Generiranje umjetnih slika keramičkih pločica

Dukić, Jana

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

GENERIRANJE UMJETNIH SLIKA KERAMIČKIH PLOČICA

Završni rad

Jana Dukić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.07.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Jana Dukić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4193, 24.07.2018.
OIB studenta:	47967888925
Mentor:	Izv. prof. dr. sc. Ivan Aleksi
Sumentor:	Izv. prof. dr. sc. Tomislav Matić
Sumentor iz tvrtke:	
Naslov završnog rada:	Generiranje umjetnih slika keramičkih pločica
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	07.07.2021.
Datum potvrde ocjene Odbora:	14.07.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 14.07.2021.

Ime i prezime studenta:	Jana Dukić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4193, 24.07.2018.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Generiranje umjetnih slika keramičkih pločica**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivan Aleksi

i sumentora Izv. prof. dr. sc. Tomislav Matić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Svrha ovog završnog rada bila je napraviti dva različita MATLAB programa koji će generirati pogreške na postojećim fotografijama keramičkih pločica. U promatranom skupu uslikanih pločica postoje dvije vrste slika: pločice bez greške i pločice s greškom. Također, uočeno je da se prilikom proizvodnje rijetko događaju pogreške. Kada se pogreške dogode, dolazi do ponavljanja nekoliko vrsta istih. Primjer takvih pogrešaka su točkaste pogreške, pogreške izlijevanja boje i slično. Upravo iz razloga što je skup slika keramičkih pločica s greškom bio vrlo malen, svega 30 slika, javila se potreba za kreiranjem slika s umjetnim pogreškama. Programi načinjeni u sklopu ovog završnog rada planiraju biti upotrebljeni za generiranje velikog broja umjetnih slika s pogreškom koje će se kasnije koristiti kao osnovni skup podataka za učenje neuronskih mreža. Pri treniranju neuronskih mreža najbolje bi bilo imati podjednak skup slika s greškom i bez greške te će se nedostatak slika s greškom upotpuniti pomoću navedenih programa [1].

U ovom završnom radu bit će prikazano generiranje jedne vrste pogrešaka na skupu ispravnih fotografija keramičkih pločica. Vrsta pogreške koja će se generirati je točkasta pogreška (engl. *dot mistake*). Proučavajući točkaste pogreške, uočeno je kako se one mogu približno opisati dvodimenzionalnom Gaussovom funkcijom. Moguće točkaste pogreške su kružna pogreška, elipsoidna pogreška te elipsoidna pogreška zakrenuta za određeni kut. Dodavanje pogreške je omogućeno s dva MATLAB programa. Prvi program omogućava ručno dodavanje proizvoljnog broja točkastih pogrešaka na pločicu, dok drugi program omogućava automatsko generiranje pogrešaka gdje se broj pogrešaka može također automatski izgenerirati ili može biti unaprijed zadan za svaku sliku.

1.1. Zadatak završnog rada

Zadatak završnog rada je bio umjetno generirati točkaste pogreške koje će odgovarati stvarnim točkastim pogreškama koje se javljaju u proizvodnji. Nakon generiranja točkaste pogreške, potrebno ju je ubaciti u stvarnu sliku pločice koja nema grešku te paziti da se pogreška doda samo na prostor pločice. Nakon dodavanja pogreške, stvorenu sliku potrebno je spremiti u zaseban direktorij.

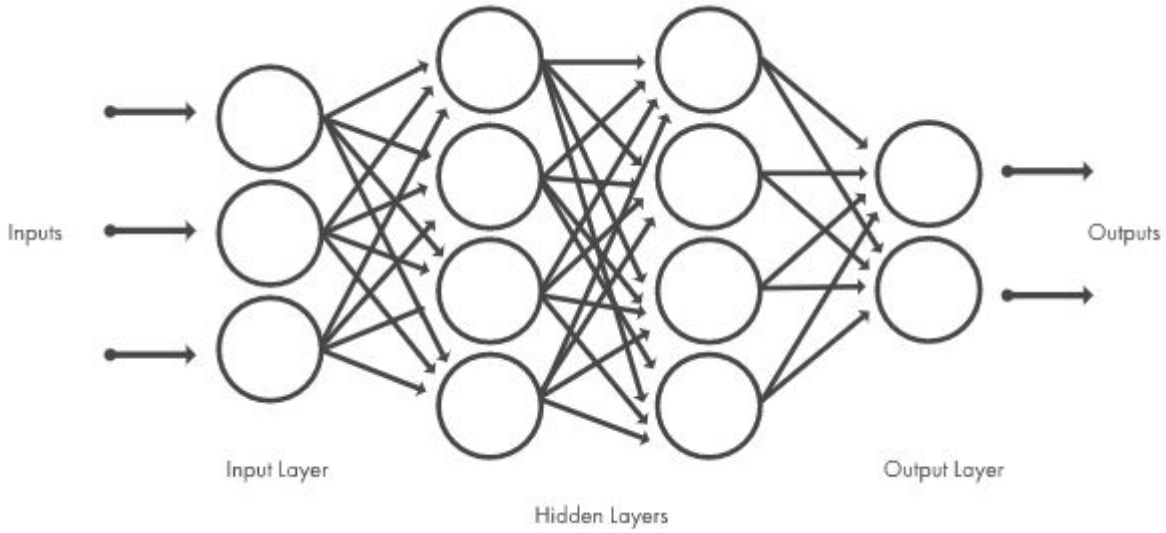
2. PREGLED POSTOJEĆIH I PRIMIJENJENIH METODA

Cilj je u budućem radu primijeniti metode dubokog učenja. Takve metode zahtijevaju puno označenih (labeliranih) slika kao dobre (bez greške) i loše (s greškom) slike. U ovom radu napravljena je priprema za tu metodu u smislu da se omogućilo umjetno generiranje slika s pogreškom, s obzirom da ne postoji dovoljno velik skup navedenih slika. U radu se koristila metoda prelaska praga osjetljivosti za uklanjanje pozadine iz keramičkih pločica, dok su umjetno generirane slike s pogreškama dobivene pomoću dvodimenzionalne Gaussove funkcije.

2.1. Metoda otkrivanja pogrešaka u slikama zasnovana na dubokom učenju

Duboko učenje predstavlja metodu strojnog učenja, koja omogućava treniranje umjetne inteligencije za predviđanje rezultata u odnosu na predani skup ulaza. Duboko učenje koristi arhitekturu neuronske mreže [2]. Neuronske mreže čine podskup strojnog učenja, čiji je naziv i struktura inspirirana ljudskim mozgom te rade na principu oponašanja načina na koji biološki neuroni međusobno signaliziraju. Svaka neuronska mreža sastoji se od tri vrste slojeva: ulazni (engl. *input layer*), skriveni (engl. *hidden layer(s)*) i izlazni (engl. *output layer*) sloj [3]. Primjer neuronske mreže prikazan je slikom 2.1. Riječ "duboko" iz naziva duboko učenje označava kako postoji više od jednog skrivenog sloja. Svaki od slojeva ima zasebnu funkciju, te kako bi se neuronska mreža što bolje istrenirala, potreban je veliki skup označenih podataka i velika snaga računala za obradu tih podataka. Metode dubokog učenja uvelike se primjenjuju jer se povećanjem skupa označenih podataka, dolazi do veće točnosti na izlazu. U skupu označenih podataka postoje podaci za treniranje i podaci za testiranje metode. Podaci za treniranje sastoje se od skupa podataka koji su ispravni i skupa podataka koji su neispravni, gdje ta dva skupa trebaju biti podjednake veličine. Proučavajući predloženi skup podataka koji se koristio u ovom radu, uočen je nedostatak neispravnih podataka, odnosno fotografija s greškom. Fotografije bez greške se smatraju ispravnim skupom podataka. Upravo zbog nedostatka fotografija s greškom, onemogućeno je daljnje istraživanje i kvalitetna primjena metode za otkrivanje pogrešaka te se zbog toga javila potreba za ovim radom. Primjeri metoda za otkrivanje pogrešaka u slikama, zasnovanih na dubokom učenju su: R-CNN, Fast R-CNN, Faster R-CNN, YOLO i slično. Jedna od najpopularnijih vrsta dubokih neuronskih mreža je poznata pod nazivom konvolucijska neuronska mreža (engl. *CNN, ConvNet*) [2]. Konvolucijska neuronska mreža konvoluira naučene značajke s unesenim podacima i koristi dvodimenzionalne konvolucijske slojeve, što ovu arhitekturu čini pogodnim za obradu dvodimenzionalnih podataka poput slika [2]. Spomenuta mreža radi na principu izvlačenja značajki izravno sa slika, gdje relevantne značajke nisu prethodno obučene, nego se uče dok mreža trenira na predanom skupu podataka. Automatizirano izdvajanje značajki čine modele dubokog učenja preciznim za zadatke računalnog vida poput klasifikacije predmeta [2]. Konvolucijske neuralne mreže nauče otkriti različite značajke slike pomoću desetak ili stotina skrivenih slojeva, gdje svaki skriveni

sloj povećava složenost naučenih značajki slike. Primjerice, prvi skriveni sloj mogao bi naučiti kako otkriti rubove, a posljednji kako otkriti složenije oblike prilagođene obliku predmeta koji pokušavamo prepoznati [2]. Upravo spomenuta klasifikacija predmeta će se nastojati primijeniti na skup fotografija keramičkih pločica, gdje će se one svrstavati u pločice s greškom i u pločice bez greške.



SL. 2.1: *Primjer neuronske mreže [2].*

2.2. Metoda prelaska praga osjetljivosti

Metoda prelaska praga osjetljivosti (engl. *thresholding method*) je jednostavna, ali vrlo učinkovita metoda razlučivanja fotografije na prvi plan i pozadinu. Ova tehnika analize je vrsta segmentacije slike koja izolira predmete pretvarajući sivo-skalirane fotografije u binarne fotografije [4]. Naravno, metoda je najučinkovitija kada fotografija ima visoku razinu kontrasta, što je slučaj kod upotrebljenog skupa podataka u ovom radu. Uobičajno pod algoritme ove metode ubrajaju se histogram i višerazinski prelazak praga osjetljivosti (engl. *multi-level thresholding*). Metoda prelaska praga osjetljivosti je vrlo važan problem prilikom automatskih analiza slike jer puno aplikacija za procesiranje slika i računalni vid koriste binarne slike (npr. crne i bijele) kao osnovni korak za daljnju obradu [5]. Najvažniji korak je odrediti vrijednost intenziteta "praga", pomoću kojeg se slike segmentiraju postavljanjem onih piksela čiji je izvorni intenzitet iznad praga u "bijeli piksel", a ostale se postavlja u "crni piksel". Na ovaj način, dobivamo bijelo crni kontrast, gdje bijeli objekt predstavlja prvi plan, dok crni predstavlja pozadinu. Navedeno je prikazano slikom 2.2b, gdje u prvi plan ulazi keramička pločica, dok u pozadinu ulazi ostatak fotografije. Prilikom korištenja ove metode, u promatranom skupu podataka, izuzetno je bitno paziti na uvjete osvjetljenja kada se fotografiraju keramičke pločice. Ukoliko osvjetljenje nije dovoljno dobro postavljeno, mogu se dogoditi smanjene razlike u kontrastima prvog plana i pozadine, gdje algoritam neće biti u mogućnosti razlučiti razliku. Također, u radu će kasnije biti objašnjeno odabiranje vrijednosti intenziteta "praga" za promatrani skup podataka.

(a) **(b)**

SL. 2.2: *Keramička pločica: prije primjene metode(a); nakon primjene metode prelaska praga osjetljivosti(b).*

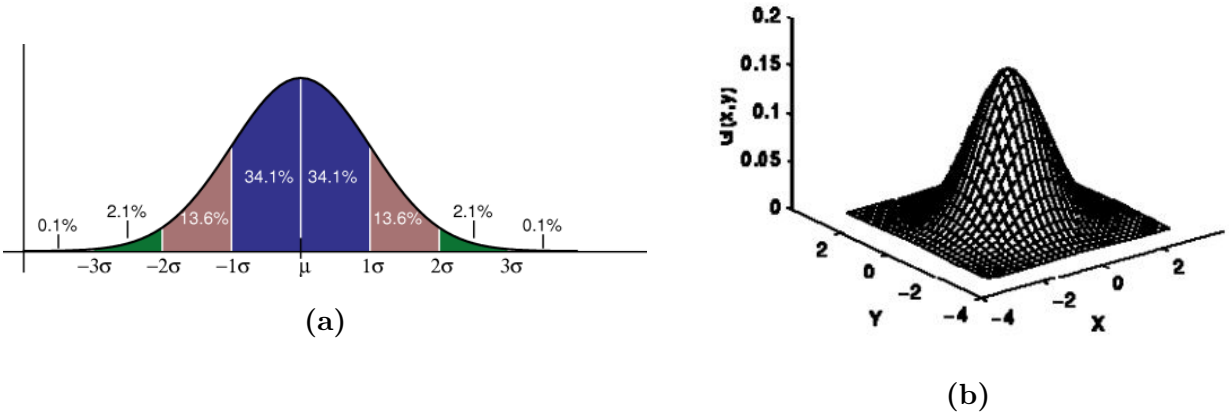
2.3. Dvodimenzionalna Gaussova funkcija

Za obradu slike često se može upotrijebiti Gaussov filter koji se temelji na Gaussovoj funkciji. Gaussov filter koristi se za zamućivanje slika te uklanjanje šumova i detalja. Standardna devijacija Gaussove funkcije ima važnu ulogu u ponašanju Gaussovog filtera [6]. Slika 2.3a prikazuje kako se 68% vrijednosti nalazi unutar +/- vrijednosti jedne standardne devijacije, dok 95% leži unutar vrijednosti dvije standardne devijacije (plavo i smeđe obojano), dok 99,7% leži unutar vrijednosti tri standardne devijacije (plavo, smeđe i zeleno). Ovo je vrlo korisno kada se definira jezgra (engl. *kernel*) za Gaussov filter [7]. U ovom radu neće se koristiti Gaussov filter kao takav nego Gaussova funkcija koja je osnova za taj filter. Kada se radi s fotografijama, potrebno je koristiti dvodimenzinalnu Gaussovu funkciju, što predstavlja umnožak jednodimenzionalne Gaussove funkcije sa samom sobom. Slika 2.3b predstavlja grafički prikaz dvodimenzionalne Gaussove distribucije s varijancom od (0,0) i standardnom devijacijom (1).

Gaussova funkcija koristi se u područjima poput definiranja raspodjele vjerojatnosti za šum ili podatke, kao zaglađujući operator i u matematici [8]. Dvodimenzionalna Gaussova funkcija prikazana je sljedećom formulom:

$$f(x,y) = A\exp\left\{-\frac{1}{2}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right)\right\} \tag{2-1}$$

gdje je u navedenoj formuli:
A ... amplituda,
μ_x ... centar funkcije u x-osi,



(a) **(b)**

SL. 2.3: *Gaussova distribucija: 1D (a); 2D (b)[8].*

μ_y ... centar funkcije u y-osi,
σ_x ... širenje krivulje duž x-osi,
σ_y ... širenje krivulje duž y-osi.

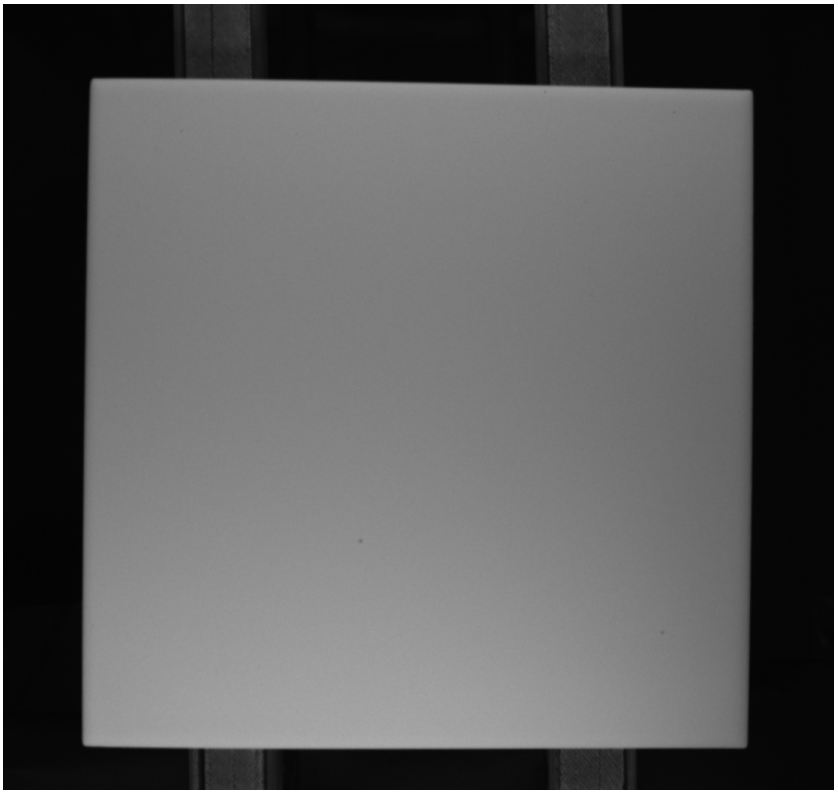
Parametar amplituda, označava koliko je dobiveni graf visok, odnosno koliko će, u promatranom slučaju, pogreška biti duboka. Ukoliko želimo nacrtati vrh Gaussove 2D funkcije u ishodištu koordinatnog sustava, tada je potrebno parametre μ_x i μ_y postaviti na vrijednost "0". Drugim riječima, parametri μ_x i μ_y diktiraju položaj najvišeg vrha Gaussove funkcije. Parametrima σ_x i σ_y upravljamo širinom zvona Gaussove funkcije, odnosno ako su parametri σ_x i σ_y veći, pogreška će biti šira. Također, veće razlike u njihovoj vrijednosti dovest će do elipsoidnog oblika. Primjena navedene Gaussove funkcije na promatrani skup podataka detaljnije je objašnjena u sljedećem poglavlju.

3. KORIŠTENE METODE

Kako bi se uspješno odradio predstavljeni zadatak, potrebno ga je raščlaniti na manje probleme koji će sustavno biti opisani ovim poglavljem. Za realizaciju programskog koda, korišten je programski jezik MATLAB.

3.1. Skup podataka

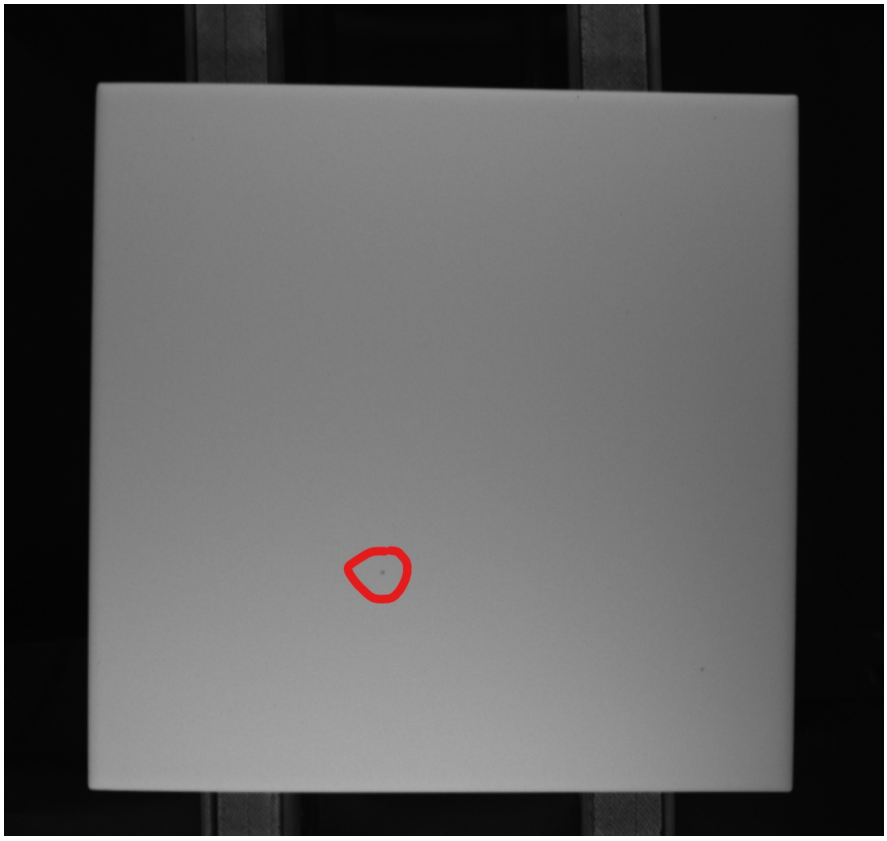
Podaci podrazumijevaju uslikane keramičke pločice na proizvodnoj liniji, spremljene s ".bmp" ekstenzijom. Svaka fotografija je bila u crno bijeloj boji, gdje se u sredini fotografije nalazi pločica svjetlije boje, dok su u pozadini vidljive dvije linije koje predstavljaju pomičnu traku u proizvodnji. Primjer fotografije prikazan je slikom 3.4. Fotografije su uslikane u stvarnom proizvodnom procesu. Na svim slikama intenzitet svjetline se kreće u rasponu od 0 do 255, gdje je 255 najsvjetlija točka, a 0 najtamnija. Korišten je skup podataka od 29 fotografija za proučavanje i analizu točkastih pogrešaka. Pod točkastom pogreškom smatra se crna ili siva mrlja kružnog ili elipsoidnog oblika na bijeloj pločici koja predstavlja udubljenje u slici. Udubljenje je na površini šire te se sužava prema dnu pločice. Kada se provuku simetrale kroz središte pogreške ona ima oblik okrenute Gaussove krivulje. Sve fotografije iz navedenog skupa izgledaju približno kao predložena fotografija, no mogu se razlikovati u položaju pločice gdje pločica može biti zakrenuta do +5 ili -5 stupnjeva.



SL. 3.4: *Primjer fotografije iz skupa podataka.*

3.2. Analiza točkaste pogreške

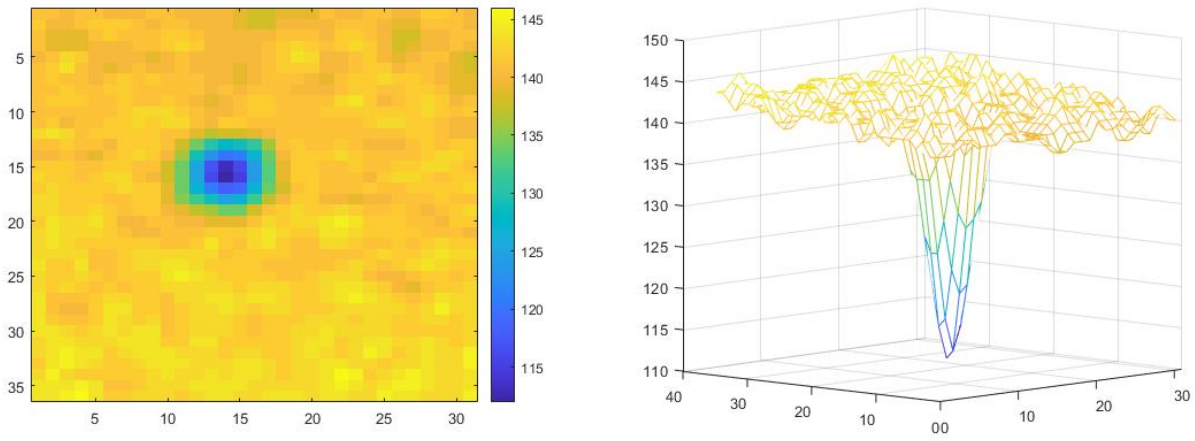
Najprije je potrebno napraviti analizu stvarnih točkastih pogrešaka koje nastaju u proizvodnji kako bi se one mogle što vjerodostojnije umjetno izgenerirati. Stoga su iz prethodno spomenutog skupa podataka od 29 fotografija pronađene sve točkaste pogreške i izvučeni su njihovi rasponi intenziteta. Postupak će biti objašnjen na primjeru slike 3.4. Najprije je pronađena točkasta pogreška nastala u proizvodnji koja je uokvirena crvenom bojom na slici 3.5. Zatim je pomoću MATLAB funkcije *imagesc(učitanaFotografija)* uvećano iscrtana prethodno uokvirena pogreška koja je prikazana na slici 3.6a. *Imagesc(učitanaFotografija)* se koristi kako bi se iscrtali pikseli istog intenziteta istom bojom, a kao parametar predaje se učitana fotografija s rasponom koji se želi iscrtati. Tako slika 3.6a prikazuje iscrtan intenzitet u rasponu od 695 do 730 retka i 460 do 490 stupca od predane učitane fotografije.



SL. 3.5: *Keramička pločica s označenom točkastom pogreškom.*

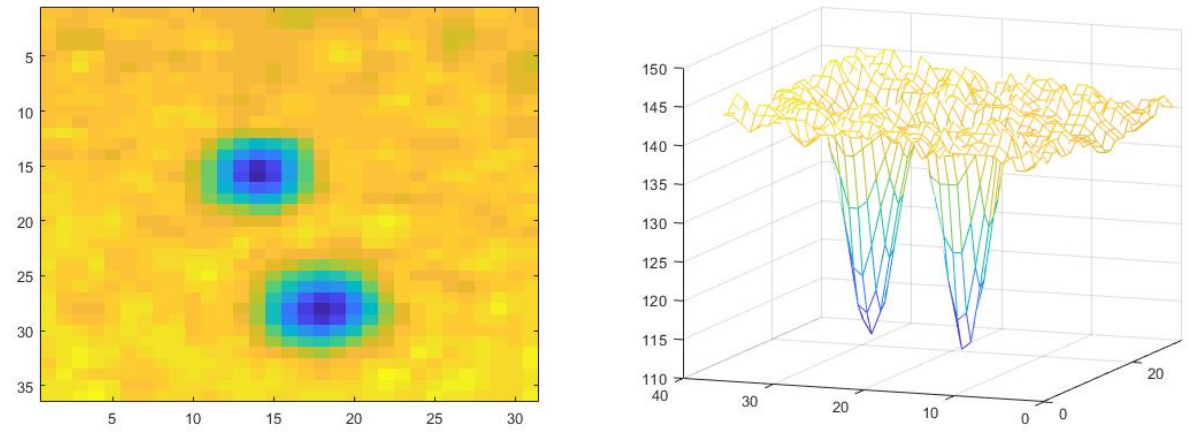
Iz slike 3.6a možemo očitati intenzitet pogreške koji iznosi od 112 do 136, gdje vrh pogreške ima intenzitet 112, dok okolina pogreške ima intenzitet od 136. Na slici 3.6b vidljiv je 3D prikaz pogreške, čiji je vrh okrenut "prema dolje", što znači kako točkasta pogreška predstavlja udubljenje u sliku. Trodimenzionalni prikaz dobiven je korištenjem MATLAB funkcije *mesh(rasponUčitaneSlike)* koja se koristi za crtanje 3D površina.

Nakon dublje analize, uočeno je kako se prikazana točkasta pogreška ponaša prema zakonu Gaussove krivulje. Naravno, kako bi se mogle opisati postojeće pogreške, za crtanje grešaka korištena je dvodimenzionalna Gaussova funkcija 2-1.



(a) **(b)**

SL. 3.6: *Točkasta pogreška: uvećana slika pogreške imagesc(A(695:730,460:490)) (a); 3D (b).*



(a) **(b)**

SL. 3.7: *Stvarna i umjetno generirana pogreška: 2D (a); 3D (b).*

Slikom 3.7 prikazan je rezultat kombiniranja vrijednosti parametara Gaussove 2D funkcije, s ciljem dobivanja pogreške koja je što sličnija pravoj pogrešci. Vidljivo je kako točkasta pogreška dobivena pomoću Gaussove funkcije dobro oponaša stvarnu pogrešku. Na slici 3.7b, lijeva krivulja predstavlja stvarnu pogrešku, dok desna predstavlja aproksimaciju stvarne pogreške Gaussovom funkcijom. Korištene su slijedeće vrijednosti parametara Gaussove funkcije: A = 31, $\mu_x = 7$, $\mu_y = 7$, $\sigma_x = 2.4$, $\sigma_y = 2.2$, i = j = 15.

Proučavajući pogreške, uočeno je kako se one mogu pojaviti i pod određenim kutom. Kako bi se omogućilo crtanje elipsoidne kružnice pod kutom, potrebno je postojeću Gaussovu funkciju 2-1 preobraziti i uvesti parametar kuta. Kut ćemo označiti s θ. Kako je riječ o koordinatnom sustavu, jednostavno ćemo točku raspisati preko zbroja odnosno razlike sinusa i kosinusa:

$$f(x,y) = A\exp\left\{-\frac{1}{2}\left(\frac{(x_m)^2}{\sigma_x^2} + \frac{(y_m)^2}{\sigma_y^2}\right)\right\} \tag{3-2}$$

gdje je:

$$x_m = (x - \mu_x) * cos(\theta) - (y - \mu_y) * sin(\theta) \tag{3-3}$$

$$y_m = (x - \mu_x) * sin(\theta) + (y - \mu_y) * cos(\theta). \tag{3-4}$$

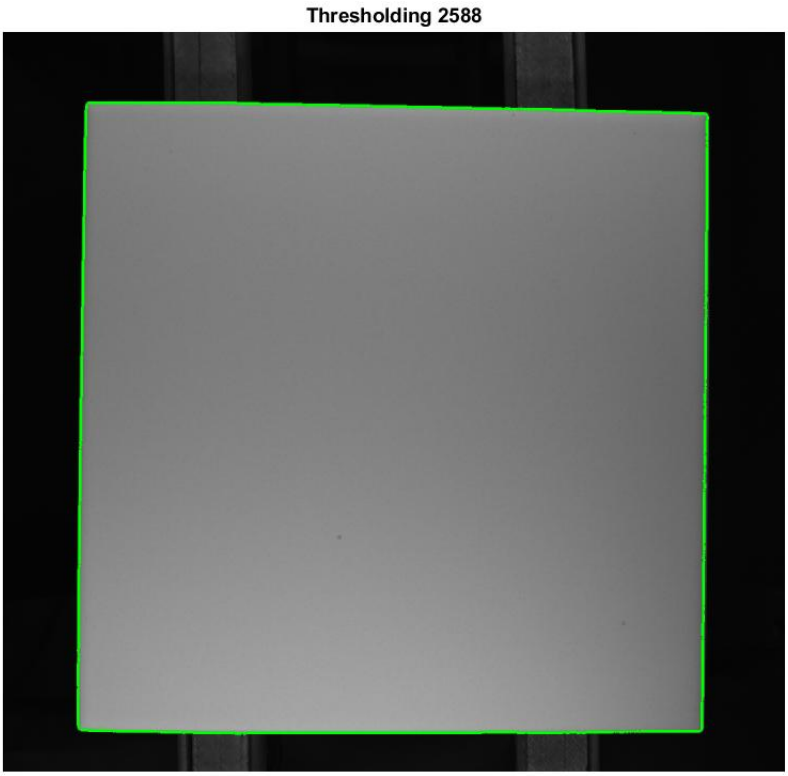
Parametri su:
A ... amplituda,
μ_x ... centar funkcije u x-osi,
μ_y ... centar funkcije u y-osi,
σ_x ... širenje krivulje duž x-osi,
σ_y ... širenje krivulje duž y-osi,
θ ... nagib kojeg je potrebno zadati u radijanima.

3.3. Analiza prepoznavanja pločice na slici

Iz priložene slike 3.4, vidljivo je kako se na fotografiji ne nalazi samo pločica te je potrebno prepoznati obrub same pločice kako se pogreške ne bi dodavale na okolno područje, već samo na unutrašnjost.

Prepoznavanje keramičke pločice omogućeno je pomoću metode prelaska praga osjetljivosti (engl. *thresholding method*). Metoda prelaska praga osjetljivosti podrazumijeva zamjenjivanje svakog piksela na fotografiji s određenom bojom, u ovom slučaju zelenom, ukoliko je intenzitet piksela veći ili manji od zadanog fiksnog intenziteta. Proučavajući postojeći skup podataka, određena je vrijednost intenziteta referentnog piksela na 80. Slikom 3.8 prikazano je pronalaženje obruba pločice na fotografiji pomoću spomenute metode. Navedenom metodom dobivena je matrica, nazvana *thisBoundary* veličine 3486x2, s parovima koordinata x i y, koji opisuju pločicu na fotografiji. Kako je pločica svijetlijeg intenziteta, koristio se znak "veće od" referentne vrijednosti. Postupak pronalaženja obruba:

1. uspoređivanje originalne slike s *threshold* vrijednosti 80,



SL. 3.8: *Rezultat prepoznavanja keramičke pločice metodom prelaska praga osjetljivosti.*

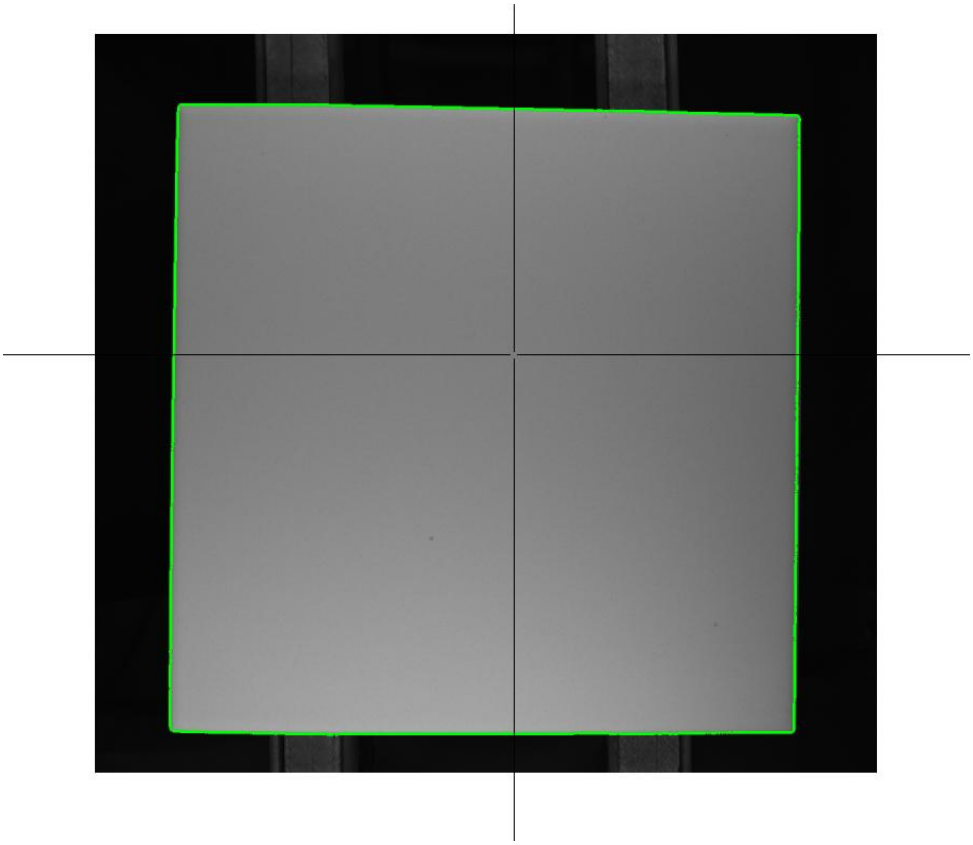
2. popunjavanje uspoređene slike s funkcijom *imfill(slika, opcija)* s opcijom "holes",
3. pronalaženje vanjskog obruba objekata u binarnoj slici s funkcijom *bwboundaries(slika)*,
4. iscrtavanje obruba pomoću *plot(x,y)* funkcije.

Korištena funkcija *imfill(slika, opcija)* s opcijom "holes" popunjava šupljine u ulaznoj slici. Šupljinom se podrazumijeva skup pozadinskih piksela do kojih se ne može doći popunjavanjem pozadine s ruba slike [9], što je u ovom slučaju promatrana pločica. Jednostavnije rečeno, ovom funkcijom se pločica odvoji od pozadine na način da se pozadina postavi na vrijednost "0". Funkcija *bwboundaries(slika)* vraća sve koordinate reda i stupaca graničnih piksela svih objekata na slici, gdje su oni pikseli s vrijednosti "0" pozadinski, a svi ostali pripadaju pronađenom objektu. Funkcijom *imfill(slika, opcija)* odvojena je pločica od pozadine, dok su funkcijom *bwboundaries(slika)* dobivene koordinate koje opisuju rub prethodno odvojene pločice. Na temelju dobivene matrice s parovima x i y koordinata obruba, moći će se provjeriti nalazi li se generirana ili odabrana koordinata unutar granica pločice, što nam je potrebno za slijedeći korak u realizaciji zadataka.

3.4. Analiza ručnog dodavanja pogreške

Prvi tip programa odnosi se na ručno dodavanje pogreške na područje pločice na fotografiji. Za odabiranje koordinata gdje će biti vrh pogreške, korištena je gotova MATLAB funkcija

ginput(n). Ginput(n) omogućava identificiranje koordinata za n točaka unutar kartezijskih, polarnih ili zemljopisnih osi [10]. Kako bi se odabrala točka, potrebno je pomaknuti pokazivač na željeno mjesto i pritisnuti tipku miša ili tipku na tipkovnici. Prestanak rada funkcije omogućava se pritiskom na tipku enter na tipkovnici. Nakon što se klikne na željeno mjesto, MATLAB vraća koordinate odabranih točaka. Dobivene koordinate su u decimalnom obliku. Na slici 3.9 može se vidjeti rezultat primjene *ginput* funkcije na fotografiju pločice.

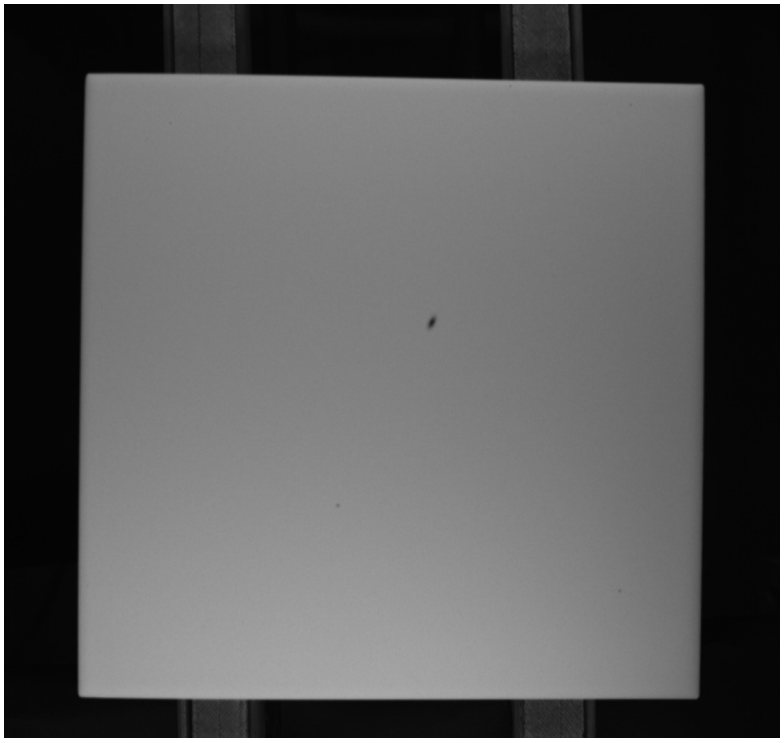


SL. 3.9: *Uključena ginput funkcija za odabir koordinata.*

Program za ručno dodavanje pogreške radi na sljedeće opisanom principu. Najprije se otvara fotografija i obavlja se prethodno opisana metoda za prelazak praga osjetljivosti za pronalazak obruba pločice. Za otvaranje fotografije koristi se MATLAB funkcija *imread(nazivSlike)*. Nakon pronalaska koordinata obruba, otvara se *while* petlja unutar koje se prvo poziva *ginput(1)* funkcija. Predani parametar u *ginput* funkciji je 1 jer se odmah želi provjeriti nalazi li se odabrani par koordinata unutar pločice prije odabiranja novog para koordinata za crtanje sljedeće pogreške. *While* petlja se završava kada ginput funkcija ne vrati koordinate, odnosno kada se prekine rad *ginput* funkcije. U suprotnom, vraćene koordinate se prvo zaokružuju s MATLAB funkcijom *round()* kako bi se te koordinate mogle proslijediti u račun Gaussove funkcije kao parametri μ_x i μ_y. Nakon navedenog, potrebno je provjeriti nalazi li se odabrani par koordinata unutar pločice, tj. unutar zelenog okvira. Postupak provjere koordinata je matematički zahtjevniji jer se u proizvodnji može dogoditi da pločica ne stoji u potpunosti okomito na trake po kojima se prevozi. Zbog zakretanja pločice do 5 stupnjeva, potrebno je za oda-

branu koordinatu buduće pogreške pronaći gdje dva pravca (jedan usporedan s x, drugi s y-osi) prolaze kroz odabranu koordinatu i sijeku obrub pločice. Najprije su pronađene dvije koordinate: x_1 i x_2 za odabranu vrijednost Y koordinate (gledajući na slici 3.9 gdje paralelni pravac s x-osi siječe zeleni rub pločice). Također, pronađene su dvije koordinate: y_1 i y_2 za zadanu X koordinatu (gledajući na slici 3.9 gdje pravac paralelan s y-osi siječe zeleni rub pločice). Zatim je provjereno nalaze li se odabrane X i Y koordinate za mjesto pogreške između prethodno pronađenih x_1, x_2, y_1 i y_2 koordinata. Ukoliko se nalaze između pronađenih koordinata, poziva se funkcija nazvana *greska(M,N,*μ_x*,*μ_y*,Intensity)*, koja kao rezultat vraća fotografiju sa slučajno izgeneriranom Gaussovom pogreškom. Ukoliko se koordinate nisu nalazile između pronađenih koordinata, tražio bi se ponovni unos slijedećeg para koordinata s *ginput* funkcijom. Fotografija sa slučajno izgeneriranom Gaussovom funkcijom je na kraju oduzeta od prvotno učitane fotografije kako bi se dobilo željeno točkasto udubljenje. Potrebno je napomenuti kako je izgenerirana fotografija s Gaussovom funkcijom iste veličine kao i učitana fotografija. Primjer slučajno izgenerirane Gaussove funkcije na praznoj slici dan je slikom 3.15. Fotografija nastala oduzimanjem originalne slike i slike s Gaussovom funkcijom se prikazuje pomoću MATLAB funkcije *imshow(nazivSlike)*. Kako je na početku spomenuto, još uvijek se nalazimo u *while* petlji te se na toj novoj slici s dodanom greškom traži slijedeći unos koordinate. Na ovaj način, moguće je unijeti proizvoljan broj pogrešaka na učitanu fotografiju. Nakon izlaska iz petlje, definiran je put do direktorija u koji se automatski sprema fotografija s generiranim pogreškama s jedinstvenim nazivom. Spremanje je omogućeno korištenjem MATLAB funkcije *imwrite(nazivSlike, nazivPutaDoFoldera)*. Slikom 3.11 prikazan je rezultat generiranja jedne pogreške na fotografiji 3.4.

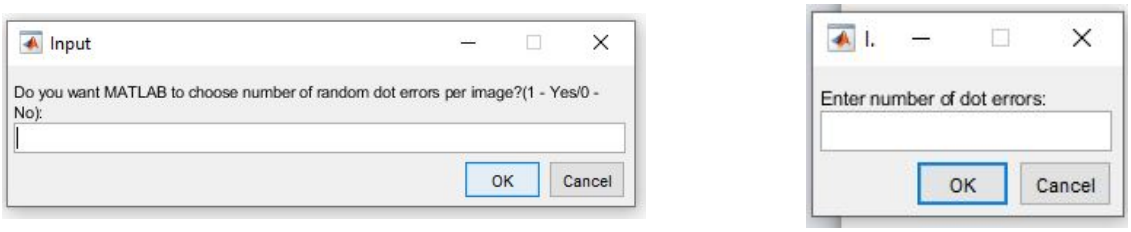
SL. 3.10: *Slučajno izgenerirana Gaussova funkcija na praznoj fotografiji.*



SL. 3.11: *Slučajno izgenerirana Gaussova funkcija na stvarnoj fotografiji.*

3.5. Analiza automatskog dodavanja pogreške

Drugi tip kreiranog programa odnosi se na automatsko dodavanje točkaste pogreške. Ovo podrazumijeva slučajno generiranje određenog broja parova koordinata za crtanje pogrešaka. Program je zamišljen na način da se prilikom pokretanja skripte prvo pojavljuje prozor, slika 3.12a, s upitom: "Do you want MATLAB to choose number of random dot errors per image?(1 - Yes/ 0 - No)". Ukoliko odgovorimo s "1", MATLAB će sam generirati slučajan broj pogrešaka za svaku sliku u predanom direktoriju. Ukoliko upišemo: "0", tada se javlja novi prozor za svaku sliku u direktoriju, slika 3.12b: "Enter number of dot errors:", gdje se traži od korisnika da unese željeni broj pogrešaka za sljedeću pročitanu sliku.

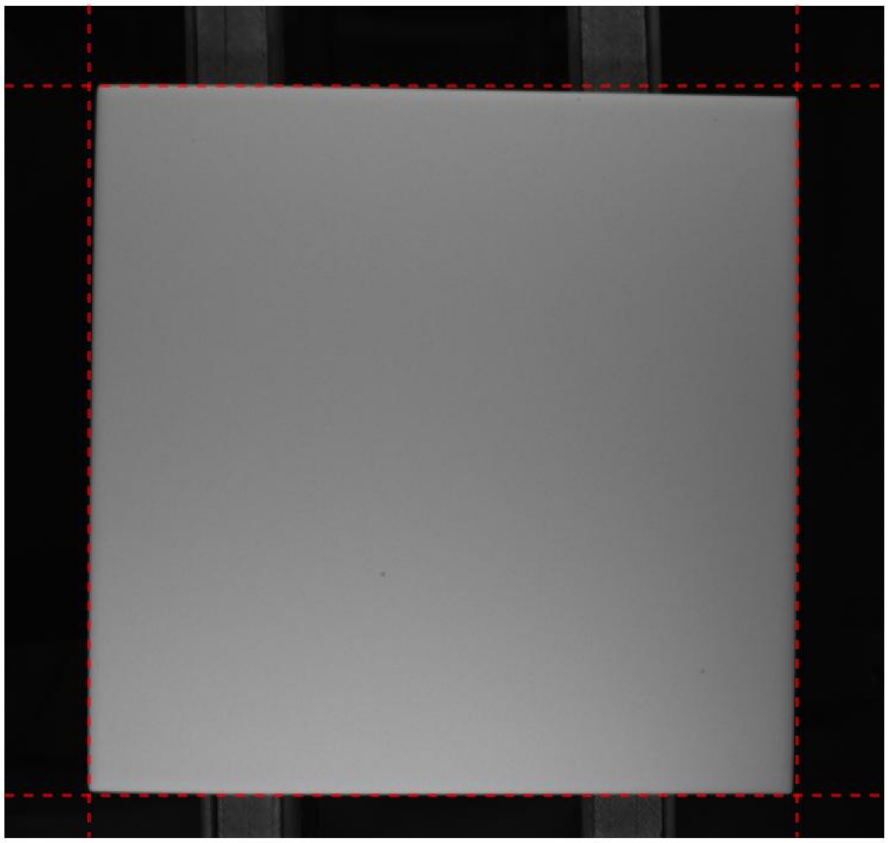


(a) **(b)**

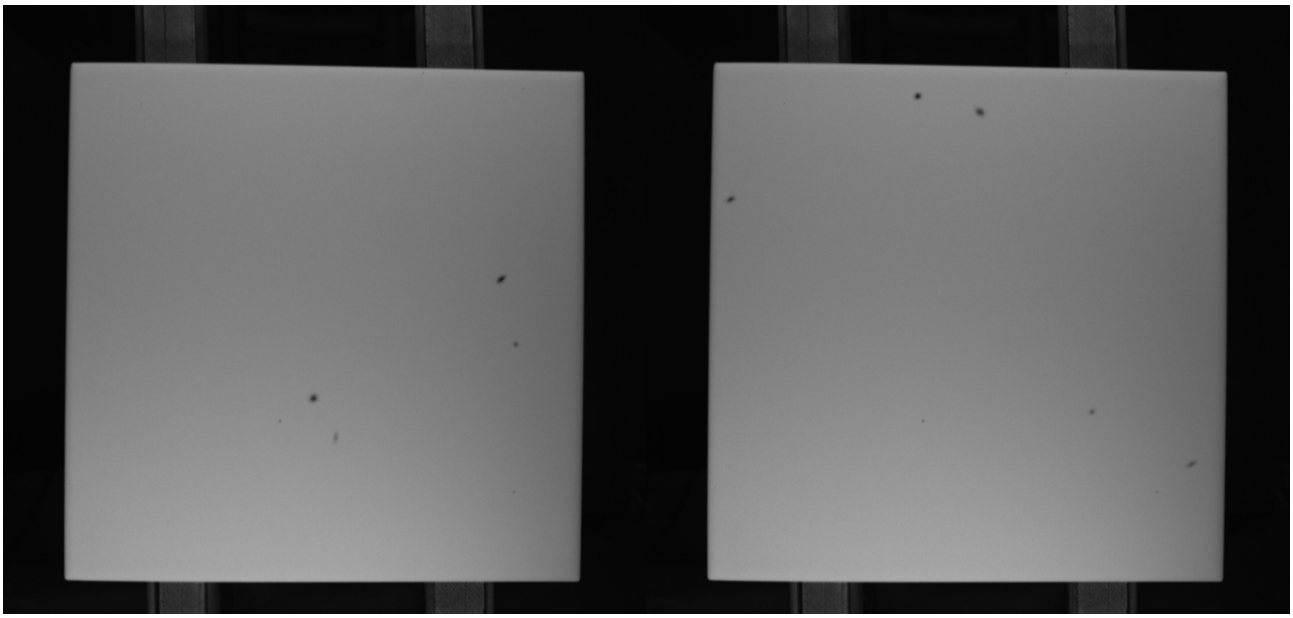
SL. 3.12: *Izbornici u MATLAB programu za: odabir načina generiranja pogrešaka (a); odabir broja pogrešaka (b).*

Najprije se u programu specificira direktorij iz kojeg će se učitavati fotografije. Prije učitavanja fotografija iz navedenog direktorija pojavljuje se prvi prozor, slika 3.12a. Nakon unosa, program ulazi u *for* petlju, koja prolazi kroz prethodno naveden direktorij i redom čita jednu po jednu fotografiju. Nakon što učita prvu fotografiju, provodi se metoda prelaska praga osjetljivosti kako bi se dobile granice pločice. Sljedeće se provjerava je li odgovor u prvom izborniku bio "1" ili "0". Ukoliko je odgovor bio "1", postavlja se privremena varijabla *answer*, koja označava broj željenih pogrešaka, na slučajno izgeneriranu vrijednost u rasponu [0, 5]. Suprotno, ukoliko je odgovor bio "0", otvara se novi prozor, slika 3.12b, te se postavlja privremena varijabla *answer* na unesenu vrijednost. Sljedeći korak je pronaći minimalnu i maksimalnu vrijednost koordinata X i Y iz *thisBoundary* matrice, kako bi se odredio raspon za slučajno generiranje koordinata pogreške. Podsjetimo se kako spomenuta matrica sadrži parove koordinata ruba pločice. Prilikom testiranja programa, javio se problem pri slučajnom generiraju koordinate u rasponu cijele slike, te se događalo da se izgenerira uzastopno 100 koordinata koje nisu unutar pločice što je dovodilo do znatnog usporenja programa. Pronalaženjem minimalne i maksimalne vrijednosti koordinata X i Y iz koordinata obruba, stvoren je četverokut unutar kojeg se svakako nalazi pločica i malo pozadine, što je prikazano slikom 3.13. Postotak pozadine unutar četverokuta ovisi o stupnju zakrivljenosti pločice na proizvodnoj traci. Na ovaj način smanjena je vjerojatnost generiranja neispravnih koordinata i očuvana je brzina programa. Nakon pronalaska granica, ulazimo u *while* petlju koja ide do veličine varijable

answer. Odnosno, ponavlja se onoliko puta koliko je uneseno ili slučajno izgenerirano željenih pogrešaka. U ovom dijelu programa odvija se generiranje i dodavanje željenog broja pogrešaka. Iduće slijedi generiranje koordinate pogreške pomoću *rand()* funkcije. Zatim slijedi provjera nalazi li se izgenerirana X i Y koordinata unutar granica pločice, te ako se nalazi, nastavlja se postupak dodavanja točkaste pogreške koji je detaljnije opisan u prethodnom poglavlju. U slučaju da se izgenerira koordinata koja se ne nalazi na području pločice, tada se privremena varijabla smanji za jedan u *while* petlji i postupak se ponavlja dok god nisu ucrtane sve željene pogreške. Slikom 3.14a prikazan je rezultat nakon odabiranja "1" na prvom izborniku, dok je slikom 3.14b prikazan rezultat nakon odabiranja "0" na prvom izborniku i broja "5" na drugom izborniku.



SL. 3.13: *Granice za slučajno generiranje koordinata.*



(a) **(b)**

SL. 3.14: *Rezultat korištenja MATLAB skripte za: automatsko generiranje 4 pogreške (a); automatsko generiranje odabranih 5 pogrešaka (b).*

3.6. Analiza funkcije za kreiranje pogreške

Kako bi programski kod bio pregledniji, postupak kreiranja fotografije s pogreškom izvučen je u zasebnu funkciju nazvanu *greska*. Funkcija *greska(M,N,*μ_x*,*μ_y*,Intensity)* prima sljedećih pet parametara:
M ... broj redova učitane fotografije,
N ... broj stupaca učitane fotografije,
μ_x ... x koordinata vrha željene greške,
μ_y ... y koordinata vrha željene greške,
Intensity ... intenzitet piksela sa sredine fotografije.
Kao što je u prethodnim poglavljima spomenuto, bilo je potrebo prilikom poziva funkcije predati i veličinu originalne fotografije kako bi navedena funkcija mogla vratiti matricu iste veličine s dodanom slučajno generiranom Gaussovom pogreškom. Ukoliko novostvorena fotografija nije iste veličine, tada se originalna fotografija i novo kreirana ne bi mogle oduzeti.
Na početku ove skripte, definiraju se rasponi za tri parametra: σ_x, σ_y i A. Rasponi za σ_x i σ_y su određeni na temelju proučavanja stvarnih pogrešaka nastalih prilikom proizvodnje.

$$\sigma_x = [2, 6] \tag{3-5}$$

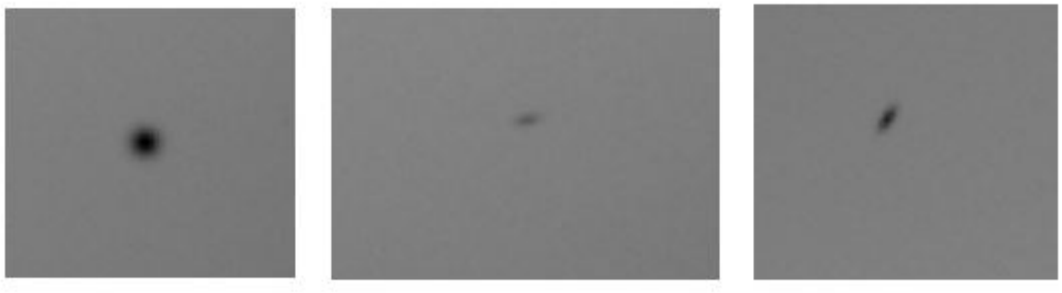
$$\sigma_y = [2, 6] \tag{3-6}$$

$$A = [0.2, 1] \tag{3-7}$$

Varijabla A predstavlja amplitudu, odnosno koliko će Gaussova krivulja biti visoka. Amplituda je postavljena kao postotak u odnosu na predani parametar *Intensity*. Na temelju definicije raspona parametara, generiraju se slučajne vrijednosti pomoću MATLAB funkcije *rand()* za navedena tri parametra. Nakon generiranja slučajnog postotka za amplitudu A, ta vrijednost se još množi s vrijednosti *Intensity* kako bi se dobila vrijednost kao postotak intenziteta srednjeg piksela. Na spomenuti način se ograničilo da pogreška svakako mora biti dublja od intenziteta srednjeg piksela. Također, stvorena je matrica s nulama pomoću funkcije *zeros(N,M)* koja će se koristiti kasnije kao prazna fotografija za dodavanje pogreške. Nakon stvaranja matrice, generira se zadnji parametar: θ, također pomoću *rand()* funkcije u rasponu: [0°, 90°]. Prisjetimo se kako θ označava kut za koji će se slučajno generirana pogreška zakrenuti. Sljedeće se prolazi pomoću dvije *for* petlje kroz prethodno stvorenu matricu s nulama i poziva se nova funkcija *rotationalGauss(c,r, theta,* σ_x*,* σ_y*, koordinateGreske)*. Varijable za prolaz kroz *for* petlje su c i r, gdje se c kreće u rasponu: [0, N], dok se r kreće u rasponu: [0, M]. Varijabla *koordinateGreske* predstavlja 1x2 matricu s vrijednostima μ_x i μ_y. Funkcija *rotationalGauss* je upravo funkcija napisana jednadžbom: 3-2. Potrebno je napomenuti kako se u funkciji *rotationalGauss* obavlja pretvorba kuta θ iz stupnjeva u radijane. Na samom kraju, potrebno je dobivenu fotografiju pretvoriti iz 'double' formata u 'uint8' format pomoću MATLAB funkcije *cast(nazivFunkcije, tip)* kako bi se mogla oduzeti od originalne slike čiji je format 'uint8'.

Tab. 3.1: *Primjeri dobivenih slučajnih pogrešaka sa svim parametrima.*

	N	M	Intensity	A	σ_x	σ_y	μ_x	μ_y	θ [°]
1.	1038	1098	126	125	5.5161	5.2616	274	690	0.0470
2.	1038	1098	126	37	2.0679	4.3204	198	675	77.6440
3.	1038	1098	126	87	2.1280	4.5195	429	328	32.6170



SL. 3.15: *Primjeri slučajno izgeneriranih pogrešaka prema parametrima iz tablici 3.1.*

Tablicom 3.1 prikazana su tri primjera generiranja pogreške. U slučaju da je parametar σ_x jednak parametru σ_y, tada se dobiva kružna pogreška. Prvim primjerom u tablici je $\sigma_x \approx \sigma_y$, te je dobivena pogreška nalik kružnoj točkastoj pogrešci. Prvom primjeru iz tablice 3.1 odgovara lijeva fotografija pogreške na slici 3.15. Ukoliko se parametri σ_x i σ_y razlikuju,

dobivamo elipsoidnu pogrešku. Također, ukoliko proučimo primjere, moguće je primijetiti kako tamnijoj pogrešci odgovara i veća amplituda, što znači da je pogreška dublja. Sva tri slučaja generiranja slučajnih grešaka načinjena su na fotografiji 3.4, zbog čega su parametri N, M i Intensity jednaki. Potrebno je objasniti i način rotiranja pogreške. Rotiranje se odvija u smjeru suprotnom od kazaljke na satu, gdje je y-os ta koja se zakreće.

4. ZAKLJUČAK

U ovom završnom radu, prikazana su dva moguća rješenja za dodavanje točkastih pogrešaka u fotografiju. Na temelju provedenog istraživanja odlučeno je koristiti Gaussovu 2D funkciju jer se korištenjem te funkcije dobiju najsličnije točkaste pogreške stvarnim pogreškama koje nastaju u proizvodnji. Pod točkastom pogreškom podrazumijevala se pogreška koja iz tlocrta izgleda kao točka dok zapravo predstavlja udubljenje u sliku. U radu je opisana Gaussova 2D formula, njezina primjena te su objašnjeni parametri funkcije i njihovi rasponi granica koji su određeni istraživanjem. Gaussovom funkcijom dobivena su dva tipa pogrešaka: kružne i elipsoidne pogreške, te je vrsta pogreške prepuštena slučajnom generiraju vrijednosti parametara u dopuštenim granicama. Također, objašnjen je postupak pronalaženja ruba pločice unutar dane fotografije pomoću metode prelaska praga osjetljivosti kako bi se ograničila mogućnost dodavanja pogrešaka samo na područje pločice. Detaljno je opisan postupak ručnog dodavanja pogrešaka pomoću MATLAB funkcije *ginput*, ali i postupak provjere nalazi li se traženi par x i y koordinata unutar pločice. Osim toga, opisan je i proces automatskog dodavanja pogrešaka, koji podrazumijeva samostalno generiranje koordinata pomoću *rand* funkcije. Stvaranjem dva različita programa s istom funkcionalnosti, omogućeno je da budući korisnici mogu odabrati na koji način žele dodati pogrešku na pločicu. Korisnici mogu samostalno klikom miša odabrati pogreške na određenim dijelovima keramičke pločice, ali ujedno mogu predati direktorij sa željenim brojem fotografija gdje će sam program izgenerirati automatski raznovrsne pogreške. Također, programima je omogućeno i automatsko spremanje novokreiranih fotografija s greškom u zasebne datoteke s jedinstvenim nazivom kako bi ih se moglo razlikovati. Smatram kako će u budućnosti najviše biti korišten program koji automatski generira pogreške jer je svrha ovog završnog rada bila načiniti program koji će generirati velike količine umjetnih slika s pogreškom koje su potrebne kao osnovni skup za učenje neuronskih mreža.

LITERATURA

[1] S. Haykin, *Neural Networks and Learning Machines*. Prentice Hall, 2009. [Online]. Available: https://cours.etsmtl.ca/sys843/REFS/Books/ebook_Haykin09.pdf

[2] T. MathWorks, "Deep learning," 1994-2021, (pristupljeno 6.7.2021.). [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html#howitworks

[3] R. Raicea, "Want to know how deep learning works? here's a quick guide for everyone," 2017, (pristupljeno 6.7.2021.). [Online]. Available: https://www.freecodecamp.org/news/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076/

[4] T. MathWorks, "Image thresholding," 1994-2021, (pristupljeno 5.7.2021.). [Online]. Available: https://www.mathworks.com/discovery/image-thresholding.html

[5] D. M. Carabias, *ANALYSIS OF IMAGE THRESHOLDING METHODS FOR THEIR APPLICATION TO AUGMENTED REALITY ENVIRONMENTS*. MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA. FACULTAD DE INFORMÁTICA UNIVERSIDAD COMPLUTENSE DE MADRID, 2012. [Online]. Available: https://eprints.ucm.es/id/eprint/16932/1/Tesis_Master_Daniel_Martin_Carabias.pdf

[6] U. Qidwai and C. Chen, *Digital Image Processing*. CRC Press, 2010.

[7] R. Sedgewick, K. Wayne, and R. Dondero, "Appendix c: Gaussian distribution," 2015, (pristupljeno 6.6.2021.). [Online]. Available: https://introcs.cs.princeton.edu/python/appendix_gaussian/

[8] "Gaussian filtering," 2010, (pristupljeno 6.6.2021.). [Online]. Available: https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf

[9] T. MathWorks, "Imfill," 1994-2021, (pristupljeno 10.6.2021.). [Online]. Available: https://www.mathworks.com/help/images/ref/imfill.html

[10] MathWorks, "Ginput," 1994-2021, (pristupljeno 10.6.2021.). [Online]. Available: https://www.mathworks.com/help/matlab/ref/ginput.html?searchHighlight=ginput&s_tid=srchtitle

SAŽETAK

Naslov: Generiranje umjetnih slika keramičkih pločica

Ovim završnim radom objašnjen je postupak dobivanja jedne vrste pogrešaka koje nastaju u stvarnoj proizvodnji keramičkih pločica. Riječ je o točkastim pogreškama koje predstavljaju udubljenje u keramičku pločicu. Analizom je primjećena sličnost udubljenja s naopako okrenutom Gaussovom krivuljom, gdje se korištenjem dvodimenzionalne Gaussove funkcije dobila gotovo identična pogreška stvarnoj. U radu je napravljena analiza stvarnih točkastih pogrešaka i opisan je postupak dobivanja granica za parametre Gaussove 2D funkcije kako bi se uspjele izgenerirati realne pogreške. Osim toga, objašnjen je proces dodavanja umjetnih pogrešaka na fotografije pločica bez greške iz stvarne proizvodnje. Na spomenutim fotografijama nisu se nalazile samo pločice, stoga se pojavio problem dodavanja pogreške isključivo na područje pločice. Ovaj problem je riješen korištenjem metode prelaska praga osjetljivosti. Kao rezultat završnog rada dobivena su dva MATLAB programa. Prvi program je osmišljen za samostalno dodavanje umjetno generiranih pogrešaka klikom miša po fotografiji pločice, dok je drugi program izrađen za smanjenu interakciju korisnika gdje se automatski generiraju pogreške na velikom skupu fotografija temeljem unosa željenog broja pogrešaka za svaku fotografiju ili prepuštanjem programu da sam sve izgenerira.

Ključne riječi:: Gaussova 2D funkcija, intenzitet piksela, metoda prelaska praga osjetljivosti, slučajno generiranje, točkasta pogreška, umjetna pogreška

ABSTRACT

Title: Generating artificial biscuit tile images

This final paper elaborates on the problem of generating one type of mistakes, which occur in a real life biscuit tile production, i.e. dot mistakes which represent a dent in a biscuit tile. The analysis has pointed to a similarity of the dent to an inverted Gaussian curve, where using the Gaussian 2D function, an almost identical mistake is generated to the real one. The paper analyzes real dot mistakes and describes the procedure of finding limits for the parameters of Gaussian 2D function in order to generate mistakes as real ones. In addition, the process of adding artificial mistakes on images of biscuit tiles without any error from real production is exemplified. The mentioned images do not only contain biscuit tiles, so there is an issue of adding mistakes only to the area of the biscuit tile. This problem is solved by using a thresholding method. The final paper results in two MATLAB programs. The first program is designed for users to independently add artificially generated mistakes by clicking on an image with a biscuit tile. In comparison, the other program is designed for a reduced user interaction with mistakes being automatically generated from a large data set of images by entering a number of desired mistakes for every image in a given file or letting MATLAB generate everything itself.

Keywords: Gaussian 2D function, pixel itensity, thresholding method, random generating, dot mistake, artificial mistake

ŽIVOTOPIS

Autor ovog predloška, Jana Dukić, rođena je 1. rujna 1999. u Osijeku. Osnovnoškolsko obrazovanje završava 2014. godine u Osnovnoj školi „Ivana Filipovića" u Osijeku. Iste godine uspisuje III. Gimnaziju Osijek koju završava 2018. godine. Nakon uspješno položene mature, 2018. godine obrazovanje nastavlja na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku smjera računarstvo. Kao student druge godine, osvojila je nagradu za postignut uspjeh u studiranju. Jana Dukić trenutno je u završetku preddiplomskog studija računarstva.

_____

Potpis autora

PRILOG 1 - RUČNO DODAVANJE POGREŠKE

```
clear all;
close all;
clc;

thisfile = 'Dist2588.bmp';
A = imread(thisfile);
originalImage = A;
imshow(A);
[M,N] = size(A);


captionFontSize = 14;
drawnow;
thresholdValue = 80;
binaryImage = originalImage > thresholdValue;
binaryImage = imfill(binaryImage, 'holes');
labeledImage = bwlabel(binaryImage, 8);
blobMeasurements = regionprops(labeledImage, originalImage, 'all');
numberOfBlobs = size(blobMeasurements, 1);
axis image;
hold on;
boundaries = bwboundaries(binaryImage);
numberOfBoundaries = size(boundaries, 1);
        thisBoundary = boundaries{1};
        plot(thisBoundary(:,2), thisBoundary(:,1), 'g', 'LineWidth', 2);
hold off;

while 1
[X_kor,Y_kor] = ginput(1);
if(isempty(X_kor))
    break;
end

    kor1 = round(X_kor);
    kor2 = round(Y_kor);

    kordinateX = zeros(1,1);
    kordinateY = zeros(1,1);
```

```
39      brojac1 = 1;
40      brojac2 = 1;
41
42       for i=1: size(thisBoundary,1)
43           if (thisBoundary(i,2) == kor1)
44               kordinateX(1,brojac1) = thisBoundary(i,1);
45               brojac1 = brojac1 + 1;
46           end
47           if (thisBoundary(i,1) == kor2)
48               kordinateY(1,brojac2) = thisBoundary(i,2);
49               brojac2 = brojac2 + 1;
50           end
51        end
52
53       if( kordinateX(1,1) > kordinateX(1,end))
54           gornjakordinataX = kordinateX(1,end);
55           donjakordinataX = kordinateX(1,1);
56       else
57           gornjakordinataX = kordinateX(1,1);
58           donjakordinataX = kordinateX(1,end);
59        end
60       if( kordinateY(1,1) > kordinateY(1,end))
61           gornjakordinataY = kordinateY(1,end);
62           donjakordinataY = kordinateY(1,1);
63       else
64           gornjakordinataY = kordinateY(1,1);
65           donjakordinataY = kordinateY(1,end);
66       end
67
68      flag1 = 0;
69      flag2 = 0;
70       if kor2 > gornjakordinataX && kor2 < donjakordinataX
71           fprintf('Izmedju X osi je tockaa\n')
72           flag1 = 1;
73       end
74      if kor1 > gornjakordinataY && kor1 < donjakordinataY
75           fprintf('Izmedju Y osi je tockaa\n')
76           flag2 = 1;
77      end
78
```

```
79      if(flag1 == 1 && flag2 == 1)
80          fprintf('Unutra jeee wuhuuuuuuuuuuuuuu!\n');
81                  fun = greska(size(A,2), size(A,1),kor2, kor1, Intensity);
82                  A = A - fun;
83                  imshow(A);
84      end
85  end
86
87  imageFolder = 'C:\Users\Korisnik\Desktop\Zavrsni rad\GinputSlikeSGreskom';
88  thisfile = thisfile(1:end-4);
89  image = sprintf('%s_WithError.bmp', thisfile);
90  fullFileName = fullfile(imageFolder, image);
91  imwrite(A, fullFileName);
```

PRILOG 2 - AUTOMATSKO DODAVANJE POGREŠKE

```
clear all;
close all;
clc;
files = dir('Surface_error');
N = length(files);

 prompt1 = {'Do you want MATLAB to choose number of random dot errors per image
     ↪ ?(1 − Yes/0 − No):'};
 dlgtitle1 = 'Input';
 %dims1 = [0 1];
 randomAnswer = inputdlg(prompt1, dlgtitle1);


for i= 3: size(files,1)
    thisfile= files(i).name
    originalImage = imread(thisfile); %thisfile
    A = originalImage;
    captionFontSize = 14;
    thresholdValue = 80;
    binaryImage = originalImage > thresholdValue;
    binaryImage = imfill(binaryImage, 'holes');
    labeledImage = bwlabel(binaryImage, 8);
    blobMeasurements = regionprops(labeledImage, originalImage, 'all');
    numberOfBlobs = size(blobMeasurements, 1);
    boundaries = bwboundaries(binaryImage);
    numberOfBoundaries = size(boundaries, 1);
        thisBoundary = boundaries{1};

    if( str2double(randomAnswer(1)) == 1)
        answer = round(1 + (5 − 1) .∗ rand(1,1))
    else
        prompt = {'Enter number of dot errors:'};
        dlgtitle = 'Input';
        answer = inputdlg(prompt, dlgtitle);
        answer = str2double(answer(1))
    end

    emergencyCounter = 0;
```

```
38
39                  max_X = 0;
40                  min_X = size(originalImage,1);
41                  max_Y = 0;
42                  min_Y = size(originalImage,2);
43                      for k = 1: size(thisBoundary,1)
44                          if( thisBoundary(k,1) > max_X)
45                              max_X = thisBoundary(k,1);
46                          end
47                          if( thisBoundary(k,1) < min_X)
48                              min_X = thisBoundary(k,1);
49                          end
50                          if( thisBoundary(k,2) > max_Y)
51                              max_Y = thisBoundary(k,2);
52                          end
53                          if( thisBoundary(k,2) < min_Y)
54                              min_Y = thisBoundary(k,2);
55                          end
56                      end
57
58
59      a = 1;
60      while a <= answer
61          randomX = min_X + (max_X - min_X) .* rand(1,1);
62          randomY = min_Y + (max_Y - min_Y) .* rand(1,1);
63          kor1 = round(randomY);
64          kor2 = round(randomX);
65
66          kordinateX = zeros(1,1);
67          kordinateY = zeros(1,1);
68          brojac1 = 1;
69          brojac2 = 1;
70          for b = 1: size(thisBoundary,1)
71              if (thisBoundary(b,2) == kor1)
72                  kordinateX(1,brojac1) = thisBoundary(b,1);
73                  brojac1 = brojac1 + 1;
74              end
75              if (thisBoundary(b,1) == kor2)
76                  kordinateY(1,brojac2) = thisBoundary(b,2);
77                  brojac2 = brojac2 + 1;
```

```
78              end
79
80          end
81           if( kordinateX(1,1) > kordinateX(1,end))
82              gornjakordinataX = kordinateX(1,end);
83              donjakordinataX = kordinateX(1,1);
84          else
85              gornjakordinataX = kordinateX(1,1);
86              donjakordinataX = kordinateX(1,end);
87           end
88          if( kordinateY(1,1) > kordinateY(1,end))
89              gornjakordinataY = kordinateY(1,end);
90              donjakordinataY = kordinateY(1,1);
91          else
92              gornjakordinataY = kordinateY(1,1);
93              donjakordinataY = kordinateY(1,end);
94          end
95         flag1 = 0;
96         flag2 = 0;
97          if kor2 > gornjakordinataX && kor2 < donjakordinataX
98              fprintf('Izmedju X osi je tockaa\n')
99              flag1 = 1;
100         end
101        if kor1 > gornjakordinataY && kor1 < donjakordinataY
102             fprintf('Izmedju Y osi je tockaa\n')
103             flag2 = 1;
104        end
105
106        if(flag1 == 1 && flag2 == 1)
107             fprintf('Unutra jeee wuhuuuuuuuuuuuuuu!\n');
108                  minIntensity = A(size(A,1)/2, size(A,2)/2);
109                  fun = greska(size(A,2), size(A,1),kor2, kor1, minIntensity);
110                  A = A - fun;
111                  %imshow(A);
112                 a = a+1;
113        else
114             fprintf('Nije unutra!\n');
115             emergencyCounter = emergencyCounter + 1;
116        end
117
```

```
118             if(emergencyCounter > 10) break; end
119
120         end
121
122     imageFolder = 'C:\Users\Korisnik\Desktop\Zavrsni rad\RandomSlikeSGreskom';
123     thisfile = thisfile(1:end-4);
124     image = sprintf('%s_WithRandomError.bmp', thisfile);
125     fullFileName = fullfile(imageFolder, image);
126     imwrite(A, fullFileName);
127     end
```