Lokalizacija vidnog polja u slikama s mikroskopa

Štern, Dora

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

Sveučilišni studij računalstva

LOKALIZACIJA VIDNOG POLJA U SLIKAMA S MIKROSKOPA

Završni rad

Dora Štern

Osijek, godina. 2023.

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

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

Osijek, 15.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Dora Štern
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4577, 28.07.2020.
OIB Pristupnika:	26883895806
Mentor:	doc. dr. sc. Hrvoje Leventić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Lokalizacija vidnog polja u slikama s mikroskopa
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rad:	Opisati primjenu i način brojanja stanica pomoću mikroskopa i hemocytometra. Opisati način lokalizacije u vidnom polju stakalca pri brojanju stanica pomoću mikroskopa i hemocytometra. Istražiti i opisati metode obrade slike za detekciju značajki u slici, koje se mogu iskoristiti za lokalizaciju vidnog polja. Razviti metodu za lokalizaciju vidnog polja u slici s mikroskopa. Rezervirano za: Dora Štern
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
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: 2 bod/boda Jasnoća pismenog izražavanja: 1 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	15.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2023.

Ime i prezime studenta:	Dora Štern
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4577, 28.07.2020.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Lokalizacija vidnog polja u slikama s mikroskopa**

izrađen pod vodstvom mentora doc. dr. sc. Hrvoje Leventić

i sumentora ,

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

U svijetu znanosti i istraživanja, mikroskopija je nezaobilazna tehnika koja nam omogućava uvid u mikroskopski svijet. Slike koje dobivamo pomoću mikroskopa često su bogate informacijama i ključne za razumijevanje različitih bioloških i kemijskih procesa. No, kao što svaki istraživač mikroskopskih slika zna, analiza ovih slika može biti iznimno zahtjevna.

Jedan od najvažnijih aspekata analize mikroskopskih slika jest lokalizacija vidnog polja, tj. precizno određivanje područja koje želimo istraživati unutar te slike. Ovaj korak je ključan jer nam omogućava fokusiranje na određene dijelove slike. Lokalizacija vidnog polja otvara vrata brojnim primjenama, uključujući brojanje i praćenje objekata te mjerenje različitih parametara koji su od znanstvenog i praktičnog značaja.

Iako ljudsko oko može intuitivno prepoznati i lokalizirati vidno polje na mikroskopskim slikama, računalima to nije tako jednostavno. Tehnološki izazovi leže u tome da se postigne razina percepcije i prepoznavanja slika slična onoj koju imaju ljudi. Zbog činjenice da računalo nema nikakvu svijest i percepciju o vanjskome svijetu, nije nimalo lako implementirati ovakav zadatak. Upravo ovdje dolazi do izražaja važnost razvoja metoda i algoritama za automatiziranu lokalizaciju vidnog polja na mikroskopskim slikama.

Svrha ovog završnog rada jest istražiti, razviti i evaluirati metodu lokalizacije vidnog polja na slikama dobivenim mikroskopom. Cilj je stvoriti alat koji će olakšati i ubrzati proces analize mikroskopskih slika, štedeći vrijeme i resurse istraživača. Ovaj alat se također može koristiti s već postojećim programskim rješenjima za brojanje stanica.

U nastavku ovog rada, detaljnije ćemo istražiti izazove s kojima se suočavamo pri lokalizaciji vidnog polja na mikroskopskim slikama te predstaviti različite metode i tehnike koje su potrebne za rješavanje ovog problema. Prvo će se opisati zapis slika u računalu te postupci koji su potrebni za razvijanje metode. Nakon toga opisati će se implementacija u odabranom programskom jeziku. Također će se provesti eksperimenti kako bi se procijenila učinkovitost razvijene metode.

1.1. Zadatak završnog rada

Zadatak završnog rada je opisati primjenu i način brojanja stanica pomoću mikroskopa i hemocitometra, opisati način lokalizacije u vidnom polju stakalca pri brojanju stanica pomoću mikroskopa i hemocitometra, istražiti i opisati metode obrade slike za detekciju značajki u slici,

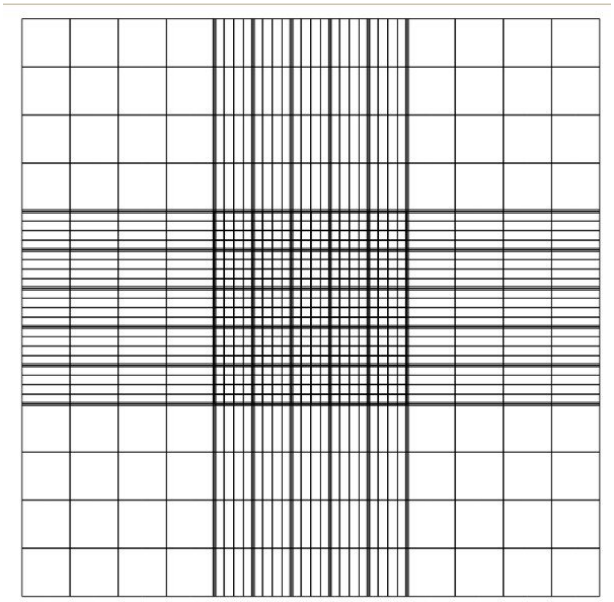
koje se mogu iskoristiti za lokalizaciju vidnog polja te razviti metodu za lokalizaciju vidnog polja u slici s mikroskopa.

2. BROJANJE I LOKALIZACIJA STANICA

Brojanje stanica pomoću mikroskopa i hemocitometra važna je tehnika koja se koristi u biološkim istraživanjima, medicinskim dijagnozama i laboratorijskim analizama. No prije brojanja stanica, potrebno je pripremiti uzorak. Ovisno o tipu hemocitometra, poželjno je imati specifičan raspon koncentracije uzorka te ovisno o vrsti stanica možda će biti potrebna obrada uzorka kao što je bojanje ili izoliranje stanica.

2.1. Hemocitometar

Hemocitometar je uređaj za brojanje stanica, isprva osmišljen i korišten za brojanje stanica krvi. Jednostavno rečen, to modificirano mikroskopsko stakalce s pravokutnim udubljenjem koje čini komoru u koju se ubrizgava suspenzija stanica. Kao što je vidljivo na slici 2.1., komora je podijeljena na devet kvadranta veličine 1 mm x 1 mm koji su onda podijeljeni na još manje cjeline.



Sl. 2.1. Mrežni raspored na hemocitometru

Pravila za brojanje stanica hemoitometrom ovise od osobe do osobe i od laboratorija do laboratorija. Prije početka brojanja stanica, potrebno je odlučiti u kojim kvadrantima će se brojati stanice i po kojim pravilima kako bi se izbjeglo brojanje iste stanice dvaput.

Postoji mnogo metoda kojima se određuje koji će se kvadranti brojati. Bitno je odabrati metodu kojom se najbolje može opisati stanje stanica na stakalcu. Najčešće korištena metoda brojanja je brojanje 4 vanjska kvadranta i središnji. Kada je u uzorku velik broj stanica, broje se svi kvadranti odozgo prema dolje, prvi red slijeva nadesno, drugi red suprotno i tako do kraja naizmjenično. Za brzu provjeru, iako tada rezultati uzorka nisu toliko reprezentativni, broje se prvi i zadnji kvadranti na dijagonali. U ovom radu koristit će se metoda u kojoj se broje 4 kutna kvadranta.

2.2. Pregled literature

Citometrija je pojam koji se odnosi na fizičke i kemijske karakteristike stanica i jedan od najvažnijih elemenata citometrije za istraživačke i kliničke svrhe je brojanje stanica. Davno je uspostavljena važnost brojanja krvnih stanica kao alat za istraživanje i kvantitativnu analizu u zdravstvu. Već više od sto godina hemocitometar se koristi za brojanje stanica i omogućava brojanje stanica ručno po niskoj cijeni i na različite načine.[1] Međutim, ručno brojanje stanica dugotrajan je proces i automatizacija istog vrlo je poželjna. U današnje vrijeme postoje dvije glavne komercijalne metode korištene kod automatskog brojanja stanica. Prva metoda je Coulterov brojač koji koristi svojstvo otpora stanice kako bi je analizirao, stanice su otkrivene kada prođu kroz otvor veličine nekoliko mikrometara. Druga metoda koristi protočni citometar koji se oslanja na princip raspršenja svjetlosti i odašiljanja fluorescencije specifično označenih stanica u suspenziji dok se kreću kroz lasersku zraku.[2] Iako je općeprihvaćeno da su ove dvije metode valjane i točne, mjerni instrumenti i priprema uzoraka jako su skupi. Zbog toga su kroz zadnjih nekoliko desetljeća napredovanjem računala i poboljšanjem optičkih i elektroničkih komponenti razvijana programska rješenja za brojanje stanica. ImageJ je javan i široko korišten alat za obradu slike kojeg je razvio američki Nacionalni institut za zdravlje. Za ovaj alat razvijen je poseban algoritam za brojanje stanica.[3]

Osim specifično programskog rješenja za analizu slika, u zadnje vrijeme široko korišten alat je strojno učenje tj. konvolucijske neuronske mreže. Stanice se ovom metodom prvo se trebaju identificirati pa se posebno provjerava koliko puta se pojavljuju.[4]

3. RAČUNALNI VID

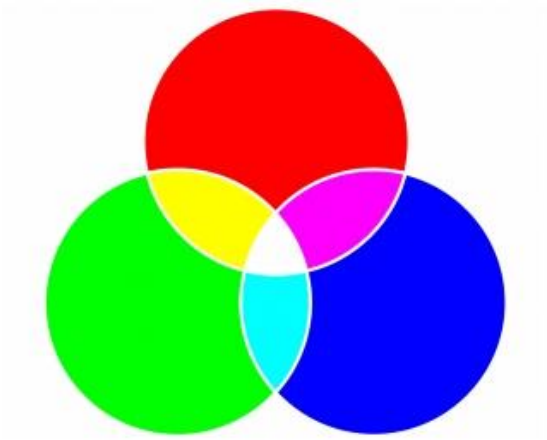
Szeliski u knjizi *Computer Vision: Algorithms and Applications (2010.)* tvrdi da računalnim vidom nastojimo opisati svijet koji vidimo u jednoj ili više slika i rekonstruirati njegova svojstva kao što su oblik, osvijetljene i opisi boja. Ljudi s lakoćom percipiraju trodimenzionalne strukture

kojima su okruženi. Kao dokaz da ljudi s lakoćom percipiraju trodimenzionalni prostor može se uzeti fotografiju više osoba gdje je ljudima lako odrediti koliko osoba ima na slici, koje su razlike u njihovom izgledu i odvojiti osobe od pozadine. Isto tako, gledajući slike s mikroskopa, ljudima je lako uočiti rešetke hemocitograma, odrediti o kojem se kvadrantu radi na slici te prebrojiti koliko svaki kvadrant ima stanica. Iako je ljudima lako odrediti značajke na slikama, računalu ovakvi zadaci predstavljaju velik problem. Zbog toga se već 1970-ih počinje se razmatrati računalni vid i kako implementirati „vizualni ulaz“ što bi bio korak za rješavanje složenijih problema.

3.1. Zapis slike u računalu

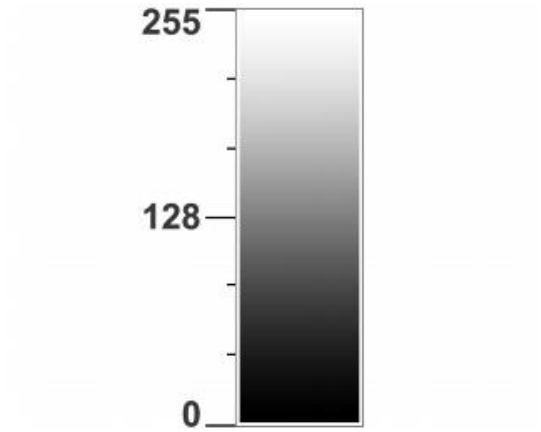
Prije obrade slike, bitno je osvijestiti na koji se način slike spremaju u računalu kako bi znali što i kako se mijenja na slici njezinom obradom. Najmanji element od kojega se sastoji digitalna slika je piksel. Pikseli su raspoređeni u redove i stupce koji predstavljaju rezoluciju slike npr. rezolucija 320x480 znači da je slika široka 320 piksela i visoka 480 piksela. U računalu pikseli se zapisuju u obliku brojeva, a za pretvaranje brojeva u boje koriste se različiti modeli boja.

Najkorišteniji je RGB model (slika 3.1.) kojemu su komponente (kanali) **crvena** (*red*), **zelena** (*green*) i **plava** (*blue*). U 24-bitnoj slici svaka komponenta je cijeli pozitivan broj u rasponu od 0 do 255. RGB je aditivan model boja, što znači da je jedna boja definirana kao kombinacija izvora svjetlosti i zbroja RGB komponenti.



Sl. 3.1. RGB model boja

Model boja u sivim tonovima[1] (slika 3.2.) ima samo jednu komponentu: **svjetlinu** (*lightness*). Njezina vrijednost također je cijeli pozitivan broj u rasponu od 0 do 255.



Sl. 3.2. Raspon modela boja u sivim tonovima

Ova dva modela bitna su jer slike obično dolaze u RGB modelu, a za obradu se slika mora pretvoriti u sliku sivih tonova. Ima više načina za pretvaranje slike iz RGB modela u sliku sivih tonova:

1. Metoda svjetline (*Lightness Method*): Uzimaju se vrijednosti komponenti s najvećom i najmanjom vrijednosti.

$$\boldsymbol{grayscale} = \frac{\min(R, G, B) \ + \ \max(R, G, B)}{2} \tag{3-1}$$

2. Metoda prosjeka (*Average Method)*: Uzima se prosječna vrijednost tri komponente.

$$\boldsymbol{grayscale} = \frac{R + G + B}{3} \tag{3-2}$$

3. Metoda osvijetljenosti (*Luminosity Method*): Najbolja metoda za pretvaranje u sliku sivih tonova. Nakon više eksperimenata i dubokih analiza, dobivena je formula:

$$\boldsymbol{grayscale} = 0.299 * R + 0.587 * G + 0.114 * B \tag{3-3}$$

[1] Eng. grayscale

Ovu metodu koriste biblioteka za obradu slike OpenCV.

3.1.1. Pretvaranje između modela boja

Kao što je ranije spomenuto, za prikaz slika na računalu koriste se različiti modeli boja, a najčešći je RGB model. Jedno od najčešćih pretvorbi koje je potrebno izvršiti nad slikama je iz RGB modela u sliku sivih tonova. Razlog ove pretvorbe je što je jednostavnije vršiti obradu i analizu slike na slikama sivih tonova.

Nakon obrade slike, ponekad je poželjno vratiti sliku u RGB model. To se postiže tako što se svakom kanalu RGB doda kanal slike sivih tonova.

Još jedna situacija u kojoj bi bilo potrebno pretvaranje između modela je ako se koristi više biblioteka za obradu slike čiji se modeli boja se razlikuju. Takav slučaj je kod Matplotlib biblioteke i OpenCV biblioteke. Kod prikazivanja slike koja je učitana pomoću OpenCV biblioteke, slika će se prikazati u BGR modelu, a kada je korištena Matplotlib biblioteka, slika se prikazuje u RGB modelu.

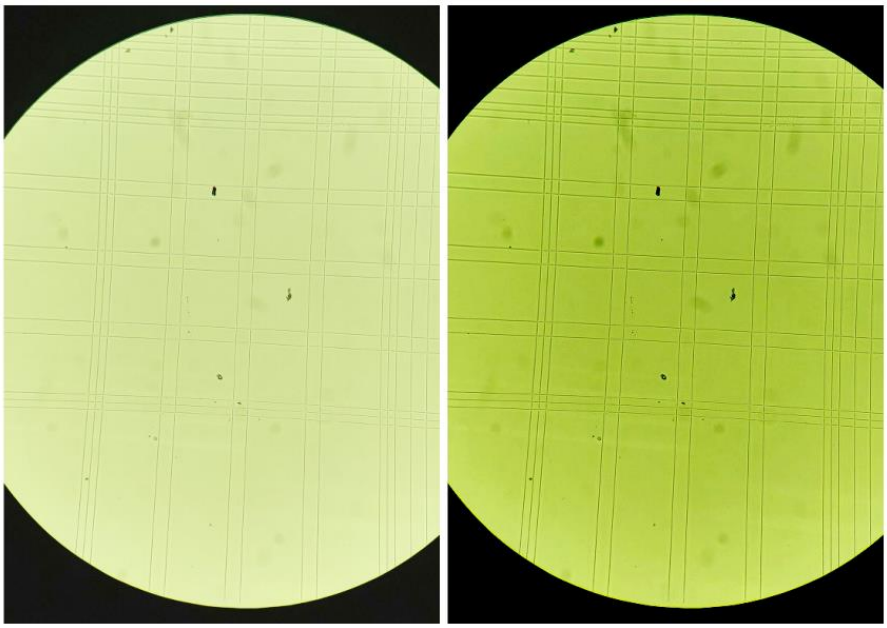
3.2. Obrada i analiza slike

3.2.1. Normalizacija

Normalizacija slike proces je u kojem se mijenja raspon vrijednosti piksela i tako se ujednačava slika. Rješenje je to problema kada na slici prevladavaju tamni/svijetli tonovi i tako sakrivaju detalje slike. Za min-max normalizaciju vrijedi formula:

$$odredišni\ kanal = 255 * \frac{izvorišni\ kanal - min}{max - min} \tag{3-4}$$

koja vrijedi za slike sivih tonova, a za slike u boji primjenjuje se na sva tri kanala. U svrhu demonstracije na slici 3.3. lijevo je namjerno povećana ekspozicija, a desno primijenjena normalizacija pa su detalji puno jasniji i čitljiviji.



Sl. 3.3. Izvorna slika (lijevo) i normalizirana slika (desno)

Iako normaliziranje otkriva detalje slike, često također ističe ili stvara šum koji nije bio na izvornoj slici. Zbog toga je u dosta slučajeva nakon normalizacije potrebno ukloniti šum koji se pojavio na slici.

3.2.2. Uklanjanje šuma[2]

Šum slike su varijacije ili poremećaji u svjetlini ili boji slike koji ne proizlaze iz prirode snimljenoga objekta[5]. Može nastati zbog elektroničkih smetnji, senzora, slabe osvijetljenosti, zbog tehničkih poteškoća pri snimanju ili obradi slike. Šum se može dodati slici(aditivan, formula 3-5) ili množiti sa slikom (multiplikativan, formula 3-6)[6].

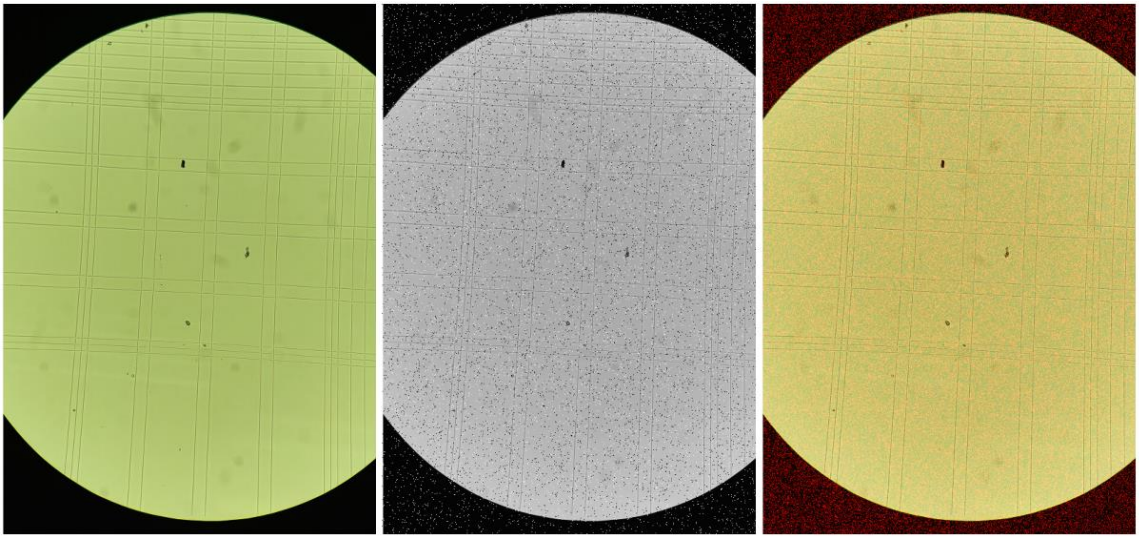
$$w(x,y) = s(x,y) + n(x,y) \tag{3-5}$$

[2] Eng. Denoising

$$w(x,y) = s(x,y) * n(x,y) \tag{3-6}$$

U ovom slučaju x, y su koordinate piksela na koje se primjenjuje šum, $s(x,y)$ je originalna slika, $n(x,y)$ je šum koji se dodaje slici i $w(x,y)$ je slika nakon primjene šuma.

Postoji nekoliko vrsta šumova, najčešći su Gaussov šum i zrnati šum. Gaussov šum je zapravo dobra aproksimacija većine realnih šumova[7], dok je zrnati šum impulsni šum i pojavljuje se kod velikih razlika u svjetlini. Za jasniju demonstraciju ova dva šuma, na slici 3.4. prikazane su slike s umjetno dodanim šumovima. Šum može uvelike otežati rad sa slikama, odnosno detekciju značajki kao što su linije ili objekti pa se na slike primjenjuju metode zaglađivanja.



Sl. 3.4. Izvorna slika (lijevo), zrnati šum (sredina), Gaussov šum (desno)

Postoji više načina zaglađivanja, ali relevantni za ovaj rad su zaglađivanje prosječnom vrijednošću i Gaussov filter te su oba jednostavni nisko propusni filteri. Općenito se smatra da je šum nasumična varijabla s prosjekom jednakim nuli[8]. Ako se analizira piksel sa šumom, njegov zbroj s nasumičnom varijablom daje originalan piksel. Koristeći zaglađivanje prosječnom vrijednošću, uzima se mali prozor u slici i velika je vjerojatnost da na slici postoji još jedan takav prozor sa sličnim vrijednostima. Za zaglađivanje se tada grupiraju slični prozori i računa se prosječna vrijednost piksela kojom onda zamjenjujemo piksele u prozorima. Gaussov filter funkcionira na sličan način, ali filtrira susjedne piksele tako da nađe težinski prosjek piksela prema Gaussovoj distribuciji.

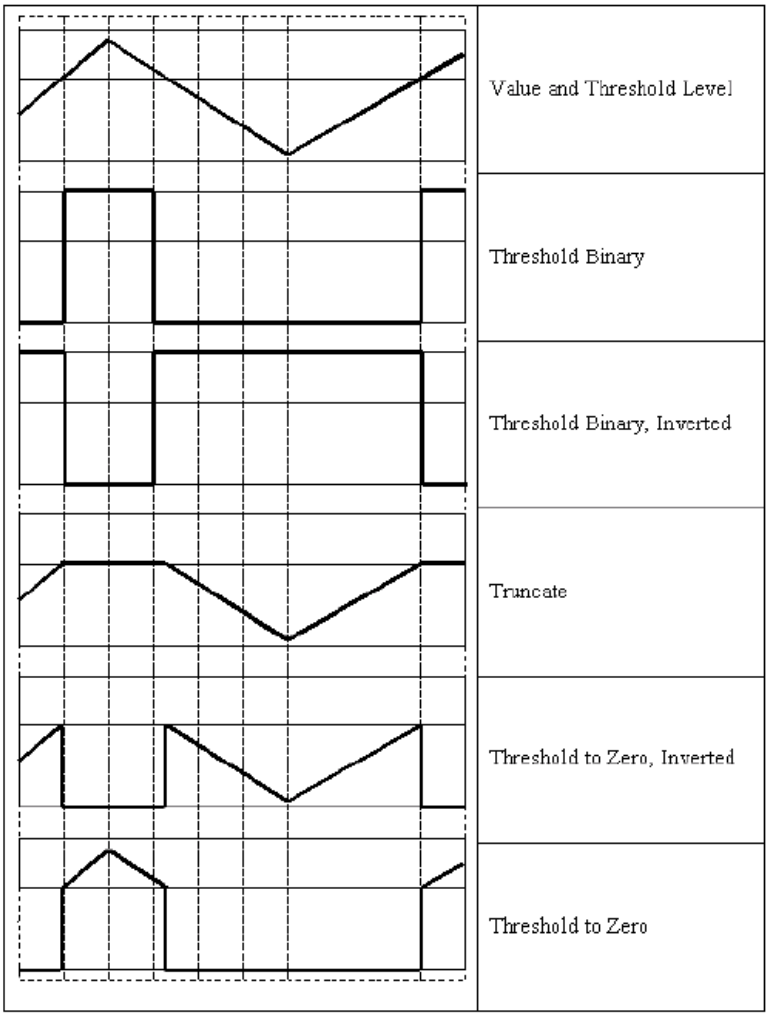
3.2.3. Segmentacija

Segmentacija slika pomaže kod dobivanja regija interesa [3] iz slike. To je proces odvajanja slika u različita područja. Izvodi se na temelju svojstava slike kao što su sličnost, diskontinuitet itd. Cilj segmentacije slika je pojednostaviti sliku za bolju analizu[9]. Ovisno o analizi koju je potrebno vršiti na slici, koriste se različite metode segmentiranja slike.

3.2.4. Prag

Prag je najjednostavniji način segmentiranja slika. Postoji više operacija koje se mogu vršiti pragom, na slici 3.5. prikazano je koje su operacije moguće i kako one mijenjaju početni signal. Za potrebe ovog rada, bit će opisan samo binarni prag.

[3] Eng. Region of interest



Sl. 3.5. Tipovi operacija s pragovima

Od slike u sivim tonovima pragom se može stvoriti binarna verzija početne slike. Binarna slika je ona koja se sastoji od piksela koji mogu poprimiti samo dvije vrijednosti, obično crnu (0) ili bijelu (255). Velika prednost binarnih slika je lako odvajanje objekata od pozadine što je vrlo poželjno kod obrade i analize slika. [10]

Postoje različiti načini određivanja praga: jednostavno, prilagodljivo[4] i Otsuovo određivanje praga. Kod jednostavnog određivanja praga, imamo vrijednost praga i na svaki piksel se primjenjuje ista vrijednost praga. Ako je vrijednost piksela niža od praga, vrijednost mu se postavlja na nulu, u suprotnom na maksimalnu vrijednost. U slučaju da slika nema dosljedno

[4] Eng. adaptive

osvijetljene u različitim područjima, promjenjiv prag može biti opravdano rješenje. Algoritam za promjenjiv prag u ovom slučaju određuje prag za piksele u malim područjima slike, pa se na različita područja primjenjuje različit prag.

Za globalnu primjenu praga potrebno je odabrati proizvoljnu veličinu koja će biti prag, a to može predstavljati problem jer se mora procijeniti koja vrijednost bi bila najbolja kao prag. Otsuov proces pretvaranja slika u binarne [5]automatski određuje vrijednost koja bi bila najbolja kao prag i primjenjuje je.

3.2.5. Otkrivanje rubova i kontura

Pojam rubova na slikama označava grupu piksela kojima se naglo promijenila razina sive boje. Otkrivanje rubova je jednostavna metoda koja prepoznaje i odvaja rubove prema tim naglim promjenama[11]. Canny operator je algoritam za otkrivanje rubova koji funkcionira tako da odabire rubne točke prag metodom pa rezultati otkrivanja rubova ovise o pragu.

Algoritam ima nekoliko koraka:

1. Otklanjanje šuma: Na sliku se primjenjuje Gaussov filter s jezgrom 5x5 jer šum može umanjiti efektivnost algoritma.
2. Računanje intenziteta gradijenta slike: Računa se pomoću tehnika gradijentnih filtera ili derivacijskih operatora, kao što su Sobelov operator ili Scharr operator. Ovi operatori procjenjuju gradijent tako što računaju razlike u intenzitetima piksela susjednih piksela u vodoravnom i okomitom smjeru.
3. Suzbijanje vrijednosti koje nisu maksimalne: Slika se pregledava po smjeru gradijenta slike i ako pikseli nisu dio lokalnog maksimuma, vrijednost im se postavlja na nulu.
4. Određivanje praga histerezom: Za ovaj korak potrebne su nam dvije vrijednost praga, minimalna i maksimalna. Svi rubovi kojima je intenzitet gradijenta veći od maksimalne vrijednosti praga sigurno će biti rubovi, a svi rubovi kojima je intenzitet gradijenta manji od minimalne vrijednosti praga neće biti rubovi pa se uklanjaju. Pikseli kojima je

[5] Eng. Otsu's Binarization

vrijednost između minimalne i maksimalne vrijednosti praga bit će rubovi ako su im susjedni pikseli rubovi, u suprotnom neće.

Konture su granice oblika s istim intenzitetom. Nakon što su otkrivene linije objekata na slici, algoritam za pronalaženje kontura pretražuje sliku red po red i piksel po piksel. Kada se pronađe piksel koji je u prvom planu binarne slike, znači da je pronađen početak nove konture i primjenjuje se algoritam za praćenje konture. Algoritam se kreće oko konture, prateći susjedne piksele u prvom planu i izvodi se dok nije došao do početne točke.

3.3. Procjena linija i oblika na slici

Nakon obrade slike, uklanjanja šuma, segmentacije i otkrivanja rubova i kontura može se procjenjivati gdje i kakve linije i oblici se nalaze na slici. Za otkrivanje linija na slici potrebni su rubovi koji se onda pretražuju kako bi se našli dijelove linija. Za svaku rubnu točku se razmatraju svi mogući dijelovi linija koji bi mogli prolaziti kroz tu točku i oni predstavljaju linije na slici.

Za otkrivanje oblika, koristi se algoritam koji opetovano pojednostavljuje konture tako da smanjuje broj točaka od kojih se sastoje, a da pri tome zadrži oblik. Ovisno o broju točaka koji ostane na kraju algoritma, može se procijeniti oblik npr. ako je broj točaka tri, moguće je da je oblik trokut, ako je četiri pravokutnik itd.

4. IMPLEMENTACIJA METODE LOKALIZACIJE U PROGRAMSKOM JEZIKU PYTHON

U prijašnja dva poglavlja detaljno su opisani postupci koje je moguće primijeniti na sliku za njezinu pred-obradu i analizu. Jednostavno rečeno, za lokalizaciju vidnoga polja na slikama s mikroskopa potrebno je primijeniti spomenute postupke specifičnim redoslijedom. Za implementaciju metode u programskom jeziku Python potrebno je pomoću metoda iz biblioteke za računalni vid provesti sljedeće korake:

- Učitavanje slike
- Normalizacija slike i prebacivanje u sive tonove (grayscale)
- Uklanjanje šuma na slici
- Određivanje binarnog praga slike
- Primjena filtera za zamućivanje

- Proširivanje praga
- Otkrivanje i proširivanje rubova na slici
- Otkrivanje linija na slici i crtanje linija na praznu sliku
- Određivanje praga, uklanjanje šuma i određivanje rubova na novoj slici
- Pronalaženje kontura
- Odvajanje kontura na pravokutnike i kvadrate
- Pronalaženje rubnih točaka kvadranta
- Brojanje kontura na svakoj strani kvadranta
- Lokalizacija kvadranta

4.1. Programski jezik Python

Metoda za lokalizaciju vidnog polja implementirana je u programskom jeziku Pythonu, za obradu i analizu slike korištena je biblioteka OpenCV. Python je široko upotrjebljavan, interpretiran, objektno-orijentiran i programski jezik visoke razine s dinamičnim semantikama[12]. Prednosti su ovog programskog jezika što je jednostavan za naučiti, jednostavan za upotrebu i jednostavno mu je postaviti okolinu za korištenje. Zbog svoje jednostavnosti, jako ga puno ljudi koristi i zbog toga podržava veliki broja biblioteka i tako se proširuju mogućnosti jezika.

4.2. OpenCV biblioteka

OpenCV je biblioteka otvorenog koda koja sadrži više stotina algoritama za računalni vid i strojno učenje. Koristi se u različitim područjima, uključujući robotiku, proširenu stvarnost, dijagnostiku medicinskih slika i sustave za nadzor.

Na početku OpenCV projekta ciljevi su bili :

- Poboljšati istraživanje vida ne samo pružajući otvoreni nego i optimizirani kod za osnovnu infrastrukturu vida.
- Proširiti znanje vida pružajući zajedničku infrastrukturu na kojoj razvojni programeri mogu graditi svoje projekte, tako da bi kod bio dostupniji i prenosiviji.
- Unaprijediti komercijalne aplikacije za vid tako da se izrađuje portabilni i optimizirani kod koji je dostupan besplatno, a za aplikacije je potrebna dozvola i nije nužno da kod bude otvoren ili besplatan.

Optimizirani algoritmi mogu se koristiti za izvršavanje zadataka poput filtriranja slika, otkrivanja rubova, prepoznavanja objekata i izdvajanja značajki te pruža mogućnost strojnog učenja s alatima TensorFlow ili PyTorch. Za razvoj metode za lokalizaciju vidnog polja u slici s mikroskopa bit će potrebno koristiti OpenCV module za procesiranje slika, učitavanje i spremanje slika.

4.3. Učitavanje slike, normalizacija, pretvaranje u sliku sivih tonova

Prvo je potrebno navesti sve biblioteke koje će se koristiti, a to su OpenCV i NumPy. OpenCV se koristi za sve operacije nad slikama kao što su učitavanje, normalizacija, uklanjanje šuma, detekcija rubova i linija, spremanje slike itd., a NumPy za stvaranje konvolucijskih matrica za filtriranje. Na slici 4.1. prikazan je kod za učitavanje slike.

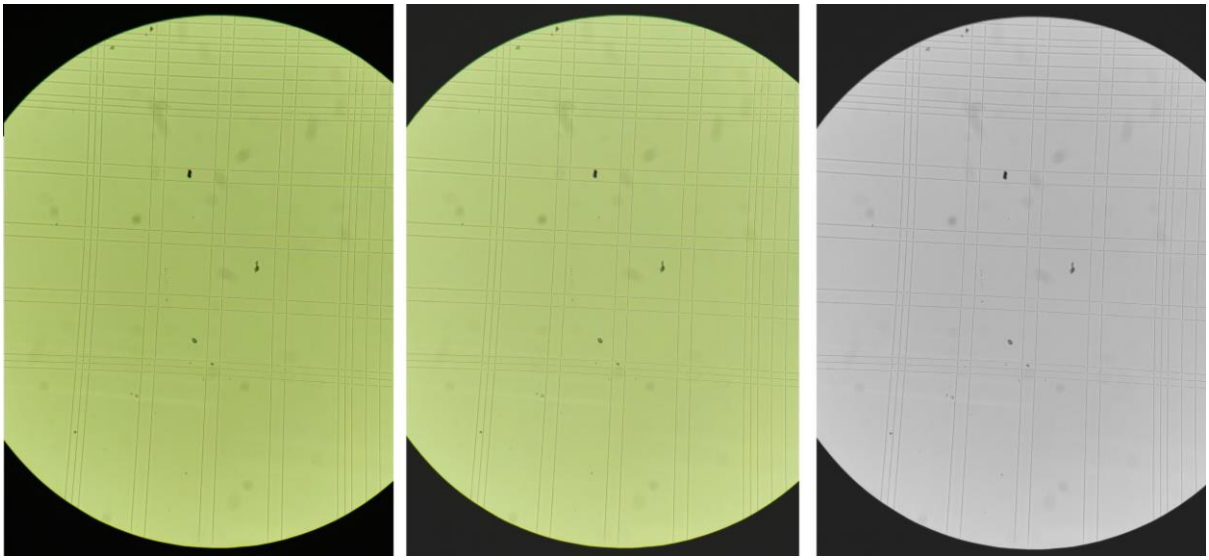
```
Linija  Kod

1:      import cv2
2:      import numpy as np
        . . .
14:     filename='pictures\IMG_20211125_163742.jpg'
15:     img = cv2.imread(filename)

17:     normalized = cv2.normalize(img, dst=None, alpha=30,
        beta=255,norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U) . . .
20:     gray = cv2.cvtColor(normalized, cv2.COLOR_BGR2GRAY)
```

Sl. 4.1. Kod za učitavanje slike, normalizaciju i pretvaranje u sliku sivih tonova

Nakon učitavanja slike, sliku je potrebno normalizirati, jer u većini slučajeva slike nisu snimljene u idealnim uvjetima osvjetljena, što može sakriti detalje slike koji su potrebni za analizu. Ne postoji idealni parametar za normalizaciju svih sliku, nego se prilagođava individualnoj slici. Treba biti oprezan jer se povećanjem parametra povećava ekspozicija i gubi se šum, ali i detalji slike. S druge strane, smanjivanje alfa parametra smanjuje se ekspozicija, detalji postaju jasniji, ali dodaje se šum koji se možda neće moći toliko lako otkloniti. Na slici 4.1. u šestoj liniji prikazan je kod korišten za normalizaciju slike, a sedmoj liniji za pretvaranje u sliku sivih tonova. Na slici 4.2. prikazana je originalna slika, slika nakon normalizacije i slika sivih tonova.



Sl. 4.2. Izvorna slika (lijevo), normalizirana slika(sredina), slika u sivim tonovima (desno)

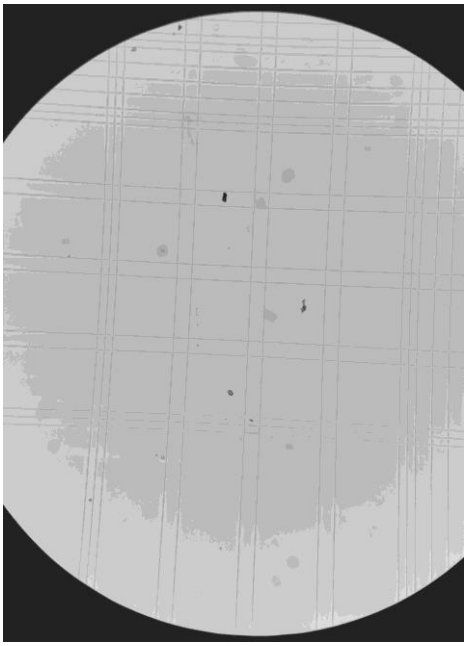
4.4. Uklanjanje šuma, određivanje binarnog praga

Iako je parametar alfa u normalizaciji bio veći od nule i tako je otklonjen dio šuma, na slici je svejedno potrebno provesti funkciju zaglađivanja, kod na slici . 4.3., a na slici 4.4. prikazana je dobivena slika.

Linija Kod

```
23:     denoised = cv2.fastNlMeansDenoising(gray, None, h=30,
        templateWindowSize=5, searchWindowSize=7)
```

Sl. 4.3. Kod za zaglađivanje slike



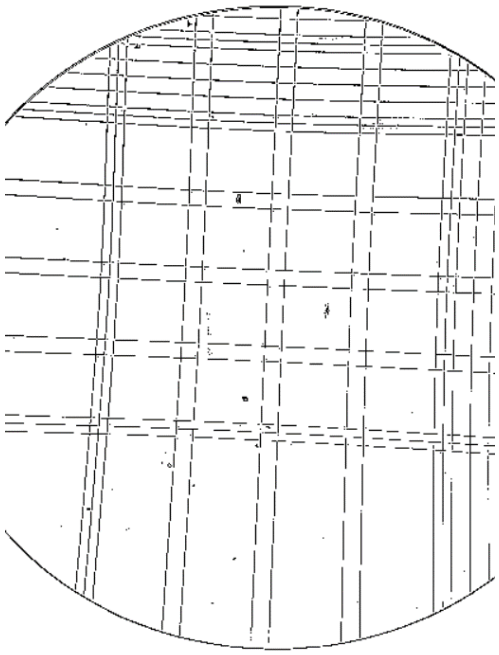
Sl. 4.4. Zaglađena slika

Nakon zaglađivanja, slika je spremna za određivanje binarnog praga. Ovo je nužan korak jer nam je cilj izdvojiti linije hemocitometra u prednji plan, a sve ostalo promatrati kao pozadinu. Za ovaj korak korišten je kod sa slike 4.5. i rezultat je slika 4.6.

```
Linija  Kod

26:     thresholded =
        cv2.adaptiveThreshold(denoised,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
                    cv2.THRESH_BINARY,9,3)
```

Sl. 4.5. Kod za određivanje praga



Sl. 4.6. Slika binarnog praga

4.5. Primjena filtera za zamućivanje, proširivanje praga

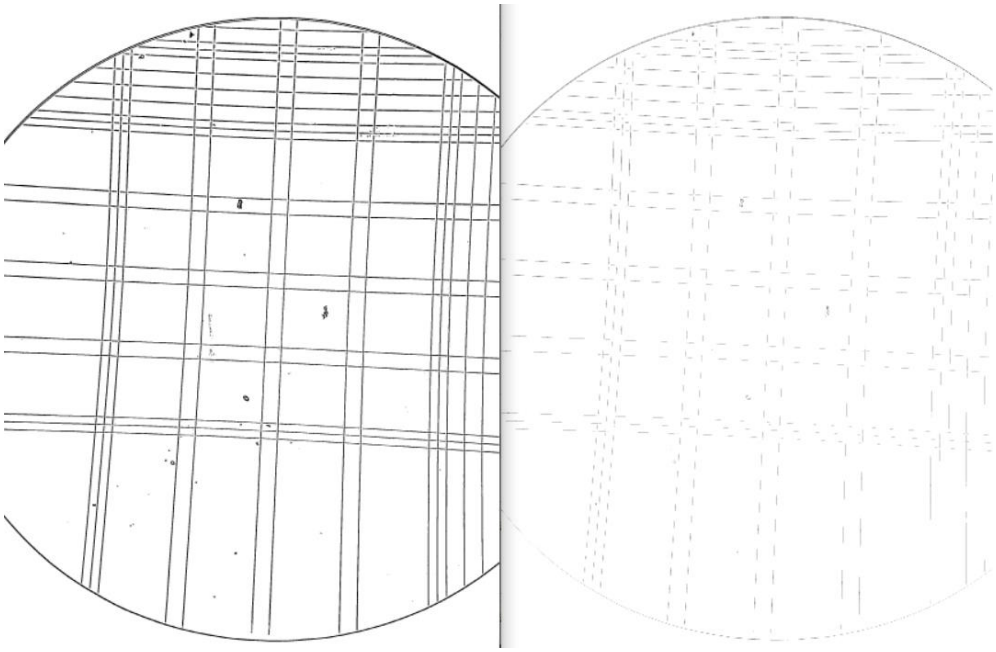
Linije na slici 4.6. su isprekidane i na slici još uvijek ima puno šuma koji je moguće ukloniti Gaussovim filterom zamućivanja [6] nakon čega se primjenjuje funkcija proširivanja linija kako bi bile vidljivije. Kod je prikazan na slici 4.7., a rezultat ovih operacija na slici 4.8.

```
Linija  Kod

35:     kernel = np.ones((5,5),np.float32)/25
36:     blurred_threshold = cv2.filter2D(thresholded,-1,kernel)
        . . .
41:     kernel = np.ones((3, 3), np.uint8)
42:     blurred_dilated_threshold = cv2.dilate(blurred_threshold, kernel,
        iterations=3)
```

Sl. 4.7. Kod primjenu Gaussovog filtera i proširivanja linija

[6] Eng. Gaussian Blur



Sl. 4.8. Zamućena slika praga (desno) i slika poslije operacije proširivanja (lijevo)

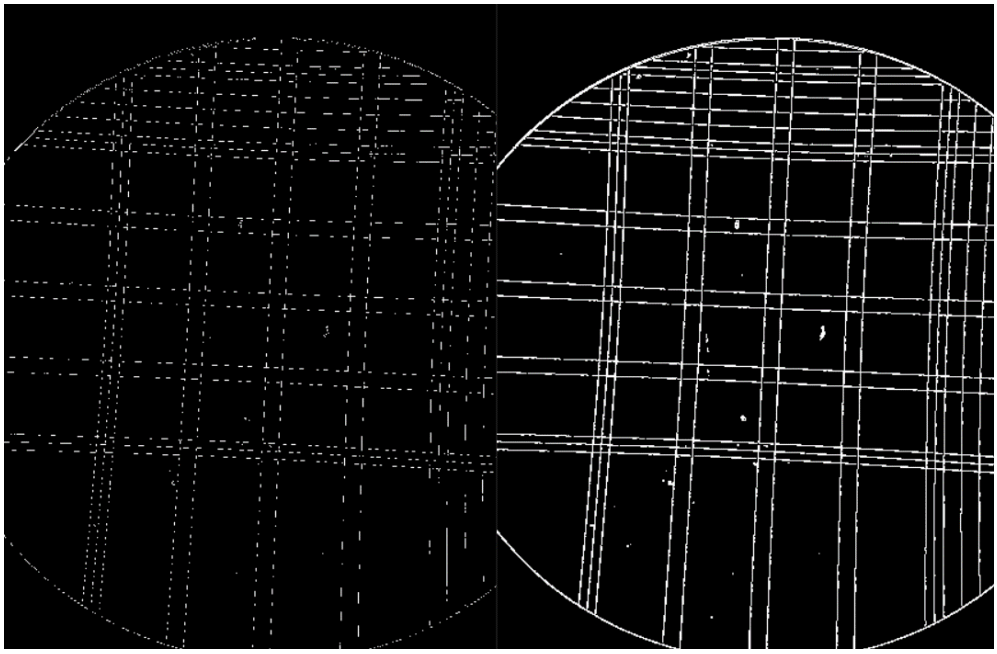
4.6. Otkrivanje i proširivanje rubova, otkrivanje i crtanje linija

Nakon otkrivanje i proširivanja praga, slika je spremna za algoritam otkrivanja rubova, kod se nalazi na slici 4.9 , a otkriveni rubovi na slici 4.10. Ovaj korak omogućava otkrivanje linija na slici. Kod za otkrivanje i crtanje linija nalazi se na slici 4.11., a rezultat crtanja linija na praznu sliku nalazi se na slici 4.12.

Linija Kod

```
50:     edges = cv2.Canny(dilated,30,60)
        . . .
52:     kernel = np.ones((3, 3), np.uint8)
53:     dilated_edges = cv2.dilate(edges, kernel, iterations=2)
```

Sl. 4.9. Kod primjene Gaussovog filtera i proširivanja na linijama

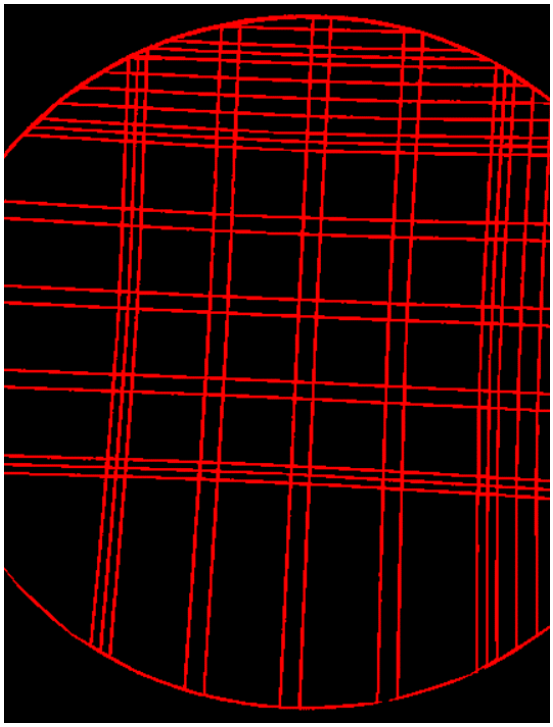


Sl. 4.10. Rubovi slike s Canny funkcijom (lijevo) i rubovi poslije operacije proširivanja (desno)

Linija Kod

```
64:     lines = cv2.HoughLinesP(dilated_edges, 1, np.pi / 180,
        threshold=50, minLineLength=100, maxLineGap=50)
65:
66:
67:     processed = img.copy()
68:     canvas = np.zeros_like(img)
69:     if lines is not None:
70:         for line in lines:
71:             x1, y1, x2, y2 = line[0]
72:             cv2.line(canvas, (x1, y1), (x2, y2), (0, 0, 255), 5)
```

Sl. 4.11. Kod za otkrivanje i crtanje linija



Sl. 4.12. Slika s linijama iz Houghova algoritma

Sada kada imamo novu sliku na kojoj su nacrtane otkrivene linije, na toj slici opet trebamo primijeniti algoritam za otkrivanje rubova kako bismo dobili oštre rubove nacrtanih linija. Kod je na slici 4.13., a dobivene konture na slici 4.14.

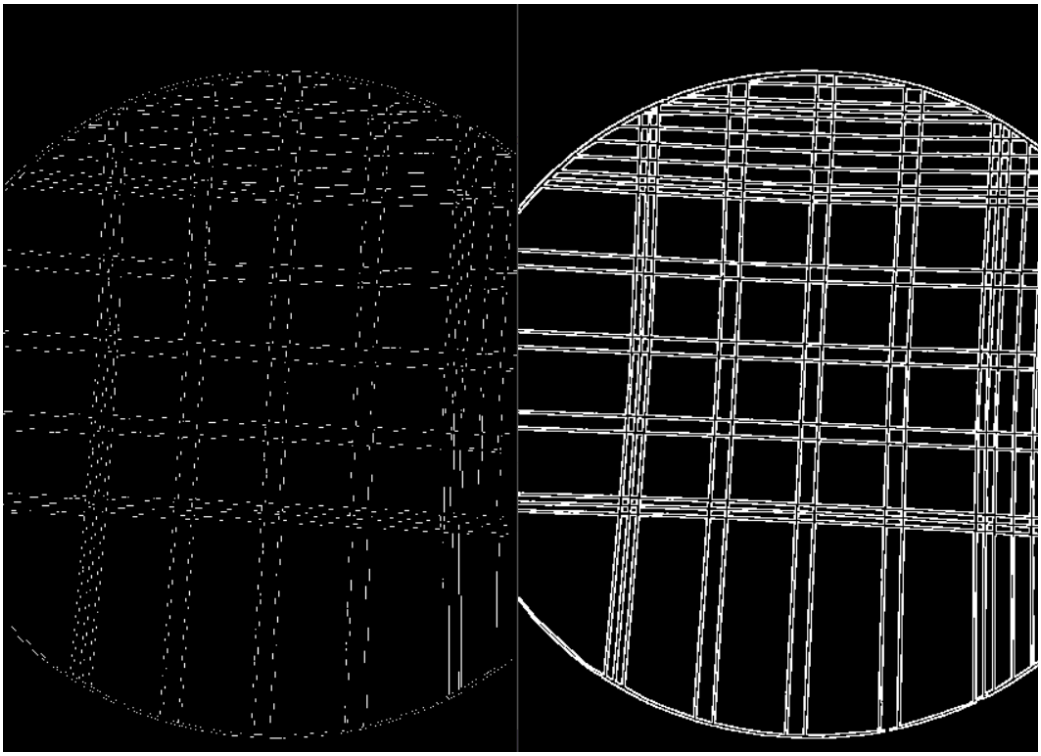
```
Linija  Kod

82:     lines_gray = cv2.cvtColor(canvas, cv2.COLOR_BGR2GRAY)
83:
84:     _, thresholded_lines_gray = cv2.threshold(lines_gray, 0, 255,
        cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        . . .
92:     denoised_lines =
        cv2.fastNlMeansDenoising(thresholded_lines_gray, None, h=30,
        templateWindowSize=5, searchWindowSize=7)
        . . .

98:     edges = cv2.Canny(gray,50,150)

100:    kernel = np.ones((3, 3), np.uint8)
101:    edges = cv2.dilate(edges, kernel, iterations=3)
```

Sl. 4.13. Kod za otkrivanje i crtanje linija



Sl. 4.14. Otkriveni rubovi (lijevo) i rubovi nakon operacije proširivanja (desno)

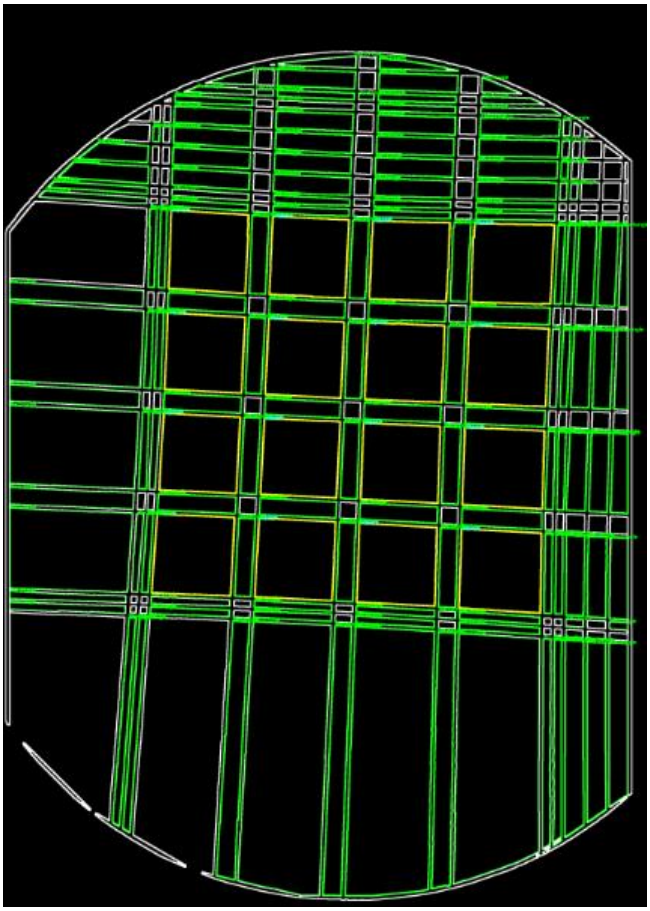
4.7. Pronalaženje kontura, odvajanje na kvadrate i pravokutnike

Na otkrivenim rubovima potrebno je primijeniti algoritam za pronalaženje kontura. Aproksimacijom kontura na minimalan broj točaka dobivaju se osnovni oblici i filtriranjem broja točaka na četiri, rezultat su oblici s četiri točke tj. pravokutnici. Kvadrati imaju omjer stranica 1:1 pa je za njihovo filtriranje potrebno pregledavati omjer stranica svakog aproksimiranog pravokutnika. Kod za filtriranje i crtanje pravokutnika nalazi se na slici 4.15., a na slici 4.16. su nacrtani pravokutnici.

```
Linija  Kod

118:    contours, hierarchy = cv2.findContours(gray,
        0,  cv2.CHAIN_APPROX_SIMPLE)
        . . .
123:    for cnt in contours:
124:        x1, y1 = cnt[0][0]
125:        approx = cv2.approxPolyDP(cnt, 0.03 * cv2.arcLength(cnt,
        True), True)
126:        if len(approx) == 4:
127:            x, y, w, h = cv2.boundingRect(cnt)
128:            ratio = float(w) / h
129:            if ratio >= 0.9 and ratio <= 1.1:
130:                if w >= 100 and h >= 100:
131:                    ratios.add(ratio)
132:                    img = cv2.drawContours(img, [cnt], -1, (0, 255,
        255), 3)
133:                    cv2.putText(img, 'Square', (x1, y1),
        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 0), 2)
134:            else:
135:                if h <= 350 and w<=400 and h>=35 and w>=35 :
136:                    rectangle_sizes.append((w, h))
137:                    cv2.putText(img, 'Rectangle', (x1, y1),
138:    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
139:                    img = cv2.drawContours(img, [cnt], -1, (0, 255,
        0), 3)
```

Sl. 4.15. Kod za otkrivanje i crtanje pravokutnika



Sl. 4.16. Slika otkrivenih pravokutnika

4.8. Pronalaženje rubnih točaka kvadranta, brojanje kontura na svakoj strani kvadranta

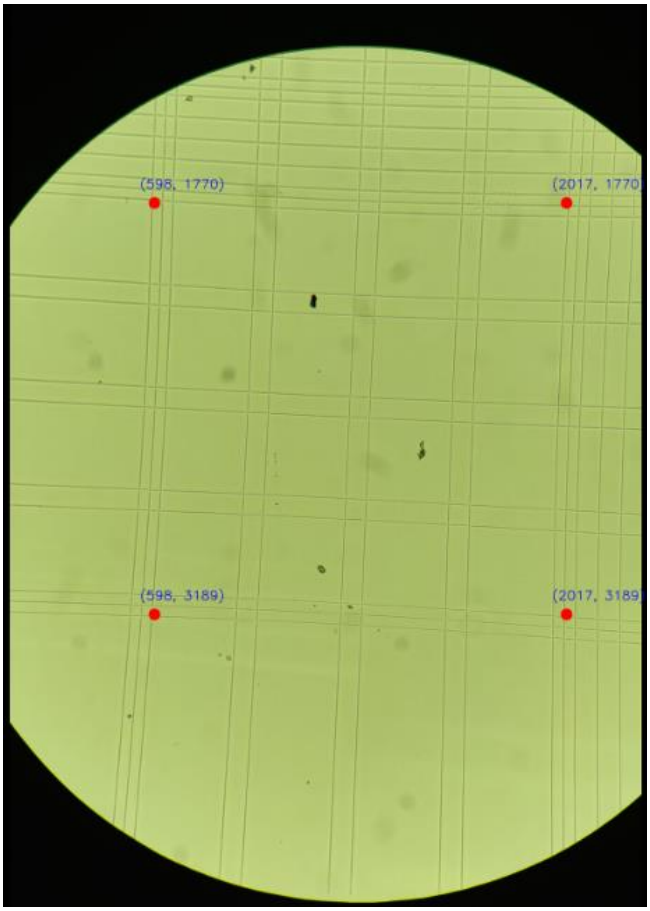
U presjeku linija na hemocitometru, kvadrati se pojavljuju na samo dva mjesta: kutnim kvadrantima i središnjem kvadrantu. Razlikuju se po veličini jer najviše linija prolazi kroz središnji kvadrant pa su kvadrati manji nego oni u kutnim kvadrantima. U slučaju da među otkrivenim kvadratima nema kvadrata većih veličina, pretpostavka je da se radi o središnjem kvadrantu. Ukoliko je program odredio da postoji barem 3 veća kvadrata, potrebno je izvršiti daljnju analizu. Veći kvadrati označeni žutom bojom na slici 4.16. služe za relativnu procjenu lokacije promatranog kvadranta. Primjenjuje se na način da od svih otkrivenih kvadrata, traži najmanju i najveću vrijednost koju poprimaju točke pravokutnika na x i y osi ako je ishodište gornji lijevi kut fotografije. Kombiniranjem ovih vrijednosti dobiju se četiri točke koje približno omeđuju kvadrant, kod za pronalazak točaka nalazi se na slici 4.17. , a na slici 4.18. su točke nacrtane na originalnoj slici.

Linija Kod

```
132:    coord=[]
        . . .
134:    for cnt in contours:
135:        x1, y1 = cnt[0][0]
136:        approx = cv2.approxPolyDP(cnt, 0.01 * cv2.arcLength(cnt,
        True), True)
137:
138:        if len(approx) == 4:
139:            approximated_rectangles.append(approx)
140:            x, y, w, h = cv2.boundingRect(cnt)
141:            ratio = float(w) / h
142:
143:            if ratio >= 0.9 and ratio <= 1.1:
144:                if w >= 200 and h >= 200:
145:                    coord.append((x,y))
146:                    coord.append((x+w,y))
147:                    coord.append((x+w, y+h))
148                     coord.append((x,y+h)))
        . . .
173:    min_x=min(sublist[0] for sublist in coord)
174:    max_x=max(sublist[0] for sublist in coord)
175:    min_y=min(sublist[1] for sublist in coord)
176:    max_y=max(sublist[1] for sublist in coord)
        . . .
183:    edge_points=[(min_x,min_y), (min_x, max_y), (max_x, min_y),
        (max_x, max_y)]

184:    for x,y in edge_points:
185:        cv2.circle(original, (x,y), 20, point_color,-1)
186:        cv2.putText(original, f'({x}, {y})', (x - 50, y - 50),
        cv2.FONT_HERSHEY_SIMPLEX, 1.5, (255, 0, 0), 2)
```

Sl. 4.17. Kod za otkrivanje i crtanje pravokutnika



Sl. 4.18. Slika aproksimiranih rubnih točaka kvadranta

4.9. Lokalizacija kvadranta

Nakon što su pronađene rubne točke kvadranta, može se provesti filtriranje sa svih strana kvadranta. Na svakoj strani kvadranta broji se koliko je manjih pravokutnika jer se veliki pravokutnici nalaze na rubovima hemocitometra i općenito ih ima manje na rubovima. U kodu na slici 4.20. gleda se koje dvije stranice imaju više pravokutnika i prema tome je određen kvadrant. U ovom slučaju, pravokutnika imaju gornja i donja strana kvadrant, pa se radi o 3 kvadrantu.

```
Linija  Kod
        . . .
194:    left_area=0
195:    right_area=0
196:    top_area=0
197:    bottom_area=0
198:    for cnt in contours:
199:        x1, y1 = cnt[0][0]
200:        approx = cv2.approxPolyDP(cnt, 0.01 * cv2.arcLength(cnt,
        True), True)
201:        if len(approx) == 4:
202:            approximated_rectangles.append(approx)
203:            x, y, w, h = cv2.boundingRect(cnt)
204:            ratio = float(w) / h
205:            if ratio >= 0.9 and ratio <= 1.1:
206:                continue
207:            else:
208:                    if (h<300 and w<300):
209:                        if(x1>min_x and x1< max_x):
210:                            if (y1<min_y):
211:                                top_area+=1
212:                            elif(y1>max_y):
213:                                bottom_area+=1
214:
215:                        elif(y1<max_y and y1>min_y):
216:                            if (x1<min_x):
217:                                left_area+=1
218:                            elif(x1>max_x):
219:                                right_area+=1
220:
221:    total_areas=[top_area, bottom_area, left_area, right_area]
```

Sl. 4.19. Kod za otkrivanje i crtanje pravokutnika

```
Linija  Kod
        . . .
223:    labels = ["top", "bottom", "left", "right"]
224:    enumerated_areas = list(zip(total_areas, labels))
225:
226:    # Sort the enumerated areas by area value in descending order
227:    sorted_areas = sorted(enumerated_areas, key=lambda x: -x[0])
228:
229:    # Print the sorted areas with labels
230:    for area, label in sorted_areas:
231:        print(f"{label}_area: {area}")
232:
233:    if sorted_areas[0][1] == "top" and sorted_areas[1][1] ==
        "left":
234:        print("4")
235:    elif sorted_areas[0][1] == "bottom" and sorted_areas[1][1] ==
        "right":
236:        print("1")
237:    elif sorted_areas[0][1] == "bottom" and sorted_areas[1][1] ==
        "left":
238:        print("2")
239:    elif sorted_areas[0][1] == "top" and sorted_areas[1][1] ==
        "right":
240:        print("3")
```

Sl. 4.20. Kod za otkrivanje i crtanje pravokutnika

4.10. Testiranje razvijene metode

Za testiranje razvijene metode odabrano je sto slika na kojima je jedan od četiri kutna kvadranta ili središnji kvadrant, za svaku opciju je odabrano dvadeset slika. Sve slike su testirane bez normalizacije. Od odabranih slika središnjeg kvadranta, uspješno su lokalizirane sve slike. Ovaj rezultat se mogao predvidjeti jer ukoliko nije pronađen kvadrat veće veličine, automatski se pretpostavlja da je riječ o središnjem kvadrantu. Rezultati testiranja na slika kutnih kvadranata redom su: 10 od 20 za prvi kvadrant, 10 od 20 za drugi kvadrant, 7 od 20 za treći kvadrant i 1 od 20 za četvrti kvadrant. Detaljni rezultati nalaze se u tablici 4.21. gdje redovi predstavljaju pet različitih opcija, a stupci rezultate primjene metode s time da su u zadnjem stupcu slike koje metoda nije mogla smjestiti niti u jednu od pet opcija. Zbrajanjem ovih vrijednosti izračunata je prosječna točnost od 48%, ali koja ne predstavlja na dobar način uspješnost metode. Stvarnost je da za sliku prvog ili drugog kvadranata metoda će biti uspješna u 50% slučajeva, za sliku treće

kvadranta bit će uspješna u 35% slučajeva i za sliku četvrtog kvadranta će biti uspješna u 5% slučajeva.

	Središnji kvadrant	***1. kvadrant***	***2. kvadrant***	***3. kvadrant***	***4. kvadrant***	
Središnji kvadrant	***20***	***0***	***0***	***0***	***0***	***0***
1. kvadrant	***3***	***10***	***0***	***1***	***0***	***6***
2. kvadrant	***4***	***0***	***10***	***0***	***0***	***6***
3. kvadrant	***3***	***1***	***0***	***7***	***0***	***9***
4. kvadrant	***5***	***1***	***1***	***0***	***1***	***12***

Sl. 4.21. Tablica s rezultatima testiranja metode

Najčešći razlog zbog kojega kvadrant nije točno lokaliziran je neujednačeno osvjetljenje na slici, koje se može za svaku sliku podesiti ručno. Još jedan razlog neuspješne lokalizacije su mutne linije hemocitometra koje nisu dovoljno oštre da metoda točno odredi kvadrate na slici. U puno slučajeva slike koje nisu točno lokalizirane imaju bijele ili prozirne linije hemocitometra koje je teže izdvojiti od pozadine. Stanice na slici također su bile velika prepreka za lokalizaciju, pogotovo za slike na kojima ih ima puno. Prilikom određivanja linija, algoritam je bio neuspješan jer stanice sijeku linije hemocitometra pa nisu cijele određene ili algoritam gleda stanicu kao liniju i time dodaje šum koji smeta algoritmu za određivanje kontura. Većina ovih problema proizlazi iz činjenice da je teško postići idealne uvjete pri fotografiranju pa se slike uvelike razlikuju jedna od druge i ne mogu se obrađivati i analizirati na isti način.

U budućem radu, valjalo bi prije svega grupirati slike prema sličnim značajkama. To znači grupirati sve slike istih boja pozadina, podijeliti ih prema definiranosti linija na slici (jesu li tamne ili svijetle, prozirne ili ne) i oštrini detalja. Na taj način, sve slike imat će podjednake uvjete pri pred-obradi pa se mogu koristiti jednaki parametri pri normalizaciji, otklanjanju šuma i primjeni metode praga. Također bi bilo dobro odijeliti slike s puno stanica od onih s malo stanica. Slike s malo stanica ne remete u toliko velikoj mjeri otkrivanje linija kao one sa puno stanica. Za takve slike bilo bi moguće razviti posebnu metodu koja otkriva stanice i otklanja ih sa binarne slike. Ovim pristupom točnost metode bi se značajno povećala.

5. ZAKLJUČAK

Svrha ovog završnog rada bila je razviti metodu lokalizacije vidnog polja na slikama s mikroskopa. Prvo je bilo potrebno opisati pojam hemocitometra i na koji se način koristi za brojanje stanica. Ljudima je jednostavno raspoznati detalje u slici, o kojem se kvadrantu hemocitometra radi i prebrojiti koliko u njemu ima stanica. Problematika ovog rada bila je kako računalo može „promatrati“ sliku na sličan način kao ljudi i analizirati što gledamo, tj. koji se kvadrant hamocitometra nalazi na slici. Ovo problematikom bavi se računalni vid.

Sama slika sastoji se od redova i stupaca piksela koje možemo mijenjati i manipulirati kako bismo dobili idealne uvjete za analizu. Postoji nekoliko koraka koje možemo napraviti za što bolju analizu slike. Sliku je potrebno prvo normalizirati, tj. izjednačiti svijetle i tamne tonove kako bi se što bolje vidjeli detalji na slici. Nakon normalizacije, ali i svaka slika prirodno ima šum koji nastaje jer uvjeti za vrijeme fotografiranja nisu idealni. Za uklanjanje šuma postoje različite metode zaglađivanja koje uklanjaju šum prema prosjeku vrijednosti piksela ili prema nekoj statističkoj razdiobi brojeva. Nakon što je šum uklonjen, cilj nam je odvojiti linije hemocitometra u prvi plan, a sve ostalo u pozadinu. To se može postići primjenom praga na sliku tj. odabrati vrijednost praga ili koristiti funkciju s automatskim određivanjem praga i odvojiti sliku na dvije vrijednosti: pikseli kojima je vrijednost niža od praga i oni kojima je vrijednost viša od praga. Kada niže vrijednosti postavimo na nulu i više vrijednosti na maksimalnu vrijednost, dobivamo binarnu sliku na kojoj su prednji plan i pozadina jasno odijeljeni. U prednjem planu slike nalaze se linije kojima je potrebno odrediti gdje se nalaze rubovi. To se postiže uspoređivanjem intenziteta piksela, a kada dođe do nagle promjene, velika je vjerojatnost da se na tom mjestu nalazi rub linije. Kada računalo uspješno otkrije rubove objekata, te rubove koristimo kako bismo primijenili algoritam za procjenu linija na slici. Dobivenu procjenu linija koristimo za otkrivanje kontura tj. oblika na slici. Svaki pravokutnik na slici hemocitometra predstavlja jednu kontura. Algoritam za procjenu oblika smanjuje broj točaka kojima se može prikazati kontura dok ne dođe do najjednostavnijih oblika, u ovom slučaju kvadrata i pravokutnika. Kod lokalizacije vidnog polja potrebno je analizirati gdje se na slici nalaze specifični pravokutnici jer nam oni otkrivaju koji kvadrant hemocitometra se nalazi na slici.

LITERATURA

[1] A. Vembadi, A. Menachery, i M. A. Qasaimeh, „Cell Cytometry: Review and Perspective on Biotechnological Advances“, *Front. Bioeng. Biotechnol.*, sv. 7, 2019, Pristupljeno: 15. rujan 2023. [Na internetu]. Dostupno na: https://www.frontiersin.org/articles/10.3389/fbioe.2019.00147

[2] F. Eren *i ostali*, „DeepCAN: A Modular Deep Learning System for Automated Cell Counting and Viability Analysis“, *IEEE J. Biomed. Health Inform.*, sv. 26, izd. 11, str. 5575–5583, stu. 2022, doi: 10.1109/JBHI.2022.3203893.

[3] J. O'Brien, H. Hayder, i C. Peng, „Automated Quantification and Analysis of Cell Counting Procedures Using ImageJ Plugins“, *J. Vis. Exp. JoVE*, izd. 117, str. 54719, stu. 2016, doi: 10.3791/54719.

[4] F. Lavitt, D. J. Rijlaarsdam, D. van der Linden, E. Weglarz-Tomczak, i J. M. Tomczak, „Deep Learning and Transfer Learning for Automatic Cell Counting in Microscope Images of Human Cancer Cell Lines“, *Appl. Sci.*, sv. 11, izd. 11, Art. izd. 11, sij. 2021, doi: 10.3390/app11114912.

[5] flip.hr, „šum slike | Struna | Hrvatsko strukovno nazivlje“. http://struna.ihjj.hr/naziv/sum-slike/44915/ (pristupljeno 26. lipanj 2023.).

[6] „Gaussian Noise“, *Hasty.ai*. https://hasty.ai/docs/mp-wiki/augmentations/gaussian-noise (pristupljeno 27. lipanj 2023.).

[7] K. Dawson-Howe, *A Practical Introduction to Computer Vision with OpenCV, Enhanced Edition*. Wiley, 2014.

[8] „Image Denoising — OpenCV-Python Tutorials beta documentation“. https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_photo/py_non_local_means/py_non_local_means.html (pristupljeno 27. lipanj 2023.).

[9] S. Sureshan, „Image Segmentation Algorithms With Implementation in Python - An Intuitive Guide“, *Analytics Vidhya*, 10. rujan 2021. https://www.analyticsvidhya.com/blog/2021/09/image-segmentation-algorithms-with-implementation-in-python/ (pristupljeno 27. lipanj 2023.).

[10] „OpenCV: Image Thresholding“. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html (pristupljeno 28. lipanj 2023.).

[11] Z. Xu, X. Baojie, i W. Guoxin, „Canny edge detection based on Open CV“, u *2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, lis. 2017, str. 53–56. doi: 10.1109/ICEMI.2017.8265710.

[12] „About Python“. https://pythoninstitute.org/about-python (pristupljeno 27. lipanj 2023.).

[13] „Image Denoising and various image processing techniques for it“. https://iq.opengenus.org/image-denoising-and-image-processing-techniques/ (pristupljeno 26. lipanj 2023.).

[14] „OpenCV: Miscellaneous Image Transformations“. https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#gaa9e58d2860d4afa658ef70a9b1115576 (pristupljeno 27. lipanj 2023.).

SAŽETAK

Naslov: Lokalizacija vidnog polja u slikama s mikroskopa

Sažetak: Ideja ovog završnog rada bila je razviti programsko rješenje za lokalizaciju kvadranata na slikama s mikroskopa. Ovaj proces uobičajeno se radi ručno pa u slučaju kada ima puno slika, može potrajati. Metoda je razvijena koristeći programski jezik *Python* i biblioteku za računalni vid OpenCV. Ova biblioteka omogućava pred-obradu slike, segmentaciju, otkrivanje rubova i kontura na slici.

Kako bi se slika pred-obradila za analizu, potrebno ju je normalizirati, ukloniti šum, izdvojiti prednji plan i pozadinu. Nakon što se izdvoji prednji plan slike, primjenjuje se algoritam za pronalaženje linija. Linije na slici definiraju zatvorene oblike koji se detektiraju algoritmom za pronalazak kontura. Krivulje se mogu svesti na minimalan broj točaka, a metoda traži one s 4 točke tj. pravokutnike. Pravokutnici se tada odvajaju od kvadrata koji služe za određivanje lokacije samog kvadranta na slici. Brojanjem kontura na svakoj strani kvadranta, kvadrant se lokalizira s obzirom na dvije strane koje imaju najviše pravokutnika.

Testiranjem metode utvrđeno je da metoda jako ovisi o pred procesiranju i ne može točno odrediti kvadrant ako nisu postignuti traženi uvjeti.

Ključne riječi: lokalizacija, računalni vid, segmentacija slike, otkrivanje rubova

ABSTRACT

Title: Field of vision localization on microscope images

Abstract: The aim of this paper was to develop a software solution for quadrant localization on microscope images. This process is usually done by hand so in case of greater number of images, this process could take a while. The method was developed using Python programming language and computer vision library OpenCV. This library enables image pre-processing, segmentation, edge detection and contour detection.

To prepare the image for analysis, it should be normalized, noise should be removed, foreground and background segmented, and line detection algorithm applied. Lines on the image define closed contours which are detected with a contour detection algorithm. Contours can be reduced to a minimal number of points, and the method searches the ones with four points i.e. rectangles. Rectangles are then separated from squares that are used to determine the location of the quadrant in the image. Counting the number of contours on each side of the quadrant, the quadrant is localized based on two sides which have the most rectangles.

While testing the method, it was determined that it strongly depends on pre-processing and the quadrant can't be localized correctly if the needed conditions aren't met.

Keywords: localization, computer vision, image segmentation, edge detection