SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 810

DETEKCIJA STANIČNE JEZGRE U MIKROSKOPSKIM SLIKAMA

Dražen Dostal

Zagreb, lipanj 2009.

Sadržaj:

1. Uvod

Za razvoj medicine i drugih znanosti je uz ostale faktore zaslužan i razvoj računarske znanosti. Računala su neizostavan dio čak i jednostavnih dijagnoza i proračuna. Ona omogućuju brzu obradu velike količine podataka te generiranje kvantitativnih procjena. Dodatnu pomoć pruža i robotika pogonjena programima za precizno seciranje, pozicioniranje i pomoć pri operacijama na mikroskopskim razinama.

Zadaci koje obavljamo računalima dijelimo na teške i rutinske. Teški su oni zadaci koje ljudi ne mogu obaviti, a rutinski su oni koji zahtijevaju puno vremena. Primjer rutinskog zadatka je ubrizgavanje raznih tvari u jezgru stanice. Taj postupak se obično ponavlja mnogo puta u određenom vremenskom periodu, a jezgra i stanica ne miruju za to vrijeme što predstavlja određene probleme. Kod ovakvih problema ljudima pomažu računala i računalni algoritmi.

Cilj ovog rada je detekcija stanice kvasca na ulaznoj slici te položaja jezgre unutar nje primjenom postupaka obrade slike i računalnog vida. Kvasci su jednostanični eukariotski organizmi koji se zbog svoje jednostavnosti često koriste pri proučavanju procesa u eukariotskim organizmima. U novije vrijeme posebno se izučava uloga mikrotubula kao dijela citoskeleta stanice u procesu centriranja jezgre tijekom interfaze [1]. Kako bi detaljno razumjeli dinamiku odvijanja tog procesa, istraživači izvode eksperimente u kojima se jezgra stanice pomiče iz središta stanice optičkom pincetom [2]. U takvim eksperimentima, dinamika pomaka jezgre stanice prati se kroz slijedove od više stotina mikroskopskih slika. Stoga je posebno pogodno položaj jezgre određivati automatski, računalnim vidom, primjenom prikladnih metoda detekcije objekata.

Ideja je da se iz video izvora ili direktno sa mikroskopa na obradu šalje niz slika. Na slikama se zatim raznim algoritmima detektira položaj i granice stanice, te konačno položaj i granice jezgre unutar te stanice. Konačni rezultat je slika na kojoj se vidi detektirana stanica oko koje je opisan lokacijski pravokutnik te locirana jezgra oko koje je opisan lokacijski krug.

U radu se koriste slijedeći algoritmi obrade slike: binarizacija, morfološke operacije skupljanja i širenja, glađenje. Uz te algoritme koristimo i postupke računalnog vida kao što su određivanje orijentacije segmentiranog objekta te detekcija citoplazme i lokalizacija jezgre. Konačni postupak detekcije možemo podijeliti u četiri koraka koji se oslanjaju na gore spomenute algoritme:

1) Pretprocesiranje (obrada slike)
2) Detekcija citoplazme
3) Određivanje orijentacije segmentiranog objekta
4) Lokalizacija jezgre

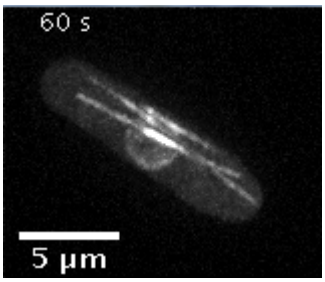
Rad je strukturiran na slijedeći način. U poglavlju 2 opisane su korišteni algoritmi i metode kojima se detektiraju stanica i jezgra. U poglavlju 3 opisana je programska izvedba algoritma. Eksperimentalni rezultati prikazani su u poglavlju 4. Konačno, u poglavlju 5 dan je završni komentar u kojem se ocjenjuje uspješnost algoritma te ističu njegove prednosti i nedostaci.

2. Korišteni algoritmi i metode

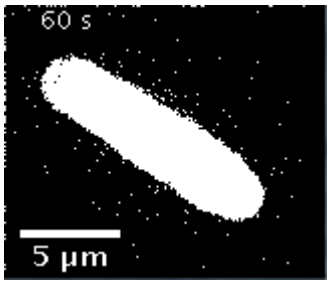
2.1. Pomoćni algoritmi

2.1.1. Binarizacija

U metodi binarizacije se logičke vrijednosti svakog piksela izlazne slike određuju u ovisnosti o odgovarajućem pikselu (elementu) ulazne slike (slika 2.1). Time dobivamo binarnu sliku na kojoj su pikseli ili bijeli ili crni (slika 2.2), nema razina sive boje. Za detaljniji opis pogledati [1,2].



2.1 Izvorna slika

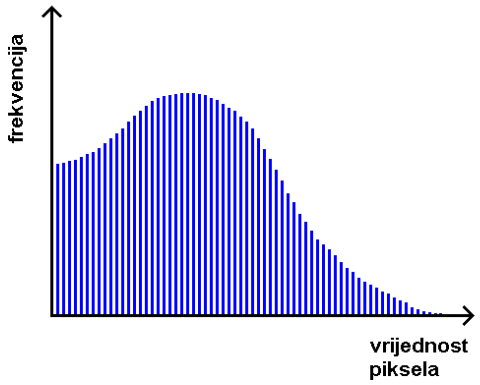


2.2 Slika nakon binarizacije

Postupak se provodi primjenom postupka usporedbe svakog piksela slike s pragom. Pretpostavimo da imamo crno-bijelu sliku s 256 razina sive boje. Ako odaberemo prag od 78, to znači da će nam svi pikseli sa iznosima manjim od praga postati crni pikseli (pikseli pozadine), a svi pikseli sa većim iznosima od praga bijeli pikseli (pikseli objekta).

Iznos praga se u našem slučaju računa iz ulaznog parametra algoritma. Taj parametar predstavlja relativnu granicu, pri čemu zadani udio govori koliko piksela od njihovog ukupnog broja zanemarujemo pri obradu (npr. ako je zadana granica 90%, to znači da uzimamo samo 10% piksela s najvećim vrijednostima).

Znači, potrebno je ispitati koje se sve sive razine piksela pojavljuju na slici i koliko ima piksela svake od tih vrijednosti, odnosno treba napraviti razdiobu frekvencije pojavljivanja piksela u ovisnosti o njihovoj diskretnoj vrijednosti. Takva razdioba se u literaturi naziva histogram slike.



2.3 Primjer histograma

Nakon što napravimo histogram, zbrajamo frekvencije pojavljivanja piksela sa rastućim vrijednostima (od 0 do 255) dok ne nakupimo zadani postotak broja piksela. Tako dobivamo apsolutni prag u obliku cjelobrojne vrijednosti iz intervala [0,255].

Zatim jednostavno prođemo svim pikselima slike te ako je piksel veći od praga postavimo ga na logičku jedinicu, ili ako je manji na logičku nulu, te ga pozicioniramo na isto mjesto na novoj slici („kopiramo“).

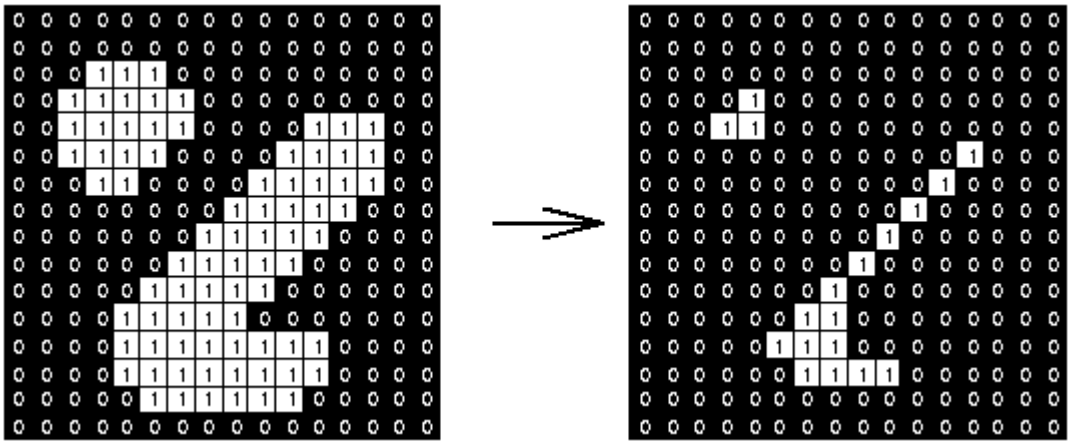
Osim ove metode usporedbe s pragom, isprobana je i Otsu-ova metoda binarizacije [9]. Ona je pokazala lošije rezultate od već spomenutog postupka, te je zbog toga odbačena.

2.1.2. Skupljanje

Sama binarizacija nije dostatna za izdvajanje stanice a pogotovo jezgre, uslijed šuma ili dijelova citoskeleta stanice (mikrotubuli). Ovdje se zato koristimo postupkom skupljanja koji u osnovi stanjuje granice objekta, tj. nagriza objekt (odatle i drugi naziv, erozija).

Postupak se provodi upotrebom filtra određene veličine. Tim filtrom prolazimo svim pikselima ulazne slike, pri čemu je promatrani piksel u središtu filtra. Na temelju susjedstva središnjeg piksela određujemo odgovarajući piksel izlazne slike. Dovoljno je da samo jedan piksel unutar granica filtra bude postavljen u logičku nulu kako bi i centralni piksel izlazne slike postavili u nulu.

Veličina filtra se zadaje kao ulazni parametar postupka. Kao primjer uzmimo veličinu filtra 3 (matrica 3x3). Prolazimo svim pikselima slike i svakome gledamo njegovo susjedstvo (piksele udaljene od njega za 1 piksel, tj. direktno susjedne). Ako je bilo koji piksel iz njegova susjedstva (u krajnjem slučaju i sam on) postavljen u nulu onda i centralni na izlaznoj slici postavljamo u nulu (slika 2.4).



2.4 Primjer skupljanja sa filtrom veličine 3

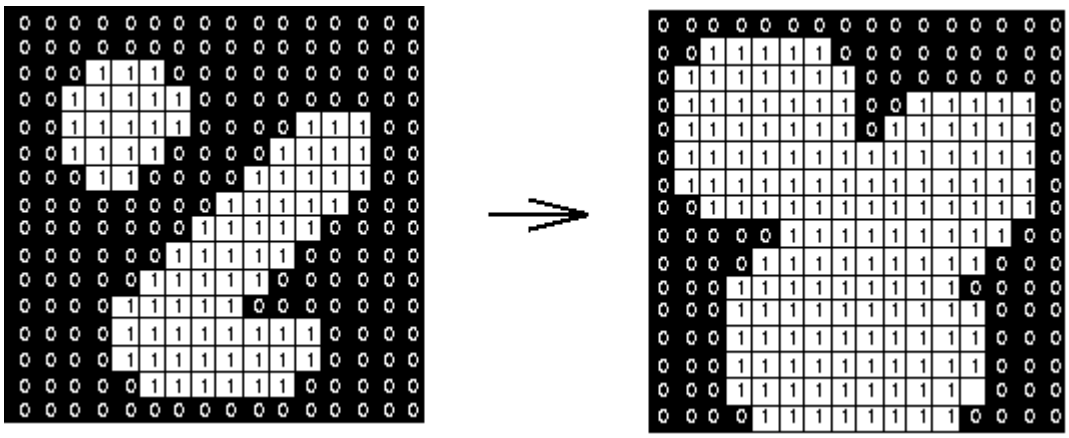
Dodatna pojašnjenja ovog postupka su dostupna u literaturi [5].

2.1.3. Širenje

Nakon skupljanja smo uklonili šum i neželjene objekte u binariziranoj slici ali smo istovremeno smanjili promatrani objekt, te raširili moguće rupe u njemu. Time promatrani objekt gubi na prepoznatljivosti. Kako bi popunili te rupe i vratili prvotnu veličinu objekta (ali ovaj put bez neželjenih objekata), uvodimo postupak širenja koji je u izvedbi dosta sličan skupljanju samo ima obrnutu logiku.

Postupak se ponovno provodi upotrebom filtra određene veličine. Tim filtrom prolazimo svim pikselima ulazne slike. Na temelju susjedstva središnjeg piksela određujemo odgovarajući piksel izlazne slike. Dovoljno je da samo jedan piksel unutar granica filtra bude postavljen u logičku jedinicu kako bi i centralni piksel izlazne slike postavili u jedinicu.

Veličina filtra se isto tako zadaje kao ulazni parametar postupka. Kao primjer uzmimo veličinu filtra 3. Prolazimo svim pikselima slike i svakome gledamo njegovo susjedstvo (direktno susjedne). Ako bilo koji piksel iz njegova susjedstva postavljen u logičku jedinicu onda i centralni na izlaznoj slici postavljamo u jedinicu (slika 2.5).



2.5 Primjer širenja sa filtrom veličine 3

Ponovno su dodatna pojašnjenja dostupna u literaturi [5].

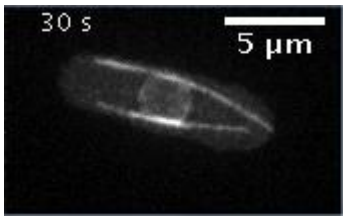
2.1.4. Glađenje

Glađenje je osnovni postupak pripreme slike za daljnju obradu. To je postupak filtriranja slike koji kao rezultat, u odnosu na originalnu sliku, daje zamagljenu sliku smanjene detaljnosti [4,5]. Glađenjem slike [6] rješavamo se štetnog šuma koji je sastavni dio slike. Šum se uglavnom sastoji od visokih frekvencija pa ga možemo ukloniti nekom vrstom linearnog niskopropusnog filtra. Ovdje koristimo diskretno filtriranje, pri čemu na ulaznu funkciju (y) primjenjujemo filtar (h) te dobivamo novu izlaznu funkciju (g). Ovdje koristimo prednost separabilnog filtra, tj. svojstvo da se dvodimenzionalni separabilni filtar može predstaviti sa dva jednodimenzionalna filtra:

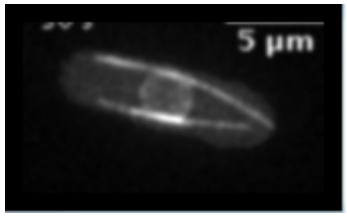
$$g(x) = \sum_{N=-\infty}^{+\infty} f(x-N)h(N)$$

Separabilni filtar koji smo koristili u ovom radu je Gaussov filtar. On ovisi o parametru sigma, tj. standardnoj devijaciji Gaussove funkcije (rasprostranjenost te funkcije). Ovdje se direktno koristi algoritam sa projekta „Cannyjev detektor rubova“. Više pojašnjenja potražite u dokumentaciji tog projekta [4] i u [7].

Ovaj postupak se koristi kako bi se zagladila slika prije binarizacije prilikom lokalizacije jezgre. Primjer izvorne i zaglađene slike se vidi na slikama 2.6 i 2.7.



2.6 Izvorna slika



2.7 Zaglađena slika, za sigma=3

2.1.5. Određivanje orijentacije segmentiranog objekta

Ovaj algoritam koristimo za određivanje orijentacije binarnog objekta. Algoritam se zasniva na analizi glavnih komponenti (engl. Principal component analysis, PCA) skupa točaka binarnog objekta [8]. Postupak je opisan u nastavku.

Prvo se traži središte skupa piksela, tj. središte objekta. Njega dobivamo računanjem srednje vrijednosti koordinata svih piksela koji su postavljeni u logičku jedinicu, tj. prema jednadžbi:

$$m_j = \frac{1}{\mathrm{N}} \sum_{i=0}^{N-1} \left(X_{j,i} \right), \ \mathrm{j} \in \{\mathrm{x}, \mathrm{y}\} \tag{1}$$

U jednadžbi (1) $X_{j,i}$ predstavlja trenutni piksel, tj. njegove koordinate ($X_{x,i}, X_{y,i}$), dok N predstavlja ukupni broj piksela na slici. Time dobivamo koordinate središta nakupine piksela stanice (m_x, m_y).

Zatim računamo smjer pružanja najvećeg odstupanja vrijednosti koordinata nakupine piksela. Time dobivamo dva okomita vektora koji predstavljaju smjer pružanja stanice po duljini i širini.

Traženje svojstvenih smjerova nakupine se sastoji od više koraka. U prvom koraku se računaju odstupanja koordinata svakog piksela od središta. Ta odstupanja spremamo u matricu B (dimenzija 2, N) pri čemu prvi redak odgovara odstupanju apscise, a drugi odstupanju ordinate:

$$B = \begin{bmatrix} X_{x,0} - m_x \ \ldots & X_{x,N-1} - m_x \\ X_{y,0} - m_y \ \ldots & X_{y,N-1} - m_y \end{bmatrix} \tag{2}$$

Dalje slijedi izračun matrice kovarijance C, koristeći podatke o odstupanju koordinata piksela objekta od središta skupa piksela. Nju računamo kao umnožak matrice B i transponirane matrice B. Matrica kovarijance nam daje podatke o raspršenju piksela objekta i računa se po formuli:

$$C = \frac{1}{N} B * B^T = \frac{1}{N} \begin{bmatrix} \sum_{i=0}^{N-1} (X_{x,i})^2 & \sum_{i=0}^{N-1} (X_{x,i} * X_{y,i}) \\ \sum_{i=0}^{N-1} (X_{y,i} * X_{x,i}) & \sum_{i=0}^{N-1} (X_{y,i})^2 \end{bmatrix} \quad (3)$$

Slijedeći korak je računanje svojstvenih vrijednosti λ iz matrice C. Slijedi izvod tog računa u obliku slijednih formula (radi jednostavnosti vrijednosti matrice C predstavljamo slovima, formula 4):

$$C = \begin{bmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{b} & \mathrm{c} \end{bmatrix} \quad (4)$$

$$C * x = \lambda * \mathrm{x} \quad (5)$$

$$(C - \lambda * \mathrm{I}) * \mathrm{x} = 0 \implies \det(C - \lambda * \mathrm{I}) = 0 \quad (6)$$

$$\lambda^2 - \lambda * (\mathrm{a} + \mathrm{c}) + \mathrm{a} * \mathrm{c} - \mathrm{b}^2 = 0 \quad (7)$$

Rješavanjem kvadratne jednadžbe (7) dobivamo dvije svojstvene vrijednosti (λ_1, λ_2). Iz njih dalje računamo svojstvene vektore (v_1, v_2). Ponovno kratki izvod prvog svojstvenog vektora:

$$v_1 = \begin{bmatrix} \mathrm{v}_{1\mathrm{x}} \\ \mathrm{v}_{1\mathrm{y}} \end{bmatrix} \quad (8)$$

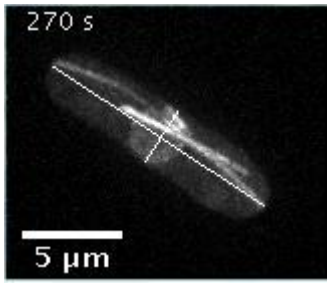
$$\left(C - \lambda_1 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) * v_1 = \begin{bmatrix} a - \lambda_1 & b \\ c & d - \lambda_1 \end{bmatrix} * \begin{bmatrix} v_{1x} \\ v_{1y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (9)$$

$$v_{1x}(a - \lambda_1) + v_{1y} * b = 0,\ v_{1x} * c + v_{1y} * (d - \lambda_1) = 0 \quad (10)$$

$$v_1 = \begin{bmatrix} v_{1x} \\ v_{1y} \end{bmatrix} = \begin{bmatrix} v_{1x} \\ v_{1x} * \frac{b}{\lambda_1 - a} \end{bmatrix} \quad (11)$$

Iz rješenja jednadžbi (11) uzimamo normirani vektor ($|\tilde{v}_1| = \frac{v_1}{\|v_1\|}$). Identičnim postupkom sa parametrom λ_2 računamo i drugi vektor.

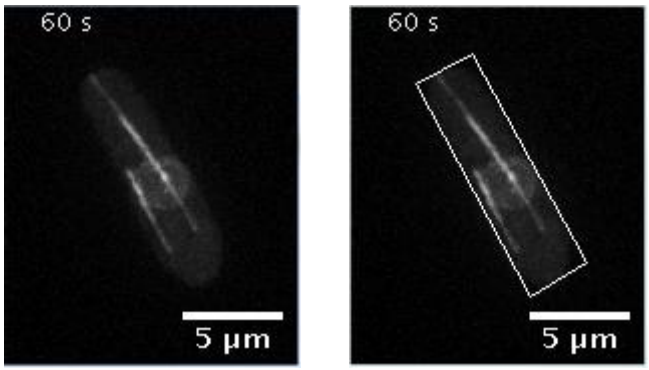
Svojstveni vektori odgovaraju pravcima pružanja stanice kao što je zorno prikazano na slici 2.13.



2.13 Pravci pružanja stanice

2.2. Detekcija citoplazme

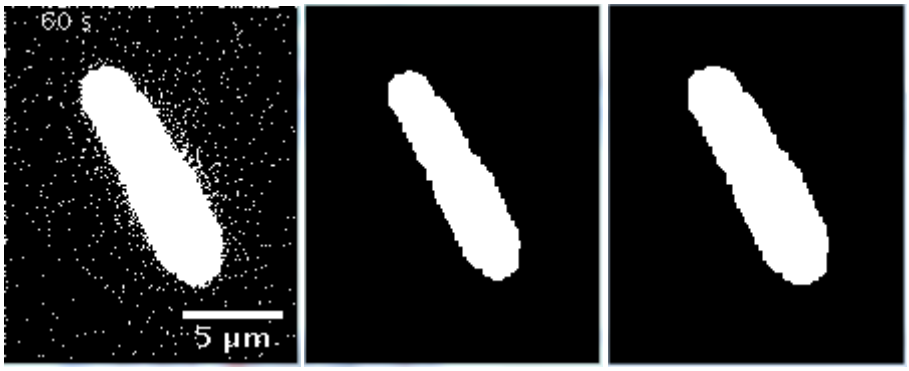
Cilj ove faze je pronalaženje položaja cijele stanice te granica stanice. Konačni rezultat je opisani pravokutnik oko stanice koji obuhvaća cijelu citoplazmu (slika 2.8).



2.8 Izvorna slika i ciljna slika

Već se prva razvijena metoda detekcije citoplazme pokazala uspješnom. To je radi toga što je citoplazma relativno istaknuta u odnosu na pozadinu te je stoga lagano već samom binarizacijom s nižim pragom utvrditi njen položaj. Kako bi se dobio što bolji rezultat još se primjenjuje i blago skupljanje te zatim širenje s istom veličinom filtra.

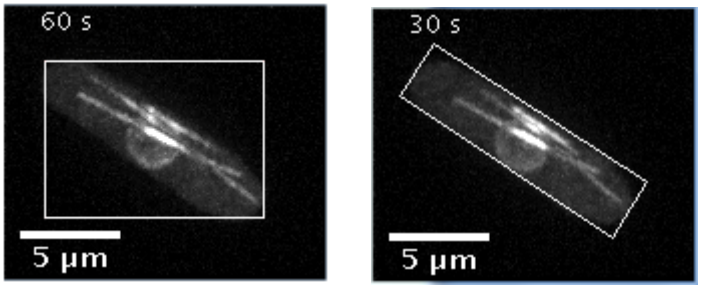
Dakle algoritam nad izvornom (crno-bijelom slikom) provodi već objašnjeni postupak binarizacije i to sa nižim relativnim pragom. Izvorna slika je prikazana na slici 2.8 a binarizirana sa relativnom granicom od 80% na slici 2.9 . Nakon binarizacije slijedi blago (npr. sa veličinom filtra 3) skupljanje koje uklanja šum i neželjene objekte na slici (slika 2.9 sredina). Skupljanjem smo smanjili pravu veličinu objekta. Kako bi to popravili provodimo širenje sa istom veličinom filtra (slika 2.9 desno). Ovako smo uklonili šum ali nismo u velikoj mjeri narušili prvotni izgled stanice.



2.9 slika nakon binarizacije (lijevo), nakon skupljanja (sredina), nakon širenja (desno)

Nakon toga je potrebno opisati pravokutnik oko citoplazme na temelju do sada obrađene slike. Jednostavan način je da se prođemo cijelom slikom te zapamtimo koordinate krajnje lijevog, krajnje desnog, najvišeg te najnižeg piksela u logičkoj jedinici. Zatim iz tih koordinata iscrtamo pravokutnik. Tako je bilo u prvoj izvedbi, druga izvedba je složenija i preciznija. Ona koristi postupak detekcije orijentacije stanice opisan u 2.1.5, te u skladu s tim orijentira i opisani pravokutnik. Jednostavno na temelju pravaca koji predstavljaju uzdužni i poprečni smjer pružanja stanice, pronalazimo granice stanice po duljini i širini u odnosu na središte. Tako dobivamo četiri točke na temelju kojih je dalje trivijalno opisati pravokutnik.

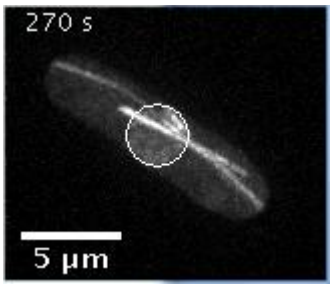
Usporedba te dvije izvedbe dana je na slici 2.10.



2.10 Prva (lijevo) i druga (desno) izvedba opisanog pravokutnika

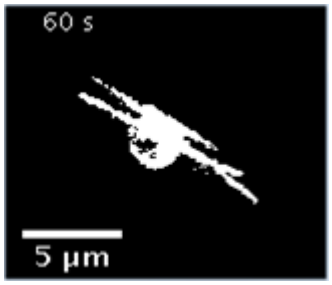
2.3. Lokalizacija jezgre

Ova faza je najzahtjevniji dio algoritma. Cilj nam je sličan detekciji citoplazme, ali postupak je mnogo složeniji. Krajnji rezultat bi bio lokalizirana jezgra predstavljena opisanim pravokutnikom ili krugom oko cijele površine koja predstavlja jezgru. Kako bismo uspjeli dobiti dobre rezultate u ovoj fazi se pozivaju sve gore spomenute metode.



2.11 Konačni cilj, lokalizirana jezgra

Najveće probleme su predstavljali mikrotubuli unutar stanice i zatamnjenje u sredini jezgre. Mikrotubuli su smetali jer su dosta svjetlije od ostatka stanice pa su nakon binarizacije ostale vidljive. Zatamnjenje u jezgri je zapravo tamnije područje u sredini jezgre koje nastaje radi sfernog oblika jezgrine ovojnice. Jezgra u sredini ima najveći odsječak kroz koji prolazi svijetlost i ovojnice su udaljenije što vodi do tamnije površine na slikama, tj. što su jezgrine ovojnice bliže to bolje reflektiraju svjetlost, radi toga su rubovi svjetliji, a sredina tamnija. To tamnije područje je nakon binarizacije imalo nekoliko crnih piksela koji bi se postupcima skupljanja i širenja samo još više naglasili, što je izazvalo loše lociranje. Oba problema se vide na slici 2.12.



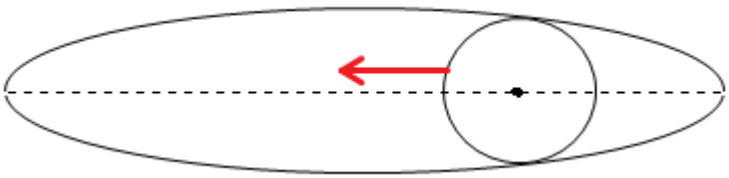
2.12 Primjer zatamnjenja (“rupe”) i mikrotubuli (niti) nakon binarizacije

Ovaj dio algoritma prošao je kroz više verzija. Prvi, ujedno i najjednostavniji postupak se temeljio samo na uporabi binarizacije, skupljanju i širenju. Ulaznu sliku smo prvo binarizirali sa višim pragom (npr. relativni prag od 93.5%) čime bi dobili istaknutu jezgru ali isto tako i spomenuti mikrotubuli i „rupu“ u sredini jezgre. Kako bi uklonili šum i mikrotubule, provodimo skupljanje. Uz

odgovarajuću veličinu filtra, ono nam uklanja veći dio mikrotubula ali isto tako i povećava „rupu“. Na kraju provodimo širenje kako bismo pokušali vratiti pravi izgled jezgre, ali to nije bilo najuspješnije. Na kraju dolazi jednostavno opisivanje pravokutnika oko jezgre koje je u ovom stadiju razvoja algoritma ostvareno kao i prva izvedba kod detekcije stanice što je objašnjeno u poglavlju 2.5. Obrtanje redoslijeda širenja i skupljanja je bio slijedeći korak, ali on nije dao ni malo bolje rezultate.

Drugi postupak temelji se na detekciji orijentacije stanice postupkom PCA koji je objašnjen u poglavlju 2.1.5.

Na obradu šaljemo binariziranu sliku, a dobivamo dva pravca koji predstavljaju uzdužni i poprečni smjer pružanja stanice. Na temelju njih se traži maksimalna kružna nakupina piksela, krećući se po dužem pravcu (pravac koji predstavlja uzdužni smjer pružanja stanice, slika 2.13). Pretpostavka je da je upravo duljina kraćeg pravca (pravac koji predstavlja poprečni smjer pružanja stanice) promjer jezgre. Jednostavno se krećemo po dužem pravcu i zbrajamo sve piksele koji su unutar radijusa na ulaznoj slici. Pozicija maksimalne vrijednosti predstavlja središte jezgre.



2.13 Traženje jezgre

Na kraju je još samo potrebno opisati jezgru pravokutnikom ili kružnicom. U prvoj izvedbi je to bio pravokutnik a kasnije kružnica. Konačni rezultat prikazan je na slici 2.11.

Ovaj postupak je dao bolje rezultate, ali nakon više eksperimentiranja dobili su se bolji rezultati uvođenjem dodatnih postupaka. Ulazna slika je prvo zaglađena pa binarizirana. Tako binariziranu sliku zatim skupljamo i širimo te tek na kraju tražimo maksimalnu kružnu nakupinu piksela.

3. Programska implementacija

3.1. Struktura programske implementacije

U programskoj implementaciji algoritma korištena je ljuska `cvsh` (computer vision shell). Ljuska je razvijena kako bi se olakšalo i pojednostavilo eksperimentiranje s algoritmima računalnog vida u programskom jeziku C++. Za dodatne informacije i detalje o korištenju ljuske konzultirati tehničku dokumentaciju projekta koji je prethodio ovom radu (Cannyjev detektor rubova) [4].

Kôd algoritma je smješten u jedanaest komponenti. Dodatno se pozivaju još dvije komponente koje su kreirane u sklopu spomenutog projekta.

`alg_NucleusDetection.cpp` je glavna komponenta koja izvršava algoritam lokalizacije stanice i jezgre. Ona sadrži klasu `alg_NucleusDetection` sa metodom `alg_nucleusDetection::process()` preko koje se ostvaruje povezivanje ljuske sa algoritmom jer se `process()` poziva svaki put kada korisnik inicira obradu slike algoritmom.

```
void alg_nucleusDetection::process (
  const img_vectorAbstract& src,
  const win_event_vectorAbstract&,
  int)
```

Datoteka `alg_NucleusDetection.cpp` izvodi algoritam lokalizacije jezgre i detekcije stanice tako što poziva pojedine funkcije smještene u komponentama:

1) `alg_NucleusDetection_binarization.cpp`
2) `alg_NucleusDetection_erosion.cpp`
3) `alg_NucleusDetection_dilation.cpp`

4) alg_Canny251_gladjenje.cpp

5) alg_NucleusDetection_locateNucleus.cpp

6) alg_NucleusDetection_drawRectangle

Funkcije su redom:

```
void binarization(
  double thRel,
  const img_wrap& imgGray,
  img_wrap& imgBinarized);

void erosion(
  int szKernelErosion,
  const img_wrap& imgInput,
  img_wrap& imgEroded);

void dilation(
  int szKernelDilation,
  const img_wrap& imgInput,
  img_wrap& imgDilated);

int filterGaussSeparatedDouble(
  double sigma,
  const img_wrap& imgGray,
  img_wrap& imgFirstPassSmooth,
  img_wrap& imgSmooth);

void locateNucleus(
  const img_wrap& imgCytoDilated,
  const img_wrap& imgGray,
  pixel& mean,
  int *radius,
  pixel& center,
  pixel& lengthVectorMax,
  pixel& lengthVectorMin,
  pixel& widthVectorMax,
  pixel& widthVectorMin);

void drawRectangle(
  win_ann& annMaska,
  const pixel& center,
  const pixel& lengthVectorMax,
  const pixel& lengthVectorMin,
  const pixel& widthVectorMax,
  const pixel& widthVectorMin);
```

Svaka funkcija predstavlja jedan algoritam pri pronalasku jezgre i stanice. Neki od tih postupaka se pozivaju više puta tokom obrade. Imena funkcija i datoteka opisuju što zapravo one rade. To je redom binarizacija, skupljanje (`erosion()`), širenje (`dilation()`), glađenje (`filterGaussSeparatedDouble()`),

lociranje jezgre (`locateNucleus()`), crtanje opisnog pravokutnika (`drawRectangle()`).

Kombiniranjem poziva tih funkcija ostvarujemo određivanje i prikazivanje položaja stanice i jezgre na slici.

Svaka od ovih šest datoteka ima pripadajuću datoteku s nastavkom .hpp u skladu s konvencijom o pravilnom strukturiranju kôda.

Većina ovih funkcija zahtijeva i određene ulazne parametre koji predstavljaju ulazne koeficijente koji utječu na obradu slike. Ti parametri se zadaju preko komandne linije ili direktno iz ljuske (naredba configure „c“). Tablica 3.1 predstavlja popis ulaznih parametara.

Parametar	Opis
`th_RelCyto`	Relativni prag za binarizaciju citoplazme (npr. 0.80).
`sz_KernelCyto`	Veličina filtra za skupljanje i širenje citoplazme (npr. 5).
`sigma`	Ulazni parametar algoritma glađenja, predstavlja točnost aproksimacije Gaussove razdiobe (npr. 3, što predstavlja točnost od 99.73%).
`th_Rel`	Relativni prag za binarizaciju jezgre (npr. 0.935).
`sz_KernelD`	Veličina filtra za širenje jezgre (npr. 13).
`sz_KernelE`	Veličina filtra za skupljanje jezgre (npr. 9).

Tablica 3.1

3.2. Opis programske implementacije

3.2.1. Binarizacija

Cijeli postupak binarizacije se provodi u glavnoj funkciji:

```
void binarization(
  double thRel,
  const img_wrap& imgGray,
  img_wrap& imgBinarized);
```

ona prima relativni prag (`thRel`) iz kojeg računamo apsolutni prag, ulaznu sliku (`imgGray`) koju koristimo kao ulaz u proračune te izlaznu sliku u koju upisujemo binariziranu sliku (`imgBinarized`).

Glavna funkcija poziva pomoćnu funkciju (`izracunGranica()`) koja vraća apsolutni prag. S tim pragom se zatim uspoređuje svaki piksel ulazne slike te ako je on veći od praga na izlaznoj slici se piksel s tim koordinatama postavi kao svijetli, tj. prednji piksel, inače se postavlja kao tamni, tj. piksel pozadine.

```
int izracunGranica(
  double thRel,
  const img_wrap& imgGray);
```

Pomoćna funkcija prima relativni prag i ulaznu sliku te iz njih računa apsolutni prag. To radi tako da se pronađe najsvjetliji piksel te na temelju njega inicijalizira vektor vrijednosti piksela koji predstavlja histogram. Zatim popuni histogram pojavljivanjima piksela određene vrijednosti (0-255). Za konačni izračun apsolutnog praga (kojega ujedno i vraća) poziva drugu pomoćnu funkciju:

```
int findThValFromThSum(
    std::vector<int>& histogram,
    double thSum,
    double thRel);
```

koja prima histogram, ukupan zbroj piksela na slici (`thSum`) te relativni prag. Na temelju tih podataka računa se apsolutni prag. Jednostavno se zbrajaju brojevi pojavljivanja piksela s istom vrijednosti dok se ne pređe zadani postotak ukupnog broja piksela predstavljen sa relativnim pragom.

3.2.2. Skupljanje

Skupljanje je programski dosta jednostavan algoritam, stoga se sav posao odvija u jednoj funkciji:

```
void erosion(
  int szKernelErosion,
  const img_wrap& imgInput,
  img_wrap& imgEroded);
```

Argumenti su veličina filtra (`szKernelErosion`), ulazna slika (`imgInput`) te izlazna slika na koju projiciramo rezultat obrade (`imgEroded`).

Funkcija prolazi svim pikselima i provjerava susjedstvo promatranog piksela i to do udaljenosti od polovine veličine filtra. Ako je bar jedan piksel iz toga susjedstva tamni piksel (vrijednost 0) onda i na izlaznoj slici piksel s istim koordinatama postavi u tamni. Npr. ako je veličina filtra 3 provjeravaju se svi pikseli direktni susjedi promatranog piksela (i promatrani piksel).

3.2.2. Širenje

Širenje je programski gotovo identično skupljanju, uz malu razliku. Sve se ponovno odvija u jednoj funkciji:

```
void dilation(
  int szKernelDilation,
  const img_wrap& imgInput,
  img_wrap& imgDilated);
```

Dakle, ulazni parametri su veličina filtra (`szKernelDilation`), ulazna slika (`imgInput`) te izlazna slika na koju projiciramo rezultat obrade (`imgDilated`).

Postupak je identičan onome kod skupljanja jedina razlika je što provjeravamo da li je barem jedan piksel iz susjedstva svijetli piksel (vrijednost 255) te ako je onda piksel s istim koordinatama na izlaznoj slici postavljamo u svijetli.

3.2.3. Glađenje

Pošto je ovo postupak koji se poziva iz vanjskih datoteka, tj. iz algoritma projekta koji prethodio ovom radu, za opis programske implementacije pogledati tehničku dokumentaciju projekta „Cannyev detektor rubova“ [4].

3.2.4. Određivanje orijentacije stanice

Orijentacija stanice je određena svojstvenim vektorima. Za izračun tih vektora koristimo funkciju `calculateEigenVector()` koja preko adrese vraća vektore predstavljene pomoću jednadžbe pravca kroz ishodište (Y=aX, pri čemu su `v1y` i `v2y` zapravo koeficijenti a, tj. kut) te središte stanice. Ulazni parametar je zapravo samo ulazna slika. Izračun se odvija onako kako je detaljno objašnjeno u poglavlju 2.1.5.

```
void calculateEigenVector(
    const img_wrap& imgCytoDilated,
    pixel& mean,
    double &v1y,
    double &v2y);
```

3.2.5. Detekcija citoplazme

Ovo je zapravo prva faza algoritma. Koristi gore objašnjene funkcije kako bi lokalizirala cijelu citoplazmu. Pošto je postupak dosta jednostavan, posao se obavlja pozivima funkcija iz glavne `alg_nucleusDetection::process()` metode.

Prvo se provodi binarizacija ulazne slike sa manjim relativnim pragom (podrazumijevana vrijednost je 0.80, tj. 80%). Izlaznu sliku zatim uz veličinu filtra (podrazumijevana vrijednost je 5) predajemo kao parametar postupku skupljanja. Nakon toga ponovno izlaznu sliku i istu veličinu filtra predajemo postupku širenja. Pozivi ovih postupaka se očituju kao pozivi gore spomenutih funkcija binarizacije, skupljanja i širenja.

Takav slijed poziva nam na izlazu daje binariziranu sliku sa dosta točnim oblikom stanice. Ta slika se zatim prosljeđuje postupku lokalizacije jezgre. Nakon izračuna svojstvenih vektora u lokalizaciji jezgre, konačno dobivamo podatke potrebne za opisivanje pravokutnika oko citoplazme što je krajnji korak algoritma.

Ti podaci su predstavljeni sa početnim i krajnjim točkama pravaca pružanja stanice koji su već objašnjeni. Na temelju tih točaka računamo četiri vrha opisanog pravokutnika te ga iscrtavamo. Posao izračuna i iscrtavanja se odvija u funkciji `drawRectangle()`.

```
void drawRectangle(
  win_ann& annMaska,
  const pixel& center,
  const pixel& lengthVectorMax,
  const pixel& lengthVectorMin,
  const pixel& widthVectorMax,
  const pixel& widthVectorMin);
```

Ovdje koristimo još i strukturu `pixel` koja nam predstavlja koordinate piksela.

```
struct pixel{
            int x;
            int y;
      };
```

Ulazni parametri su referenca na anotaciju (`annMaska`) koju lijepimo na konačnu izlaznu sliku. Centar stanice (`center`) koji smo dobili u postupku računanja svojstvenih vektora. Početne i krajnje točke pravaca pružanja stanice.

Koordinate vrhova računamo tako da vrhove pravca pružanja u širinu translatiramo za razliku koordinata početne (i krajnje) točke pravca pružanja u duljinu i koordinata centra. Time dobivamo četiri točke koje jednostavno spojimo s linijama. Sve to spremimo u anotaciju (`annMaska`) koji kasnije projiciramo na sliku konačnog rezultata.

3.2.6. Lokalizacija jezgre

Lokalizacija jezgre je druga faza algoritma. Ova faza je složenija od prve faze jer se pri lociranju jezgre pojavljuju već spomenuti problemi mikrotubula i zatamnjenja središta jezgre, te je potrebno uvesti dodatne postupke za kvalitetnu lokalizaciju.

Prvo se provodi glađenje ulazne slike (podrazumijevana vrijednost `sigma` je 3). Zatim se provodi binarizacija zaglađene slike sa višim relativnim pragom (podrazumijevana vrijednost je 0.935, tj. 93.5%). Izlaznu sliku zatim uz veličinu filtra (podrazumijevana vrijednost je 13) predajemo kao parametar postupku širenja. Nakon toga ponovno izlaznu sliku i manju veličinu filtra (podrazumijevana vrijednost je 9) predajemo postupku skupljanja. Pozivi ovih postupaka se očituju kao pozivi gore spomenutih funkcija binarizacije, skupljanja i širenja.

Tako dobivamo izlaznu sliku na kojoj je do neke točnosti očuvan oblik jezgre. Tu sliku zatim predajemo postupku traženja jezgre `locateNucleus()`.

```
void locateNucleus(
  const img_wrap& imgCytoDilated,
  const img_wrap& imgGray,
  pixel& mean,
  int *radius,
  pixel& center,
  pixel& lengthVectorMax,
  pixel& lengthVectorMin,
  pixel& widthVectorMax,
  pixel& widthVectorMin);
```

Ulazni parametri su već spomenuta pripremljena slika (`imgCytoDilated`), ulazna crno-bijela slika (`imgGray`), te koordinate medijana (`mean`), radijusa (`radius`), koordinate centra jezgre (`center`), te koordinate početnih i završnih točaka pravaca pružanja stanice. Adrese se predaju kako bismo imali povratne vrijednosti jer ih koristimo u daljnjoj obradi (npr. iscrtavanju opisnog pravokutnika).

Funkcija prvo računa svojstvene vektore i za to koristi pomoćnu funkciju `calculateEigenVectors()` koja je objašnjena u poglavlju 3.2.4. Ona nam vraća pravce pružanja stanice.

Sada kada imamo pravce pružanja popunjavamo vektore koordinatama piksela koji predstavljaju pravce pružanja stanice kako bi ih kasnije mogli po potrebi iscrtati, ali i zbog razloga što ćemo se šetati po njima pri traženju jezgre. Jednostavno pronađemo krajnje piksele objekta u smjerovima pružanja stanice i popunimo vektore koordinatama piksela koji određuju pravce pružanja stanice (linije određene koordinatama krajnjih piksela objekta). Taj posao se odvija pozivom pomoćne funkcije `fillVectors()`.

```
void fillVectors(
    const img_wrap& imgCytoDilated,
    double v1y,
    double v2y,
    pixel mean,
    std::vector<pixel> &lengthVector,
    pixel& widthVectorMax,
    pixel& widthVectorMin);
```

Njoj dodatno predajemo referencu na vektor koji popunjavamo spomenutim pikselima te adrese početne i krajnje točke koje predstavljaju pravac pružanja u širinu koje također postavljamo u funkciji.

Sada u glavnoj funkciji imamo sve podatke koji su nam potrebni za lociranje jezgre. Radijus računamo kao polovinu duljine pravca pružanja stanice u širinu. Lociranje jezgre provodimo tražeći maksimalnu kružnu nakupinu piksela krećući se po pravcu koji predstavlja pružanje stanice u duljinu. To radimo tako da se pozicioniramo u sve piksele koji predstavljaju uzdužni pravac pružanja stanice (koordinate se nalaze u `lengthVector`). Na svakoj poziciju prebrojavamo koliko je piksela objekta unutar kružnice određene radijusom (širinom stanice). Ondje gdje je ta suma najveća nalazi se jezgra te tada pamtimo trenutne koordinate.

4. Eksperimentalni rezultati

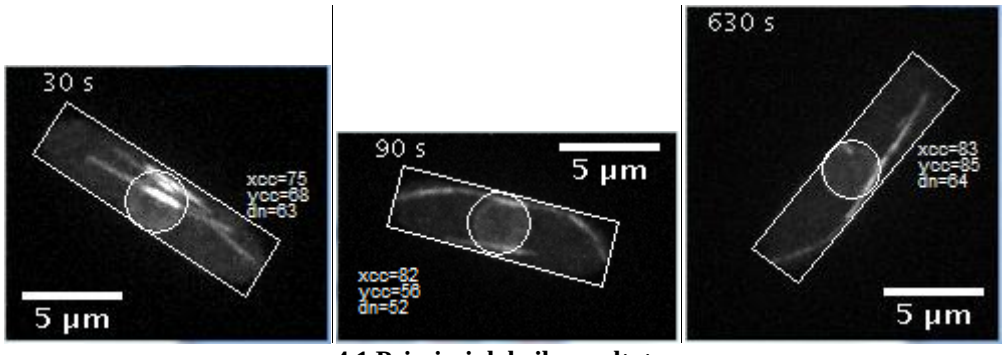
4.1. Organizacija rezultata

Kod testiranja ovog algoritma jedini testni parametar bi bio preciznost lokalizacije stanice i jezgre. Stoga se slijedeći rezultati dijele na:

1) Potpuno točna detekcija
2) Neprecizna detekcija citoplazme
3) Neprecizna lokalizacija jezgre

4.2. Potpuno točna detekcija

Potpuno točna detekcija se očituje kao u potpunosti zaokružena jezgra i citoplazma koja je u potpunosti okružena lokalizacijskim pravokutnikom. Na rezultatima su dodatno prikazane koordinate središta stanice (središte nakupine piksela citoplazme) te odmak središta jezgre od kraja citoplazme. Nekoliko primjera je prikazano na slici 4.1.



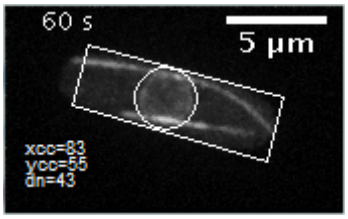
4.1 Primjeri dobrih rezultata

Na prikazanim primjerima se najbolje vidi točno detektirana citoplazma opisana pravokutnikom te točno lokalizirana jezgra opisana kružnicom.

4.3. Neprecizna detekcija citoplazme

Neprecizna detekcija citoplazme predstavlja pogrešku u detekciji koja se očituje kao odstupanje pravokutnika opisanog citoplazmi od idealnog položaja.

Primijećen je jedan tip pogreške vezan uz položaj opisanog pravokutnika oko citoplazme. Ta pogreška se očituje kao pojava da su dijelovi citoplazme izvan pravokutnika što je prikazano na slici 4.2. Razlog tome je tamniji dio citoplazme koji nije preživio binarizaciju, skupljanje i širenje (slika 4.3). Stoga je pravac pružanja stanice po duljini nešto kraći te je i ukupna duljina opisanog pravokutnika manja. Ova greška se pojavila na 27 slika iz uzorka od 260 slika. Većina tih slika sa pogreškom pripada istoj seriji slika na kojima su stanice puno tamnije od ostatka slika tako da bi za tu seriju trebalo postaviti druge ulazne parametre kako bi se uspješno proveo algoritam. Ako se ta serija slika izdvoji iz testa, dobivamo pojavljivanje greške na 2 slike od 205 slika što je zadovoljavajuće.



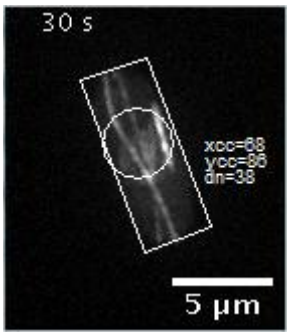
4.2 Primjer neprecizne detekcije citoplazme

4.3 Razlog pogreške

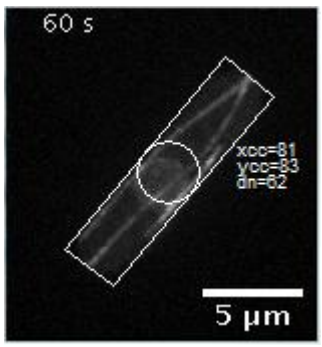
4.4. Neprecizna lokalizacija jezgre

Neprecizna lokalizacije jezgre predstavlja pogrešku u lokalizaciji koja se očituje kao odstupanje kružnice opisane jezgri od idealnog položaja.

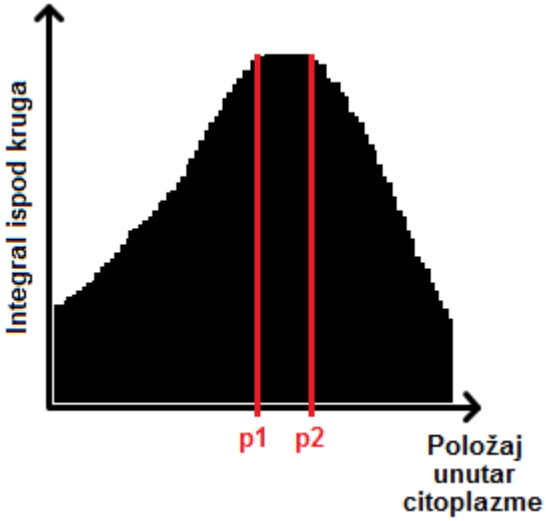
Primijećene su dvije vrste odstupanja opisane kružnice oko jezgre od pravog položaja jezgre. Prvi je kada opisuje više od same jezgre (slika 4.4) što je rezultat pretpostavki da je širina stanice upravo i promjer stanice te da je jezgra kružnog oblika. Drugi položaj je kad je veličina dobra ali je položaj kružnice odmaknut od idealne sredine jezgre (slika 4.5). Razlog tomu su mikrotubuli (niti) unutar stanice koji prežive pokušaje uklanjanja i zatim narušavaju položaj maksimalne kružne koncentracije svijetlih piksela pri traženju jezgre. Ove pogreške su uočene na 51 slici od 260 slika. Ponovno većina slika pripada istoj seriji slika, što zahtijeva promjenu ulaznih parametara algoritma kako bi se dobili bolji rezultati. Ako tu seriju izdvojimo iz testa, dolazimo do boljeg postotka. Greška se tada javlja na 13 slika od 165 testnih slika. Pri testiranju je u obzir uzeta tolerancija odstupanja od 10% radijusa jezgre (što u većini slučajeva odgovara otklonu od 2 piksela). Na slikama 4.6 i 4.7 se vide histogrami koji predstavljaju sume piksela na određenim pozicijama na uzdužnom pravcu pružanja stanice za primjere na slikama 4.4 i 4.5. Na histogramima su dodatno označene dobivene pozicije (p1) i prave pozicije (p2) jezgre u odnosu na rub stanice.



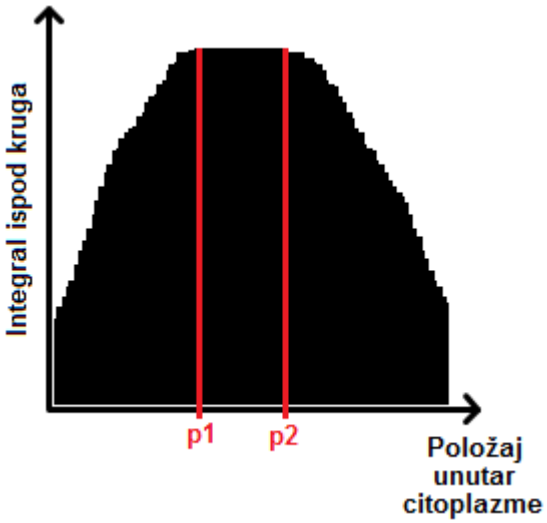
4.4 Primjer prevelike kružnice



4.5 Primjer pomaknute kružnice



4.6 Histogram za primjer sa slike 4.4



4.7 Histogram za primjer sa slike 4.5

5. Zaključak

Predstavljen je postupak detekcije stanične jezgre u mikroskopskim slikama. Postupak se temelji na dvije pretpostavke i to (i) da je jezgra kružnog oblika (ne eliptičnog), te (ii) da je radijus jezgre upravo onoliki kolika je i širina stanice. Ukoliko pretpostavke nisu zadovoljene, postupak neće moći proizvesti precizan rezultat. Postupak zahtijeva dosta ulaznih parametara o kojima ovisi konačni rezultat. Razlog tome je to što se koristi više postupaka pripreme slike za obradu. To omogućuje obradu širokog spektra ulaznih slika uz naravno modifikaciju tih ulaznih parametara.

Rezultati su pokazali točnu detekciju citoplazme i lokalizaciju jezgre u većini slučajeva. Pogreške se javljaju kad nisu zadovoljene pretpostavke, kada je stanica slabo vidljiva ili kada je jezgra gotovo istog intenziteta kao i ostatak stanice. Uz toleranciju odstupanja od 10%, spomenute greške su se javile na 13 slika od 165 slika.

Prosječna brzina obrade slike je reda 300 ms, što omogućava nešto više od obrade 3 slike u sekundi (3fps). To je dostatno jer obradu nije potrebno vršiti u stvarnom vremenu.

Iako je jezgra na svim slikama uvijek locirana i to barem 80% nje je okruženo lokacijskom kružnicom, ostaje mjesta za poboljšanja. Moguće je uvesti optimizaciju s bogatijim modelom (zarotiranom elipsom) koja bi umanjila pojavu pogrešaka uzrokovanih pretpostavkom da je jezgra okrugla. Praćenje pozicije jezgre na slikama se isto prezentira kao moguće poboljšanje. U postupku praćenja bismo pratili jezgru na nizu slika te bi na temelju prošlih pozicija mogli odrediti približni trenutni položaj. Problem mikrotubula bi se mogao riješiti uporabom upravljivog filtra [10], pri čemu bismo detektirali mikrotubule te ih zatim potisnuli.

6. Literatura

[1] P.T. Tran, L. Marsh, V. Doye, S. Inoué, and F. Chang: *A Mechanism for Nuclear Positioning in Fission Yeast Based on Microtubule Pushing*, The Journal of Cell Biology, 153:2, pp 397-412

[2] Tolic-Nørrelykke Iva M, Sacconi Leonardo, Stringari Chiara, Raabe Isabel, Pavone Francesco S. Title: *Nuclear and division-plane positioning revealed by optical micromanipulation*. Current biology, 15:1212-1216.

[3] Canny, J., *A Computational Approach To Edge Detection*, 1986., PAMI, volume 8, issue 6.

[4] Bašić, Š., Čepo, P. Š., Dodolović, I., Dostal, D., Grbić, S., Gulić, M., Horvatin, I., Louč, S., Sučić, I. *Cannyjev detektor rubova*, tehnička dokumentacija dodiplomskog projekta, Fakultet elektrotehnike i računarstva, 2008.

[5] Parker, J. R., *Algorithms For Image Processing And Computer Vision*, Wiley 1997.

[6] *Wikipedia, the free encyclopedia*, "Gaussian blur", 31 March 2009; http://en.wikipedia.org/wiki/Gaussian_blur, 18.04.2008.

[7] J. Canny, *Finding edges and lines in images*, tech. report AITR-720, Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.

[8] Duda, R., Hart, P., Stork, D., *Pattern Classification*, Wiley, 2001.

[9] M. Sezgin i B. Sankur (2004). "Survey over image thresholding techniques and quantitative performance evaluation". *Journal of Electronic Imaging* **13** (1): 146–165. doi:10.1117/1.1631315.

[10] Majić, A., *Pronalaženje prometnog traka korištenjem upravljivih filtara*, završni rad br. 391, Fakultet elektrotehnike i računarstva, 2009.

Detekcija stanične jezgre u mikroskopskim slikama

Sažetak

Razvijen je postupak pronalaženja stanice i jezgre kvasca u mikroskopskim slikama. Postupak se sastoji od dvije faze: detekcije citoplazme i lokalizacije jezgre. U izvedbi su korišteni algoritmi binarizacije, skupljanja, širenja i glađenja. Krajnji rezultat je položaj stanice predstavljen opisanim pravokutnikom i položaj jezgre predstavljen opisanom kružnicom. Algoritam je ostvaren u programskom jeziku C++. Postupci obrade slike su detaljno pojašnjeni kao i njihova programska implementacija. Postignuti eksperimentalni rezultati su prikazani i komentirani.

Ključne riječi

Računalni vid, detekcija objekata, binarizacija, skupljanje, širenje, glađenje, detekcija citoplazme, detekcija jezgre kvasca.

Nucleus detection in microscopic images

Summary

Method for yeast cell detection in microscopic images is developed. The method is divided into two phases: cytoplasm detection and nucleus localization. These phases are implemented by using algorithms of binarization, erosion, dilation and Gaussian blur. The end result is the position of the cytoplasm represented by a bounding rectangle and the position of the nucleus represented by a bounding circle. Method implementation is realized using C++. The employed image processing algorithms are explained in detail. The obtained experimental results are presented and discussed.

Keywords

Computer vision, object detection, binarization, erosion, dilation, Gaussian blur, cytoplasm detection, yeast nucleus localization.