SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 132

SIMULACIJA MODELA RASTA EPITELNOG TKIVA

Lovro Nuić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 132

SIMULACIJA MODELA RASTA EPITELNOG TKIVA

Lovro Nuić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 132

Pristupnik:	**Lovro Nuić (0036514276)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Simulacija modela rasta epitelnog tkiva**

Opis zadatka:

Proučiti metodu podjele prostora Voronoievim dijagramom te čestične sustave. Proučiti rast MDCK II tkiva. Razmotriti mogućnost prikaza rasta staničnog tkiva Voronoievim dijagramom. Implementirati podjelu prostora Voronoievim dijagramom temeljenu na sustavu čestica. Parametre sustava čestica trenirati tako da je konačni prikaz Voronoievim dijagramom usporediv s rastom stvarnog tkiva. Ostvariti prikaz simuliranog modela. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranih algoritama. Izraditi odgovarajući programski proizvod. Koristiti programski jezik C++. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 132

Simulacija modela rasta epitelnog tkiva

Lovro Nuić

Zagreb, srpanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 132

Pristupnik:	**Lovro Nuić (0036514276)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Simulacija modela rasta epitelnog tkiva**

Opis zadatka:

Proučiti metodu podjele prostora Voronoievim dijagramom te čestične sustave. Proučiti rast MDCK II tkiva. Razmotriti mogućnost prikaza rasta staničnog tkiva Voronoievim dijagramom. Implementirati podjelu prostora Voronoievim dijagramom temeljenu na sustavu čestica. Parametre sustava čestica trenirati tako da je konačni prikaz Voronoievim dijagramom usporediv s rastom stvarnog tkiva. Ostvariti prikaz simuliranog modela. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranih algoritama. Izraditi odgovarajući programski proizvod. Koristiti programski jezik C++. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

SADRŽAJ

1. Uvod

Znanstveno proučavanje pojava u prirodi temelji se na provođenju eksperimenata. Iz rezultata eksperimenata pokušavamo zaključiti koje zakonitosti vrijede u nekom sustavu. Kako bi razumjeli procese koji se nalaze u pozadini eksperimenta poželjno je moći sve pojedine dijelove dobro kontrolirati. Zbog toga se u istraživanju rasta tkiva primjenjuju standardizirane stanične kolonije kao što je MDCK II (*Madin-Darby Canine Kidney*) tkivo. MDCK II tkivo je vrsta epitelnog tkiva u distalnoj tubi bubrega koker španijela. Ovo tkivo karakterizira jednoslojni rast na dvodimenzionalnoj podlozi, te se stoga i svrstava u monoslojna tkiva.

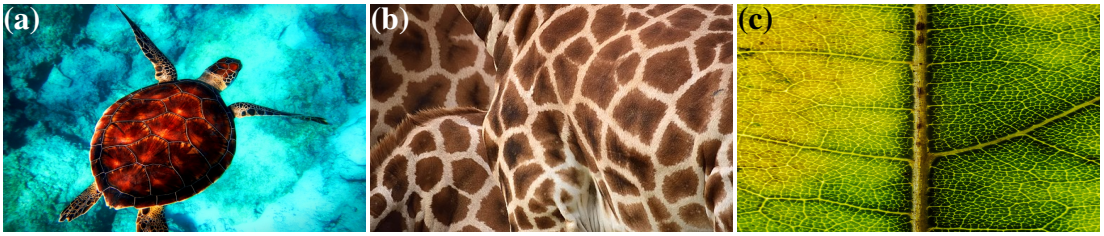
Količina podataka koje izmjerimo prilikom provođenja eksperimenata često premašuju mogućnosti ručne obrade. Takva disproporcija u vremenu potrebnom za ručnu obradu rezultata i dostupnog vremena prevladava se razvojem automatiziranih alata koji mjerenje pojedinih često promatranih dijelova ubrzavaju. Kod promatranja rasta tkiva jedan takav alat je svakako i označavanje jezgri stanica. Ovakvim alatom možemo detaljno analizirati osnovna svojstva stanica iz slika tkiva. Novi napreci u području dubokog učenja omogućuju nove pristupe tom problemu. U drugom dijelu ovog rada proučit ćemo arhitekturu U-Net koju ćemo primijeniti za segmentiranje jezgri stanica.

Simulacije svakako su jedan od bitnih alata predviđanja i modeliranja pojava u prirodi. Često se dogodi da različite aspekte iz prirode ne možemo ostvariti zbog premale računalne moći koje pruža sklopovlje. Neke od tih problema možemo riješiti paralelizacijom na za to specifičnom sklopovlju. Razvoj napredne sklopovske podrške kao što su grafičke procesorske jedinice omogućuje različite načine obrade, ali i generiranja velike količine podataka.

Rezultati analize strukture tkiva pokazuju da Voronoijev dijagram kojemu su generatori jezgre stanica dobro aproksimira raspodjelu prostora između stanica. Zbog tog razloga Voronijevi dijagrami su često ključni dio simulacije rasta tkiva. U mnogim modelima koristi se Voronoijevi dijagrami generirani točkama, dok ćemo u ovom radu proučavati Voronoijeve dijagrame jezgri koje ćemo aproksimirati elipsama.

2. Voronoijevi dijagrami

Promatranjem prirode vrlo često naiđemo na zanimljive uzorke. Jedan takav uzoraka može se vidjeti na pigmentu kože žirafe 2.1b, na oklopu kornjače 2.1a ili na rasporedu lisnih žila2.1c. Svi navedeni uzorci su jako slični i postavlja se pitanje može li se napraviti takav uzorak.



Slika 2.1: Uzorci koje nalazimo u prirodi: (a) oklop morske kornjače, (b) pigment kože žirafa, (c) nervatura lista

Uzmemo li skup s konačnim brojem različitih točaka i podijelimo prostor tako da svaku točki prostora pridružimo njoj najbližu točku iz skupa dobit ćemo uzorak jako sličan željenom. Ovakvu jednostavnu konstrukcija koristilo je mnogo ljudi u povijesti, stoga nije čudo da se postoji više naziva. Kako je u literaturi najčešći naziv Voronoijev dijagram koristi ćemo ga i u ovom radu.

Zanimljivo je da ovakav uzorak promatralo mnogo ljudi kroz povijest i da povezuje rješenja problema u različitim znanstvenim područjima. Među zanimljivijim povijesnim dokumentima na kojima se pojavljuju nacrtani Voronoijevi dijagrami je i karta dijela Londona koju je napravio John Snow u 19. stoljeću. On je pratio slučajeve kolere u Londonu te ih bilježio na karti. Pretpostavivši da se zaraza prenosi putem onečišćene vode, na kartu je označio područja koja su bila najbliža pojedinim javnim pumpama za vodu. Većina slučajeva zaraze nalazila se u području oko crpke za vodu u ulici Broad Street, te nakon njezinog uklanjanja broj zaraza se smanjio. Među prvim opsežnim istraživanjima uzorka je rad Georgy Voronoy koji je proučavao podjelu

n-dimenzionalnog prostora s točkama cjelobrojnih koordinata. [13]

2.1. Matematička definicije pojmova

Uvesti ćemo prema literaturi [5] i [18] matematičku definiciju Voronoijeva dijagrama i neka svojstva koja ćemo koristiti u radu.

U ovom radu promatrat ćemo Voronoijev dijagram u dvodimenzionalnom euklidskom prostoru stoga možemo definirati udaljenost dvije točke p_1 i p_2 kao:

Definicija 2.1.1 (Udaljenost točaka p i q)**.**

$$d(p_1, p_2) = \sqrt{(p_1.x - p_2.x)^2 + (p_2.y - p_1.y)^2} \tag{2.1}$$

Definicija 2.1.2 (Voronoijeva ćelija)**.** Neka je $P = \{p_1, p_2, \dots p_n\}, n \geq 2, n \in \mathbb{N}$, skup od n različitih točaka u prostoru. Neka su točke p_i i p_j takve da $p_i \neq p_j$ za $i \neq j$, $i, j \in \{1, 2, \dots, n\}$. Voronoijevu ćeliju od p_i, definiramo oznakom $Vor(p_i)$ prema

$$Vor(p_i) = \{x \mid d(x, p_i) < d(x, p_j), \forall j \neq i, j \in \{1, \dots, n\}\} \tag{2.2}$$

Definicija 2.1.3 (Voronoijev dijagram)**.** Skup svih Voronoijevih ćelija točaka iz P nazivamo Voronoijev dijagram od P te ga označavamo s $Vor(P)$. Dakle $Vor(P) = \{Vor(p_1), Vor(p_2), \dots, Vor(p_n)\}$. Točke iz skupa P nazivamo generatori Voronoijeva dijagrama.

Neka su p i q dvije točke u prostoru $\mathbb{N}^2$, tada definiramo $h(p, q)$ kao poluravninu, u kojoj je točka p, koja nastaje kada ravninu podijelimo simetralom dužine $\overline{\mathrm{pq}}$. Također $h(q, p)$ je poluravnina u kojoj se nalazi točka q. Vidimo da je $r \in h(p, q)$ akko $d(r, p) < d(r, q)$, stoga Voronoijevu ćeliju možemo definirati i kao:

$$Vor(p_i) = \cap_{1 \leq j \leq, i \neq j} h(p_i, p_j) \tag{2.3}$$

Za točku q definiramo najveći prazni krug od q u odnosu na točke iz P, kao najveći krug s točkom q u centru koji ne sadrži nijednu drugu točku iz P, te ga označavamo se $C_P(q)$.

Navodimo sljedeće teoreme bez dokaza. Za $Vor(P)$ gdje je P skup od $n \geq 3, n \in \mathbb{N}$ točki generatora vrijedi:

Teorem 2.1.1. *Broj vrhova u* $Vor(P)$ *je najviše* $2n-5$ *i broj bridova je najviše* $3n-6$.

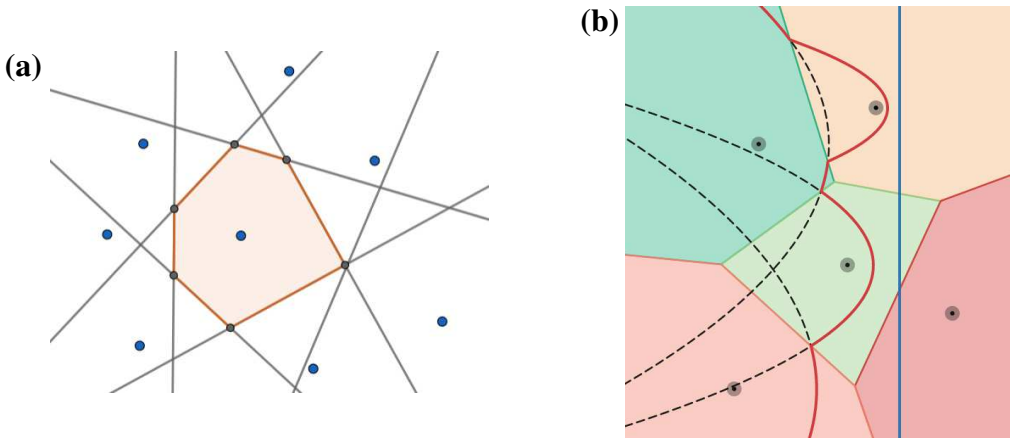
Teorem 2.1.2. *Točka q je vrh od $Vor(P)$ akko najveći prazni krug $C_P(q)$ sadrži tri ili više točaka na kružnici.*

Teorem 2.1.3. *Simetrala dužina između generatora p_i i p_j je brid od $Vor(P)$ akko postoji točka q koja se nalazi na simetrali dužine takva da $C_P(q)$ sadrži na rubu samo generatore p_i i p_j.*

2.2. Algoritmi izgradnje

Postoje mnogo različitih algoritama izgradnje dvodimenzionalnih Voronoijeva dijagrama, ali navesti ćemo neke najkorištenije. Podijelit ćemo algoritme u dvije kategorije u ovisnosti o preciznosti izračuna na egzaktne i diskretizirane Voronoijeve dijagrame.

2.2.1. Egzaktni algoritmi



Slika 2.2: Egzaktni algoritmi: (a) algoritam grubom silom, (b) Fortune algoritam

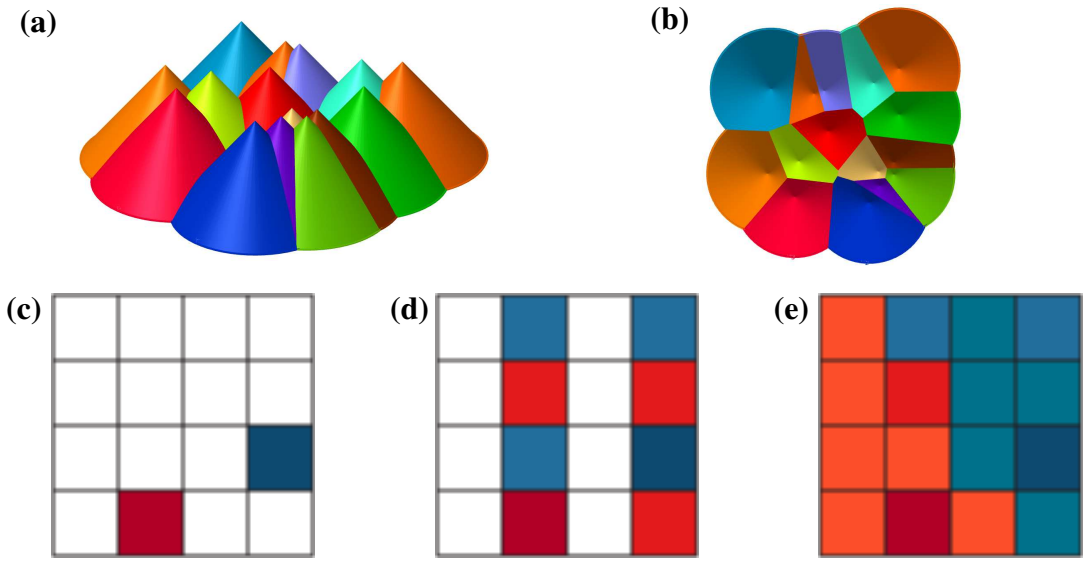
Vjerojatno najjednostavniji način izgradnje koristi navedeno svojstvo 2.3. Na slici 2.2a je prikazana Voronoijeva ćelija jednog generatora nastala presjekom poluravnina. Ovakav način određivanja Voronoijeva dijagrama je izrazito spor jer mu je složenost $\mathcal{O}(n^2)$ te se zbog toga u praksi ne koristi.

Među najpoznatijim algoritmima izgradnje je Fortune-ov algoritam, na slici 2.2b, koji spada u kategoriju algoritama pomicanjem linije. Algoritam prati događaje kroz koje prolazi linija za skeniranje usporedna s y osi. Na polovištu između svake točke koja se nalazi lijevo od linije za skeniranje definira se parabola. Također algoritam pamti u binarnom stablu liniju vanjskih parabola koja je rub svih parabola koje se

nalaze s lijeve strane linije za skeniranje. Događaji su kada linija za skeniranje pređe preko nove točke ili prilikom promjene parabola na liniji vanjskih parabola. Ovakav algoritam ima složenost $\mathcal{O}(n \log n)$, te se zbog toga često koristi u praksi. [5]

2.2.2. Aproksimacijski algoritmi

Aproksimacijski algoritmi često su korišteni kada postoji potreba samo za prikazom Voronoijevih dijagrama. Algoritmi koji se mogu izvesti u grafičkoj protočnoj strukturi pogodni su u takvim situacijama.



Slika 2.3: Aproksimacijski algoritmi: (a, b) Algoritam stožaca , (c) Jump Flooding korak 1, (d) Jump Flooding korak 2, (e) Jump Flooding korak 3

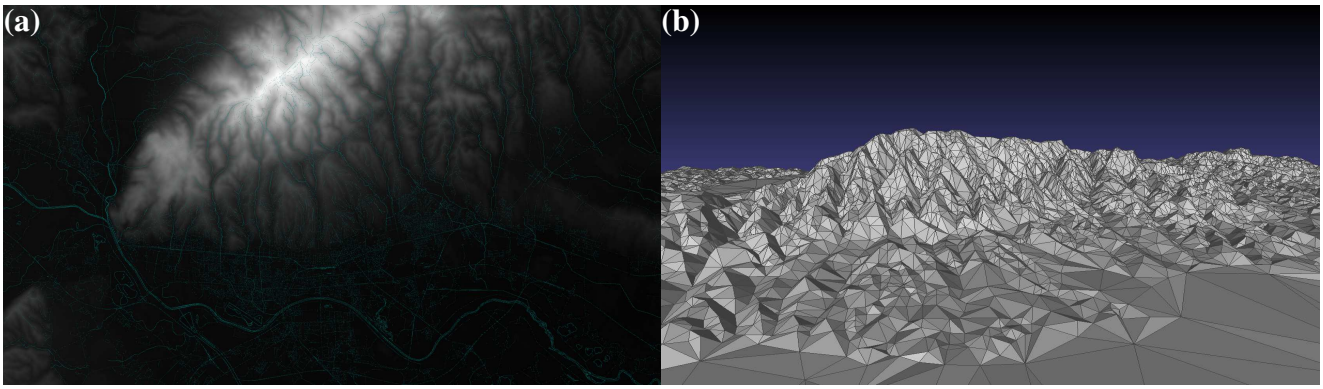
Algoritam stožaca koristi postojeću implementaciju Z-spremnika u grafičkoj protočnoj strukturi iscrtavajući beskonačne trodimenzionalne stošce u točkama generatorima (slika 2.3a). Iscrtavanjem scene ortografskom perspektivom iz pozicije okomite na ravninu u kojoj se nalaze točke generatori dobivamo Voronoijev dijagram (slika 2.3b). Univerzalnost ovog rješenja nam omogućuje da za generatore koristimo poligone, a kao trodimenzionalno tijelo koristimo oblik dobiven unijom stožaca iz svih točaka na poligonu. Ovakva inačica Voronoijeva dijagrama bit će obrađena u poglavlju 4.

Sjenčar fragmenata također možemo koristiti za izgradnju Voronoijeva dijagrama pomoću algoritma "Jump Flooding"(JFA) iz rada [15] . Točke generatore Voronoijeva

dijagrama mapiramo različitim bojama na kvadratnu teksturu veličine potencije broja 2. Potom se provodi niz iteracija za svaki slikovni element teksture. U svakoj iteraciji veličina koraka je prepolovljena veličina koraka u prethodnoj iteraciji, a početni korak je pola veličine teksture. Svaki slikovni element u iteraciji provjerava boju svojih susjeda udaljenih za točno veličinu koraka u toj iteraciji u 8 smjerova. Za već obojene susjede se potom uspoređuje duljina trenutnog slikovnog elementa s točkom generatorom kojoj pripada boja susjeda. Nakon provedenih svih koraka zaključno s korakom veličine 1, dobivena tekstura je pobojana Voronoijevim dijagramom. Primjer provedbe algoritma JFA za aproksimaciju, Voronoijev dijagram s dvije točke i teksturom veličine 16 slikovnih elemenata je prikazan na slikama 2.3c - 2.3e. Ovakav algoritam može se paralelizirati, a i sama složenost je $\mathcal{O}(n \log_2 n)$ što je daleko bolje od naivnog algoritma. Problem ovakvog algoritma je veličina potrebne teksture pri velikom broju točaka generatora, jer je za svaku točku generatora potreban barem jedan početni slikovni element.

3. Delaunayova triangulacija

Ideju Delaunayove triangulacije uveo je Voronoi u svom radu opisujući strukturu odnosa između susjeda u Voronoijevom dijagramu. Ipak Delone prvi definira takvu triangulaciji pomoću svojstva prazne sfere. Svoj rad objavljuje na francuskom jeziku i potpisuje se kao Delaunay zbog čega se u literaturi pojavio naziv Delaunayeva triangulacija, kojeg ćemo koristiti u ovom radu. [13]



Slika 3.1: Delaunay triangulacija kod visinskih mapa: (a) visinska mapa Grada Zagreba, (b) trokutasta mreža dobivena 2.5D Delaunay triangulacijom

Slika 3.1a prikazuje visinsku mapu područja Grada Zagreba i okolice. Visinska mapa je zapravo funkcija $M : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ koja pozicijama u području prostora pridodjeljuje nadmorsku visinu. Prilikom spremanja u teksturu obično se najvećoj nadmorskoj visini pridodijeli bijela boja, dok najnižoj crna boja. Na ostale nadmorske visine se primjeni bilinearna funkcija. Visinska mapa 3.1a ima samo konačan broj izmjerenih pozicija u području što nam otežava mogućnost trodimenzionalnog prikaz područja. Pojavljuje se potreba za nekom vrstom aproksimacije nadmorske visine. Za pozicije prostora možemo izgraditi triangulaciju u ravnini, te potom podići pozicije u trodimenzionalan prostor prema njihovim nadmorskim visina. Problem koji nastaje je da postoji velik broj mogućih načina triangulacije točaka, ali ispostavlja se da zapravo

Delaunayeva triangulacija u ovom problemu daje najprirodniji prikaz (primjer slika 3.1b). [5].

3.1. Matematičke definicije pojmova

Uvesti ćemo prema literaturi [5] matematičku definiciju Delaunayove triangulacije i neka svojstva koja ćemo koristiti u radu.

Definicja 3.1.1 (Triangulacija T)**.** Neka je $P = \{p_1, p_2, \dots p_n\}, n \geq 2, n \in \mathbb{N}$, skup točaka. Triangulacija $T(P)$ je neusmjereni planaran povezan pravocrtan graf kojem su vrhovi točke iz skupa P, a dodavanjem jednog dodatnog proizvoljnog pravocrtnog brida, koji već nije u grafu, narušava se planarnost.

Može se pokazati da su sve unutarnje strane triangulacije trokuti, a vanjski obrub triangulacije čini konveksnu ljusku skupa P.

Teorem 3.1.1. *Neka je $P = \{p_1, p_2, \dots p_n\}, n \geq 2, n \in \mathbb{N}$, skup točaka. Neka je k broj točaka koje se nalaze na konveksnoj ljusci $T(P)$. Tada sve triangulacije imaju $2n - 2 - k$ trokuta i $3n - 3 - k$ bridova.*

Ovaj teorem koristimo prilikom rezervacije memorijskog prostora potrebnog za spremanje $T(P)$.

Teorem 3.1.2 (Delaunayeva triangulacija DT(T))**.** *Skup trokuta koji nastaje kada se spoje točke generatori susjednih Voronoijevih ćelija naziva se Delaunayeva triangulacija DT(T).*

Delaunayeva triangulacija DT(T) također je triangulacija koja ima maksimalan minimalan kut triangulacije.

3.2. Algoritmi izgradnje

Prema [12] algoritme izgradnje Delaunayeva triangulacije možemo podijeliti u kategorije:

- lokalno poboljšanje - algoritam proizvoljno dobivenu trokutastom mrežom transformiramo mrežu do Delaunaye triangulacije primjenjujući lokalni Delaunay test i okrećući bridove

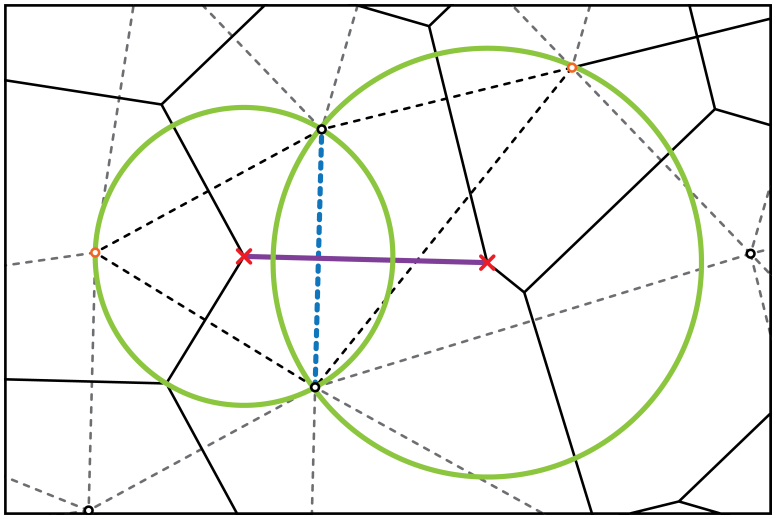
– preslikavanje u višu dimenziju - točke se preslikaju u prostor za jedan dimenzije više. Izračuna konveksna ljuska u novom prostori koja se potom projicira na početni prostor

– inkrementalno umetanje - točke postepenim umetanjem u postojeću Delaunayevu triangulaciju, koju nakon svake umetnute točke popravljamo do Delaunayove.

– podijeli pa vladaj - točke se podijele u grupe za koje dobijemo Delaunayeve triangulacije, te se potom takve grupe spajaju

Zbog potreba za velikim brojem Voronoijevih ćelija u simulacijama tkiva istražene su mogućnosti paralelizacije. Kako u simulacijama tkiva dolazi samo do malih promjena u Delaunayevom dijagramu između svaka dva koraka, razvoj je usmjeren na paralelizaciju algoritama iz kategorije lokalnog poboljšanja postojećeg dijagrama umjesto uzastopnom izgradnjom novog. Odabrani algoritam opisan je i implementiran u radu [14] te će se on i koristiti u ovom radu.

3.3. Povezanost s Voronoijevim dijagramom



Slika 3.2: Dualnost Voronoi dijagram - Delaunay triangulacija: plavi brid - Delaunayev brid, zelene kružnice - opisane kružnice kroz točke brida i njemu nasuprotnih žutih vrhova, crveni križići - centri zelenih kružnica i vrhovi Voronoijevih ćelija

Slika 3.2 prikazuje dualnost Voronoijeva dijagrama i Delaunyeve triangulacije. Odaberemo li brid iz Delaunayeve triangulacije označenog isprekidanom plavom bojom (označimo ga s b_1). Neka taj brid spaja točke p_1 i p_2. Označimo i suprotne vrhove tom bridu sa op_1 i op_2 tada iz centara opisane kružnice trokuta $\triangle p_1 p_2 op_1$ i centar opi-

sane kružnice trokuta $\triangle p_1 p_2 o p_2$ čine susjedne vrhove Voronoijeva dijagrama za točke generatore p_1 i p_2.

4. Voronoijeva dijagram skupova

U ovom poglavlju proučit ćemo izgradnju Voronoijeva dijagrama skupova u dvodimenzionalnom prostoru kojem su generatori podskupovi ravnine. Istraženo je i implementirano rješenje na grafičkom procesoru koje je nadogradnje na izabrani algoritam iz prethodnog poglavlja. Specifični Voronoijev dijagram kojem su generatori elipse bit će iskorišten u poglavlju 5. prilikom simulacije rasta tkiva.

4.1. Matematička definicija

Definirajmo prvo udaljenost točke p do nekog podskupa ravnine.

Definicija 4.1.1 (Udaljenost točke p i podskupa A ravnine)**.** Neka je $p \in \mathbb{R}^2$ i neka je S neprazni podskup od $\mathbb{R}^2$. Udaljenost točke p i podskupa ravnine S definiramo s:

$$d(p, A) = \inf\{dist(p, x) \colon x \in A\} \tag{4.1}$$

Neka je S skup svih podskupova ravnine $S = \{S_0, S_1, \ldots, S_n\}$ za koje želimo izgraditi Voronoijev dijagram. Sada možemo definirati Voronoijevu ćeliju od S_i s oznakom $Vor(S_i)$.

Definicija 4.1.2 (Voronoijeva ćelija od S_i)**.**

$$Vor(S_i) = \{x \mid d(x, S_i) < d(x, S_j), \forall j \neq i, j \in \{1, \ldots, n\}\} \tag{4.2}$$

Voronoijev dijagram skupova tada je unija svih Voronoijevih ćelija te to označavamo s $Vor(S) = \{Vor(S_1), Vor(S_2), \ldots, Vor(S_n)\}$.

4.2. Algoritam izrade dijagrama

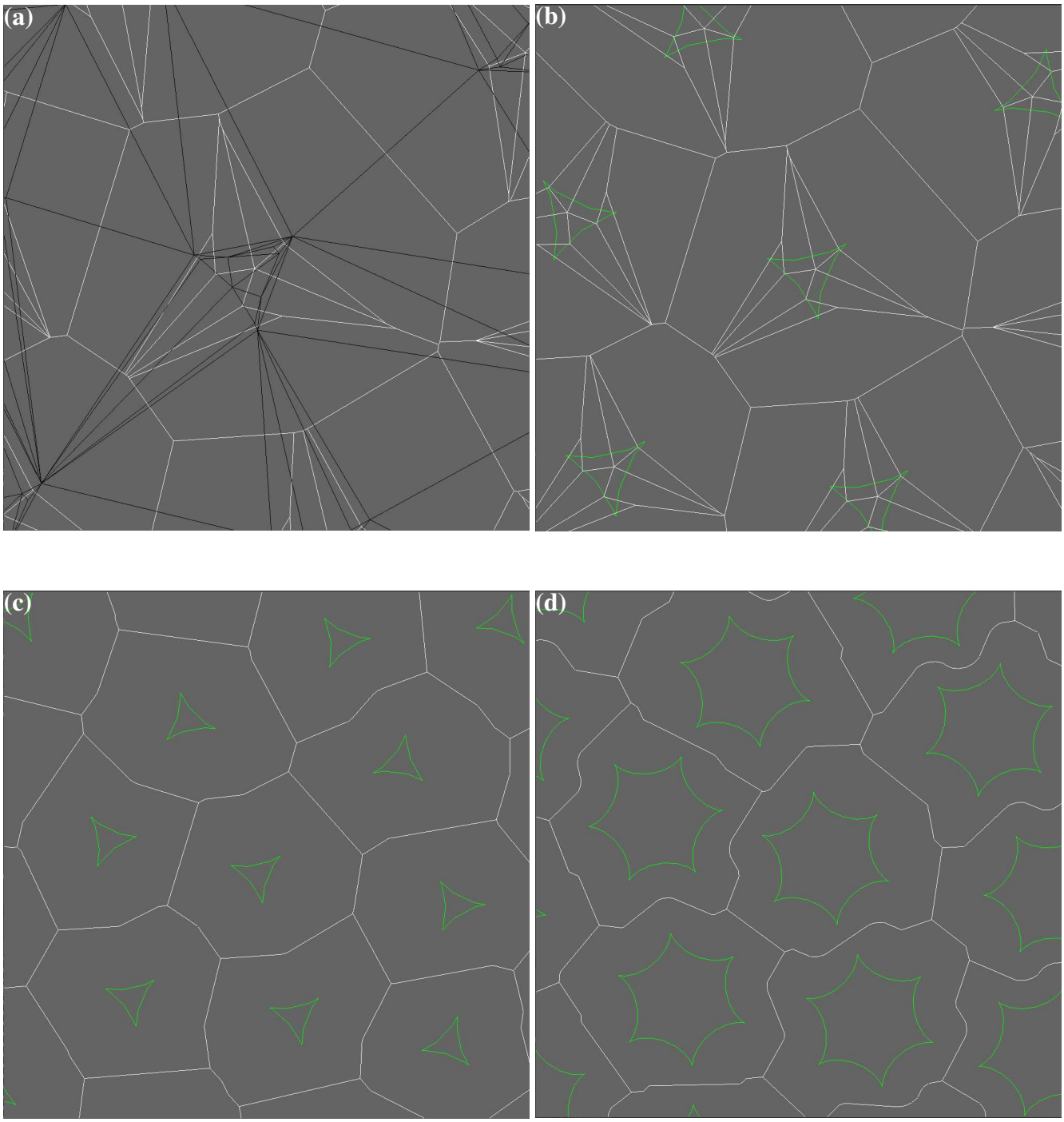
Aproksimaciju Voronoijev dijagram skupova možemo dobiti tako da rub prostora S aproksimiramo dužinama. Sve točke dužina koristimo kao generatore Voronoijeva dijagrama. Dobiveni dijagram sastoji se od Voronoiji ćelija na koje nazivamo osnovnim

u ovom kontekstu. Ovakvim načinom aproksimacija svake Vronoijeve ćelije podskupa ravnine zapravo se sastoji od većeg broja osnovnih Voronojevih ćelija. Osnovne Voronoijeve ćelije je potrebno spojiti kako bismo dobili aproksimaciju Voronijeve ćelije skupa.

Predloženi način generiranja aproksimacije Voronoijeva dijagrama skupova prikazat ćemo na skupu hipocikloida. Hipocikloida je krivulja koja je definirana parametarskim jednadžbama:

$$x(\theta) = r(k-1)\cos\theta + r\cos\left((k-1)\theta\right) \tag{4.3}$$

$$y(\theta) = r(k-1)\sin\theta - r\sin\left((k-1)\theta\right) \tag{4.4}$$



Slika 4.1: Voronoij dijagrami skupova hipocikloida:, bijeli bridovi - Voronoijevi dijagrami, crni bridovi - Delaunayeva triangulacija, zeleni bridovi - podskupovi ravnine

Prostor na slikama (4.1a-4.1c) je podijeljen na hipocikloide s parametrom $k = 3$ koje su proizvoljno rotirane u prostoru. Svaka hipocikloida je aproksimirana s 10 točaka prema parametarskoj jednadžbi 5.2. Točke koje su susjedne na rubu hipocikloide

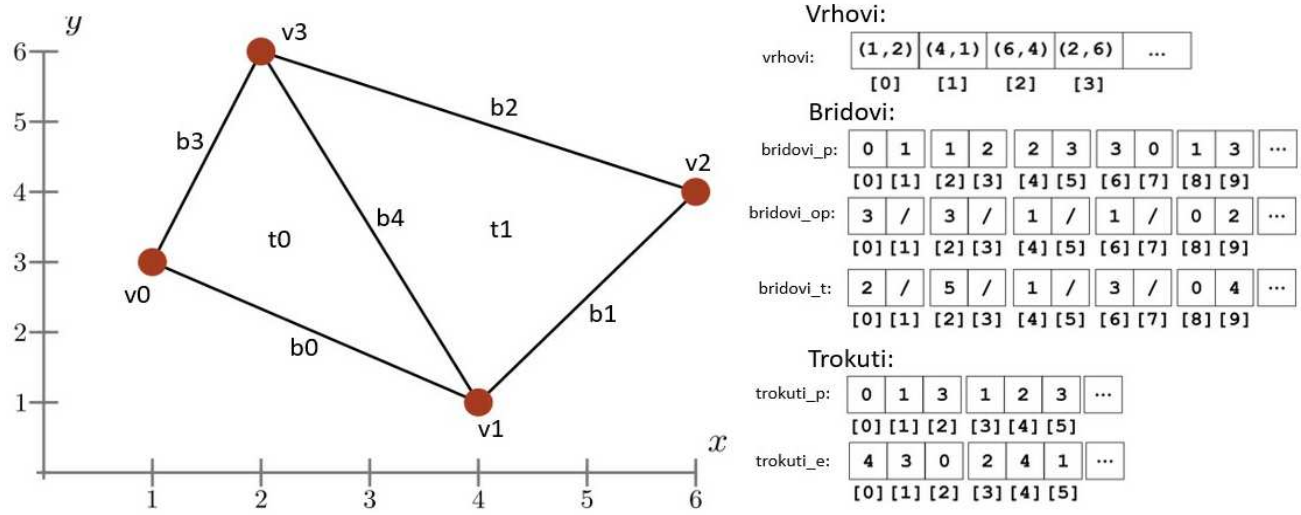
su povezane dužinama. Tako povezane dužine omeđuju prostor unutar hipocikloide.

Prvi korak prilikom generiranja Voronoijeva dijagrama skupa je izrada Delaunayeve triangulacije točaka koje aproksimiraju rub. Na slici 4.1a je prikazan Delaunayeva triangulacija promatranih hipocikloida. Možemo primijetiti da imamo bridove koji su međusobno jako blizu što može izazvati različite probleme prilikom pomicanja skupa generatora što ćemo proučiti u poglavlju 5.

Voronoijev dijagram točaka konstruiramo primjerom svojstva povezanosti s Delaunayevom triangulacija. Dobivene osnovne Voronoijeve ćelije prikazane su na slici 4.1b s bijelim bridovima, dok su zelenom bojom prikazani rubovi hipocikloida za koje želimo izgraditi Voronoijev dijagram. U svakoj osnovnoj ćeliji nalazi se jedna točka ruba neke hipocikloide, dok unija osnovnih Voronoijevih ćelija iste hipocikloide čini traženu Voronoijevu ćeliju. Rezultat unije prikazan je na slici 4.1c.

Kod jednostavnih oblika s malim brojem točaka koje koristimo za aproksimaciju ruba podskupa prostora možemo dobiti zadovoljavajuće rezultate, što možemo vidjeti i na slici 4.1d gdje je za generatore uzeta hipocikloida s parametrom $k = 6$. Zbog veće složenosti takve cikloide povećan je broj točaka s kojima aproksimiramo njezin rub na 30.

4.3. Struktura podataka



Slika 4.2: Struktura podataka korištene u algoritmu izgradnje Delaunayeve triangulacije

Grafički procesor je arhitekturski SIMD procesor što znači da jednu instrukciju izvršava na više memorijskih podataka paralelno. Stoga se pri dohvaćanju memorijske

lokacije dohvaćaju se velik broj elementa koji su neposredno jedan pored drugoga. To svojstvo je uzeto u obzir korištenjem rasporeda memorije strukturom nizova, a ne kako je uobičajeno nizom struktura.

Struktura podataka trokutaste mreže identična je predloženoj u radu [14], ali navest ćemo je ponovno kako bih mogli objasniti dodatne algoritme.

Poligonalna mreža sastoji se od vrhova, bridova i trokuta. Niz vrhova sastoji se od vektora koordinata vrhova poligonalne mreže. Bridovi se sastoje od tri niza. Niz bridovi_p predstavlja indeks vektora koje taj brid povezuje. Tako na primjer brid b_i povezuje vrhove s indeksima bridovi_p[2 * i] i bridovi_p[2 * i + 1]. Niz bridovi_op sadrži informacije o indeksima nasuprotnih vrha toga brida, dok u nizu bridovi_t su spremljene vrijednosti pozicije pripadnog nasuprotnog vrha tom bridu u nizu trokuti_p. Niz trokuti_p predstavlja indekse vrhova trokuta, dok niz trokuti_e predstavlja niz indeksa brida koji je nasuprotni vrhu na toj poziciji u tom trokutu.

Iako ovakva struktura ima redundanciju podataka, ona je nužna zbog potrebe za uklanjanjem i umetanjem vrhova u Delaunayevu triangulaciju.

4.4. Izračun morfoloških mjera

4.4.1. Susjedne ćelije u dijagramu

Definirajmo prvo što znači da su dvije ćelije susjedi:

Definicja 4.4.1 (Susjedi Voronoijeve ćelije skupa S_i)**.** Neka su $Vor(S_i)$ i $Vor(S_j)$, $i \neq j$, takve da dijele rub u $Vor(S)$. Tada kažemo da su Voronoijeve ćelije skupova S_i i S_j susjedi.

Algoritam za određivanje susjeda u Voronoijevu dijagramu opisan je u radu [14]. Ovakav algoritam ipak ne možemo koristiti za Voronoijeve dijagrame skupova zato što međusobno susjedne ćelije mogu biti višestruko povezane u odgovarajućem Delaunayevom dijagramu. Stoga trebamo modificirati algoritam kako bi bio ispravan za Voronoijeve dijagrame skupova.

Kako bismo eliminirali ponavljanje vrijednosti prilikom izračuna susjeda koristit ćemo strukturu raspršeni skup(unrodered_set) iz biblioteke STDGPU. Ova struktura ima svojstvo da unutar memorije grafičkog procesora možemo unositi vrijednosti proizvoljan broj puta, dok prilikom paralelnog čitanja iz strukture svaka vrijednost je jedinstvena.

Izračun susjeda započinje pridodjeljivanjem dretve svakom bridu u trokutastoj mreži. Za svaki brid znamo koje vrhove povezuje u Delaunayevom dijagramu. Ako dobiveni indeks vrha $cell_1$ je veći od indeksa vrha $cell_2$ onda indekse vrhova okrenemo. Manji indeks pomaknemo na gornju polovicu 64 bitnog broja, dok se veći indeks nalazi na donjoj polovici 64 bitnog broja. Na kraju algoritma imamo u strukturi susjedi_set sve susjede u dijagramu.

Algorithm 1 Izračun susjeda u Voronoijevom dijagramu skupova

Require: $Mesh(P)$ je Delaunayeva triangulacija $DT(P)$

Ensure: susjedi_set je skup svih jedinstvenih susjeda

```
1: procedure IZRAČUNSUSJEDA(m, susjedi_set)
2:     for idx ← 0 to |m.edges| do
3:         n ← broj točaka aproksimacije ruba potprostora
4:         cell_1 ← m.bridovi_p[2 * idx]/n
5:         cell_2 ← m.bridovi_p[2 * idx + 1]/n
6:         if cell_1 > cell_2 then
7:             swap(cell_1, cell_2)
8:         susjedi_set.insert(cell_1 << 32 + cell_2)
```

4.4.2. Površina, opseg i centar mase

Algoritam za izračun površine i opsega opisan u radu [14] također moramo prilagoditi izračunu za Voronoijeve dijagrame skupova. Stoga je implementirani modificirani algoritam. Svaka Voronoijeva ćelija točkastog generatora može se triangulirati tako da se za trokute uzmu susjedni vrhovi od $Vor(p_j)$ i sami generator p_j. To znači da možemo triangulirati i Voronoijevu ćeliju skupova jer se ona sastoji od više osnovnih Voronoijevih ćelija točkastih generatora. Neka je a_j površina jednog trokuta triangulirane Voronoijeve ćelije točkastog generatora p_i koji pripada rubu podskupa ravnine S_k. Također neka je c_j centar mase tog trokuta. Tada se centar mase $\vec{C_{S_k}}$ Voronoijeve ćelije podskupa ravnine S_k može izračunati prema formuli:

$$\vec{C_{S_k}} = \frac{1}{A_{C_{S_i}}} \sum_{i=0}^{m} \sum_{j=0}^{n} \vec{c}_j a_j \tag{4.5}$$

Površinu Voronoijeve ćelije S_k možemo izračunati kao sumu svih osnovnih Voronoijevih ćelija točaka na rubu području S_k, dok je za opseg potrebno pripaziti da samo sumiramo vanjski rub osnovnih ćelija. Taj uvjet možemo ostvariti ako samo sumiramo duljinu između Voronoijevih vrhova osnovnih ćelija kada Delaunayev brid povezuje

različite podskupove prostore S_k i S_l. U pseudo kodu algoritma 2 svakom bridu pridružimo dretvu te paralelno računamo površinu, opseg i centar mase Voronoijevog dijagram skupova.

Algorithm 2 Izračun površine, opsega, i centra mase Voronoijevog dijagrama skupova

Require: $T(P)$ je Delaunayeva triangulacija $DT(P)$

Ensure: $povrsina$ - niz površina od pojedinačne ćelije iz $Vor(S)$

Ensure: $opseg$ - niz opsega od pojedinačne ćelije iz $Vor(S)$

Ensure: $centar\_mase$ - niz centara mase od pojedinačne ćelije $Vor(S)$

1: **procedure** IZRAČUNSUSJEDA(m, $povrsina$, $opseg$, $centar\_mase$)
2: **for** $idx \leftarrow 0$ **to** $|m.bridovi|$ **do**
3: $n \leftarrow$ broj točaka aproksimacije ruba potprostora
4: $p_1 \leftarrow m.bridovi\_p[2 * idx]$
5: $p_2 \leftarrow m.bridovi\_p[2 * idx + 1]$
6: $op_1 \leftarrow m.bridovi\_op[2 * idx]$
7: $op_2 \leftarrow m.bridovi\_op[2 * idx + 1]$
8: $cell_1 \leftarrow p_1/n$
9: $cell_2 \leftarrow p_2/n$
10: $voronoi\_vrh_1 \leftarrow$ centarOpisaneKružnice(p_1, p_2, op_1)
11: $voronoi\_vrh_2 \leftarrow$ centarOpisaneKružnice(p_1, p_2, op_2)
12: $area_1 \leftarrow$ izračunajPovršinuTrokuta(c_1,c_2,p_1)
13: $area_2 \leftarrow$ izračunajPovršinuTrokuta(c_1,c_2,p_2)
14: $opseg_{12} \leftarrow$ izračunajDuljinuDužine($voronoi\_vrh_1$,$voronoi\_vrh_2$)
15: $centar\_mase_1 \leftarrow (voronoi\_vrh_1 + voronoi\_vrh_2 + m.vrhovi[p_1]) * (area_1/3)$
16: $centar\_mase_2 \leftarrow (voronoi\_vrh_1 + voronoi\_vrh_2 + m.vrhovi[p_2]) * (area_2/3)$
17: atomicAdd($area[cell_1]$, $area_1$)
18: atomicAdd($area[cell_2]$, $area_2$)
19: atomicAdd($centar\_mase[cell_1]$, $centar\_mase_1$)
20: atomicAdd($centar\_mase[cell_2]$, $centar\_mase_2$)
21: **if** $cell_1 \neq cell_2$ **then**
22: atomicAdd($opseg[cell_1]$, $opseg_{12}$)
23: atomicAdd($opseg[cell_2]$, $opseg_{12}$)

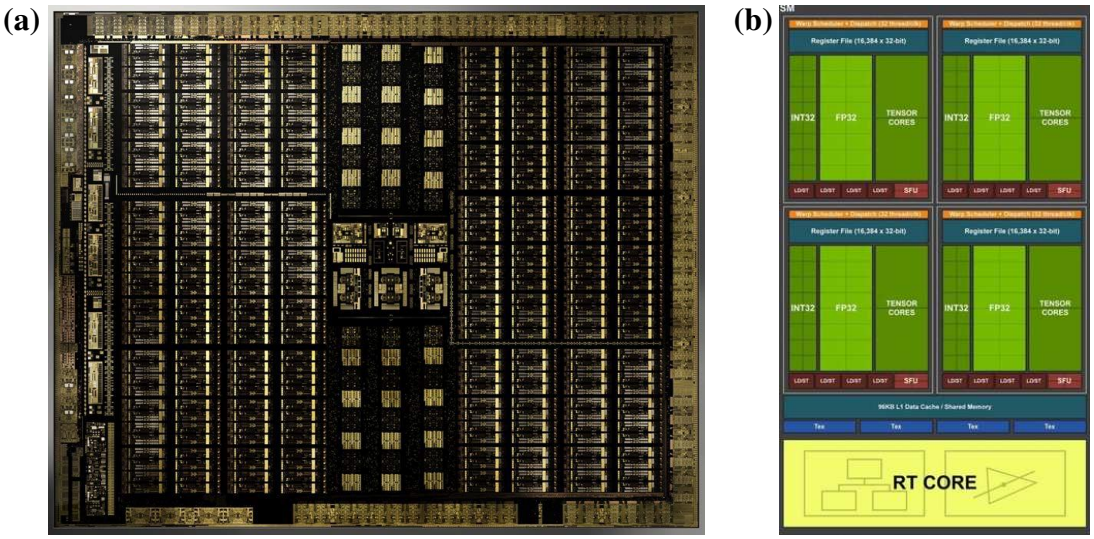
Potrebno je napomenuti da u prikazanom algoritmu 2. izračunati centri masa nisu

podijeljeni s ukupnom površinom Voronoijeve ćelije skupa te je to potrebno izvršiti u drugom kernelu.

4.5. Implementacija i rezultati

4.5.1. Grafička procesorska jedinica

Grafičke procesorske jedinice je vrsta procesora posebno specijaliziranog na izvršavanje paraleliziranih izračuna. Prve grafički procesori bili su samo modelirani su kao među spremnik okvira za iscrtavanje na rasterski zaslon. Prve implementacije grafičkog protočnog sustava su se počele pojavljivati početkom 1990. godine, dok je potpuna implementacije napravljena 1999. godine. Grafičke kartice te generacije imale su protočni sustav te se jednom poslani podaci nisu više mogli mijenjati. Idućih nekoliko generacija je dodavalo mogućnosti programiranja pojedinih dijelova protočnog sustava, kao što su na primjer sjenčari. Izlaskom Nvidijine GeForce 8 serije grafičkih kartice 2006. godine otvara se mogućnost potpunog paralelnog programiranja na grafičkoj procesorskoj jedinici [8]. Na (slici 4.3a) prikazan je komad silicija na kojem se nalazi čip novije generacije Nvidijine grafičke kartice. Na ovom konkretnom čipu se nalazi 72 "streaming multiprocessora" koji svaki sadrži po 128 CUDA jezgre.



Slika 4.3: Dijelovi arhitekture moderne grafičkog procesora: (a) Turing T102 dijagram procesora, (b) jedna jezgra grafičkog procesora

CUDA paralelna platforma

CUDA je paralelna platforma prvi put objavljena 2006. godine kao nadogradnja na programske jezike C, C++ i Fortran. Ova platforma omogućava jednostavno programiranje paralelnih algoritama na procesorima grafičke protočne jedinice. Kako grafička procesorska jedinica može sadržavati i preko tisuću mikroprocesora, CUDA je postala odličan alat za korištenje slobodne računalne snage na mikroprocesorima grafičkog procesora i van domene računalne grafike.

Za ovaj rad je odabrana CUDA kao platforma za programiranje jezgri grafičkog procesora zbog sintakse slične C-u i dostupnog sklopovlja.

Pozivanje kernel programa

Pojedini programe na grafičkom procesoru nazivamo kernelom koji se unutar CUDA programske platforme piše u obliku funkcije anotirane s direktivom __global__.

Kernel se unutar C++ koda poziva slično pozivu "funkcije" ali se odmah nakon naziva kernela dodaje i oznake «<gridDim, blockDim»>. Primjer poziva kernel funkcije:

kernel_fun<<<32, 128>>>(arg1);

Pozvat će "funkciju" koju će izvršavati na 32 bloka gdje je svaki blok veličine 128 dretvi. Stoga ovakav poziv će u pokrenuti 4096 kada se oslobode. Važno je napomenuti da unutar svakog kernela dostupan nam je redni broj dretve koja je pokrenuta te tako možemo raditi radnje na različitim podacima unutar dretve.

Thrust i STDGPU programske biblioteke

Thrust je programska biblioteka za C++ koja omogućava korištenje dinamičkih struktura u memoriji grafičkog procesora, a pruža i veliku kolekciju najkorištenijih algoritama na grafičkom procesoru.

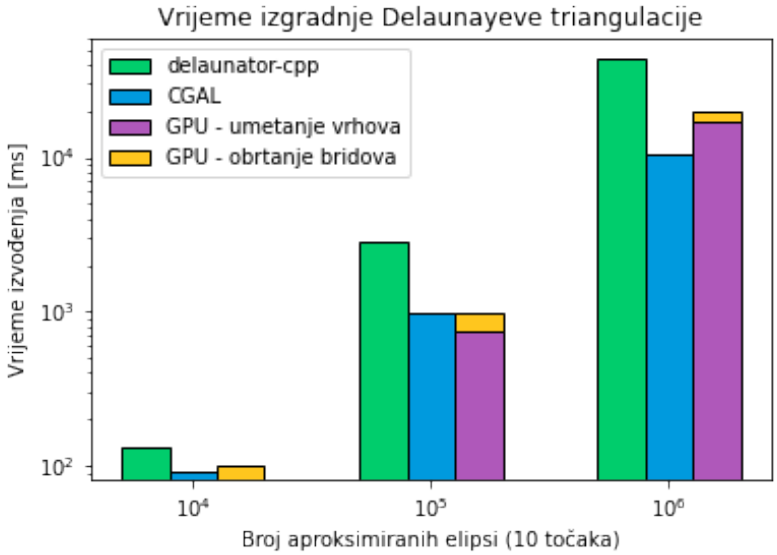
STDGPU [19] [20] je biblioteka koja implementira nekoliko struktura iz standardne biblioteke C++-a te ih je moguće koristiti u kernelu.

Obje biblioteke su korištene u izradi ovog rada, te su pripomogle smanjenju duljine koda.

4.5.2. Evaluacija algoritma

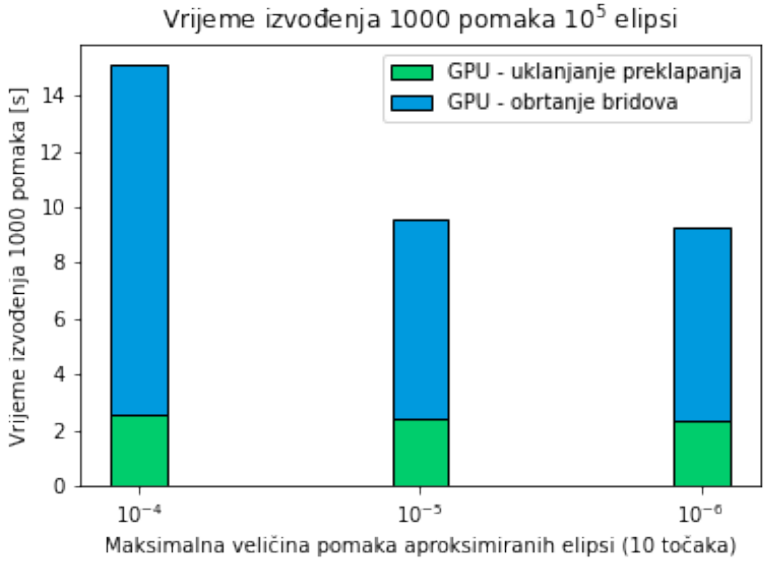
Prilikom evaluacije implementiranog algoritma korišteno je računalo s Intel Xeon E5-2650v2 procesorom i grafičkom karticom NVIDIA RTX 2060 Super. Brzina izvođena je uspoređena najpoznatijom otvorenim bibliotekama za izradu geometrijskih struktura

CGAL. Također algoritam je uspoređen i s delaunator-cpp koji implementacija često korištenog algoritma iz programskog jezika JavaScript.



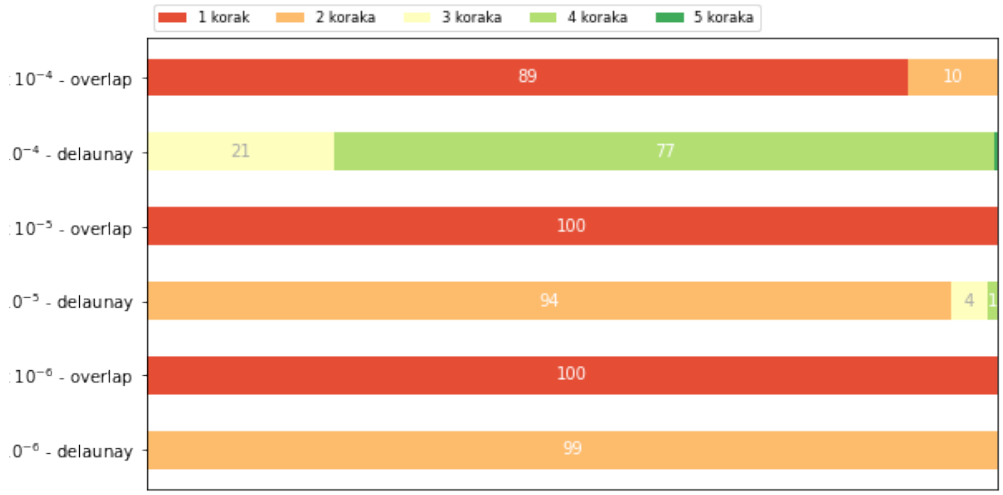
Slika 4.4: Usporedba vremena izvođenja izgradnje Voronoijevog dijagrama skupova za različit broj elipsi

Kako bi usporedili brzine izvođenja algoritama mjerili smo vrijeme potrebno za izgradnju Delaunayeve triangulacije kojoj su generatori elipse. Centri elipsa su odabrani iz uniformne te je pokrenut algoritam izgradnje Delaunayeve triangulacije. Biblioteka CGAL i delaunator-cpp nemaju podršku za izgradnju Voronoijevih dijagrama skupova, ali njihovu brzinu možemo predvidjeli korištenjem jednakog broja točaka generatora koje se koriste u predloženom algoritmu. Na slici 4.4 vidimo da biblioteka CGAL ima najmanje vrijeme izvođenja i duplo je brža od naše implementacije. Zanimljivo je primijetiti da algoritam obrtanja bridova koje popravlja triangulaciju do Delaunayeve u našem algoritmu traje izuzetno malo. Većina vremena potrošena je na izgradnju proizvoljne triangulacije. Ovakvo svojstvo je poželjno jer primarni cilj našeg algoritma je održavanje ispravne Delaunayeve triangulacije kroz niz malih pomaka točaka generatora.



Slika 4.5: Vrijeme izvođenja prilikom održavanja Delaunayeve triangulacije sa različitim pomacima vrhova

U idućem ispitivanju želimo usporediti predloženi algoritam prilikom niza koraka u kojima nad točkama generatorima Delaunayeve triangulacije radimo male pomake. U simulacijama rasta epitelnog tkiva često je potrebno održavati Delaunayevu triangulaciju prilikom niza koraka u kojima stanice rade male pomake. Kako bi ispitali takav način rada predloženog algoritma na ravninu su uniformnom distribucijom izabrani centri elipsa, ali takvi da nisu udaljeni bliže od 0.01. Potom je praćeno vrijeme izvođenja održavanja Delaunayeve triangulacije kroz 1000 pomaka gdje su elipse pomicane s različitim udaljenostima. Na slici 4.5 vidimo da veći pomaci dovode do veće deformacije trianulacije, zbog čega je potrebno napraviti veći broj paralelnih koraka popravke Delaunayeve triangulacije. Za pomake maksimalne duljine 10^{-4} potrebno vrijeme za održavanje Delaunayeve triangulacije tijekom 1000 koraka je oko 15 sekundi. Ako taj rezultat usporedimo s vremenom potrebnim da CGAL bibliotekom izgradimo Delaunayevu triangulaciju 1000 puta vidimo da je ubrzanje našeg algoritma oko 40 puta. Ovakav rezultat je iznimno poželjan jer upravo održavanje Delaunayeve triangulacije prilikom malih pomaka će nam biti potrebno u simulaciji rasta epitelnog tkiva.



Slika 4.6: Očekivana broj koraka prilikom postupka popravke Delaunayeve triangulacije

Slika 4.6 prikazuje očekivani količinu paralelnih koraka prilikom malih pomaka različite veličine iz prethodno provedenog eksperimenta. Vidimo da pri većim pomacima potrebno je napraviti više paralelnih popravaka Delaunayeve triangulacije zbog čega i vidimo razliku u brzini izvođenja na slici 4.5. Važno je napomenuti da pri velikim pomacima ovakav algoritam ne može popraviti triangulaciju do Delunayeve te je potrebno izgraditi Delaunayevu triangulaciju ispočetka.

5. Simulacija rasta MDCK II tkiva generiranih elipsama

U radu [7] je pokazano da ako za generatore Voronoijevog dijagrama uzmemo jezgre MDCK II tkiva pogreška u morfološkim mjerama Voronoijeva dijagrama naspram mjera stvarnog tkiva manja od 10%. Iz čega možemo zaključiti da takva vrsta Voronoijeva dijagrama dobro aproksimira pravo tkivo stanica. Stoga u ovom poglavlju motivirani rezultatima iz prethodnog rada objasnit ćemo potrebne promjene i implementirati nadogradnju nad dvočestičnim modelom rasta stanica tkiva opisanu u radu [14]. Simulaciju ćemo nadograditi koristeći Voronoijeve dijagrame skupova gdje će generatori biti elipse.

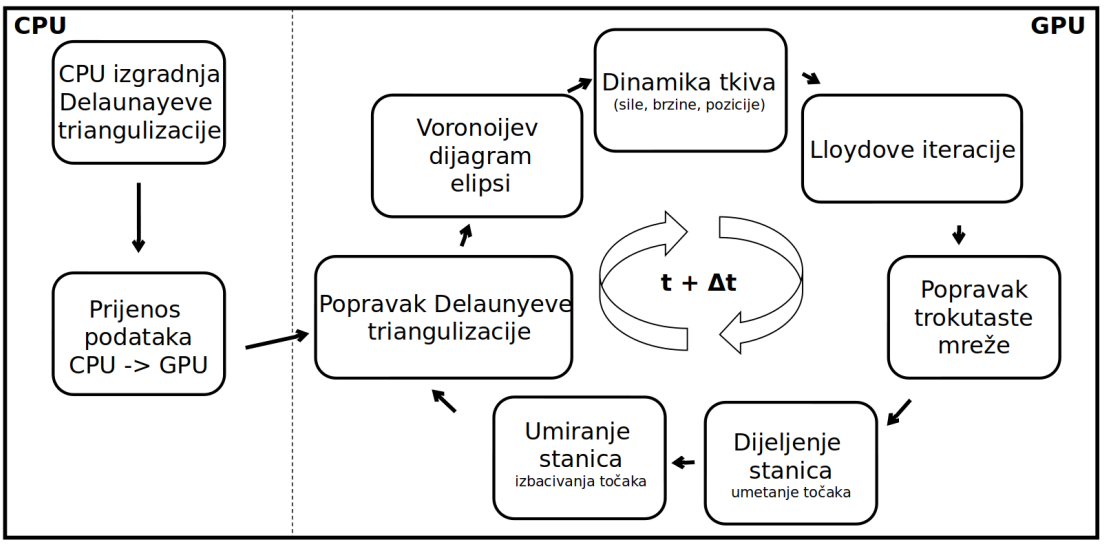
Definicja 5.0.1 (Elipsa u prostoru)**.** Neka je $\vec{c}$ točka u kojoj se nalazi centar elipse u prostoru te neka je α kut koji elipsa zatvara s pozitivnim smjerom osi x. Kut gledamo u smjeru obrnuto od kazaljke na satu počevši od pozitivnog smjera osi x. Neka su a i b dužine poluosi elipse, i neka je $t \in [0, 2\pi]$ Tada možemo definirati parametarsku jednadžbu elipse s:

$$x(t) = a\cos(t)\cos(\alpha) - b\sin(t)\sin(\alpha) + c.x \tag{5.1}$$

$$y(t) = a\cos(t)\sin(\alpha) + b\sin(t)\sin(\alpha) + c.y \tag{5.2}$$

U simulaciji ćemo za jezgru stanice koristiti elipse koje su orijentirane u smjeru čestica pojedine stanice. Zbog jednostavnosti će sve elipse imati jednake veličine poluosi te ćemo taj konstantni parametar dodati u tablicu parametara korištenih u simulaciji (tablica 5.1).

5.1. Tijek simulacije



Slika 5.1: Korakci simulacije rasta tkiva

Simulacija se sastoji od dva glavna dijela (slika 5.1. Prvi dio je inicijalizacije struktura podataka koji se događa na procesoru. Za izgradnju prvotne Delaunayeve triangulacije koristi se implementacija generatora Delaunayeve triangulacije [11] zbog slične strukture trokutaste mreže. Inicijalizirano tkivo s trokutastom mrežom Delaunayeve triangulacije prenosi se na memoriju grafičkog procesora. Na grafičkom procesoru se potom izvršava određeni broj koraka simulacije, te se na kraju podatci tkiva prebacuju na procesor.

5.2. Algoritam popravka Delaunayeve triangulacije okretanjem

Algoritam popravka Delaunayeve triangulacije je identičan algoritmu opisanom u radu [14], ali ga zbog potpunosti ponovno ukratko navodim.

Lokalni Delaunay test nije zadovoljen ako suma kuteva pri vrhovima suprotnim od brida veća od $180°$, jer ako je kut veći od $180°$ to znači da bi brid mogli okrenuti i dobiti lokalno triangulaciju koja ima veći minimalni kut u nekom vrhu od trokuta.

Za svaki brid planarne triangulacije ispitamo valjanost lokalnog Delaunayevog testa. Ako brid nije lokalno Delaunayev nad takvim bridom trebamo provesti okretanje. Kako nam u jednom trokutu Delaunayeve triangulacije više bridova može nezadovoljavati lokalni Delaunayev uvjet moramo paziti da dva brida istog trokuta ne

okrenemo u isto vrijeme. To činimo s nizom s_t koji nam služi kao zastavica ako je neki brid uspio rezervirati susjedni trokut. Brid koji uspije rezervirati oba trokuta okrenemo. Operacija okretanja brida je zamjena vrhova u strukturi trokutaste mreže sa suprotnim vrhovima tog brida, također moramo i popratne podatke u trokutastoj mreži osvježiti.

Algorithm 3 Delaunay triangulacija popravak okretanjem kernel

Require: $Mesh(P)$ je planarna triangulacija $T(P)$

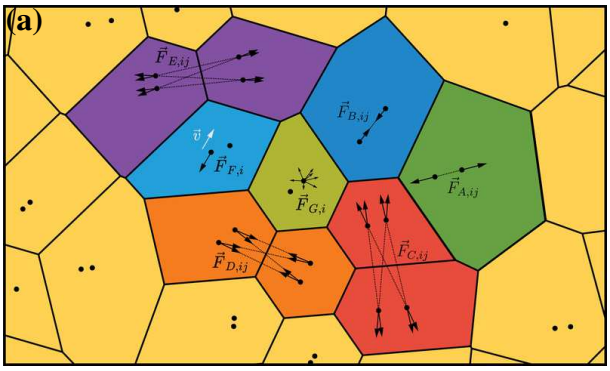
Ensure: $Mesh(P)$ je popravljena triangulacija do $DT(P)$

```
1: procedure DelaunayPopravakOkretanjemKernel(m, s_t, stop)
2:     for idx ← 0 to |m.edges| do
3:         e ← m.edges[idx]
4:         if DelaunayLocalAngleTest(e) then
5:             r_1 ← atomicOr(s_t[e.t_1/3], 1)
6:             r_2 ← atomicOr(s_t[e.t_2/3], 1)
7:             stop ← false
8:             if r_1 == 0 && r_2 == 0 then
9:                 flipEdge(e)
```

Obrtanje se događa paralelno za sve bridove za koje ne vrijedi lokalnost Delaunaya i koji su uspjeli zauzeti svoje susjedne trokute. Pokazano je da će se svaka trokutasta mreža pretvoriti u Delaunayevu triangulaciju u najviše $\log n$ koraka [6].

5.3. Dinamika tkiva

(a)



(b) Sile između čestica unutar stanice

$$\vec{F}_{A,ij} = \frac{B}{(r_{ij} + r_0)^2} \hat{r_{ij}} \tag{5.3}$$

$$\vec{F}_{B,ij} = -\gamma_c \omega(r_{ij}) \hat{r}_{ij} (\vec{v}_j - \vec{v}_i) \hat{r}_{ij} \tag{5.4}$$

$$\omega(r_{ij}) = \begin{cases} 1 - \frac{r_{ij}}{r_t} & r_{ij} < R_t \\ 0 & r_{ij} \geq R_t \end{cases}$$

(c) Sile između čestica susjednih stanica

$$\vec{F}_{C,ij} = \begin{cases} -\frac{f_a \hat{r_{ij}}}{r_{ij}} & r_{ij} \geq R_{pp} \text{ i } r_{ij} \leq R_{pp} * f_d \\ 0 & \text{else} \end{cases} \tag{5.5}$$

$$\vec{F}_{D,ij} = -\begin{cases} \gamma_\perp \\ \gamma_\parallel \end{cases} \times \omega(r_{ij}) \hat{r}_{ij} (\vec{v}_j - \vec{v}_i) \hat{r}_{ij} \tag{5.6}$$

$$\vec{F}_{E,ij} = \begin{cases} f_v((\frac{R_{pp}}{r_{ij}})^5 - 1)\hat{r}_{ij} & r_{ij} \leq R_{pp} \\ 0 & r_{ij} > R_{pp} \end{cases} \tag{5.7}$$

(d) Sile na svaku česticu unutar stanice

$$\vec{F}_{F,i} = -\gamma_s \vec{v}_i \tag{5.8}$$

$$\vec{F}_{G,i} = f_r(\cos\phi, \sin\phi)^T \tag{5.9}$$

Slika 5.3: Sila unutar modela rasta epitelnog tkiva: (a) slikovni prikaz sila modela, (**b**) sile između čestica unutar stanice, (**c**) sile između čestica susjednih stanica stanice, (**d**) sile na svaku česticu unutar stanice

Ideja modeliranja rasta tkiva s česticama predstavljena je u radu [17], gdje je za svaku stanicu korištena jedna čestica. U ovom modelu rasta tkiva opisana je s dvije čestice koje predstavljaju diobeno vreteno stanice, što omogućava jednostavnije definiranje proliferacije. Sile koje djeluju u modelu inspirirane su radom [2], a cijeli model koji ovdje koristimo je detaljnije opisan u [14]. Ovdje ćemo navesti ukratko sile koje dje-

luju u tkivu i način na koje promjena u načinu generiranja Voronoijevu dijagramu utječe na izračun sila.

Između dva diobena vretena u stanici djeluje sila rasta opisana formulom 5.3. Samu jačinu rasta kontroliramo parametrom B. Suprotstavljena sili rasta je sila trenja 5.4 koja nastaje kao posljedica gibanja čestica unutar stanice. Parametar $\omega(r_{ij})$ se koristi za određivanje dosega disipativne sile, dok je γ_c određuje koeficijent sile trenja.

Zbog interakcije s okolinom također su definirane i sila $\vec{F}_{F,i}$ formulom 5.8 koja opisuje trenje s podlogom. Parametar γ_s je koeficijent trenja s podlogom. Također kako bi definirali kaotično gibanje uzrokovano toplinom uvodimo i silu $\vec{F}_{G,i}$ formulom 5.9, gdje je θ proizvoljni kut, a parametar f_r je uzet proizvoljno iz Gaussove raspodijele oko 0 sa standardnom devijacijom kBT.

Način izračuna unutarstaničnih sila u modelu je ostao isti kao i radu [14].

Definirajmo sad i sile koje djeluju između susjednih stanica u Voronoijevim dijagramu. Kako je sad jezgra definirana elipsom, moramo koristiti algoritam 1 iz poglavlja 4 da bi jednostruko izračunali susjede u jednom koraku simulacije.

Za međudjelovanje čestica susjednih stanica definiramo disipativnu silu $\vec{F}_{D,ij}$ zadanu formulom 5.6 i adhezijsku sili $\vec{F}_{C,ij}$ zadanu formulom 5.5. Disipativna sila ima različite parametre izvanstaničnog trenja za paralelnu $\gamma_{\parallel}$ i okomitu komponentu $\gamma_{\perp}$, dok je parametrom f_a definirana jačina adhezijske sile i parametrom R_pp doseg.

Kao posljedica volumne ekskluzije dodajemo silu $\vec{F}_{E,ij}$ koja djeluje na malom prostoru oko same stanice i onemogućuje preklapanje stanica.

5.4. Određivanje površine, opsega i Lloydove iteracije

Prilikom uvođenja Voronoijeva dijagrama generiranog elipsama jezgri mijenja se način raspodijele prostora. Ako su dimenzije elipse blizu krugu, Voronoijev dijagram nije jako različit od Voronoijeva dijagrama generiranog točkama,ali moramo uzeti u obzir da je to samo poseban slučaj i pravilno računati površinu, opseg i centar mase Voronoijeve ćelije generirane elipsom. Algoritam za izračunavanje potrebnih mjera implementiran je i objašnjen u poglavlju /refchap:setVoronoiDijagram te ćemo ga ovdje iskoristiti.

Lloydov algoritam je proces relaksacije Voronoijeva dijagrama do centroidnog Voronoijevog dijagrama kojemu su točke generatori Voronoijevih ćelija u centru mase same ćelije. Ovaj algoritam je definiran za Voronoijeve dijagrame generirane točkama, ali za Voronoijeve dijagrame elipsi nije definiran. U simulaciji se elipse pomiču pos-

tepeno interacijama prema centru mase ali to ne mora dovesti do centroidnog Voronoijevog dijagrama.

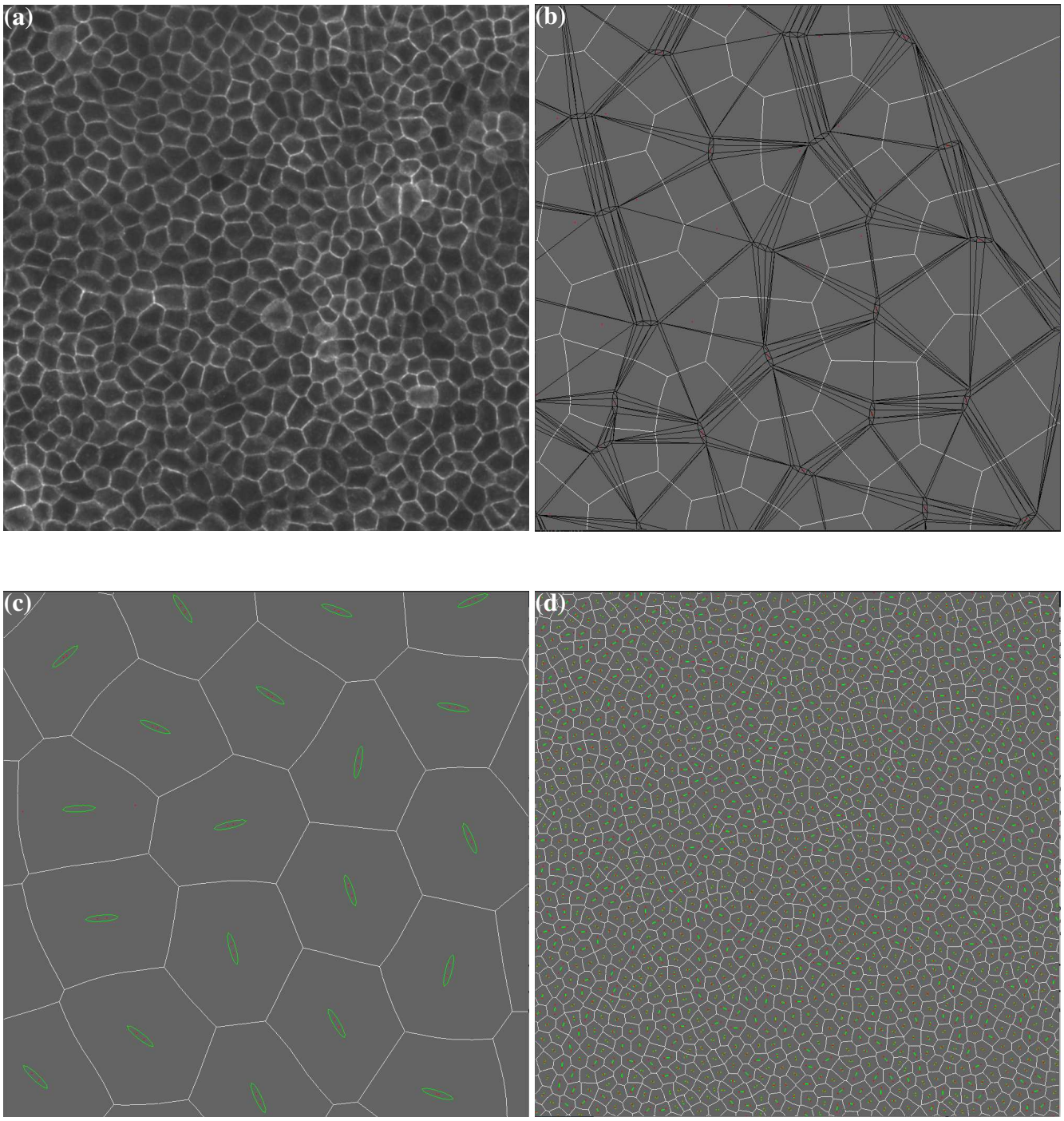
5.5. Rezultati simulacije

Tablica 5.1: Parametri simulacije rasta epitelnog tkiva

Simbol	Značenje	Korištena vrijednost
B	koeficijent rasta	4.0
r_0	ograničavajući faktor rasta	1.12
γ_c	koeficijent unutarstaničnog trenja	1.1
R_t	doseg disipativnih sila	0.8
f_a	koeficijent adhezije	10.0
R_{pp}	doseg potencijala	1.0
f_d	koeficijent vezan uz doseg adhezije	1.4
$\gamma_\perp$	faktor izvanstaničnog trenja okomitog na r_{ij}	0.15
$\gamma_\parallel$	faktor izvanstaničnog trenja paralelnog na r_{ij}	0.2
f_v	koeficijent volumne ekskluzije	150.0
γ_s	koeficijent trenja stanice sa supstratom	0.1
k_BT	intenzitet termalnog utjecaja	0.424
k_a	stopa smrti stanica	0.001
R_{c1}	donja granica stanične diobe	0.75
R_{c2}	gornja granica stanične diobe	1000
r_c	udaljenost novonastale čestice stanične diobe	1.12×10^{-5}
m	masa čestice	5.76×10^{-4}
dt	interval vremena po koraku	10^{-5}
L	jačina Lloydovih iteracija	0.5×10^{-5}
$A_{cutborder}$	površina nakon koje se stanica smatra rubnom	18.34356
A_{border}	površina šesterokuta na rubu	12.22904
$elipsa_a$	duljina a poluosi elipsi jezgri	0.5
$elipsa_b$	duljina b poluosi elipsi jezgri	0.1

Model simulacije je uspješno nadograđen Voronoijevim dijagramom skupova. Parametri korišteni za simulaciju definirani su u tablici 5.1. Slika 5.4b prikazuje Delaunayevu triangulaciju tkiva. Vidimo da su bridovi vrlo blizu jedni drugima što može ponekad izazvati prijelaz vrha i preko dva ili više brida. Takve deformacije trokutaste

mreže ne možemo popraviti algoritmom otklanjanja preklapanja iz rada [14]. Takve slučajeve možemo otkriti i popraviti ponovnim izračunom Delaunayeve triangulacije na procesoru. Ovako popravljanje trokutaste mreže dogodi se svakih otprilike 10000 koraka što je zadovoljavajuća stabilnost. Na slici 5.4c vidimo zelenom bojom obojene jezgre stanica, dok su crvenom bojom označene čestice diobenog vretena. Dobiveni Voronoijev dijagram malo se razlikuje od točkastog Voronoijevog dijagram zbog trenutne definicije veličine elipsa koja zbog trenutnog protokola proliferacije stanica nije moguće postavljati na veći veličinu.



Slika 5.4: Prikaz simulacije rasta epitelnog tkiva

6. Segmentacija jezgri stanica u tkivu

Istraživanje o svojstvima stanica u različitim uvjetima iziskuje obradu velike količine podataka. Prebrojavanje stanica prilikom testiranja lijekova, ili praćenje broja stanica prilikom rastezanja tkiva samo su neki od primjera koje bi automatska segmentacija mogla ubrzati. Velika raznolikost u slikama stanica koji su rezultat različitosti u tipovima tkiva ili razlike u mikroskopima korištenim za slikanje stanica dodatno pridonose težini problema. Zbog navedenih razloga postoji velika potreba za razvojem automatiziranog načina obrade slika stanica.

Prvi automatizirani računalni sustavi za označavanje stanica građeni su sredinom 1950. godina kako bi pripomogli masovnom testiranju na raka vrata maternice. Takvi sustavi su pokušavali su raditi segmentaciju na temelju praga boje. Ovakva metoda temelji se na pretpostavci da stanice imaju različitu boju od podloge na kojoj su slikane. Stoga postavljanjem određenog praga možemo segmentirati stanice [10]. Naravno ovakav algoritam ne može niti približno pokriti cijelu varijabilnost slika tkiva, stoga je potrebno razviti kompleksnije metode. Moderniji sustavi kao što su CellProfiler [9] već izuzetno dobro segmentiraju stanice, zbog čega se često koriste u praksi. Automatskom obradom podataka smanjujemo pristranost koja bi mogla nastati prilikom ručnog označavanja. Trenutno postoji mnogo različitih tehnika kako pristupiti segmentaciji, ali fokus ovog rada će biti područje neuro računarstva i korištenje umjetnih neuronskih mreža.

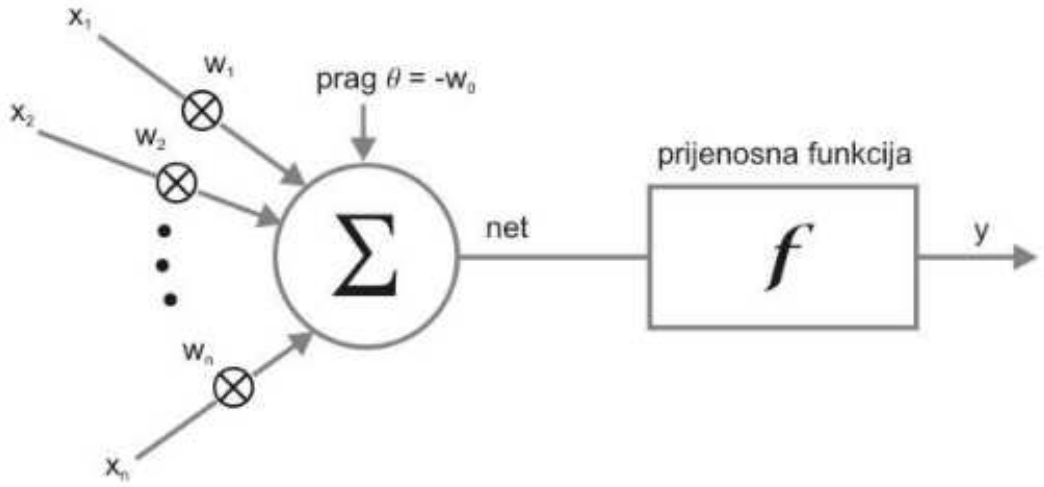
6.1. Umjetna neuronska mreža

Težnja da mogućnosti umjetne inteligencije budu usporedive s inteligencijom čovjekom potaknulo je proučavanje razmišljanja čovjeka. Neuronske mreže zapravo su inspirirane neuronskim stanicama u mozgu čovjeka. Procjena je da čovjek ima više od 10^11 neuronskih stanica u mozgu od kojih je svaka prosječno povezana s 10^4 drugih [3]. Preko tih veza neuroni primaju i šalju različite količine energije. Zanimljivo je da neuroni ne prosljeđuju tu energiju automatski dalje nego je prikupljaju do određene

granice te potom prosljeđuju svoju određenu razinu energije. Smatra se da mozak uči podešavanje jačina veza između neurona. Ovakav jednostavan model dovoljan nam je da probamo simulirati takav način učenja znanja. U nastavku rada će biti obrađeni dijelove samo blisko vezane uz arhitekturu koju ćemo koristiti za segmentaciju stanica.

6.1.1. Umjetni neuron

Prvi matematički model biološkog neurona je McCulloch-Pitts model, tzv. Threshold Logic Unit. Ovakav vrlo jednostavan model je za ulaze $x_1, \ldots, x_n$ ima vrijednosti 0 ili 1. Također težine veza bridova poprimaju vrijednosti 1 ili -1 što predstavlja ekscitacijski ili inhibicijski učinak. Također takav neuron ima prag θ nakon kojeg ako je suma ulaza pomnožena s težinom korespondentnih bridova manja od θ izlaz iz neurona je 0, dok je inače izlaz 1. Ovakav neuron može modelirati različita logička vrata, ali danas se koristi njegova općenitija verzija.



Slika 6.1: Općeniti model umjetnog neurona: slika preuzeta iz rada [3].

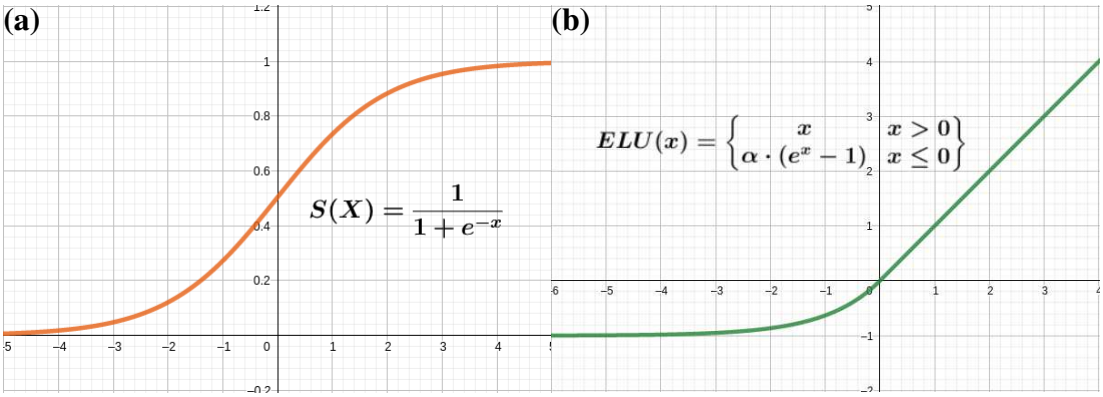
Kod općenitog modela umjetnog neurona (Slika 6.1) ulazne vrijednosti $x_1, x_2, \ldots, x_n$ mogu biti u različitim intervalima kao što su naprimjer [-1,1], [0, 1] itd. Uobičajeno je neuron povezati s dodatnim ulazom koji prag neurona i definiran je definiran s $x_0 = 1$ i $w_0 = \theta$. Suma umnožaka pripadnih težina w_i i ulaza x_i dovedenih na ulaze neurona modelira utjecaj signala dobivenih na dendrite neurona. Dobivena suma je na slici 6.1 označena s net. Na dobivenu net vrijednost potom se primjeni prijenosna funkcija f čiji rezultat je izlaz y iz modela neurona. Ovakvu definiciju možemo definirati i

formulom:

$$y = f(\sum_{i=0}^{n} x_i \cdot w_i) \tag{6.1}$$

6.1.2. Prijenosne funkcije

Prijenosna funkcija ili aktivacijska funkcija uvodi nelinearnost u model neuronske mreže. Uvođenje nelinearnosti omogućuje da neuronska mreža nauči složenije zadatke. Pokazano je da dvoslojna neuronska mreža s nelinearnom aktivacijskom funkcijom može aproksimirati svaku kontinuiranu funkciju $y = f(x)$ do tražene preciznosti [4]. Kako ćemo težine u neuronskoj mreži učiti algoritmom propagacije pogreške unazad, poželjno je da prijenosna funkcija derivabilna.



Slika 6.2: Aktivacijske funkcije: (a) sigmoida, (b) eksponencijalno-linearna

Jedna od najpoznatijih prijenosnih funkcija je svakako sigmoida definirana sa:

$$S(X) = \frac{1}{1 + e^{-x}} \tag{6.2}$$

Derivacija sigmoide je:

$$S'(X) = S(x) \cdot (1 - S(x)) \tag{6.3}$$

Ova prijenosna funkcija danas se rijetko koristi u neuronskim mrežama, osim u slučaju binarne klasifikacije. Kako neuronska mreža u ovom radu treba provesti segmentaciju jezgri slike, u završnom sloju modela korištena je sigmoida.

U ovom radu ćemo koristiti eksponencijalno-linearnu aktivacijsku (ELU) funkciju koja je definirana s:

$$ELU(x) = \begin{cases} x & x > 0 \\ \alpha \cdot (e^x - 1) & x \leq 0 \end{cases} \tag{6.4}$$

Njena derivacija je:

$$ELU'(x) = \begin{cases} 1 & x > 0 \\ \alpha \cdot e^x & x < 0 \end{cases} \tag{6.5}$$

ELU je aktivacijska funkcija koja je slična poznatoj ReLU aktivacijskoj funkciji, s razlikom da u području negativne domene se polagano približava vrijednosti parametra $-\alpha$. Zbog nelinearnog negativnog dijela definicije funkcije treniranje neuronske mreže je sporije, ali kontinuiranost funkcije i odsustvo problema umiranja neurona ponekad uzrokuje bolje rezultate. Za potrebe ovog rada koristiti se vrijednost $\alpha = 1$.

6.2. Funkcija gubitka

Funkcija gubitka koristi se za određivanje pogreške u nadziranom učenju između izlaza modela kojeg promatramo i traženog izlaza. Takva mjera pridodjeljuje vrijednost modelu koliko se izlaz razlikuje od traženog izlaza. U ovom radu koristit ćemo binarnu unakrsnu funkciju definiranu sa:

$$H = -\frac{1}{n}\sum_{i=1}^{n}(y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i) \tag{6.6}$$

y_i je procijenjena vrijednost razreda iz modela, dok je $\hat{y}_i$ stvarna vrijednost razreda iz skupa podataka.

Ovakvu definiciju funkcije gubitka možemo koristiti zbog toga što izlaz iz neuronske mreže treba biti segmentirana slika koja se sastoji od dva razreda. Slikovni element je jezgra stanice ili slikovni element nije jezgra stanice.

6.3. Optimizacija parametara

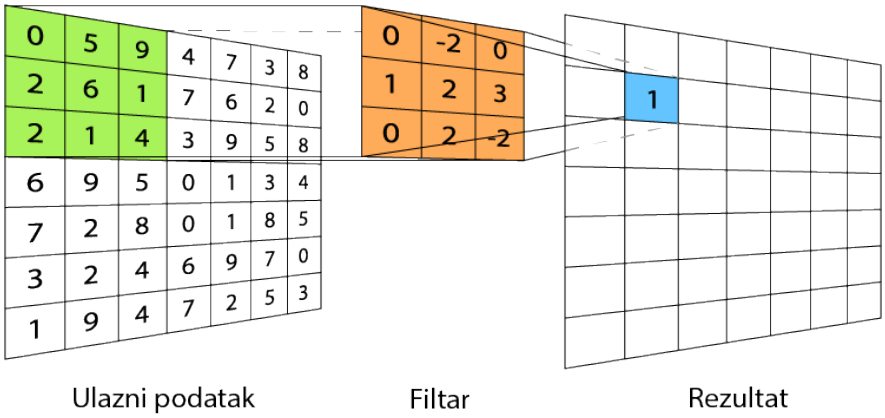
Neuronska mreža se uči na podacima prilagođavanjem parametara težine bridova i vrijednosti praga pojedinog neurona. Najčešći način optimizacije parametara je uporabom metoda koje se temelje na derivacijama svi parametara. Parcijalne derivacije pojedinog parametra mogu se efikasno dobiti algoritmom propagacije pogreške unazad, što nam omogućava korištenje tehnika iz grupe baziranih na metodi stohastičkog gradijentnog spusta. U ovom radu koristi se metoda ADAM koji je nadogradnja na metodu stohastičkog gradijentnog spusta jer koristi informaciju i o prijašnjim gradijentima.

6.4. Konvolucijske neuronske mreže

Višeslojne neuronske mreže često su korišteni alat prilikom obrade velike količine podataka i rješavanja naprednih zadataka u različitim područjima znanosti. Potpuno povezane neuronske mreže s više slojeva i većim brojem neurona povećavaju ekspresivnost modela što je svakako poželjno kod rješavanja sve kompleksnijih zadataka. U različitim područijima veličina potrebnih potpuno povezanih neuronskih mreža prešla je mogućnosti koje su dostupne s trenutnom kompjuterskom moći. Zbog navedenih razloga različite arhitekture su predmet istraživanja.

Konvolucijske neuronske mreže jedan su od većih proboja u području, a donijele su veliki napredak u područjima računalnog vida, obrade slike i prirodnog jezika. Naziv konvolucijske neuronske mreže dolaze od operacije konvolucije koja se dešava u konvolucijskim slojevima mreže. Slojevi sažimanja također su karakteristični za konvolucijske neuronske mreže, dok se potpuno povezani slojevi često koriste pri kraju neuronske mreže. Bitno je napomenuti da pretpostavka konvolucijskih mreža prostorna neovisnost značajki koje promatramo. Na primjer, pozicija čaše na slici nije predmet promatranja nego postojanje čaše [1].

6.4.1. Konvolucijski sloj

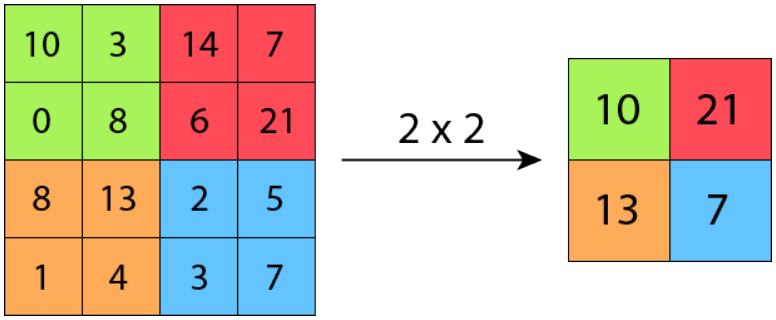


Slika 6.3: Konvolucijski sloj

Konvolucijski sloj sastoji se od skupa filtara koji se primjenjuju na ulazne podatke. Filtar je obično kvadratni element neparne dimenzije koji je manji od volumena ulaznog podatka. Na primjer, za ulazni podatak veličine 128x128x3 možemo koristiti filtar veličine 5x5x3. Konvolucija svakog filtra i ulaznog podatka je izlaz iz konvolucijskog sloja. Prilikom izračuna konvolucije zapravo pomičemo filtar po ulaznom podatku i

računamo umnožak svakog odgovarajućeg elemenata ulaznog podatka i filtra. Jedan takav korak prikazan je na slici 6.4 gdje je izračunata konvolucija gornjeg lijevog kuta ulaznog podatka i filtra veličina 3x3x1. Primijenimo li ovakav postupak s pomakom 1 po cijelom ulaznom podatku dobiti ćemo izlaz iz konvolucijskog sloja koji je manji od ulaznog podatka. Ponekad želimo održati veličinu podatka stoga se oko ulaznog podatka dodaje dopuna koja najčešće ima vrijednost 0. Dakle, konvolucijski sloj definiran je skupom filtara s parametrima koje treniramo na podacima, pomakom i dopunom ako je koristimo.

6.4.2. Sloj sažimanja



Slika 6.4: Sloj maksimalnog sažimanja 2x2

Sloj sažimanja smanjuje prostornu veličinu volumena unutar mreže. Najčešće korišteno je maksimalno sažimanje u kojem se prostor podijeli na potprostore veličine kxk. Kao predstavnik svakog potprostora uzima se najveći element u tom potprostoru. Na slici 6.4. prikazan je sloj maksimalnog sažimanja veličine 2x2 koji će biti korišten i u ovom radu. Slojeve sažimanja koristimo za smanjenje broj parametara u sljedećim slojevima i ubrzanje treniranja neuronsku mrežu.

6.4.3. Sloj transponirane konvolucije

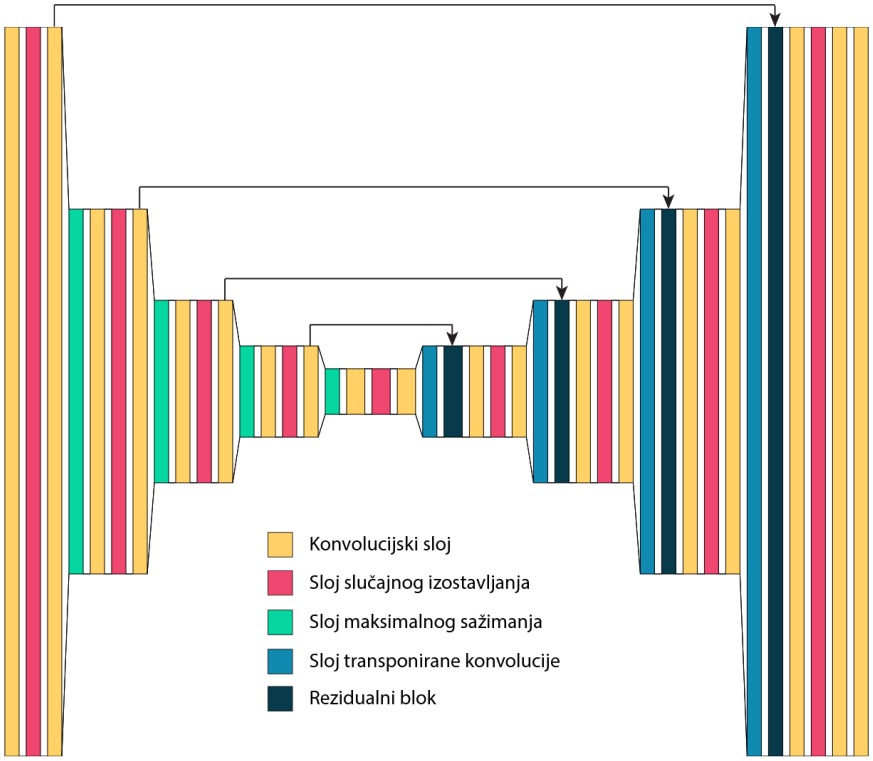
Povećanje prostorne veličine volumena može se postići bilinearnom interpolacijom ili interpolacijom najbližeg susjeda. Ipak u ovom radu korištene su transponirane konvolucije koje se smatraju operacijama obrnutim konvolucijama. Svakako bitno je napomenuti da transponirane konvolucije nisu inverzi operacije konvolucije. Transponirana konvolucija sadrži filtar koji je obično kvadratni element. Filtar se množi sa svakim elementom ulaznog podatka i pritom se pomiče za definirani pomak. Za filtar veličine 2x2 i pomak 2x2 transponirana konvolucija će zapravo pomnožiti svaki element s filtrom te tako dobivene nove elemente spojiti jednu pored druge. Ovakav odabir filtra

i pomaka prostorna dimenzija bit će udvostručena. Također kao i kod konvolucije u ovom sloju parametre filtra učimo na podacima.

6.4.4. U-Net arhitektura

U-Net arhitektura neuronske mreže predložena je u radu [16] za semantničku segmentaciju različitih slika iz područja biomedicine. Predložena arhitektura popularna je zbog mogućnosti da s malim skupom označenih podataka utrenirana mreža dobro segmentira objekte promatranja. Semantička segmentacija je pridodjeljivanje svakog slikovnog elementa nekom od razreda promatranja. Važno je napomenuti da semantička segmentacija ne odjeljuje instance objekta unutar razreda.

Arhitektura U-Neta (Slika 6.5) kombinacija je konvolucijske neuronske mreže i autoenkodera koja se sastoji od dva dijela: kodera koji čini kontrakcijski dio mreže i enkodera koji čini ekspanzijski dio mreže. Kontrakcijski dio mreže sadrži uzastopne blokove od dva konvolucijska sloja s filtrom 3x3 i slojem maksimalnog sažimanja veličine 2x2. Ekspanzijski dio mreže sadrži uzastopne blokove koji se sastoje od transponirane konvolucije, rezidualnog bloka i dva konvolucijska sloja. Filtar transponirane konvolucije je veličine 2x2 s pomakom 2x2, a filtar konvolucijskog sloja je veličine 3x3. Rezidualni blokovi povezuju slojeve u ekspanzijskom dijelu mreže s kontrakcijskim dijelom čime se pospješuje prijenos informacije o lokalizacija objekata koja se izgubila u kontrakcijskom dijelu mreže. U arhitekturu su dodani i slojevi slučajnog izostavljanja kako bi se spriječila pretreniranost neuronske mreže.



Slika 6.5: U-Net arhitektura neuronske mreže

6.5. Implementacija i rezultati

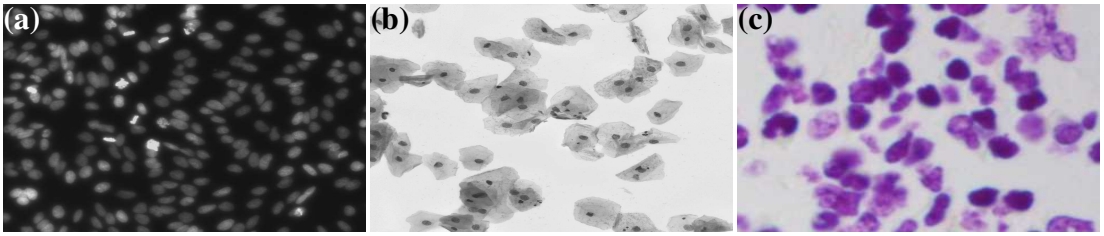
6.5.1. Programsko okruženje

Za implementaciju modela automatskog označavanja jezgri stanica korišten je programski jezik Python s bibliotekama za obradu slika i strojno učenje. Biblioteka TensorFlow zajedno s bibliotekom Keras korištena je za ostvarenje modela U-Net neuronske mreže kao i za treniranje modela nad skupom podataka. Biblioteka scikit-image korištena je za obradu slika i za Watershed algoritam, a je biblioteka opencv korištena za prilagođavanje elipsa na segmentirane jezgre stanica.

6.5.2. Skup podataka

Podaci korišteni za treniranje neuronske mreže dobiveni su s natjecanja Data Science Bowl. Data Science Bowl je godišnje natjecanje održano na platformi Kaggle s ciljem rješavanja teških problema iz različitih područja obrade podataka. Na natjecanju održanom 2018 godine postavljen je problem automatizacije segmentiranja jezgri stanica različitih tkiva. U skupu podataka pretežno su slike veličine 256x256 slikovnih

elemenata, ali postoji i dio većih veličina. Skup podataka podijeljen je u skup za treniranje veličine 670 slika i skup za testiranje veličine 3019 slika. Svaka slika iz skupa za treniranje također sadrži i pripadne maske svake jezgre na slici.

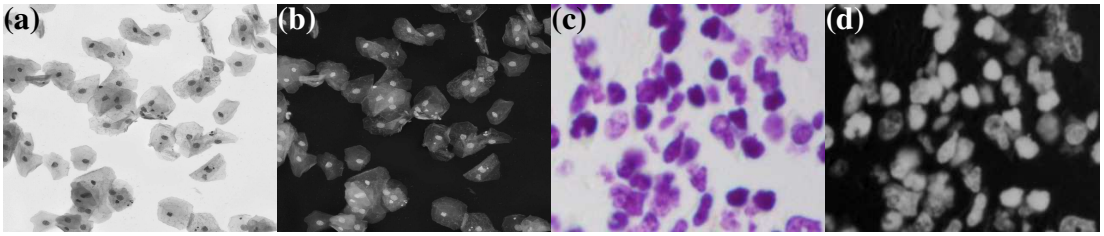


Slika 6.6: Različita tkiva u skupu podataka

Skup podataka karakterizira velika raznolikost, zbog čega se često i nakon natjecanja koristi prilikom procjene koliko dobro neki model generalizira. Na slici 6.6 prikazani su predstavnici grupa koji su najčešći u skupu podataka. Slike iz grupe 6.6a su najčešće i čine preko 75% skupa podataka, dok je grupa slika 6.6c druga po zastupljenost. Također u skupu podataka se u manjem broju nalaze i slike tkiva koje nisu prikazane na slici 6.6.

6.5.3. Predobrada i postobrada podataka

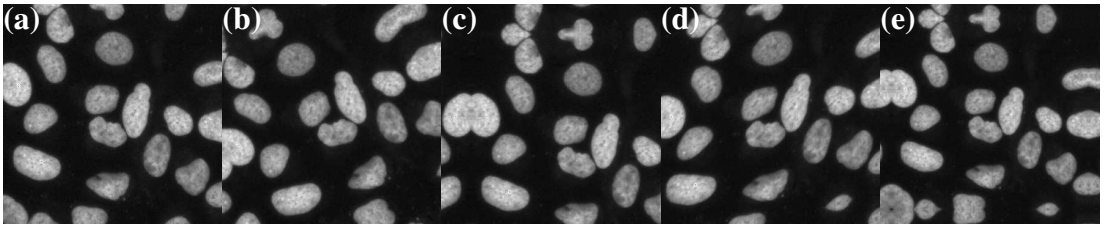
Skup podataka za treniranje sastoji se od slika različitih veličina što nije povoljno kada koristimo neuronsku mrežu arhitekture U-Net. Kako je odabrano da model na ulazu prima slikovne podatke veličine 256x256 slikovna elementa sve su slike zbog jednostavnosti skalirane na traženu veličinu. Izlaz iz U-Net arhitekture u ovom slučaju maska veličine 256x256 slikovna elementa što znači da ćemo dobivene maske trebati skalirati na originalnu veličinu početne slike.



Slika 6.7: Invertiranje boje na slikama svijetlih tkiva

Kako se skup podataka sastoji pretežno od tamnih slika gdje se jezgre stanica ističu svjetlijom bojom (Slika 6.6a), nad svjetlijim slikama (Slika 6.7a i Slika 6.7c primijenjeno je invertiranje boje. Također zbog jednakog razloga slike u boji pretvorene su crnobijele i normalizirane. Rezultati ovog postupka vidljivi su na slikama 6.7b i 6.7d.

Skup podataka za treniranje sastoji se od relativno malog broja slika zbog čega je bitno napraviti augmentaciju podataka. Augmentacija podataka je postupak kreiranja novih slika uz pomoć jednostavnih transformacija slika iz skupa podataka. U ovom radu odlučeno je napraviti nekoliko različitih transformacija prikazanih na slici 6.8. Korištene transformacije su: rotacija, translacija, smicanje, povećavanje i smanjivanje.



Slika 6.8: Uzorci koje nalazimo u prirodi: (a) oklop morske kornjače, (b) pigment kože žirafa, (c) nervatura lista

Svaka slika maske u dobivenim podacima označuje točno jednu jezgru na slici. Kako bi mogli trenirati neuronsku mrežu UNet maske su spojene u zajedničku masku za svaku sliku.

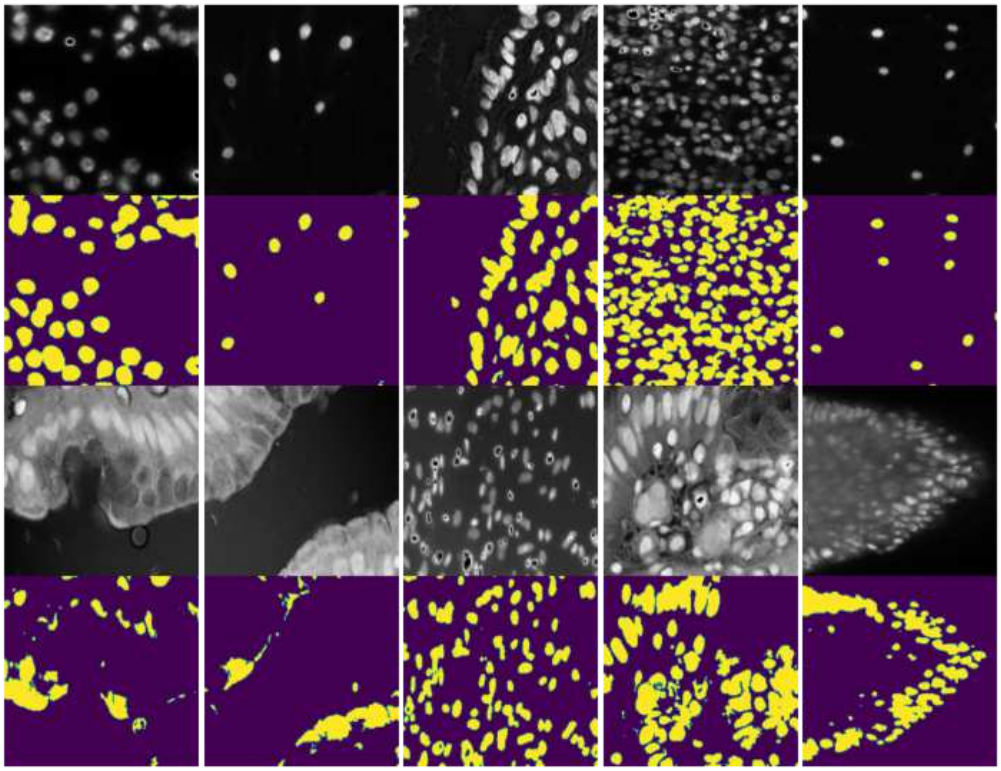
6.5.4. Rezultati

Prilikom učenja neuronske mreže U-Net korišten je ADAM s korakom učenja 10^{-4}. Težine neuron postavljene su prema odrezanoj normalnoj distribuciji, a za aktivacijsku funkciju koristi se eksponencijalno-linearna funkcija ELU. Skup podataka za treniranje podijeljen je u omjeru 90:10 kako bismo dobili skup za validaciju koji je korišten za ocjenjivanje modela prilikom promjena različitih hiperparametara neuronske mreže. Učenje neuronske mreže provedeno je u grupama veličine 32, dok je u svakoj epohi korišteno 128 koraka. Vrijeme potrebno za treniranje jedne epohe bilo je oko 1 i pol minute. Prilikom učenja praćene su metrike točnosti i funkcija gubitka. Prilikom problema segmentacije možemo imati visoku točnost, a da model uopće dobro ne označava tražene objekte. Takva situacija se može dogoditi kada su objekti koje želimo označiti relativno mali naspram pozadine zbog čega i prazno označena slika ima visoku razinu točnosti. Stoga je praćena i metrika Dice koeficijent koja se za logičke podatke izračunava prema formuli:

$$DSC = \frac{2 * TP}{2 * TP + FP + FN} \tag{6.7}$$

TP - točno pozitivan, FP - lažno pozitivan, FN - lažno negativan

Na kraju treniranja dobili smo točnost od 0.9772 na skupu za validaciju. Također funkcija gubitka iznosila je 0.05392, dok je Dice koeficijent iznosio 0.8812. Iz ovih metrika možemo uočiti da je neuronska mreža zadovoljavajuće naučila označavati jezgre stanica. Kako bi se spriječila pretreniranost u model su dodani slojevi izbacivanja što je rezultirano da je funkcija gubitka na skupu za treniranje prilikom treniranje neuronske mreže bila veća nego na skupu za validaciju. Također kako bi spriječili pretreniranost zapamćeni su samo modeli koji su poboljšavali funkciju gubitka na skupu za validaciju.



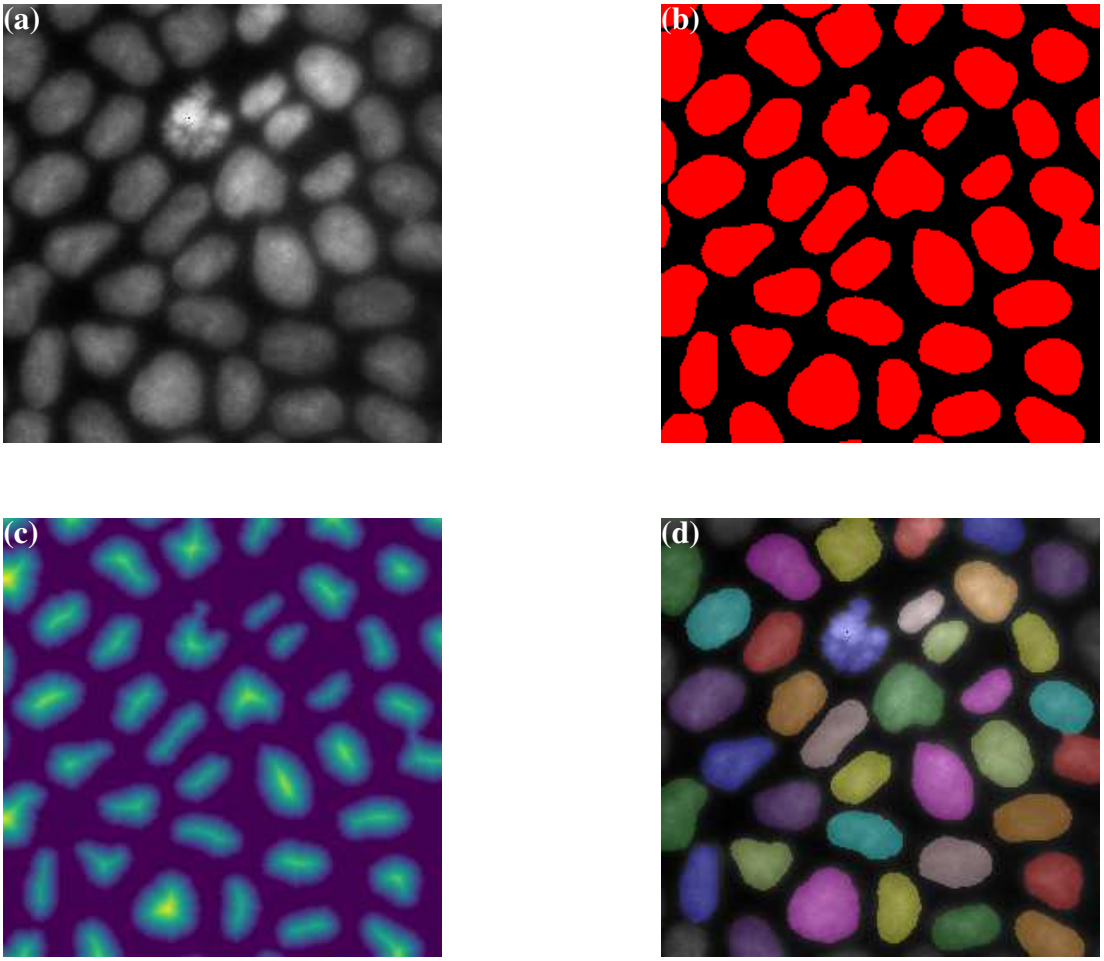
Slika 6.9: Rezultati označavanja slika na skupu za testiranje

Slika 6.9 prikazuje rezultate označavanja slika na skupu za testiranje. Možemo uočiti da je neuronska mreža uspješno naučila označavati gornji dio slika na kojima se nalazi tkivo male gustoće i jezgre stanica su eliptičnog oblika s velikim kontrastom naspram neposredne okoline. Takvo tkivo je upravo i MDCK II tkivo za koje smo izradili simulaciju rasta. S druge strane možemo uočiti da slike donjeg reda nisu ispravno označene zbog čega ovakvu mrežu ne možemo koristiti za analizu takvih tkiva. Slike koje su loše označene imaju jako veliku gustoću tkiva ili su jezgre stanice okružene svijetlim dijelovima ostatka stanice. U samom skupu podataka ovakvih slika tkiva je bilo izrazito malo što je vjerojatno i razlog za slabu učinak modela.

Rezultati modela također su i predani na platformu Kaggle za evaluaciju gdje je rezultat bio 0.38466, što je zapravo i očekivani rezultat za jednostavni model U-Net arhitekture. Ovakav rezultat odgovara poziciji oko 480 mjesta od ukupno 742 tima koji su se originalno natjecali na 2018 Data Science Bowl natjecanju. Najbolje rangirani tim osvojio je natjecanje s rezultatom 0.63164, gdje su koristili U-Net arhitekturu spojenu s nekoliko drugih neuronskih mreža. Kao metrika za evaluaciju modela korištena je mean Iou koja je kvocijent presjeka semantičke maske i oznake te unije oznake i maske.

Watershed algoritam

Prilikom korštenja U-Net arhitekture dobivamo semantičko označavanje jezgara stanica u kojem je označeno gdje se nalaze jezgre stanica, ali instance stanica nisu odvojene. Ovakav izlaz iz modela nije pogodan prilikom izračunavanja različitih mjera na razini jedne stanice. Stoga je potrebno semantičko označavanje pretvoriti u označavanje instanci. Ovakav zahtjev možemo postići korištenjem Watershed algoritma što je prikazano na slici 6.10.



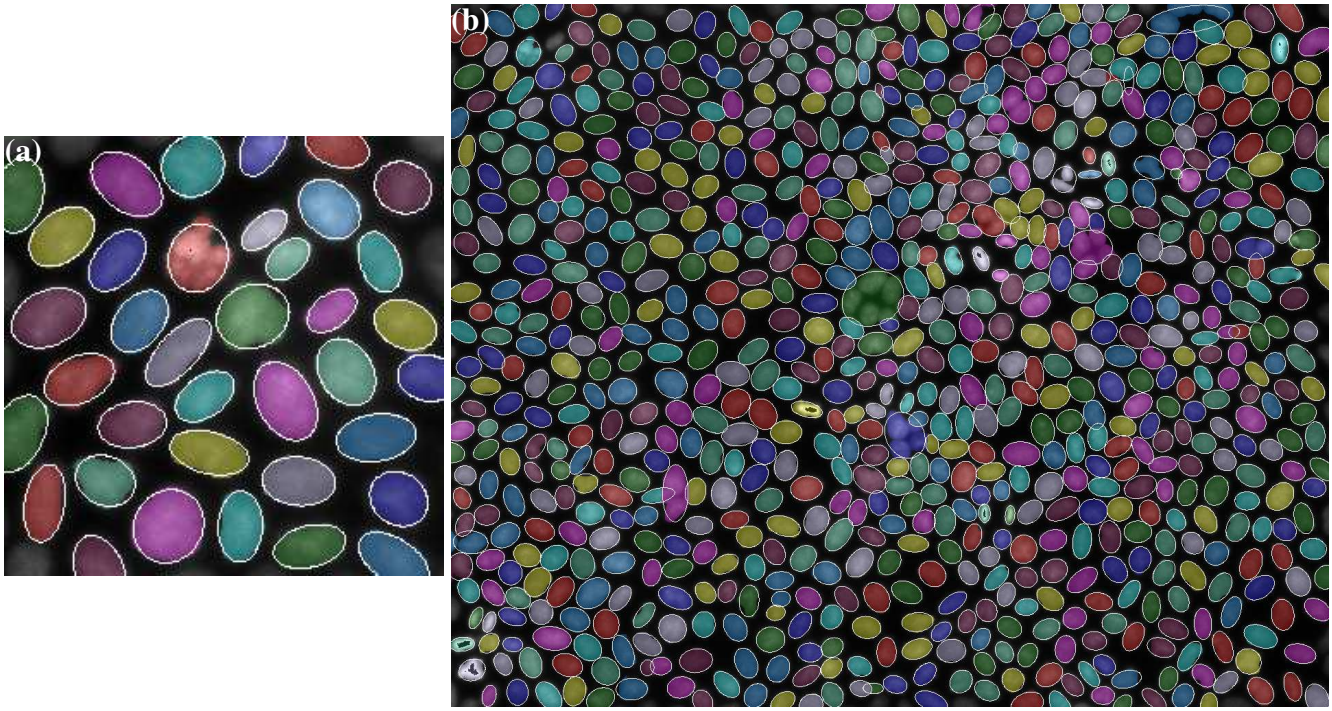
Slika 6.10: Primjena algoritma Watershed: (a) slika koju želimo označiti, (b) označena slika primjenom neuronske mreže, (c) udaljenost centra do ruba označene jezgre, (d) odvojene jezgre algoritmom Watershed

Postupak označavanja instanci započinjemo segmentacijom slika jezgri stanica (Slika 6.10a) koje dovedemo na ulaz utrenirane neuronske mreže. Rezultat segmentacije je slika 6.10b. Za takvu dobivenu sliku označimo središta segmentiranih dijelova ma-

ske uz pomoć algoritma peak_local_max iz biblioteke SciPy. Ovaj algoritam traži maksimume udaljenosti za dijelove maske do ruba maske te tako označava središta segmentiranih dijelova. Potom izračunamo mapu udaljenosti od pojedinog dijela te dobivamo sliku 6.10c. Nad ovakvim podatkom pokrenemo algoritam Watershed koji zapravo postepeno popunjava udoline u mapi udaljenosti. Kao rezultat dobivena je slika 6.10d na kojoj vidimo da je postupak segmentacije instanci uspješno proveden. Ovakav postupak je izrazito uspješan kod tkiva s malom gustoćom dok pri velikim gustoćama nismo u mogućnosti odvojiti pojedine jezgre stanica.

Prilagođavanje elipsa na segmentirane jezgre

Kako smo u ovom radu proučavali izgradnju simulacije rasta epitelnog tkiva u kojem su jezgre stanica modelirane elipsama provest ćemo i prilagodbu elipsi na označene slike tkiva.



Slika 6.11: Prilagođene elipse na jezgre stanica

Postupak prilagodbe elipsi koristi označene slike nakon obrade Watershed algoritmom, gdje se za svaku označenu jezgru stanice traži aproksimacija jezgre elipsom uz pomoć algoritma iz biblioteke opencv. Na slici **??** prikazana su dva primjera prilagođavanja elipse na MDCK II tkivo. Vidimo da rezultati u dobroj mjeri aproksimiraju

jezgre stanica, zbog čega ih možemo kao početnu konfiguraciju koristiti i za samu simulaciju rasta epitelnog tkiva.

7. Zaključak

Tema ovog završnog rada je proučavanje MDCK epitelnog tkiva. Kroz tri dijela povezana su različita područja kao što su simulacije čestica, neuro-računarstvo, računalna geometrije, biologija i fizika.

U prvom dijelu završnog rada definirani su Voronoijevi dijagrami skupova i Delaunayevi dijagrami koji su usko povezani sa strukturom stanica u tkivu. Proučeni su različiti načini izgradnje Voronoijevih dijagrama, a potom su uvedeni Delaunayevi dijagrami kao dualni grafove Voronoijevih dijagrama. Proširena je definiciju Voronoijevih dijagram s generatorima koji su podskupovi ravnine. Implementiran je efikasan paralelan način izgradnje Voronoijevih dijagrama na grafičkoj procesnoj jedinici i algoritmi izračuna morfoloških mjera u Voronoijevim dijagramima. Algoritmi su implementirani u CUDA programskom okruženju.

Drugi dio završnog rada posvećen je nadogradnji modela rasta epitelnog staničnog tkiva. Razvijeno je grafičko sučelje za prikaz simulacije rasta tkiva s Voronoijevim dijagram i prilagođeni su algoritmi u simulaciji da omogućuju predstavljanje staničnih jezgri elipsama.

Zadnji dio rada završnog rada pokušali smo segmentirati jezgre stanica tkiva putem dostupnog otvorenog skupa podataka. Proučen je dio teorijske pozadine neuronskih mreža i trenirana je neuronsku mrežu arhitekture U-Net. Segmentirane su slike MDCK tkiva koje su referentni podaci prilikom izrade simulacije rasta tkiva. Dobiveni model je odlična podloga za nastavak razvoja programa za segmentaciju jezgri.

LITERATURA

[1] Saad Albawi, Tareq Abed Mohammed, i Saad Al-Zawi. Understanding of a convolutional neural network. U *2017 International Conference on Engineering and Technology (ICET)*, stranice 1–6. Ieee, 2017.

[2] Markus Basan, Jacques Prost, Jean-François Joanny, i Jens Elgeti. Dissipative particle dynamics simulations for biological tissues: rheology and competition. *Physical biology*, 8(2):026014, 2011.

[3] Bojana Dalbelo Bašić, Marko Čupić, i Jan Šnajder. Umjetne neuronske mreže. *Fakultet elektrotehnike i računarstva*, 2008.

[4] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[5] Mark De Berg, Marc Van Kreveld, Mark Overmars, i Otfried Schwarzkopf. Computational geometry. U *Computational geometry*, stranice 1–17. Springer, 1997.

[6] Ferran Hurtado, Marc Noy, i Jorge Urrutia. Parallel edge flipping. U *CCCG*, 1998.

[7] Sara Kaliman, Maxime Hubert, Carina Wollnik, Lovro Nuić, Damir Vurnek, Simone Gehrer, Jakov Lovrić, Diana Dudziak, Florian Rehfeldt, i Ana-Sunčana Smith. On the mechanical regulation of epithelial tissue homeostasis. *bioRxiv*, 2021.

[8] Chris McClanahan. History and evolution of gpu architecture. *A Survey Paper*, 9, 2010.

[9] Claire McQuin, Allen Goodman, Vasiliy Chernyshev, Lee Kamentsky, Beth A Cimini, Kyle W Karhohs, Minh Doan, Liya Ding, Susanne M Rafelski, Derek Thirstrup, et al. Cellprofiler 3.0: Next-generation image processing for biology. *PLoS biology*, 16(7):e2005970, 2018.

[10] Erik Meijering. Cell segmentation: 50 years down the road [life sciences]. *IEEE Signal Processing Magazine*, 29(5):140–145, 2012.

[11] msokalski. delabella. `https://github.com/msokalski/delabella`, 2018.

[12] Ashwin Nanjappa. *Delaunay triangulation in R3 on the GPU*. Doktorska disertacija, 01 2012.

[13] Atsuyuki Okabe. Spatial tessellations. *International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology*, stranice 1–11, 2016.

[14] Luka Rogić i Lovro Nuić. Utjecaj optimizacije staničnih oblika na dvočestični model rasta epitelnih tkiva. 2020.

[15] Guodong Rong i Tiow-Seng Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. U *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, stranice 109–116, 2006.

[16] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. U *International Conference on Medical image computing and computer-assisted intervention*, stranice 234–241. Springer, 2015.

[17] Néstor Sepúlveda, Laurence Petitjean, Olivier Cochet, Erwan Grasland-Mongrain, Pascal Silberzan, i Vincent Hakim. Collective cell motion in an epithelial sheet can be quantitatively described by a stochastic interacting particle model. *PLoS Comput Biol*, 9(3):e1002944, 2013.

[18] Jakov Šiller Lovrić. *Statistička analiza strukture Voronojevih ćelija generiranih slučajno pakiranim elipsama i primjena na stanična tkiva*. Doktorska disertacija, University of Zagreb. Faculty of Science. Department of Mathematics, 2016.

[19] P. Stotko. stdgpu: Efficient STL-like Data Structures on the GPU. arXiv:1908.05936, Kolovoz 2019. URL `https://arxiv.org/abs/1908.05936`.

[20] P. Stotko, S. Krumpen, M. B. Hullin, M. Weinmann, i R. Klein. SLAMCast: Large-Scale, Real-Time 3D Reconstruction and Streaming for Immersive Multi-

Client Live Telepresence. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2102–2112, Svibanj 2019.

Simulacija modela rasta epitelnog tkiva

Sažetak

Radi boljeg razumijevanja rast tkiva često modeliramo jednoslojni epitel, čiji je jedan od predstavnika Madin-Darby Canine Kidney (MDCK) tkivo. U ovom završnom radu razvijen je paralelni algoritam izgradnje Voronoijeva dijagrama skupova. Implementirani algoritam primijenjen je na jezgru oblika elipse u dvočestičnom modelu stanice. Simulaciji rasta MDCK tkiva izvedena je disipativnom dinamikom čestica uz optimizaciju strukture stanice Lloydovim algoritmom. Nadalje, provedeno je učenje U-Net konvolucijske neuronske mreže za segmentaciju jezgri stanica tkiva.

Ključne riječi: Voronoijev dijagram skupova, epitelna tkiva, segmentacija jezgri stanica, disipativna dinamika čestica, GPGPU programiranje

Simulation of Epithelial Tissue Growth

Abstract

In order to better understand tissue growth, a single-layer epithelium, represented by Madin-Darby Canine Kidney (MDCK) tissue, is often modeled. In this final paper, a parallel algorithm for constructing the Set Voronoi tessellation is developed. The implemented algorithm is applied to an ellipse-shaped nucleus in a two-particle cell model. Simulation of MDCK tissue growth was performed by dissipative particle dynamics with optimization of cell structure by Lloyd's algorithm. Furthermore, the U-Net convolutional neural network is employed to predict tissue cell nucleus segmentation.

Keywords: Set Voronoi Tessellation (SVT), epithelial tissue, nuclei segmentation, DPD simulation, GPGPU programming