

Proceduralno generiranje celic z uporabo statističnega modela

Erik Rakušek, er8261@student.uni-lj.si

Povzetek

V tem članku je predstavljena implementacija statističnega modela za generiranje celic. Statistični model smo izdelali na podlagi mikroskopskih slik celičnih membran. Iz slik smo izluščili posamezne celice z uporabo različnih metod procesiranja slik. Nato smo glede na članek [VTL*04] ustvarili statistični model s pomočjo katerega smo generirali nove celice. Kljub temu da smo delali s pomankljivimi podatki, se nam je uspelo približati realni obliki celic.

Kategorije in ključne besede: [Računalniška grafika]: procesiranje slik—proceduralno generiranje modelov

1. Uvod

V sklopu predmeta Napredna računalniška grafika (NRG) smo implementirali statistični model za generiranje oblik celičnih membran na podlagi članka Vrtovec et al. [VTL*04]. Model smo implementirali v programskejem jeziku Python. Celoten projekt je dostopen na [GIT] z MIT licenco. Model je zasnovan na podlagi podatkov o oblikih celičnih membran, zato smo morali najprej pridobiti te podatke. Podatke o celičnih membranah smo izluščili iz slik, ki smo jih pridobili iz baze podatkov o celicah Allan Cell institute [CEL]. V tej bazi so na voljo slike celičnih prelezov, kar nam je prišlo prav pri rekonstrukciji celic v trodimenzionalnem okolju.

Iz pridobljenih slik smo morali najprej izluščiti posamezno celico in ji določiti nekaj mejnih točk na njenem robu (angl. *landmarks*). Z uporabo teh točk smo nato ustvarili povprečno obliko celice. Glede na povprečno obliko celice oziroma povprečne koordinate mejnih točk smo ustvarili kovariančno matriko nad katero smo nato izvedli algoritmom PCA. Ta algoritem nam vrne lastne vrednosti in lastne vektorje, ki jih nato lahko uporabimo pri generiranju mejnih vrednosti novih celic. Pri generiranju novih celic dodamo še neko naključno vrednost, ki povzroči razlikovanje na novo generiranih celic med seboj. Z mejnimi vrednostmi nove celice lahko le-to zapišemo v binarno .raw datoteko, da si jo lahko ogledamo v programu kot je [LBM18]. Takšno binarno datoteko smo ustvarili tako, da smo se sprehodili čez vse rezine celice in v binarno datoteko zapisali stanje vsake točke v objektu. Torej ali je v notranosti celice ali izven nje.

2. Pridobivanje podatkov

Slike celic smo pridobili s spletnne strani Allan Cell institute [CEL]. Podatki, ki jih ponujajo na tem spletnem mestu so zelo obsežni, zato smo izluščili le tiste slike, ki smo jih dejansko potrebovali za analizo celičnih membran. Najprej smo s pomočjo orodja Fiji [SACF*12] pregledali pridobljene podatke in ugotovili, da se slike celičnih membran nahajajo na kanalu 4, vidne pa so približno med rezinami 15 in 55. To znanje smo uporabili v pythonu tako, da smo z uporabo knjižnice *czifile* prebrali podatkovno datoteko in filtrirali vse nepomembne podatke, tako da so nam ostali le uporabni podatki oziroma slike.

3. Procesiranje slik

Ker so bile slike zelo raznolike med seboj nam ni uspelo najti popolnoma avtomskega generalnega načina ekstrakcije mejnih točk za vse celice. Zato smo se odločili, da si pomagamo z ročno analizo celic in tako najprej ročno določimo koordinate središča celice, katero obdelujemo. Ročno določeno središče celice mora biti v notranosti celice v vseh rezinah za boljše rezultate. Nato smo se lotili obdelave slik vsako rezino posebej. Za procesiranje slik smo uporabili OpenCV knjižnico v Pythonu. Najprej smo sliko pretvorili v sivinsko sliko. Nato smo nad sliko izvedli algoritmom *FloodFill* takoj za tem pa še *Dilate* s čimer smo dobili sliko z rdeče pobarvano celico brez šuma v njeni notranosti.

Sliko nato filtriramo tako, da odstranimo vse sivine na sliki. Tako dobimo binarno masko, kjer je celica pobarvana s črno barvo, vse ostalo pa z belo. Nad takšno sliko smo nato izvedli *Canny* algoritmom za iskanje robov, ki nam vrne sez-

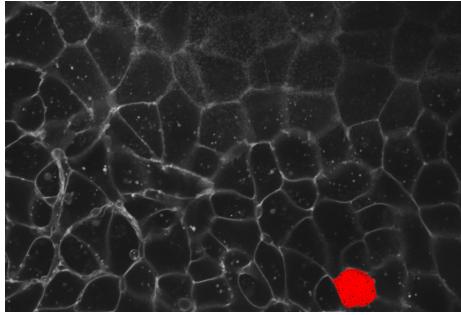


Figure 1: Slika celice po uporabi *FloodFill*.

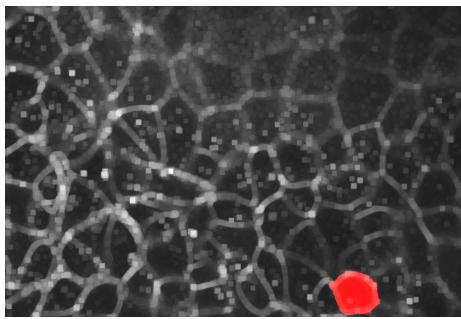


Figure 2: Slika celice po uporabi *Dilate*.

nam vseh točk na robu celice. Iz tega seznama smo nato vzeli naključnih n mejnih točk (pomembno je da z vsake rezine vzamemo enako število mejnih točk).

Ker slike rezin celic niso bile popolne smo imeli veliko težav pri iskanju posamezne celice znotraj slike oziroma pri uporabi algoritma *FloodFill*. Zato smo pri slikah, kjer s *FloodFill* nismo dobili celice ali pa nam jih je označilo preveč, uporabili koordinate mejnih točk iz prejšnje rezine.

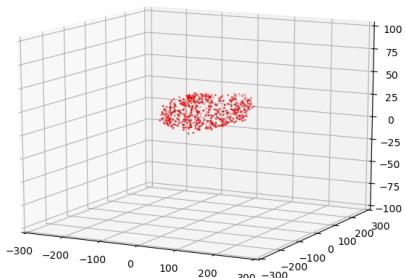


Figure 3: Mejne točke celice.

4. Statistični model oblik celic

V prejšnjem koraku smo vse mejne točke neke celice shranjevali v seznam točk. Ko smo obdelali in izluščili mejne točke dovolj velikega vzorca celic smo začeli z izdelavo statističnega modela oblike celice. Zato smo najprej izračunali povprečno obliko celice po formuli 1.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad (1)$$

kjer je x_i seznam mejnih točk posamezne celice.

Nato smo z uporabo formule 2 tvorili kovariančno matriko nad katero izvedemo PCA algoritem. Tako izračunamo lastne vrednosti in lastne vektorje matrike C .

$$C = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^\top \quad (2)$$

Nato lahko izračunamo mejne točke nove celice po formuli 3.

$$x' = \bar{x} + (r * d * \sqrt{\lambda_i} * q_i), \quad (3)$$

kjer je r naključno število med 0 in 1, d parameter razlike mejnih točk od povprečne celice, λ_i največja lastna vrednost in q_i njej pripadajoč lastni vektor. Največjo lastno vrednost vzamemo zato ker nam omogoči generiranje najbolj raznolikih novih celic.

5. Zapis v binarno datoteko

Z uporabo mejnih točk nove celice lahko zapišemo binarno .raw datoteko. To naredimo tako, da se sprehodimo čez vse rezine celice in v vsaki rezini ustvarimo poligon iz njej pripadajočih mejnih točk.

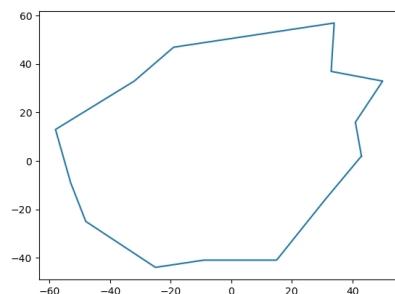


Figure 4: Poligon med mejnimi točkami ene rezine.

Ker mejne točke, ki jih dobimo iz statističnega modela

niso razvrščene, jih pred definiranjem poligona razvrstimo v smeri urinega kazalca. Ko imamo definiran poligon se sprehodimo čez vse točke po x in y koordinatnih oseh in za vse točke izven poligona zapišemo v datoteko binarno vrednost 0, za vse točke na meji poligona binarno vrednost 128 in za vse točke znotraj poligona binarno vrednost 255.

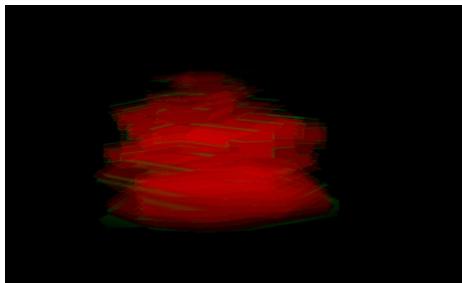


Figure 5: Primer celice prikazane v 3D prikazovalniku [LBM18]

[GIT] Proceduralcellshapemodel. <https://github.com/erikrakuscek/ProceduralCellShapeModel>. [Online; accessed 14-May-2020]. 1

[LBM18] LESAR Ž., BOHAK C., MAROLT M.: Real-time interactive platform-agnostic volumetric path tracing in webgl 2.0. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology* (2018), pp. 1–7. 1, 3

[SACF*12] SCHINDELIN J., ARGANDA-CARRERAS I., FRISE E., KAYNIG V., LONGAIR M., PIETZSCH T., PREIBISCH S., RUEDEN C., SAALFELD S., SCHMID B., ET AL.: Fiji: an open-source platform for biological-image analysis. *Nature methods* 9, 7 (2012), 676–682. 1

[VTL*04] VRTOVEC T., TOMAŽEVIČ D., LIKAR B., TRAVNIK L., PERNUŠ F.: Automated construction of 3d statistical shape models. *Image Analysis & Stereology* 23, 2 (2004), 111–120. 1

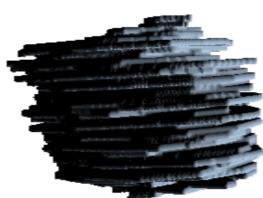


Figure 6: Primer celice prikazane v 3D prikazovalniku [LBM18]

6. Ugotovitve in nadaljnje delo

V okviru seminarske naloge smo uspešno ustvarili novo obliko celice iz podatkov o celicah v realnem življenju. Čeprav so naše na novo ustvarjene celice na pogled precej bolj oglate kot referenčne celice pa so dober približek le-tem. V nadalnjem delu bi lahko še dopolnili algoritem za zapisovanje celic v binarne datoteke tako, da bi ustvarili bolj gladke prehode med različnimi rezinami. Prav tako bi lahko implementacijo še optimizirali saj trenutni način zapisovanja binarnih datotek traja okoli pet minut za eno celico. Nadgradili bi lahko tudi procesiranje slik ter s tem odstranili potrebo po ročnem določanju parametrov algoritma.

References

[CEL] Allan cell institute database, membrane (caax). <https://www.allencell.org/data-downloading.html>. [Online; accessed 14-May-2020]. 1