uz cijenu nešto sporijeg izvođenja, daje kvalitetnije rezultate [34].

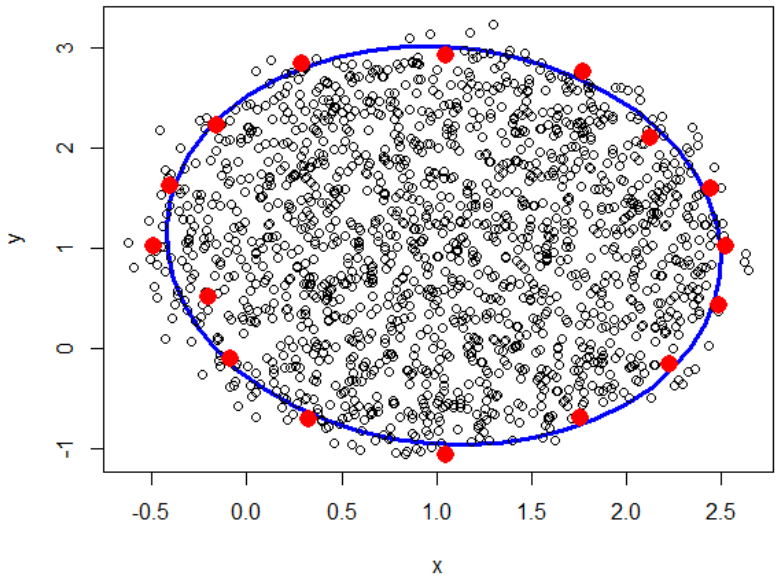
Ako se u procijenjene rubne točke želi optimalno smjestiti elipsa, problem postaje puno složeniji. Potrebno je promatrati elipsu kao krivulju drugog reda (koniku) implicitno definiranu polinomom

$$F(\mathbf{a}; \mathbf{x}) = \mathbf{a} \cdot \mathbf{x} = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{5.8}$$

gdje su $\mathbf{a} = \left[a \ b \ c \ d \ e \ f \right]^T$ i $\mathbf{x} = \left[x^2 \ xy \ y^2 \ x \ y \ 1 \right]^T$. Dodatno, kako bi se skup krivulja definiran izrazom (5.8) ograničio samo na elipse, potrebno je postaviti uvjet $b^2 - 4ac < 0$. Funkcija $F(\mathbf{a}; \mathbf{x_i})$ naziva se algebarska udaljenost točke (x_i, y_i) od krivulje drugog reda zadane jednadžbom $F(\mathbf{a}; \mathbf{x}) = 0$. Problem optimalnog smještanja elipse može se promatrati kao minimizacija sume kvadrata algebarskih udaljenosti

$$D_A(\mathbf{a}) = \sum_{i=1}^{N} F(\mathbf{a}; \mathbf{x}_i)^2 \tag{5.9}$$

s obzirom na N zadanih točaka x_i. Efikasno rješenje ovog netrivijalnog problema predložili su A. Fitzgibbon, M. Pilu i R. B. Fisher [20]. Primjenom tog algoritma na točke sa Slike 5.4 dobivaju se parametri $\mathbf{a} = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T$ elipse koja je prikazana plavom bojom na Slici 5.5.



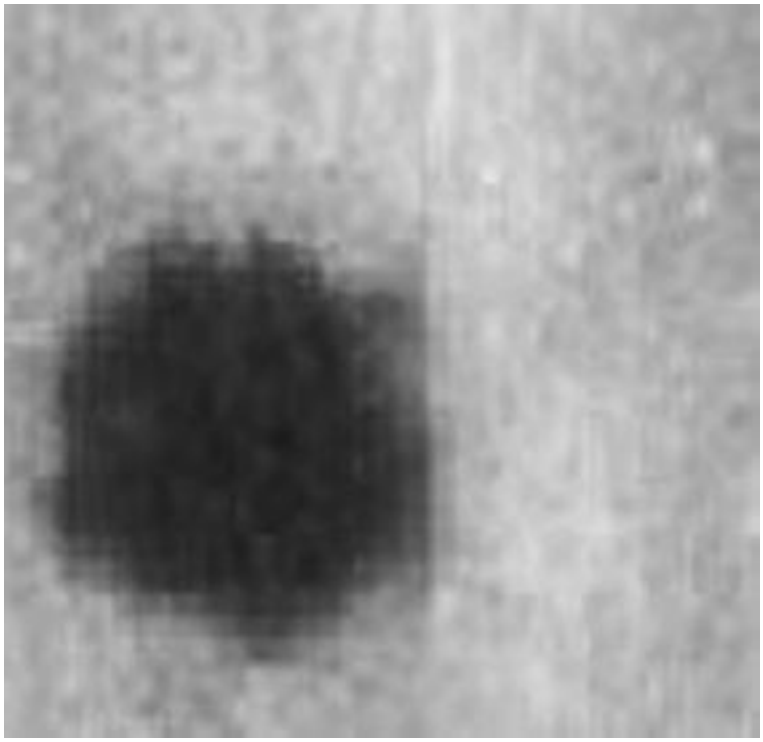
Slika 5.5 Optimalno smještena elipse

Kao završni korak, trivijalno je iz parametara elipse izračunati njezinu površinu

$$P = \frac{2\pi}{\sqrt{4ac - b^2}} \ . \tag{5.10}$$

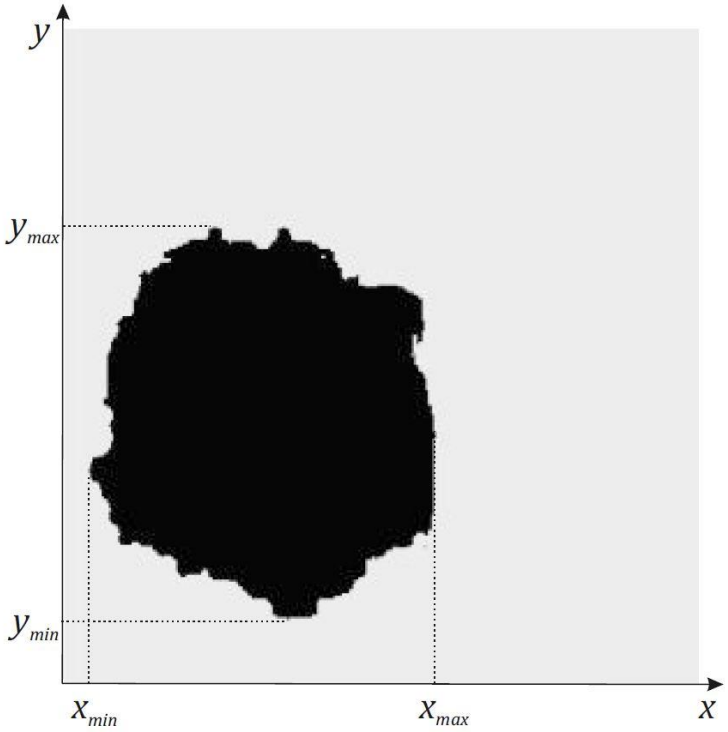
5.2. Procjena površine objekta snimljenog na slici

Postupak opisan u prethodnom poglavlju može, uz dodatne modifikacije, poslužiti i za procjenu površine kružnog ili elipsoidnog objekta prikazanog na slici. Za ilustraciju, na Slici 5.6 prikazana je snimka kolonije crnih gljivica (preuzeto iz [35]) čiju je površinu potrebno procijeniti.



Slika 5.6 Kolonija crnih gljivica (preuzeto iz [35])

Ako će se prilikom procjene skupa rubnih točaka koristiti *ALGORITAM 1*, tj. podjela slike na horizontalne i vertikalne pruge, može se primijetiti da će postojati pruge koje uopće ne zahvaćaju objekt. Kada se na takvim prugama izvrši procjena širine objekta dobit će se po par procijenjenih rubnih točaka, ali one će biti daleko od objekta i u konačnici će znatno pokvariti kvalitetu procjene površine. Iz tog je razloga potrebno na neki način izolirati područje slike na kojem je objekt. Kako bi se to postiglo, pikseli slike će biti razvrstani u dva segmenta – objekt i pozadina. Za to se koristi dobro poznati *k-means* algoritam [36] kojim će se na temelju svjetline svakog piksela slike taj piksel svrstati u jedan od dva segmenta. Slika 5.7 prikazuje tu podjelu, s time da su pikseli koji pripadaju segmentu objekta predstavljeni tamnijom bojom.



Slika 5.7 Snimka kolonije crnih gljivica razvrstana u dva segmenta

Koristeći skup koordinata piksela koji su svrstani u segment objekta sada je potrebno odrediti granice objekta $x_{min}, x_{max}, y_{min}, y_{max}$. Kako bi se izbjeglo da eventualne stršeće točke segmenta objekta (engl. *outliers*) naruše kvalitetu određivanja graničnih vrijednosti, potrebno je takve stršeće točke eliminirati. Promatraju se koordinate piksela – uređeni parovi (x_i, y_j) $i = 1,\ldots,m,\ j = 1,\ldots,n$ takvi da vrijedi

$$\begin{aligned} x_1 < x_2 < \ldots < x_m, \\ y_1 < y_2 < \ldots < y_n. \end{aligned} \tag{5.11}$$

Ako se sa C označi skup piksela koji pripadaju segmentu objekta, za graničnu vrijednost x_{min} uzet će se najmanji x_i za koji vrijedi

$$\exists y_{p_0} : \left(x_i, y_{p_0}\right) \in C,\ \exists y_{p_1} : \left(x_{i+1}, y_{p_1}\right) \in C, \ldots,\ \exists y_{p_k} : \left(x_{i+k}, y_{p_k}\right) \in C \tag{5.12}$$

gdje je $k > 0$ proizvoljan parametar koji određuje osjetljivost na stršeće točke. Analogno tome, za graničnu vrijednost x_{max} uzima se najveći x_i za koji vrijedi

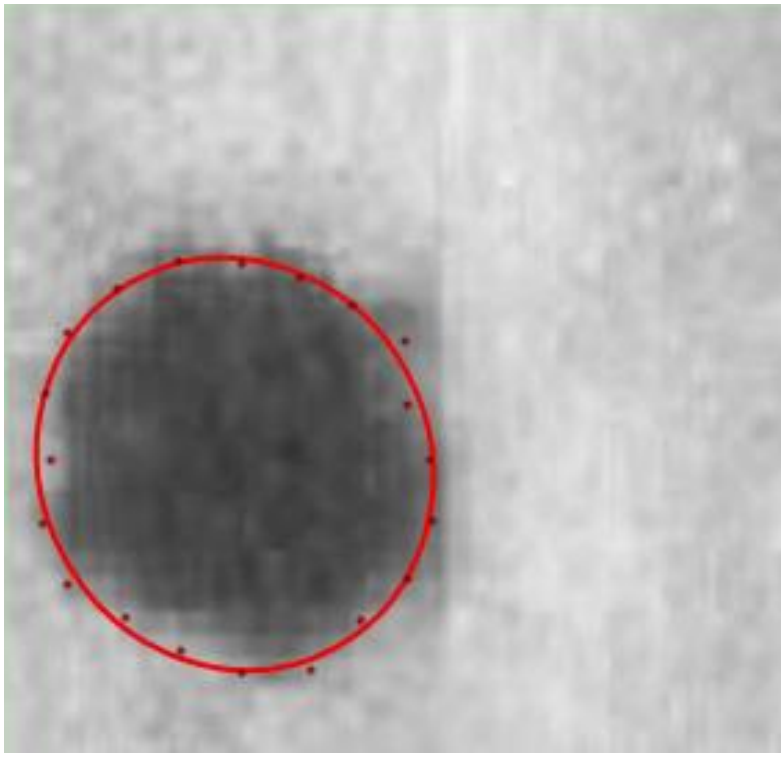
$$\exists y_{q_0} : \left(x_i, y_{q_0}\right) \in C,\ \exists y_{q_1} : \left(x_{i-1}, y_{q_1}\right) \in C, \ldots,\ \exists y_{q_k} : \left(x_{i-n}, y_{q_k}\right) \in C. \tag{5.13}$$

Na identičan način potrebno je procijeniti i granice y_{min}, y_{max}. Nadalje će se za podjelu objekta na horizontalne pruge u *ALGORITAM 1* koristiti područje omeđeno granicama y_{min} i y_{max}, odnosno za podjelu na vertikalne pruge područje omeđeno granicama x_{min} i x_{max}.

Idući je zadatak dio slike objekta omeđen prethodno utvrđenim granicama pretvoriti u matricu logičkih vrijednosti *podaci*, postupkom opisanim u poglavlju 4. Zatim se, dalje prateći *ALGORITAM 1* i koristeći metode opisane u poglavlju 4, procijeni skup rubnih točaka te se naposljetku u te točke optimalno smjesti kružnica ili elipsa i izračuna njezina površina.

Za slučaj zvjezdaste podjele objekta u svrhu procijene rubnih točaka (*ALGORITAM 2*), nužno je odrediti središnju točku objekta D_c. U tu svrhu može ponovo poslužiti *k-means* algoritam jer se njime, osim grupiranja točaka u segmente, izračunavaju i središta svakoga segmenta. Dakle, kako bi se odredila točka D_c potrebno je, kao i kod primjene *ALGORITAM 1*, sve piksele slike razvrstati po njihovoj svjetlini u dva segmenta i za točku D_c uzeti središte segmenta piksela koji predstavljaju objekt. Dalje je, kao i u prethodnom slučaju, sliku objekta potrebno pretvoriti u matricu logičkih vrijednosti *podaci*, pratiti daljnje korake u *ALGORITAM 2*, optimalno smjestiti kružnicu ili elipsu u procijenjene rubne točke te izračunati njezinu površinu.

Procijenjene rubne točke i optimalno smještena elipsa koja opisuje objekt snimljen na Slici 5.7 prikazan je na Slici 5.8.



Slika 5.8 Snimka kolonije crnih gljivica s procijenjenim rubnim točkama i optimalno smještenom elipsom

6. Programska implementacija predloženih algoritama

Predložene metodologije za procjenu duljine presjeka objekta s pravcem i procjenu površine kružnog ili elipsoidnog objekta opisane u poglavljima 3, 4 i 5 implementirana je u programskom jeziku R [37].

Programski jezik R izdan je kao softver otvorenog koda pod GNU licencom, a nastao je na temeljima programskog jezika S koji je razvila tvrtka Bell Laboratories (bivši AT&T). Pruža široku paletu statističkih metoda kao što su linearno i nelinearno modeliranje, klasični statistički testovi, analiza vremenskih nizova, segmentiranje itd. Aplikacije napisane u R-u mogu se izvršavati na svim modernim operativnim sustavima (Windows, različite Linux, UNIX i BSD distribucije, MacOS). Sama jezgra R-a nudi samo osnovne programske funkcionalnosti, no razvojna okolina je iznimno proširiva korištenjem službenog (The Comprehensive R Archive Network – CRAN [38]) i neslužbenih repozitorija programskih paketa.

Programska implementacija metodologija opisanih u ranijim poglavljima realizirana je kao R paket *LeArEst* [39] koji je izdan u CRAN repozitoriju i može se od tamo preuzimati i upotrebljavati. Četiri glavna modula paketa bit će opisana u nastavku.

6.1. Implementacija algoritma za procjenu širine uniformne distribucije s aditivnom greškom

Ovaj modul predstavlja implementaciju rezultata objavljenih u radovima [28, 29, 30] i omogućuje procjenu širine uniformne distribucije iz numerički zadanih jednodimenzionalnih podataka koji su mjereni s aditivnom greškom različitih statističkih distribucija. Implementiran je kao funkcija *lengthest* programskog paketa *LeArEst*.

Ulazni parametri funkcije *lengthest* su:

- vektor polazišnih podataka,
- statistička distribucija aditivne greške,
- varijanca aditivne greške,

- metoda procjene varijance greške (koristi se ako je varijanca nepoznata) i
- zahtijevana razina pouzdanosti procijenjenog intervala pouzdanosti.

Funkcija kao rezultat daje:

- procijenjenu polu-širinu uniformne distribucije (parametar a u izrazu (3.2)),
- procijenjenu varijancu greške, ukoliko ona nije eksplicitno zadana i
- procijenjeni interval pouzdanosti rezultata.

Ova će se funkcija koristiti prilikom procjenjivanja duljine presjeka objekta s pravcem i procjenjivanja površine objekta u nastavku.

6.2. Implementacija algoritma za procjenu površine numerički zadanog kružnog ili elipsoidnog objekta

Metodologija opisana u poglavlju 5.1 implementirana je u ovom modulu paketa kao funkcija *areaest*. Pomoću nje je moguće procijeniti površinu elipse ili kružnice koja opisuje objekt predstavljen skupom dvodimenzionalnih točaka. Procjenu rubnih točaka moguće je izvršavati koristeći podjelu ulaznih točaka na horizontalne i vertikalne pruge (*ALGORITAM 1*) ili zvjezdastu podjelu (*ALGORITAM 2*).

Posebna je pažnja posvećena optimizaciji algoritma. Može se primijetiti da su, kod oba predložena algoritma, procjene rubnih točaka iz različitih skupova $\overline{C_k}$ međusobno neovisni zadaci. Iz tog je razloga u modulu implementirana mogućnost paralelnog izvršavanja više procjena rubnih točaka ukoliko računalo na kojem se aplikacija izvršava raspolaže s više procesorskih jezgri. To je izvedeno koristeći R pakete *parallel* (u novijim verzijama R-a nalazi se u jezgri), *doParallel* [40] i *foreach* [41]. Navedene pakete odlikuje mogućnost paralelnog izvršavanja na svim modernim operacijskim sustavima – na Windows operacijskom sustavu interno se koristi funkcionalnost *snow*, dok se na različitim izvedenicama UNIX i Linux operacijskih sustava koristi funkcionalnost *fork* [42]. Kako bi za vrijeme izvršavanja procijene rubnih točaka operacijski sustav ostao responzivan, kod paralelnog izvršavanja funkcije *areaest* ne koristi se sve raspoložive jezgre, već se jedna uvijek ostavlja na raspolaganje operacijskom sustavu.

Za rješavanje problema optimalnog smještaja kružnice ili elipse u procijenjene rubne točke koristi se paket *conicfit* [43], odnosno njegove funkcije *CircleFitByPratt* i *EllipseDirectFit*. U funkciji *CircleFitByPratt* implementirana je Prattova metoda za optimalno smještanje kružnice [32], dok se u funkciji *EllipseDirectFit* nalazi implementacija metode za optimalno smještanje elipse koju su predložili autori A. Fitzgibbon, M. Pilu i R. B. Fisher [20].

Funkcija *areaest* paketa *LeArEst* ima sljedeće ulazne parametre:

- vektor dvodimenzionalnih točaka koje opisuju objekt,
- statistička distribucija aditivne greške,
- varijanca aditivne greške,
- metoda procjene varijance greške (koristi se ako je varijanca nepoznata),
- metoda podjele objekta (horizontalne i vertikalne pruge ili zvjezdasta podjela),
- broj parova rubnih točaka koje je potrebno procijeniti,
- vrsta krivulje kojom je potrebno predstaviti objekt (kružnica ili elipsa),
- logički parametar koji određuje hoće li se procjena rubnih točaka izvršavati sekvencijalno ili paralelizirano i
- logički parametar koji određuje hoće li se skup ulaznih točaka, procijenjene rubne točke i optimalno smještena kružnica ili elipsa grafički prikazati.

Rezultati funkcije su:

- procijenjena površina objekta,
- vektor procijenjenih rubnih točaka,
- polumjer kružnice ili poluosi elipse koja opisuje objekt i
- opcionalni grafički prikaz ulaznih točaka, procijenjenih rubnih točaka i optimalno smještene kružnice ili elipse (Slika 5.5).

Cijela je funkcija *areaest* opisana pseudo-kodom u *ALGORITAM 3*.

ALGORITAM 3 (*pseudo-kod algoritma za procjenu površine numerički zadanog kružnog ili elipsoidnog objekta*)

Funkcija *areaest*(*točke, distribucija, varijanca, metodaProcjeneVarijance, metodaPodjele, brojParovaRubnihTočaka, vrstaKrivulje, paralelizirati, nacrtati*)

Izračunaj najmanje i najveće x i y koordinate iz točaka (x_{min}, y_{min}, x_{max}, y_{max})

Izračunaj *udaljenostPruga* = (x_{max} - x_{min}) / *brojParovaRubnihTočaka*

Ako je *metodaPodjele* = horizontalne i vertikalne pruge onda

Za svaki *x* od x_{min} do x_{max} u koracima *udaljenostPruga*

Izračunaj *točkePruge* kao podskup od *točke* s uvjetom *x* - *udaljenostPruga*/2 < *točkePruge*(*x*) < *x* + *udaljenostPruga*/2

Izračunaj *vektorUdaljenosti* svih točaka iz *točkePruge* po y koordinati od y_{min}

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni rubne točke za *vektorUdaljenosti* i dodaj ih u *rubneTočke*

Za svaki *y* od y_{min} do y_{max} u koracima *udaljenostPruga*

Izračunaj *točkePruge* kao podskup od *točke* s uvjetom *y* - *udaljenostPruga*/2 < *točkePruge*(*y*) < *y* + *udaljenostPruga*/2

Izračunaj *vektorUdaljenosti* svih točaka iz *točkePruge* po x koordinati od x_{min}

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni rubne točke za *vektorUdaljenosti* i dodaj ih u *rubneTočke*

Ako je *metodaPodjele* = zvjezdasta onda

Izračunaj *centralnuTočku* skupa *točkePruge*

Za svaki *kutRotacije* od 0 do π u koracima π/*brojParovaRubnihTočaka*

Izračunaj *rotiraneTočke* rotiranjem *točke* oko *centralneTočke* za *kutRotacije*

Izračunaj *točkePruge* kao podskup od *rotiraneTočke* s uvjetom *y* - *udaljenostPruga*/2 < *točkePruge*(*y*) < *y* + *udaljenostPruga*/2

Izračunaj *vektorUdaljenosti* svih točaka iz *točkePruge* po x koordinati od x_{min}

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni rubne točke za *vektorUdaljenosti*, zarotiraj ih oko *centralneTočke za* - *kutRotacije* i dodaj ih u *rubneTočke*

Ako je *vrstaKrivulje* = elipsa onda

Optimalno smjesti elipsu u *rubneTočke* i izračunaj njezinu površinu

Ako je *vrstaKrivulje* = kružnica onda

Optimalno smjesti kružnicu u *rubneTočke* i izračunaj njezinu površinu

Ako je *nacrtati* = ISTINA onda

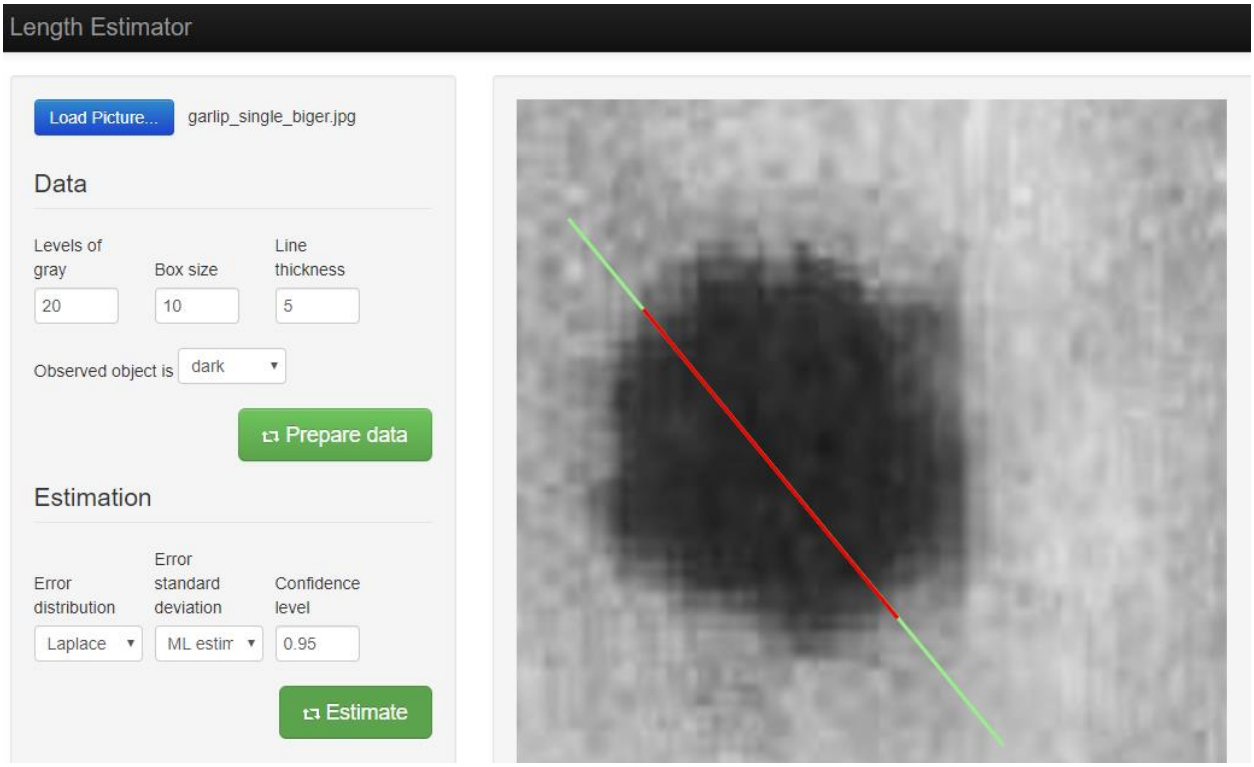
Nacrtaj koordinatni sustav, unesi *točke* i optimalno smještenu krivulju

6.3. Implementacija algoritma za procjenu duljine presjeka objekta snimljenog na slici s proizvoljnim pravcem

Prilikom implementacije algoritma za procjenu duljine presjeka objekta s pravcem odabrano je da se ona realizira kao web aplikacija. U R programskom jeziku to je najjednostavnije izvesti koristeći paket *shiny* [44] koji omogućava kreiranje web aplikacija koristeći velik broj unaprijed pripremljenih gradivnih blokova (engl. *widget*). Međutim, ograničenja koja nameće besplatna verzija toga paketa presudila su da se za izgradnju web aplikacije koristi paket *opencpu* [45] koji pruža veću fleksibilnost, ali zahtjeva poznavanje HTTP protokola, HTML prezentacijskog jezika i skriptnog jezika JavaScript. Paket o*pencpu* pruža pouzdano dvosmjerno sučelje (engl. Application Programming Interface – API) između R-a i web stranice, temeljeno na JavaScript skriptnom jeziku i tehnologiji AJAX [46]. Dodatni razlog za korištenje ovog paketa je i mogućnost budućeg razdvajanja klijentske i poslužiteljske strane aplikacije. Naime, paket omogućuje da se na klijentskom računalu pokrene sama web aplikacija, a svi računsko intenzivni zadaci izvrše na fizički udaljenoj, procesorski snažnijoj radnoj stanici.

Nakon pokretanja web aplikacije za procjenu duljine presjeka objekta s pravcem (Slika 6.1) potrebno je prvo učitati sliku objekta. Zatim se označe proizvoljne dvije točke na slici kako bi se iscrtao pravac koji presijeca objekt. Idući korak je podešavanje parametara procjene:

- broj nijansi sive boje (paletu nijansi boja na slici moguće je diskretizirati u svrhu ubrzanja izvođenja algoritma),
- dimenzija kvadratne matrice logičkih vrijednosti kojom je predstavljen svaki piksel (parametar *dim_logic* u poglavlju 4),
- širina prostora oko pravca kojeg treba uzeti u obzir prilikom procjene (parametar *širina_linije* u poglavlju 4),
- odabir je li promatrani objekt svijetao ili taman (za slučaj tamnog objekta promatra se negativ slike),
- statistička distribucija aditivne greške,
- varijanca aditivne greške ili,
- metoda procjene varijance greške (koristi se ako je varijanca nepoznata).



Slika 6.1 Web aplikacija za procjenu duljine presjeka objekta s proizvoljnim pravcem

Nakon procjenjivanja duljine presjeka ispisuje se:

- procijenjena duljina presjeka objekta s pravcem, izražena u pikselima i postotku širine cijele slike,
- procijenjena varijanca greške, ukoliko ona nije eksplicitno zadana i
- procijenjeni interval pouzdanosti rezultata.

Dodatno, procijenjena duljina presjeka na slici se označava crvenom bojom radi očitijeg uvida u dobiveni rezultat.

Pseudo-kod ove funkcije dan je u *ALGORITAM* 4.

ALGORITAM *4 (pseudo-kod algoritma za procjenu duljine presjeka objekta snimljenog na slici s proizvoljnim pravcem)*

Funkcija *startweb-lengthest*(*slika*, *početnaTočka*, *završnaTočka*, *brojNijansiSive*, *dimLogic*, *širinaLinije*, *objektSvijetao*, *distribucija*, *varijanca*, *metodaProcjeneVarijance*)

Učitaj podatke o svjetlini svih piksela slike u dvodimenzionalnu matricu *slika*

Ako je *objektSvijetao* = FALSE onda

Za svaki piksel slike (i, j)

slika(*i*, *j*) = 255 – *slika*(*i*, *j*)

Kreiraj logičku matricu *podaci* dimenzije (*visinaSlike* * *dimLogic*, *širinaSlike* * *dimLogic*)

Ispuni matricu *podacima* vrijednostima *FALSE*

Za svaki *i* od 0 do *visinaSlike*

Za svaki *j* od 0 do *širinaSlike*

Ispuni dio matrice *podaci* (od *i* * *dimLogic* do (*i*+1) * *dimLogic*,

od *j* * *dimLogic* do (*j*+1) * *dimLogic*)

uniformno raspodijeljenim TRUE vrijednostima kojih će biti

proporcionalno vrijednosti *podaci*(*i*, *j*)

Izračunaj koeficijente *pravca* koji prolazi *početnomTočkom* * *dimLogic* i

završnomTočkom * *dimLogic*

Za svaki *i* od $početnaTočka_x$ * *dimLogic* do $završnaTočka_x$ * *dimLogic*

Za svaki *j* od $početnaTočka_y$ * *dimLogic* do $završnaTočka_y$ * *dimLogic*

Ako je (udaljenost koordinata (*i*, *j*) od *pravca* manja od *širinaLinije*/2 * *dimLogic*) i

(*podaci*(*i*, *j*) = TRUE) onda

Izračunaj *ortogonalnuProjekciju* točke *podaci*(*i*, *j*) na *pravac*

Izračunaj *udaljenostTočke ortogonalnaProjekcija* od

početnaTočka * *dimLogic*

Dodaj *udaljenostTočke* u *vektorUdaljenosti*

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni *rubneTočke* za *vektorUdaljenosti*

Za svaku *rubnuTočku* iz *rubnihTočaka*

$rubnaTočka_x = rubnaTočka_x$ / *dimLogic*

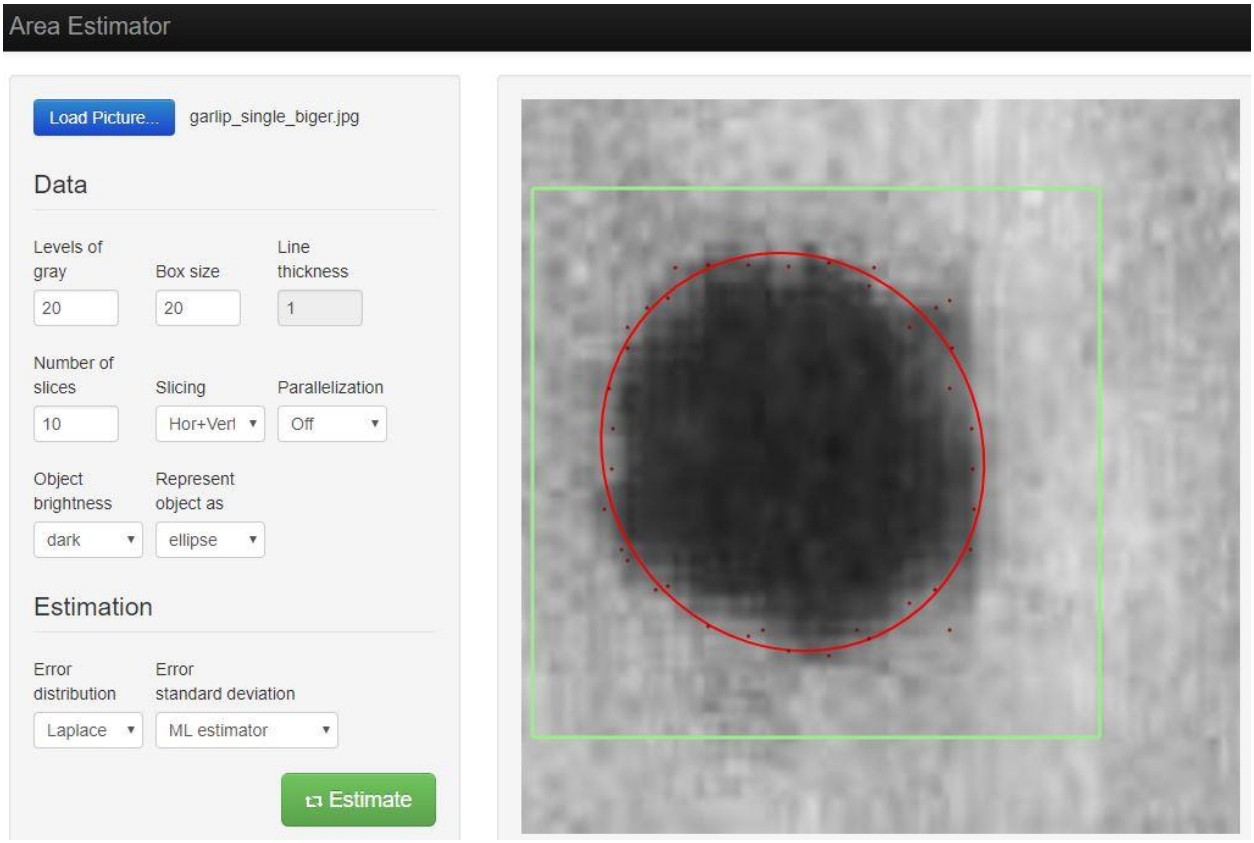
$rubnaTočka_y = rubnaTočka_y$ / *dimLogic*

Ucrtaj na slici *rubneTočke* i spoji ih crvenom linijom

6.4. Implementacija algoritma za procjenu površine kružnog ili elipsoidnog objekta snimljenog na slici

Korisničko sučelje za ovaj algoritam izvedeno je na isti način kao i za prethodno opisani, kao web aplikacija. Implementacija algoritma u potpunosti prati metodologiju predstavljenu u potpoglavlju 5.2. Za segmentiranje slike objekta na segmente objekta i pozadine korištena je funkcija *k-means* koja se u novijim verzijama R-a nalazi u njegovoj jezgri. Kao i kod implementacije algoritma za procjenu površine numerički zadanog objekta (potpoglavlje 6.2) omogućeno je paralelno izvršavanje procjena različitih rubnih točaka koristeći pakete *parallel*, *doParallel* i *foreach*. Optimalno smještanje kružnice ili elipse u rubne točke postignuto je, kao i u potpoglavlju 6.2, korištenjem funkcija *CircleFitByPratt* i *EllipseDirectFit* iz paketa *conicfit*.

Po pokretanju web aplikacije za procjenu površine kružnog ili elipsoidnog objekta (Slika 6.2) potrebno je prvo učitati sliku objekta. Zatim je potrebno označiti područje interesa na slici, odnosno dio slike na kojem se nalazi objekt čiju je površinu potrebno procijeniti, na način da se na slici označe dva nasuprotna vrha pravokutnika koji ga omeđuje. U daljnjim koracima promatrat će se isključivo prethodno omeđeni dio slike.



Slika 6.2 Web aplikacija za procjenu površine kružnog ili elipsoidnog objekta snimljenog na slici

Potom je potrebno podesiti parametre procjene:

- broj nijansi sive boje,
- dimenzija kvadratne matrice logičkih vrijednosti kojom je predstavljen svaki piksel (parametar *dim_logic*),
- širina prostora oko pravca kojeg treba uzeti u obzir prilikom procjene, ukoliko je odabrana zvjezdasta podjela objekta (parametar *širina_linije*),
- broj parova rubnih točaka koje je potrebno procijeniti,
- metoda podjele objekta (horizontalne i vertikalne pruge ili zvjezdasta podjela),
- sekvencijalno ili paralelizirano procjenjivanje rubnih točaka,
- odabir je li promatrani objekt svijetao ili taman,
- vrsta krivulje kojom je potrebno predstaviti objekt (kružnica ili elipsa),
- statistička distribucija aditivne greške i
- metoda procjene varijance greške.

Nakon procjene površine u aplikaciji se ispisuje:

- procijenjena površina objekta, izražena u pikselima i postotku površine čitave slike,
- središte elipse koja reprezentira objekt, duljina njezinih poluosi i kut rotacije, ukoliko je odabrano da se objekt predstavlja elipsom, ili
- središte kružnice koja reprezentira objekt i njezin polumjer, ukoliko se objekt predstavlja kružnicom.

Na učitanu sliku se dodatno crvenom bojom ucrtavaju procijenjene rubne točke i krivulja koja aproksimira objekt.

Opis funkcije *startweb-areast* u pseudo-kodu prikazan je u *ALGORITAM* 5.

ALGORITAM *5 (pseudo-kod algoritma za procjenu površine kružnog ili elipsoidnog objekta snimljenog na slici)*

Funkcija *startweb-areaest*(*slika*, *gornjaLijevaTočka*, *donjaDesnaTočka*, *brojNijansiSive*, *dimLogic*, *širinaLinije*, *brojParovaRubnihTočaka*, *vrstaKrivulje*, *paralelizirati*, *objektSvijetao*, *metodaPodjele*, *distribucija*, *metodaProcjeneVarijance*)

Učitaj podatke o svjetlini svih piksela u dvodimenzionalnu matricu *slika*

Ako je *objektSvijetao* = FALSE onda

Za svaki piksel slike (*i*, *j*)

slika(*i*, *j*) = 255 – *slika*(*i*, *j*)

Segmentiraj dio slike omeđen kvadratom kojeg određuju *gornjaLijevaTočka* i *donjaDesnaTočka* u dva segmenta koristeći *k-means* algoritam

Odredi početne i završne koordinate segmenta objekta x_{min}, x_{max}, y_{min}, y_{max} postupkom opisanim u (5.12) i (5.13)

Kreiraj logičku matricu *podaci* dimenzije ((x_{max} - x_{min}) * *dimLogic*, (y_{max} - y_{min}) * *dimLogic*)

Za svaki *i* od x_{min} do x_{max}

Za svaki *j* od y_{min} do y_{max}

Ispuni dio matrice *podaci* (od (x_{min} - *i*) * *dimLogic* do (x_{min} - *i* + 1) * *dimLogic*, od (y_{min} - *j*) * *dimLogic* do (y_{min} - *j* + 1) * *dimLogic*) uniformno raspodijeljenim TRUE vrijednostima kojih će biti proporcionalno vrijednosti *podaci*(*i*, *j*)

Izračunaj *udaljenostPruge* = (x_{max} - x_{min}) * *dimLogic* / *brojParovaRubnihTočaka*

Ako je *metodaPodjele* = horizontalne i vertikalne pruge onda

Za svaki *x* od x_{min} * *dimLogic* do x_{max} * *dimLogic*

Izračunaj *točkePruge* kao podskup od *podaci* s uvjetom *x* - *udaljenostPruge*/2 < *točkePruge*(*x*) < *x* + *udaljenostPruge*/2

Izračunaj *vektorUdaljenosti* svih točaka iz *točkePruge* po *y* koordinati od y_{min}

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni rubne točke za *vektorUdaljenosti* u dodaj ih u *rubneTočke*

Za svaki *y* od y_{min} * *dimLogic* do y_{max} * *dimLogic*

Izračunaj *točkePruge* kao podskup od *podaci* s uvjetom *y* - *udaljenostPruge*/2 < *točkePruge*(*y*) < *y* + *udaljenostPruge*/2

Izračunaj *vektorUdaljenosti* svih točaka iz *točkePruge* po *x* koordinati od x_{min}

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni rubne točke za *vektorUdaljenosti* u dodaj ih u *rubneTočke*

Ako je *metodaPodjele* = zvjezdasta onda

Izračunaj *centralnuTočku* segmenta objekta

Za svaki *kutRotacije* od 0 do π u koracima π/*brojParovaRubnihTočaka*

Izračunaj *rotiraniPodaci* rotiranjem elemenata matrice *podaci* oko *centralneTočke* za *kutRotacije*

Izračunaj *točkePruge* kao podskup od *rotiraniPodaci* s uvjetom *y* - *udaljenostPruge*/2 < *točkePruge*(*y*) < *y* + *udaljenostPruge*/2

Izračunaj *vektorUdaljenosti* svih točaka iz *točkePruge* po *x* koordinati od x_{min}

Centriraj *vektorUdaljenosti* po aritmetičkoj sredini

Procijeni rubne točke za *vektorUdaljenosti*, zarotiraj ih oko *centralneTočke* za - *kutRotacije* i dodaj u *rubneTočke*

Za svaku *rubnuTočku* iz *rubnihTočaka*

$rubnaTočka_x$ = $rubnaTočka_x$ / *dimLogic*

$rubnaTočka_y$ = $rubnaTočka_y$ / *dimLogic*

Ako je *vrstaKrivulje* = elipsa onda

Optimalno smjesti elipsu u *rubneTočke* i izračunaj njezinu površinu

Ako je *vrstaKrivulje* = kružnica onda

Optimalno smjesti kružnicu u *rubneTočke* i izračunaj njezinu površinu

Ucrtaj na slici *rubneTočke* i optimalno smještenu krivulju

7. Usporedba predloženih algoritama s postojećima

Algoritmi čija je implementacija opisana u poglavlju 6 (u daljnjem tesktu: *LeArEst*) detaljno su testirani na problemu procjene površine kružnih objekata snimljenih RGB-D kamerom koji koristi strukturirano osvjetljenje. Podacima dobivenim od RGB-D kamere dodane su različite količine šuma te se na takvih podacima vršila usporedna statistička analiza predloženih algoritama i algoritama opisanih u poglavlju 2.

7.1. Snimanje kružnih objekata i generiranje ulaznih podataka

U ovom postupku korištena je RGB-D kamera Asus Xtion PRO LIVE [47] (Slika 7.1).

Slika 7.1 Asus Xtion PRO LIVE RGB-D kamera

Spomenuti uređaj opremljen je infracrvenim CMOS projektorom koji na snimani objekt projicira strukturirani uzorak. Kamera potom skenira taj uzorak i na osnovu njegovog izobličenja računa udaljenosti. Također, snima se i klasična slika objekta. Na osnovu podataka o udaljenosti može se, uz pomoć parametara dobivenih od proizvođača, generirati dvodimenzionalna matrica $\mathbf{D}$ čiji elementi odgovaraju točnim udaljenostima pojedinih piksela snimljene slike od kamere. Od te se matrice nadalje, za potrebe procjene površine, kreira crno-bijela slika na način da je svjetlina svakog piksela slike proporcionalna udaljenosti odgovarajuće točke od kamere.

Na sljedećim je slikama prikazano:

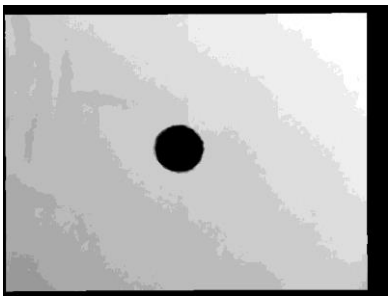
Slika 7.2: fotografija jednog od objekata kojemu se procjenjuje površina gornje plohe,

Slika 7.3: RGB slika objekta dobivena RGB-D kamerom,

Slika 7.4: dubinska slika objekta generirana iz matrice $\mathbf{D}$.

Slika 7.2 Fotografija objekta konzerva *čija se površina gornje plohe procjenjuje*

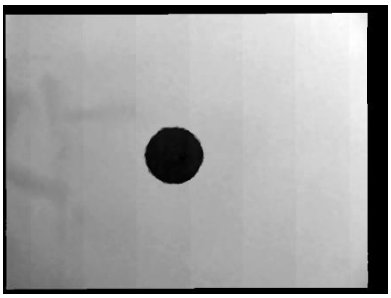
Slika 7.3 RGB slika objekta konzerva *s RGB-D kamere*



Slika 7.4 Dubinska slika objekta konzerva

Slika 7.5 Fotografija objekta CD stalak *čija se površina gornje plohe procjenjuje*

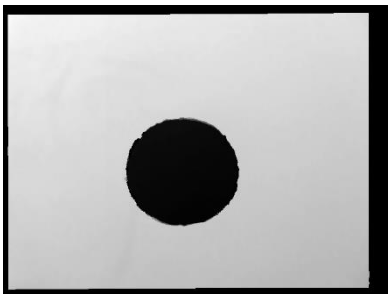
Slika 7.6 RGB slika objekta CD stalak *s RGB-D kamere*



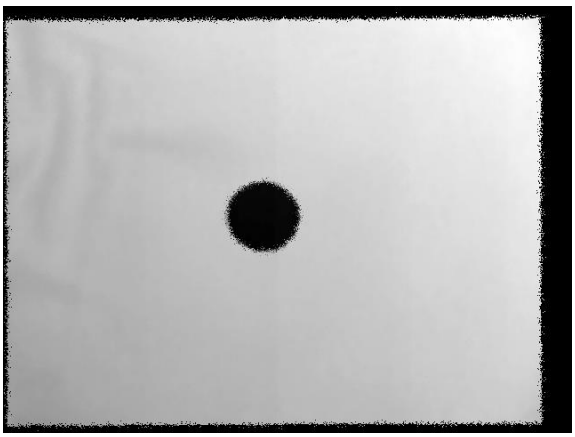
Slika 7.7 Dubinska slika objekta CD stalak

Slika 7.8 Fotografija objekta koš za smeće *čija se površina gornje plohe procjenjuje*

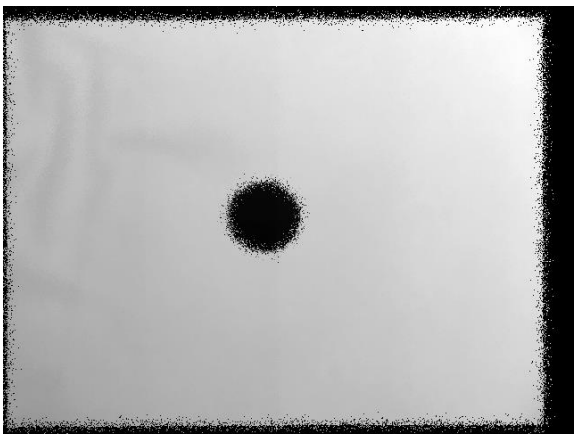
Slika 7.9 RGB slika objekta koš za smeće *s RGB-D kamere*



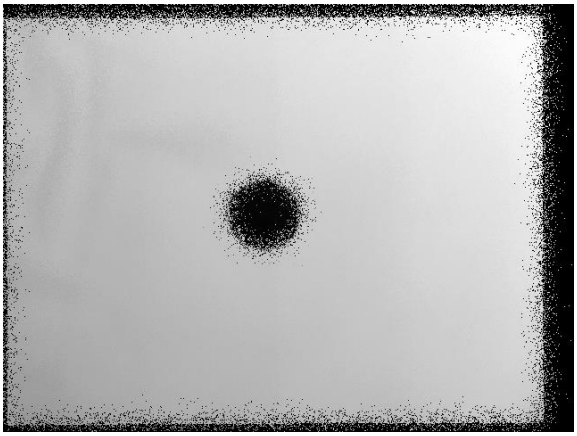
Slika 7.10 Dubinska slika objekta koš za smeće



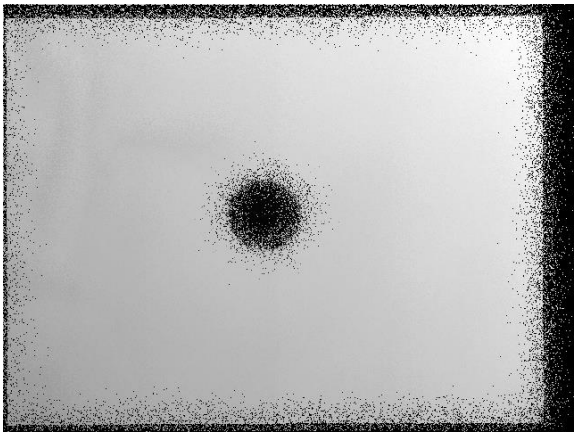
Slika 7.11 Dubinska slika objekta konzerva *s aditivnom greškom normalne distribucije, sd = 2*



Slika 7.12 Dubinska slika objekta konzerva *s aditivnom greškom normalne distribucije, sd = 5*



Slika 7.13 Dubinska slika objekta konzerva s aditivnom greškom normalne distribucije, sd = 10



Slika 7.14 Dubinska slika objekta konzerva s aditivnom greškom normalne distribucije, sd = 15

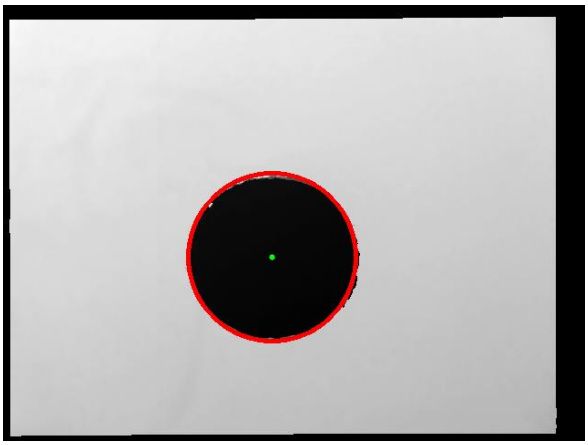
Na Slici 7.4 može se primijetiti crni obrub koji nastaje zbog hardverskog ograničenja objektiva kamere koji ne može procijeniti udaljenosti rubnih segmenata slike, no ta činjenica neće utjecati na kvalitetu procjene površine gornje plohe objekta.

Osim objekta prikazanog na Slici 7.2 (*konzerva*), identična je metodologija primijenjena na procjenu površina gornje plohe *CD stalka* (Slika 7.5) i *koša za smeće* (Slika 7.8).

Kako bi se ispitala kvaliteta procjene površine različitih algoritama pri radu s nejasnim slikama, na slike koje vizualiziraju udaljenosti piksela od objektiva kamere dodane su aditivne greške normalne distribucije, različitih standardnih devijacija (2, 5, 10 i 15), kao što je za objekt *konzerva* prikazano na Slikama 7.11 – 7.14. Za svaku standardnu devijaciju generirano je po 1000 slika, kako bi se eliminirala statistička pogreška i dobili pouzdaniji pokazatelji kvalitete procjene ispitivanih algoritama.

7.2. Metodologija testiranja

Ključan dio postupka procjene površine gornje plohe objekta jest detekcija kružnice na slici koja vizualizira udaljenost objekta od objektiva kamere (Slika 7.15). Konkretnije, zadaća algoritma za detekciju kružnice jest procjena središta i polumjera kružnice koja najpreciznije obuhvaća prikazani objekt. Pomoću tih podataka se potom iz matrice $\mathbf{D}$ izolira skup točaka koje reprezentiraju objekt.



Slika 7.15 Detektirana kružnica na jednoj od dubinskih slika objekta

Budući da ravnina u kojoj je smještena promatrana ploha snimljenog objekta nije nužno okomita na liniju kojom ga je kamera snimila, sljedeći je zadatak nalaženje parametara ravnine koja najbolje opisuje trodimenzionalne točke za koje je algoritam zaključio da pripadaju promatranom objektu (točke unutar crvene kružnice na Slici 7.15). U tu se svrhu prvo koristi RANSAC algoritam [48, 49] – iterativna metoda koja u ovom slučaju služi za početnu, grubu procjenu parametara ravnine. Na taj je način dobivena početna aproksimacija parametara za sljedeći korak – optimalno smještanje ravnine metodom najmanjih kvadrata [50] koja daje konačne parametre ravnine. Nakon opisanih radnji raspolažemo s točkom u prostoru koja predstavlja središte detektirane kružnice, polumjerom detektirane kružnice i parametrima ravnine na kojoj ta kružnica leži. Iz navedenih je podataka moguće jednoznačno odrediti stvarnu površinu kruga opisanog detektiranom kružnicom.

U prethodnim poglavljima predloženi algoritam za procjenu površine kružnih ili elipsoidnih objekata *LeArEst* uspoređen je sa algoritmima *Houghova transformacija*, *EDCircles* i *Fornaciari*, opisanima u poglavlju 2. Svi su alternativni algoritmi na testnim slikama detektirali kružnice. Iz

toga je razloga algoritam *LeArEst* konfiguriran na način da aproksimira objekt kružnicom (potpoglavlje 5.1.2), kako bi statistička usporedba bila vjerodostojna.

7.3. Rezultati testiranja

Prilikom samoga testiranja problem je predstavljala nedostupnost izvornog koda algoritma *EDCircles*, a pokušaj kontaktiranja autora s molbom za ustupanje izvornog koda nije urodio plodom. Nadalje, nije bilo moguće pronaći funkcionalnu implementaciju algoritma s raspoloživim izvornim kodom. Iz navedenih je razloga korištena desktop verzija *EDCircles* aplikacije [51], no, budući ona nema mogućnost automatizacije i skriptiranja, bilo je nemoguće u razumnom roku obaviti ispitivanja na svih 1000 generiranih slika za svaku količinu šuma za svaki promatrani objekt. Stoga se prilikom usporedbe s *EDCircles* algoritmom promatralo po 20 nasumično odabranih generiranih slika za svaki slučaj.

Još jedan problem koji je proizišao prilikom testiranja je obavezno podešavanje dvaju parametara kod algoritma *Houghova transformacija* (vezanih uz detekciju ruba *Canny* algoritmom i veličinu akumulatorske matrice), dok su ostale promatrane metode bezparametarske. Budući bi bilo vremenski veoma zahtjevno podešavati parametre posebno za svaku generiranu sliku, parametri su postavljeni jednom za svaki skup testnih slika tako da algoritam najpreciznije detektira kružnicu i s njima je izvršen cijeli skup testiranja za zadani objekt i za zadanu standardnu devijaciju greške.

Kvaliteta procjene površine bit će iskazana pomoću korijena srednje kvadratne pogreške – RMSE (engl. *Root Mean Square Error*) definirane kao:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(A(x_i) - \hat{A}\right)^2} \tag{7.1}$$

gdje je:

- $A(x_i)$: procijenjena površina plohe za ulazne podatke x_i,
- $\hat{A}$: stvarna površine plohe,
- n: broj testnih slika.

7.3.1. Rezultati za 20 nasumično odabranih slika iz skupa generiranih slika

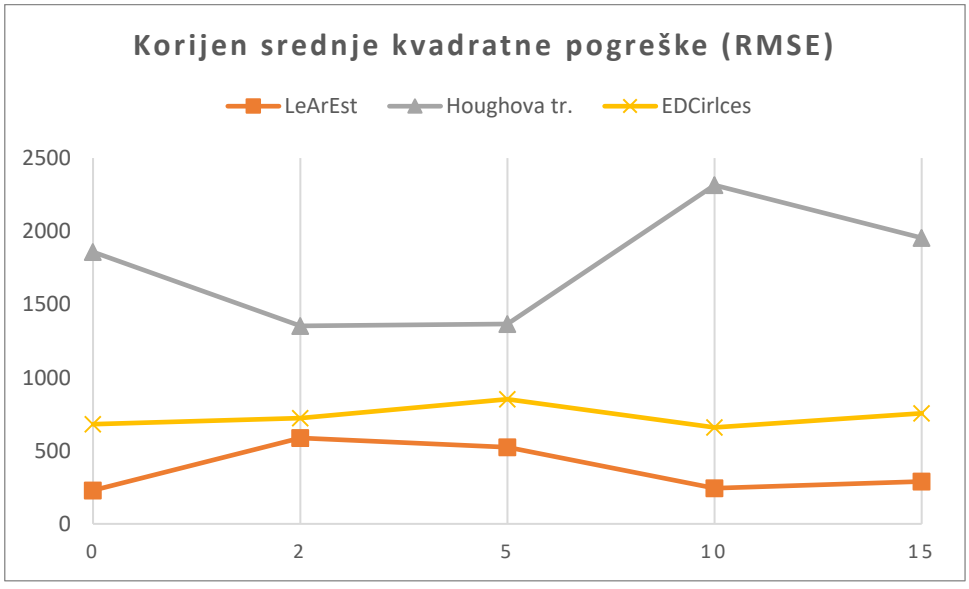
7.3.1.1. Objekt konzerva

U Tablica 7.1 prikazani su korijeni srednje kvadratne pogreške za objekt *konzerva*. Na dubinsku sliku objekta dodane su aditivne greške normalne distribucije standardnih devijacija $2, 5, 10$ i 15. Za svaku od navedenih devijacija generirano je 20 testnih slika ($n = 20$ u jednadžbi (7.1)).

	Korijen srednje kvadratne pogreške (RMSE)			
sd	**LeArEst**	**Houghova tr.**	**EDCircles**	**Fornaciari**
0	229	1858	681	
2	587	1353	723	
5	523	1365	852	
10	243	2315	659	
15	289	1955	755[1]	

Tablica 7.1 Korijeni srednje kvadratne pogreške za objekt konzerva *(20 slika)*

Rezultati iz prethodne tablice vizualizirani su na Slici 7.16.



Slika 7.16 Graf korijena srednje kvadratne pogreške za objekt konzerva *(20 slika)*

Iz rezultata se može primijetiti da algoritam *Fornaciari* nije detektirao kružnicu niti na jednoj slici. Nadalje, *EDCircles* nije detektirao kružnicu na tri od ukupno dvadeset slika s najvećom promatranom količinom šuma ($standardna\ devijacija = 15$).

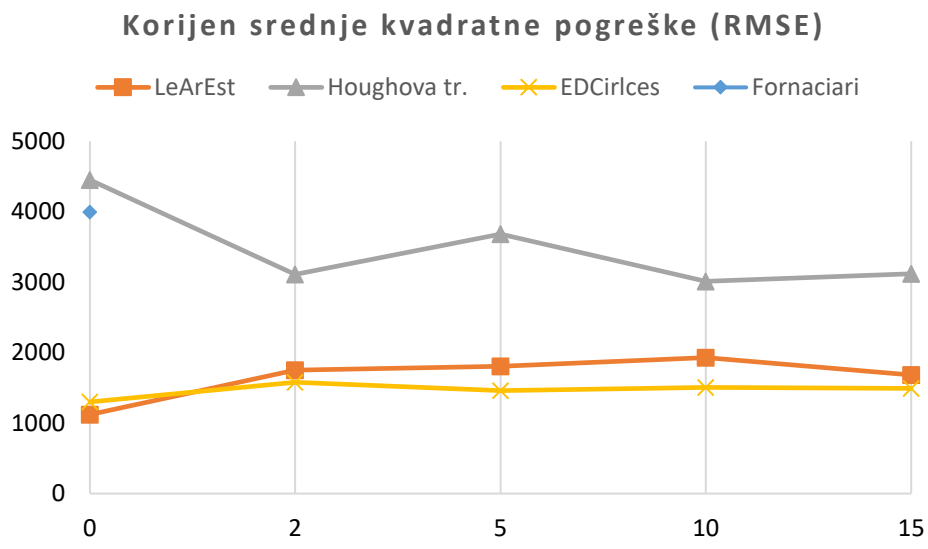
[1] Za tri promatrane slike metoda nije detektirala kružnicu

7.3.1.2. *Objekt* CD stalak

Korijeni srednje kvadratne pogreške za objekt *CD stalak* za različite količine šuma i različite algoritme prikazani su u Tablici 7.2 i vizualizirani na Slici 7.17.

	Korijen srednje kvadratne pogreške (RMSE)			
sd	**LeArEst**	**Houghova tr.**	**EDCircles**	**Fornaciari**
0	1121	4450	1300	3994
2	1749	3107	1579	
5	1805	3682	1459	
10	1929	3011	1506	
15	1682	3117	1494	

Tablica 7.2 Korijeni srednje kvadratne pogreške za objekt CD stalak *(20 slika)*



Slika 7.17 Graf korijena srednje kvadratne pogreške za objekt CD stalak *(20 slika)*

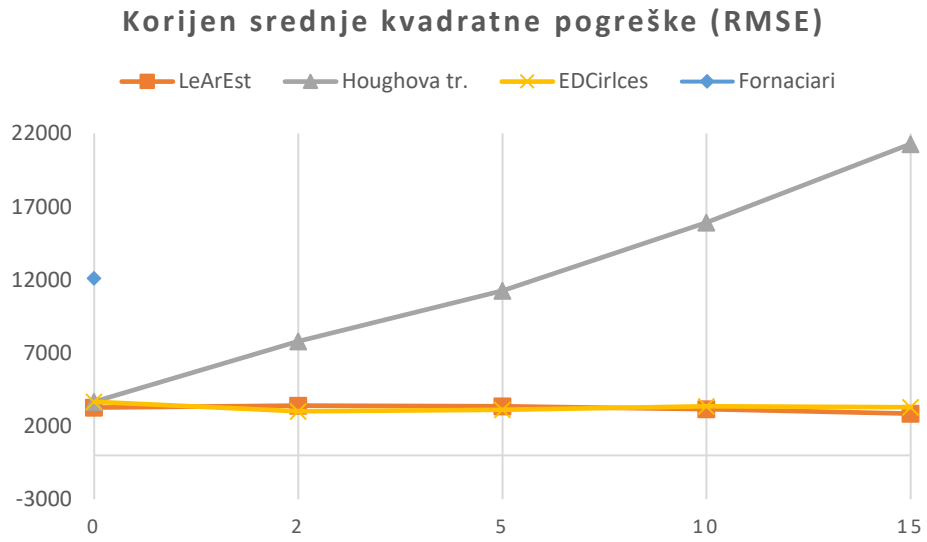
Iz rezultata je vidljivo da algoritmi *LeArEst* i *EDCircles* daju bliske rezultate. *Houghova transformacija* daje manje precizne rezultate, dok je algoritam *Formaciari* detektirao kružnicu samo na jasnoj slici bez aditivne greške.

7.3.1.3. *Objekt* koš za smeće

Rezultati za objekt *koš za smeće* za različite količine šuma i različite algoritme prikazani su u Tablici 7.3 i Slici 7.18.

	Korijen srednje kvadratne pogreške (RMSE)			
sd	**LeArEst**	**Houghova tr.**	**EDCircles**	**Fornaciari**
0	3277	3667	3660	12102
2	3402	7789	3014	
5	3363	11263	3109	
10	3159	15908	3354	
15	2855	21283	3285[2]	

Tablica 7.3 Korijeni srednje kvadratne pogreške za objekt koš za smeće *(20 slika)*



Slika 7.18 Graf korijena srednje kvadratne pogreške za objekt koš za smeće *(20 slika)*

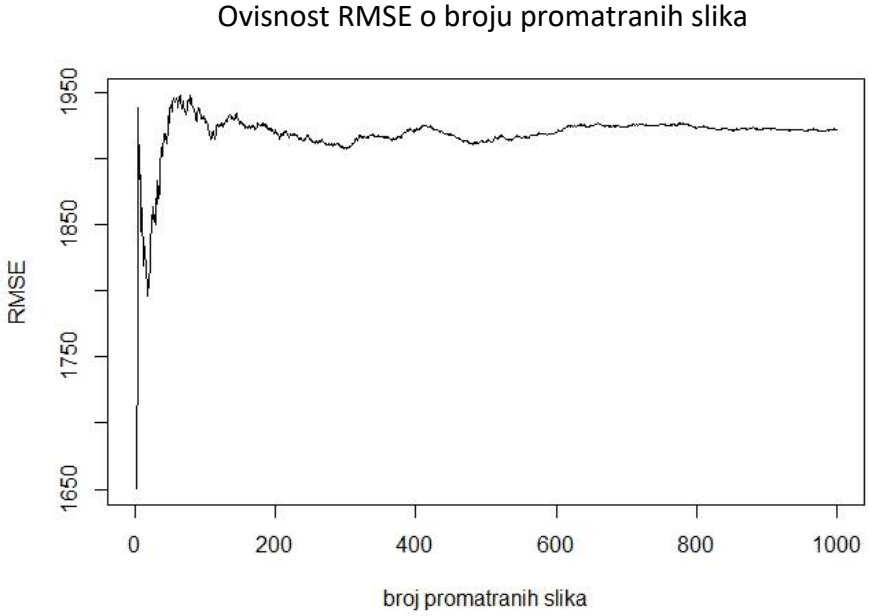
I kod ovog objekta može se primijetiti da algoritmi *LeArEst* i *EDCircles* daju bliske rezultate, dok su rezultati *Houghove transformacije* puno lošiji. Iz Slika 7.4, 7.7 i 7.10 može se primijetiti da je objekt *koš za smeće* na slici puno veći od ostala dva promatrana objekta, pa se može zaključiti da *Houghova transformacija* daje lošije rezultate kod procjene površina većih objekata na slikama s aditivnom greškom.

7.3.2. Rezultati za skup od 1000 generiranih slika

Prilikom statističke obrade u prethodnom odjeljku nametnulo se pitanje koliko izoliranje dvadeset nasumično odabranih slika iz cijelog skupa od tisuću generiranih slika utječe na kvalitetu statističke analize. Kako bi se to provjerilo, promotrena je ovisnost RMSE o broju promatranih

[2] Za jednu promatranu sliku metoda nije detektirala kružnicu

slika za *LeArEst* algoritam. Na Slici 7.19 nalazi se grafički prikaz te ovisnosti za objekt *CD stalak* s aditivnom greškom standardne devijacije 5.



Slika 7.19 Ovisnosti RMSE o broju promatranih slika za LeArEst *algoritam (objekt* CD stalak *s aditivnom greškom standardne devijacije 5)*

Iz prethodne slike primjećuje se da je vrijednost korijena srednje kvadratne pogreške (RMSE) relativno nestabilna za mali broj promatranih slika. Iz tog će se razloga u nastavku promatrati vrijednosti RMSE za svih 1000 generiranih slika, no bez rezultata *EDCircles* algoritma koje, zbog nemogućnosti automatizacije, nije bilo moguće izračunati u razumnom roku.

7.3.2.1. *Objekt* konzerva

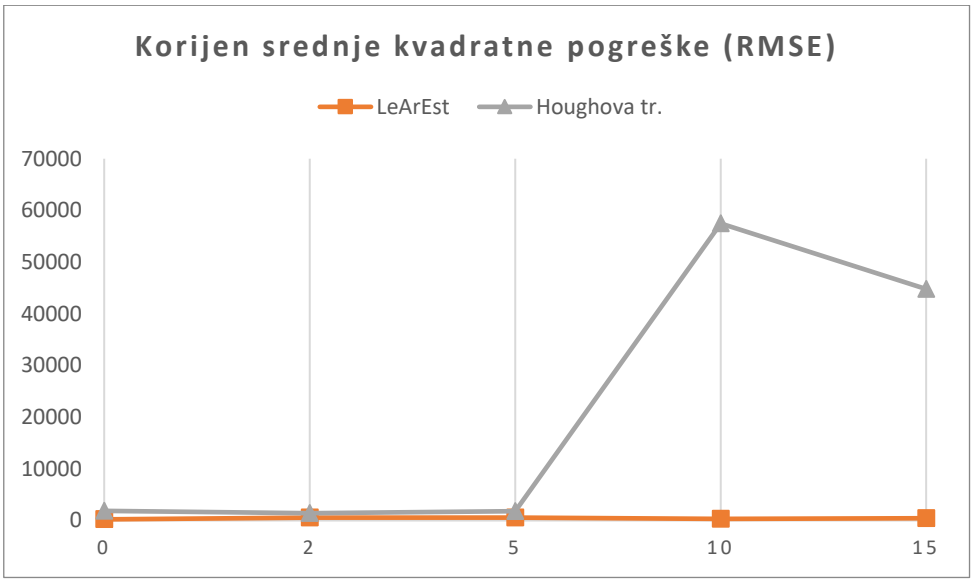
U Tablici 7.4 prikazani su korijeni srednje kvadratne pogreške za objekt *konzerva*. Korištena metodologija je identična onoj u potpoglavlju 7.3.1.1, no ovdje je iskorišteno svih 1000 generiranih slika ($n = 1000$ u jednadžbi (7.1)).

	Korijen srednje kvadratne pogreške (RMSE)		
sd	**LeArEst**	**Houghova tr.**	**Fornaciari**
0	229	1858	
2	565	1417	
5	541	1777	
10	339	57464	

15	440	44785	

Tablica 7.4 Korijeni srednje kvadratne pogreške za objekt konzerva *(1000 slika)*

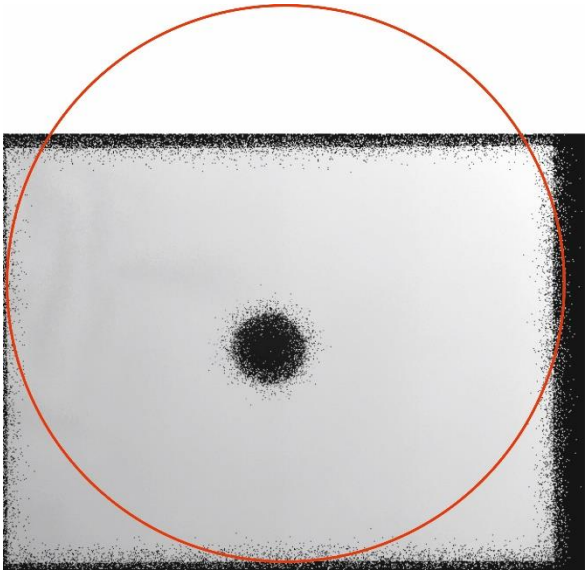
Rezultati iz Tablice 7.4 vizualizirani su na Slici 7.20.



Slika 7.20 Graf korijena srednje kvadratne pogreške za objekt konzerva *(1000 slika)*

Kao i za promatranih 20 slika, algoritam *Fornaciari* nije detektirao kružnicu niti u jednom slučaju.

Iz Tablice 7.4 i Slike 7.20 može se primijetiti da su vrijednosti RMSE za promatrana dva algoritma kod standardnih devijacija manjih ili jednakih 5 istog reda veličine i bliske su vrijednostima kod promatranih 20 nasumično odabranih slika (potpoglavlje 7.3.1.1). Međutim, kod standardnih devijacija 10 i 15 dolazi do naglog skoka RMSE vrijednosti kod *HT* algoritma. Analizom dobivenih rezultata uočava se da je do te pojave došlo zbog toga što za neke slike *HT* algoritam, uz empirijski postavljene vrijednosti dvaju ulaznih parametara, vrlo loše detektira kružnicu (Slika 7.21). Ta se pojava pokušala izbjeći modifikacijom ulaznih parametara, no to nije dalo smislene rezultate.



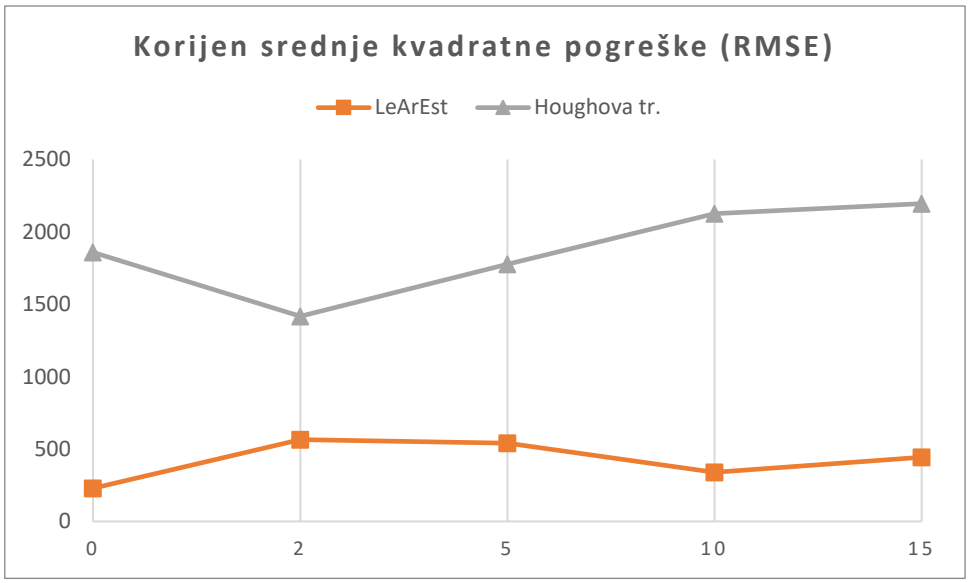
Slika 7.21 Loše procijenjena kružnica kod HT algoritma za objekt konzerva *i sd = 10*

Iako je takvih stršećih vrijednosti u odnosu na ukupan broj slika relativno malo (za sd = 10 od ukupno 1000 slika, 10 ih ima procijenjenu površinu veću od dvostruke realne, a 7 za sd = 15), iznimno loše procijenjene površine kružnica na tim slikama znatno kvare konačne RMSE vrijednosti.

Nadalje je izvršena usporedba *HT* i *LeArEst* algoritama bez slika za koje *HT* algoritam daje stršeće vrijednosti procijenjene površine. Takve su slike izbačene iz skupa testnih podataka za oba algoritma. Rezultati su dani u Tablici 7.5 i Slici 7.22.

	Korijen srednje kvadratne pogreške (RMSE)		
sd	**LeArEst**	**Houghova tr.**	**Fornaciari**
0	229	1858	
2	565	1417	
5	541	1777	
10	339	2125	
15	443	2195	

Tablica 7.5 Korijeni srednje kvadratne pogreške za objekt konzerva *(1000 slika) bez slika za koje HT algoritam daje stršeću vrijednost procijenjene površine*



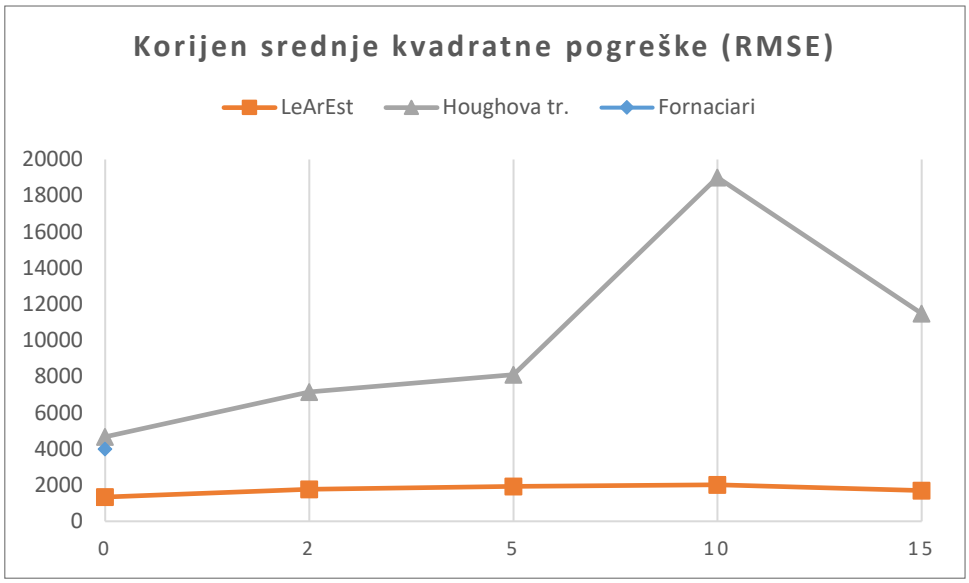
Slika 7.22 Graf korijena srednje kvadratne pogreške za objekt konzerva (1000 slika) bez slika za koje HT algoritam daje stršeću vrijednost procijenjene površine

7.3.2.2. *Objekt* CD stalak

Tablica 7.6 i Slika 7.23 prikazuju korijene srednje kvadratne pogreške za objekt *CD stalak* s različitim standardnim devijacijama aditivne greške i različitim algoritmima procjene površine, kada se u analizu uzme svih 1000 generiranih slika.

	Korijen srednje kvadratne pogreške (RMSE)		
sd	**LeArEst**	**Houghova tr.**	**Fornaciari**
0	1341	4670	3994
2	1776	7145	
5	1922	8098	
10	2019	19009	
15	1708	11477	

Tablica 7.6 Korijeni srednje kvadratne pogreške za objekt CD stalak *(1000 slika)*

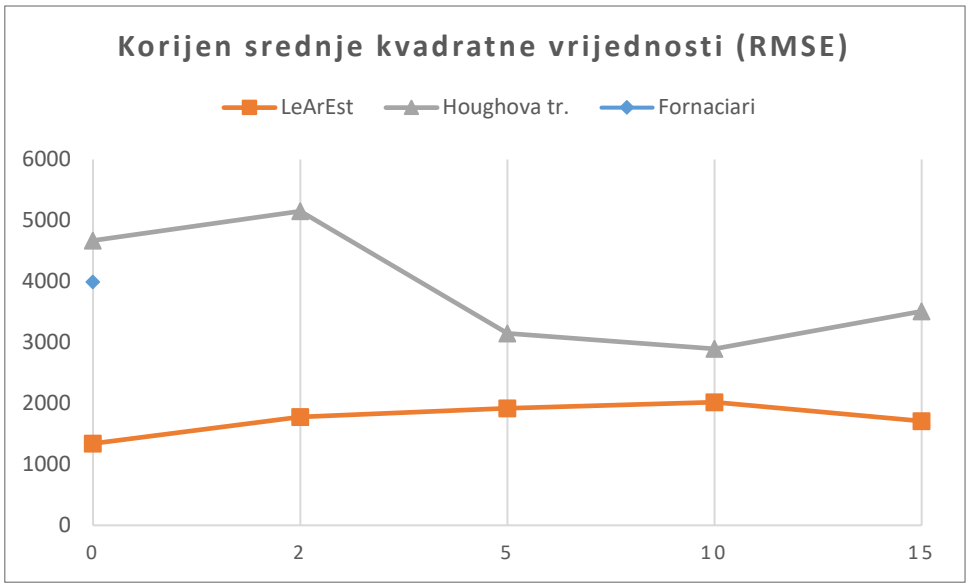


Slika 7.23 Graf korijena srednje kvadratne pogreške za objekt CD stalak *(1000 slika)*

Za ovaj objekt algoritam *Fornaciari* detektira kružnicu samo kada slici nije dodana aditivna greška. *HT* algoritam daje znatno lošije rezultate (veću RMSE vrijednost) za sve vrijednosti standardne devijacije, uz napomenu da je kod sd = 2, sd = 5 i sd = 15 prisutna jedna stršeća vrijednost procijenjene površine, a 4 kod sd = 10. Ako se slike za koje *HT* algoritam daje stršeće vrijednosti izbace iz skupa testnih podataka, dobiju se korijeni srednje kvadratne pogreške prikazani u Tablici 7.7 i Slici 7.24.

	Korijen srednje kvadratne pogreške (RMSE)		
sd	**LeArEst**	**Houghova tr.**	**Fornaciari**
0	1341	4670	3994
2	1776	5152	
5	1922	3148	
10	2019	2894	
15	1708	3512	

Tablica 7.7 Korijeni srednje kvadratne pogreške za objekt CD stalak *(1000 slika) bez slika za koje HT algoritam daje stršeću vrijednost procijenjene površine*



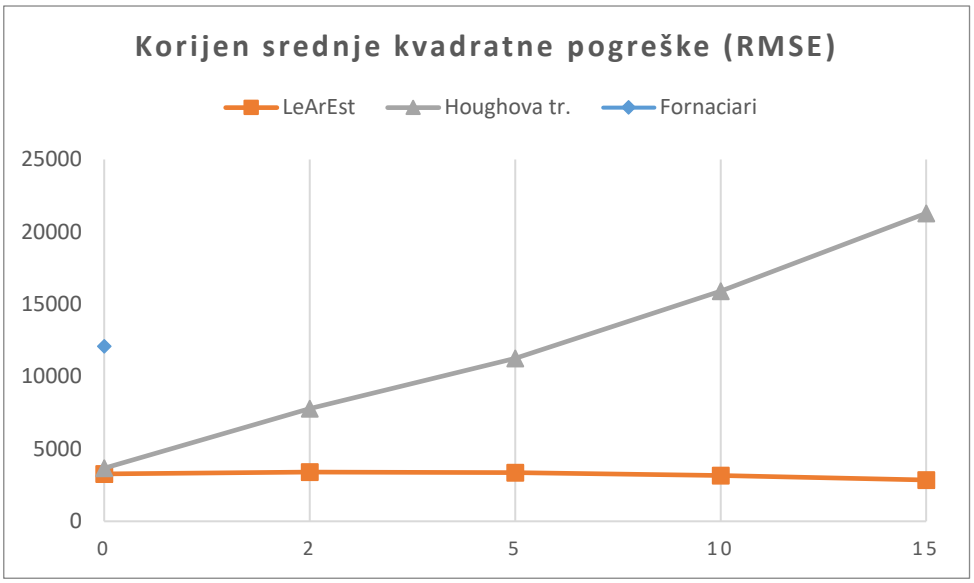
Slika 7.24 Graf korijena srednje kvadratne pogreške za objekt CD stalak *(1000 slika) bez slika za koje HT algoritam daje stršeću vrijednost procijenjene površine*

7.3.2.3. *Objekt* koš za smeće

Rezultati statističke analize procjene površine kružnica na 1000 generiranih slika za objekt *koš za smeće* za različite standardne devijacije greške i različite algoritme procjene prikazani su u Tablici 7.8 i Slici 7.25.

	Korijen srednje kvadratne pogreške (RMSE)		
sd	**LeArEst**	**Houghova tr.**	**Fornaciari**
0	3277	3667	12102
2	3402	7789	
5	3363	11263	
10	3159	15908	
15	2855	21283	

Tablica 7.8 Korijeni srednje kvadratne pogreške za objekt koš za smeće *(1000 slika)*



Slika 7.25 Graf korijena srednje kvadratne pogreške za objekt koš za smeće *(1000 slika)*

Za razliku od prethodna dva promatrana objekta, kod ovoga se za *HT* algoritam nisu pojavile stršeće vrijednosti procijenjene površine, tako da su vrijednosti RMSE vrijednosti bliske onima kod 20 promatranih slika.

7.4. Analiza rezultata

Dok je kod *LeArEst* algoritma red veličine RMSE vrijednosti manje ovisan o aditivnoj grešci za sve promatrane objekte, kod *HT* algoritma i objekta velike površine (*koš za smeće*) RMSE vrijednost znatno raste s porastom standardne devijacije aditivne greške. Kod manjih objekata RMSE vrijednost za *HT* algoritam manje ovisi o standardnoj devijaciji greške, no u svakom slučaju ona je lošija u odnosu na *LeArEst* algoritam. Nadalje, veliki problem kod *HT* algoritma je pojava stršećih vrijednosti u procijenjenim površinama. Uvidom u rezultate za skup od 1000 generiranih slika može se primijetiti da je broj stršećih vrijednosti procijenjene površine obrnuto proporcionalan površini objekta. Napravljena je statistička analiza za skup podataka iz kojeg su uklonjene slike za koje *HT* algoritam daje stršeću vrijednost procijenjene površine. Iz te je analize vidljivo da čak i u tom slučaju *LeArEst* algoritam preciznije procjenjuje površinu objekta na slici.

EDCircles algoritam uglavnom daje rezultate usporedive sa *LeArEst* algoritmom. Međutim, kod tog se algoritma pojavio problem što kod slika s aditivnim greškama velikih standardnih devijacija često ne detektira kružnicu (15% slika za objekt *konzerva* i 5% slika za objekt *koš za smeće*).

Algoritam *Fornaciari* nije usporediv s ostala tri promatrana algoritma jer ne uspijeva detektirati kružnicu čim je na slici prisutna bilo kakva aditivna greška. Kod najmanjeg objekta (*konzerva*) ovaj algoritam nije detektirao kružnicu niti na slici bez aditivne greške.

8. Zaključak

Ova se disertacija bavi problemom procjene površine kružnog ili elipsoidnog objekta kada je on snimljen na slici koja je loše kvalitete, odnosno na kojoj je prisutan šum. Korištenjem metodologije temeljene na prirodnom parametarskom modelu za procjenu nosača uniformne distribucije mjerene s greškom razvijen je softver *LeArEst*. Tim je softverom moguće procijeniti duljinu presjeka objekta s proizvoljnim pravcem i procijeniti površinu objekta zadanog numerički kao skup dvodimenzionalnih točaka ili očitanog sa slike. Softver je razvijen kao paket za programski jezik R te ga je moguće slobodno preuzeti i koristiti iz središnjeg repozitorija R paketa (CRAN).

U prvom izvornom znanstvenom doprinosu ove disertacije izrađen je algoritam i programski modul za procjenu duljine presjeka proizvoljnog pravca i objekta na slici snimljenoj s aditivnom greškom. Algoritam promatra piksele slike na kojoj je prikazan objekt u okolici pravca i stvara novi pseudoprostor logičkih točaka na način da se svakom pikselu slike pridružuje kvadratna matrica logičkih točaka. Svaka se takva matrica potom uniformno ispunjava s logičkim vrijednostima *istina* kojih će biti proporcionalno svjetlini piksela slike kojem je pridružena. Te se logičke matrice zatim spajaju u veliku matricu točaka te se točke koje predstavljaju logičke vrijednosti *istina* u okolici pravca koji siječe objekt ortogonalno projiciraju na sam pravac. Konačno, promatraju se udaljenosti ortogonalnih projekcija od neke proizvoljne početne točke na pravcu i tako kreira histogram udaljenosti točaka iz kojeg se može procijeniti širina uniformne distribucije. Povratkom iz pseudoprostora logičkih točaka u početni prostor piksela slike, širina uniformne distribucije spomenutog histograma pretvara se u duljinu presjeka pravca i objekta. Predstavljeni algoritam realiziran je kao web aplikacija u programskom paketu *LeArEst*.

U drugom izvornom znanstvenom doprinosu razvijen je algoritam za procjenu površine kružnog ili elipsoidnog objekta na slici snimljenoj s aditivnom greškom. Opisani algoritam promatra presjeke objekta te, koristeći rezultate prvog znanstvenog doprinosa, procjenjuje rubne točke objekta. Predstavljane su dvije različite metode za presijecanje objekta i opisane njihove specifičnosti. Nakon izračuna skupa rubnih točaka koriste se dobro poznate metode za optimalno smještanje kružnice ili elipse u procijenjeni skup točaka. U konačnici se iz parametarski izražene

kružnice ili elipse izračuna njezina površina – procijenjena površina promatranog objekta. Algoritam je realiziran u paketu *LeArEst* u dva oblika: kao funkcija, za slučaj da je objekt numerički opisan skupom dvodimenzionalnih točaka, i kao web aplikacija, za slučaj da je objekt snimljen na slici.

Treći izvorni znanstveni doprinos daje objektivnu ocjenu predloženog algoritma za procjenu površine usporedbom s drugim algoritmima na referentnim podacima. U analizi je korištena RGB-D kamera kojom je snimljeno nekoliko objekata s kružnim plohama. Od tih snimaka kreirane su crno-bijele slike na način da je svjetlina svakog piksela slike proporcionalna udaljenosti odgovarajuće točke objekta od objektiva kamere. Slikama je potom dodan šum različitih standardnih devijacija. Algoritam razvijen u sklopu drugog znanstvenog doprinosa korišten je za procjenu površina kružnih ploha objekata iz slika sa šumom. Također, te su površine procijenjene i korištenjem nekoliko iz literature najpoznatijih algoritama za detekciju kružnica. Budući da je realnu površinu ploha bilo moguće izmjeriti, napravljena je detaljna statistička usporedba svih algoritama korištenjem korijena srednje kvadratne greške (RMSE). Pokazalo se da u većini slučajeva *LeArEst* algoritam kvalitativno najpreciznije detektira kružnicu na slikama na kojima je prisutan šum, a samim time i najpreciznije određuje površinu kružnih ploha snimljenih objekata.

Algoritam predstavljen u ovoj disertaciji bilo bi moguće eventualno daljnje poboljšati kako bi procjenjivao površine objekata koji nisu nužno kružni ili elipsoidni. U tu svrhu bi se, nakon procjene skupa rubnih točaka kao što je u radu prethodno opisano, objekt reprezentirao konveksnim poligonom kako bi se korištenjem neke od iz literature poznatih metoda (npr. [52]) procijenila površina tog poligona. Još jedno potencijalno poboljšanje je razdvajanje klijentskog od poslužiteljskog dijela web aplikacija. Trenutno se oba ova dijela izvršavaju na istom, klijentskom računalu, međutim odvajanjem poslužiteljskog dijela aplikacije na za to predodređenu snažniju radnu stanicu brzina izvođenja web aplikacije postala bi praktički neovisna o snazi računala na kojem je pokrenut njezin klijentski dio.

9. Literatura

[1] I. Bankman, Handbook of medical image processing and analysis, Elsevier, 2008.

[2] S. E. Ruzin, Plant microtechnique and microscopy. Vol. 198, New York: Oxford University Press, 1999.

[3] D. J. Daniels, Ground penetrating radar. Vol. 1, Iet, 2004.

[4] M. A. Farooque i J. S. Rohankar, »Survey on various noises and techniques for denoising the color image,« *International Journal of Application or Innovation in Engineering & Management (IJAIEM),* svez. 2, br. 11, pp. 217-221, 2013.

[5] P. V. Hough, »Method and means for recognizing complex patterns«. Patent U.S. Patent No. 3,069,654, 1962.

[6] R. D. Duda i P. E. Hart, »Use ofthe Hough transform to detectlinesand curvesin pictures,« *Commun. ACM,* svez. 15, br. 1, 1972.

[7] A. Rosenfeld, »Picture processing by computer,« *ACM Computing Surveys (CSUR),* svez. 1, br. 3, pp. 147-176, 1969.

[8] C. Akinlar i C. Topal, »EDCircles: A real-time circle detector with a false detection control,« *Pattern Recognition,* svez. 46, br. 3, pp. 725-740, 2013.

[9] M. Fornaciari, A. Prati i R. Cucchiara, »A fast and effective ellipse detector for embedded vision applications,« *Pattern Recognition,* svez. 47, br. 11, pp. 3693-3708, 2017.

[10] D. H. Ballard, »Generalizing the Hough transform to detect arbitrary shapes,« u *Readings in computer vision*, Elsevier, 1987, pp. 714-725.

[11] J. Canny, A computational approach to edge detection, Elsevier, 1987, pp. 184-203.

[12] D. Shaked, O. Yaron i N. Kiryati, »Deriving stopping rules for the probabilistic Hough transform by sequential analysis,« *Computer Vision and Image Understanding,* svez. 63, br. 3, pp. 512-526, 1996.

[13] L. Xu, E. Oja i P. Kultanen, »A new curve detection method: randomized Hough transform (RHT),« *Pattern recognition letters,* svez. 11, br. 5, pp. 331-338, 1990.

[14] J. H. Han, L. T. Kóczy i T. Poston, »Fuzzy hough transform,« *Pattern recognition letters,* svez. 15, br. 7, pp. 649-658, 1994.

[15] C. Akinlar i C. Topal, »Edge drawing: a combined real-time edge and segment detector,« *Journal of Visual Communication and Image Representation,* svez. 23, br. 6, pp. 862-872, 2012.

[16] C. Akinlar i C. Topal, »EDPF: a real-time parameter-free edge segment detector with a false detection control,« *International Journal of Pattern Recognition and Artificial Intelligence,* svez. 26, br. 1, p. 1255002, 2012.

[17] F. J. J. Clarke i D. J. Parry, »Helmholtz reciprocity: its validity and application to reflectometry,« *Lighting Research & Technology,* svez. 17, br. 1, pp. 1-11, 1985.

[18] R. Bullock, »Least squares circle fit,« 2006. [Mrežno]. Available: https://dtcenter.org/met/users/docs/write_ups/circle_fit.pdf. [Pokušaj pristupa 12. 6. 2018.].

[19] C. Akinlar i C. Topal, »EDLines: a real-time line segment detector with a false detection control,« *Pattern Recognition Letters,* svez. 32, br. 13, pp. 1633-1642, 2011.

[20] A. Fitzgibbon, P. Maurizio i R. B. Fisher, »Direct least square fitting of ellipses,« *IEEE Transactions on pattern analysis and machine intelligence,* svez. 21, br. 5, pp. 476-480, 1999.

[21] L. M. J. M. M. A. Desolneux, From Gestalt Theory to Image Analysis: A Probabilistic Approach, Springer, 2008.

[22] L. M. J. M. M. A. Desolneux, »Meaningful alignments,« *International Journal of Computer Vision,* svez. 40, br. 1, pp. 7-23, 2000.

[23] I. Sobel, »An isotropic 3x3 image gradient operator,« *Machine vision for three-dimensional scenes,* pp. 376-379, 1990.

[24] M. Crowder, »Deconvolution Problems in Nonparametric Statistics by Alexander Meister,« *International Statistical Review,* svez. 78, br. 1, pp. 142-143, 2010.

[25] H. Schneeweiss, »Estimating the endpoint of a uniform distribution under measurement errors,« *Central European Journal of Operations Research,* svez. 12, br. 2, pp. 221-231, 2004.

[26] M. Benšić i K. Sabo, »Border estimation of a disc observed with random errors solved in two steps,« *Journal of computational and applied mathematics,* svez. 229, br. 1, pp. 16-26, 2009.

[27] M. Benšić i K. Sabo, »Border estimation of a two-dimensional uniform distribution if data are measured with additive error,« *Statistics,* svez. 41, br. 4, pp. 311-319, 2007.

[28] M. Benšić i K. Sabo, »Estimating a uniform distribution when data are measured with a normal additive error with unknown variance,« *Statistics,* svez. 44, br. 3, pp. 235-246, 2010.

[29] M. Benšić i K. Sabo, »Estimating the width of a uniform distribution when data are measured with additive normal errors with known variance,« *Computational statistics \& data analysis,* svez. 51, br. 9, pp. 4731-4741, 2007.

[30] M. Benšić i K. Sabo, »Uniform distribution width estimation from data observed with Laplace additive error,« *Journal of the Korean Statistical Society,* svez. 45, br. 4, pp. 505-517, 2016.

[31] M. Benšić, P. Taler, S. Hamedović, E. K. Nyarko i K. Sabo, »LeArEst: Length and Area Estimation from Data Measured with Additive Error,« *The R Journal,* svez. 9, br. 2, pp. 461-473, 2017.

[32] I. Kåsa, »A circle fitting procedure and its error analysis,« *IEEE Transactions on instrumentation and measurement,* svez. 1001, br. 1, pp. 8-14, 1976.

[33] V. Pratt, »Direct least-squares fitting of algebraic surfaces,« *ACM SIGGRAPH computer graphics,* svez. 21, br. 4, pp. 145-152, 1987.

[34] A. Al-Sharadqah i N. Chernov, »Error analysis for circle fitting algorithms,« *Electronic Journal of Statistics,* svez. 3, pp. 886-911, 2009.

[35] T. Garlipp i C. H. Müller, »Detection of linear and circular shapes in image analysis,« *Computational statistics & data analysis,* svez. 51, br. 3, pp. 1479-1490, 2006.

[36] J. A. Hartigan i M. A. Wong, »Algorithm AS 136: A k-means clustering algorithm,« *Journal of the Royal Statistical Society. Series C (Applied Statistics),* svez. 28, br. 1, pp. 100-108, 1979.

[37] »R: The R Project for Statistical Computing,« [Mrežno]. Available: https://www.r-project.org/. [Pokušaj pristupa 14. 8. 2018.].

[38] »The Comprehensive R Archive Network,« [Mrežno]. Available: https://cloud.r-project.org/. [Pokušaj pristupa 14. 8. 2018.].

[39] »LeArEst: Border and Area Estimation of Data Measured with Additive Error,« [Mrežno]. Available: https://cran.r-project.org/package=LeArEst. [Pokušaj pristupa 14. 8. 2018.].

[40] »doParallel: Foreach Parallel Adaptor for the 'parallel' Package,« [Mrežno]. Available: https://cran.r-project.org/package=doParallel. [Pokušaj pristupa 16. 8. 2018.].

[41] »foreach: Provides Foreach Looping Construct for R,« [Mrežno]. Available: https://cran.r-project.org/package=foreach. [Pokušaj pristupa 16. 8. 2018.].

[42] S. Weston i R. Calaway, »Getting Started with doParallel and foreach,« [Mrežno]. Available: https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf. [Pokušaj pristupa 16. 8. 2018.].

[43] »conicfit: Algorithms for Fitting Circles, Ellipses and Conics Based on the Work by Prof. Nikolai Chernov,« [Mrežno]. Available: https://cran.r-project.org/package=conicfit. [Pokušaj pristupa 16. 8. 2018.].

[44] »shiny: Web Application Framework for R,« [Mrežno]. Available: https://cran.r-project.org/package=shiny. [Pokušaj pristupa 16. 8. 2018.].

[45] »opencpu: Producing and Reproducing Results,« [Mrežno]. Available: https://cran.r-project.org/package=opencpu. [Pokušaj pristupa 16. 8. 2018.].

[46] J. J. Garrett, »Ajax: A New Approach to Web Applications,« 18. 2. 2005.. [Mrežno]. Available: http://adaptivepath.org/ideas/ajax-new-approach-web-applications/. [Pokušaj pristupa 16. 8. 2018.].

[47] »Xtion PRO LIVE | 3D Sensor,« ASUS Global, [Mrežno]. Available: https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/. [Pokušaj pristupa 20. 11. 2018.].

[48] R. C. Bolles i M. A. Fischler, »A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data,« *IJCAI,* svez. 1981, pp. 637-643.

[49] O. Galo, R. Manduchi i R. Abbas, »CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data,« *Pattern Recognition Letters,* svez. 32, br. 3, pp. 403-410, 2011..

[50] V. Schomaker, J. Waser, R. A. Marsh i G. Bergman, »To fit a plane or a line to a set of points by least squares,« *Acta crystallographica,* svez. 12, br. 8, pp. 600-604, 1959.

[51] »EDCircles Downloads,« [Mrežno]. Available: http://ceng.eskisehir.edu.tr/CV/downloads/downloads.aspx. [Pokušaj pristupa 26. 11. 2018.].

[52] E. W. Weisstein, »Polygon Area,« MathWorld - A Wolfram Web Resource, [Mrežno]. Available: http://mathworld.wolfram.com/PolygonArea.html. [Pokušaj pristupa 23. 1. 2019.].

10. Sažetak

Detekcija kružnog ili elipsoidnog objekta snimljenog na slici aktualan je problem u segmentu računalnog vida. U literaturi je predstavljeno nekoliko metoda koje se bave tim problemom, no pokazalo se da one ne daju zadovoljavajuće rezultate kod slika na kojima je prisutan šum. U sklopu istraživanja predstavljenog u ovom radu primijenjen je statistički model za procjenu širine uniformne distribucije za podatke zagađene aditivnom greškom kako bi se spomenutom problemu pristupilo na inovativan način. Inicijalno je razvijena metoda za procjenu duljine presjeka objekta s proizvoljnim pravcem. Njezinim je korištenjem moguće procijeniti skup rubnih točaka kružnog ili elipsoidnog objekta na slici. U taj se skup točaka potom optimalno smješta kružnica ili elipsa te izračuna njezina površina - aproksimacija površine samoga objekta. Moguće je također procijeniti i površinu kružnog ili elipsoidnog objekta predstavljenog numerički skupom točaka u ravnini.

Predstavljena metoda je implementirana i javno objavljena kao paket za programski jezik R. Izvršeno je njezino opsežno testiranje na problemu procjene površine objekata snimljenih RGB-D kamerom. Snimkama s kamere je dodavana različita količina šuma, te su rezultati procjene površine uspoređeni s dobivenima od ranije poznatih metoda. Analiza rezultata testiranja pokazala je da ovdje predstavljena metoda daje kvalitativno najbolje rezultate procjene površine objekata snimljenih na slikama sa šumom.

Ključne riječi: detekcija kružnice i elipse, procjena površine, slika sa šumom

11. Abstract

Detection of circular or ellipsoidal object on an image is a current issue in computer vision. Several methods that address this problem have been previously presented, but it turned out that they do not give satisfactory results when dealing with noisy images. As part of the research presented in this paper, a statistical model for estimating the width of uniform distribution for data contaminated with additive error was applied in order to approach mentioned problem in an innovative manner. Initially, a method for length estimation of intersection of the object with an arbitrary line has been developed. It is possible to estimate the set of edge points of the circular or ellipsoidal object by using this method. Further, a circle or an ellipse is fitted in that set of points, and its area is calculated, which approximates the area of on object itself. It is also possible to estimate the area of a circular or ellipsoidal object represented by a set of numerical points in the plane.

The presented method was implemented and publicly released as a package for the programming language R. The method has been extensively tested on the problem of estimating the area of objects recorded using RGB-D camera. Different noise levels were added to the captured images, and estimation results were compared with the ones obtained by a several established methods. The test results analysis showed that the method presented in this paper gives qualitatively the best results of area estimation when dealing with noisy images.

Keywords: circle or ellipse detection, area estimation, noisy image

12. Extended Abstract

Detection of circular or ellipsoidal object on an image is a current issue in computer vision. This problem occurs, for example, when the object is captured by a fluorescent microscope, a ground penetrating radar, X-ray or ultrasound machines, etc. The dimensions of an object can be calculated from its edges, using some edge detection techniques, but it is not an easy task. Several methods that address this problem have been previously presented, but it turned out that they do not give satisfactory results when dealing with noisy images. As part of the research presented in this paper, a statistical model for estimating the width of uniform distribution for data contaminated with additive error was applied in order to approach mentioned problem in an innovative manner.

Initially, a method for length estimation of intersection of the object with an arbitrary line has been developed. The algorithm captures the pixels of an image in the vicinity of the line that intersects the object and creates a pseudo space of boolean points in such a way that each pixel of the image is joined with the square boolean matrix. Each boolean matrix is then uniformly fulfilled with the values of *true* proportional to the brightness of the associated pixel. The matrices are then merged into single large boolean matrix and the points (values of *true*) nearby the intersecting line are orthogonally projected to the line. Finally, the distances of such created projected points to an arbitrary point on the line are used to create a histogram of distances. Using that histogram, the width of the uniform distribution, i.e. the length of intersection of the object with the line is estimated.

The presented method is further used to estimate the area of the circular or ellipsoidal object. The object is being cut with the series of lines and in such manner the set of edge points is being estimated. Two different methods for cutting the object were presented. A circle or an ellipse is then fitted into the set of edge points using some of the well-known curve fitting algorithms. Finally, the area of such created circle or ellipse is calculated, which is the approximation of the observed object's area. It is worth mentioning that it is also possible to estimate the area of a circular or ellipsoidal object represented by a set of numerical points in the plane using this method.

The presented methods were implemented and publicly released as a package for the programming language R. The methods have been extensively tested on the problem of estimating the area of objects recorded using RGB-D camera. Different noise levels were added to the captured images, and estimation results were compared with the ones obtained by a several established methods. The test results analysis showed that the methods presented in this paper give qualitatively the best results of area estimation when dealing with noisy images.

Keywords: circle or ellipse detection, area estimation, noisy image

13. Životopis

Petar Taler rođen je 30. travnja 1978. u Osijeku. Nakon završene osnovne škole u Belišću i prirodoslovno-matematičke gimnazije u Valpovu upisuje Elektrotehnički fakultet u Osijeku gdje diplomira 2002. godine pod mentorstvom doc. dr. sc. Darka Fischera s temom „Huffmanovo stablo“. Nakon toga upisuje poslijediplomski magistarski studij i magistrira 2010. godine pod mentorstvom prof. dr. sc. Gorana Martinovića s temom „Izgradnja raspodijeljenog računalnog sustava za učinkovito izvođenje paralelnih programa“.

Zaposlen je na Odjelu za matematiku Sveučilišta u Osijeku prvo kao sistem-inženjer (2002.-2008.), a potom kao voditelj odjeljka za informatiku i računalnu mrežu (2008.-). Sudjeluje u znanstvenom istraživanju na projektu „The optimization and statistical models and methods in recognizing properties of data sets measured with errors“ od 2017. godine. Izabran je za asistenta u naslovnom zvanju (2008.-2016.) te predavača u naslovnom zvanju (2016.-) na Odjelu za matematiku, gdje održava vježbe iz kolegija Uvod u računarstvo i Uvod u programiranje na Sveučilišnom preddiplomskom studiju matematike i Sveučilišnom nastavničkom studiju matematike i informatike, te predavanja i vježbe iz kolegija Web programiranje i primjene na Sveučilišnom nastavničkom studiju matematike i informatike i Sveučilišnom diplomskom studiju matematike.

Stekao je industrijske certifikate MCP (*Microsoft Certified Professional*, 2005.) i CCNA (*Cisco Certified Network Associate*, 2007.).

Tijekom poslijediplomskog studija objavio je pet radova u znanstvenim časopisima i dva rada u zbornicima skupova s međunarodnih konferencija.

Njegovo područje istraživanja obuhvaća primijenjenu statistiku, segmentiranje velikih skupova podataka i distribuirano računalstvo.