

TDK dolgozat

Szikszai László

Debreceni Egyetem
Tudományegyetemi Karok
Informatika kar
Komputergrafika és Képfeldolgozás Tanszék

**Egy képfeldolgozási eljárás sejtklaszterek
dinamikájának mérésére**

Dr. habil. Hajdu András
tanszékvezető, egyetemi docens

Szikszai László
programtervező-informatikus MSc, 2. évf.

Debrecen
2012

Tartalomjegyzék

1. Bevezetés.....	4
2. Technikai körülmények, követelmények	5
3. Előfeldolgozás.....	6
4. Háttér és klaszterek szegmentálása	10
4.1. Szegmentációs technikák.....	10
4.2. Fuzzy C-means osztályozás	11
4.1. FCM szegmentálás	13
5. Klaszterek szegmentálása	14
6. Információ kinyerés.....	17
7. Információk feldolgozása	17
7.1. Információ kinyerés és feldolgozás.....	18
8. Összefoglalás	19
9. Irodalomjegyzék.....	21
10. Függelék.....	22
10.1. Fontosabb forráskódok.....	22
10.1.1. Színes kép intenzitásképpé konvertálása.....	22
10.1.2. Kép kontrasztjának mérése.....	22
10.1.3. FCM szegmentálás	22

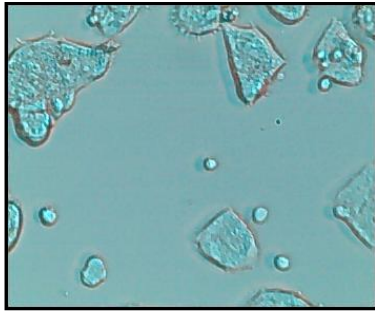
10.1.4.	Adatok kinyerése és feldolgozása.....	23
10.1.5.	Információk megjelenítése	24
10.1.6.	Mintavideó feldolgozása.....	24
11.	Saját munka	25
12.	Köszönetnyilvánítás	27

1. Bevezetés

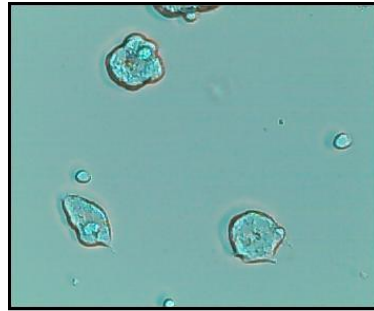
Különböző sejtek, sejtklaszterek (sejtcsoportok) különbözőképpen viselkednek az egyes fehérjék hatására, másként változnak bizonyos tulajdonságaik, pl.: kerület, terület, állábak száma, stb.

A kísérletekben kettő fehérje hatását vizsgálják vastagbél rákos sejteken (LS174T human, 1. ábra).

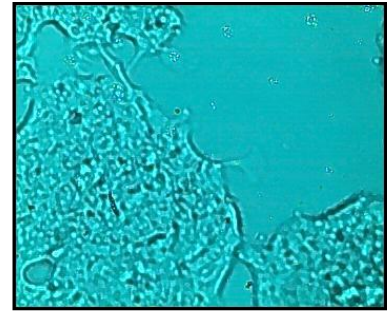
- **IFN- γ** : A fehérje mennyisége a szervezetben gyulladás, illetve autoimmun betegségek jelenlétében megnövekszik. A sejtklaszterek az IFN- γ hatására kompaktabbak lesznek, összetömörülnek, nincs állábuk, vagy nagyon kevés állábat növesztenek. (2. ábra)
- **TNF- α** : A fehérjét az NK sejtek (limfociták) termelik, sejtmigráció szabályozásában vesz részt. Hatására a sejtklaszterek „szétterülnek”, sok állábat növesztenek. (3. ábra)



1. ábra - kontroll mintáról készült kép



2. ábra - IFN mintáról készült kép



3. ábra - TNF mintáról készült kép

A feladat a látottak számszerűsítése, a mintákról készült felvételek feldolgozása, elemzése, a sejtklaszterek megfelelő tulajdonságainak automatikus kinyerése és a kinyert adatok feldolgozása.

Például:

1. Kerület, terület, „tömeg” és ezek arányának követése az időben, sebességtér számolása.
2. Megjelenő állábak növekedési sebességének, irányeloszlásának, hosszúság szerinti eloszlásának és az egyes álláb-szomszédok távolságának meghatározása.

2. Technikai körülmények, követelmények

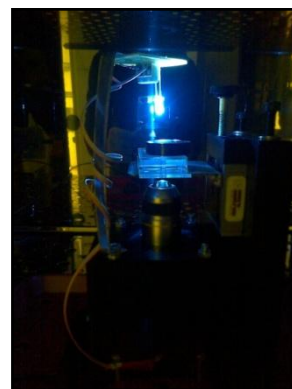
Az egyes mintákról és az azokban levő sejtekről/sejtklaszterekről a következőképpen készítünk felvételeket:

1. A mintát felülről egy fehér LED világítja meg, mely elé kék színszűrőt helyezünk. Azért ezt a színszűrőt használjuk a megvilágításhoz, mert ezen hullámhossz-tartományú fény halad át optimálisan az egyes sejtklasztereken.
2. A mintán áthaladó fény egy – a minta alatt elhelyezkedő – 10x-es nagyítású mikroszkóp objektívén halad át, melynek optikai zoom-ja megfelelő ahhoz, hogy kiértékelhető képet adjon a mintában levő sejtklaszterekről.
3. Az objektívből kilépő fény egy – optikájától megfosztott – Genius Eye 110 típusú webkamera CCD érzékelőjére vetül.
4. Az érzékelt képet a webkamera egy PC-re továbbítja, ahol MATLAB program segítségével percenként egy – RGB24 formátumú és 352x288 pixel méretű – képet készítünk. Az elkészült képekből a könnyebb feldolgozás és tárolás végett egy videót állítunk elő.

A felvételek készítésére szolgáló szerkezet (4-5. ábra) egy inkubátorban van elhelyezve, melyben a hőmérséklet és a levegő széndioxid tartalma is szabályozható, ami a minta megfelelő vizsgálatához szükséges.



4. ábra - a felvételt készítő berendezés



5. ábra - a felvételt készítő berendezés működés közben

Az 1-3. ábrán látható a három különböző mintáról készített felvétel. Megfigyelhető, hogy mind a három felvételen az egyes sejtklaszterek inhomogének, míg a sejtek közötti tápoldal (a háttér) eléggé homogén. Továbbá több dolog is megfigyelhető a képen, melyekre magyarázat, hogy a felvételt készítő „szerkezet”, az egyes alkotórészek nincsenek pontosan beállítva:

1. Mivel a megvilágításra egy pontforrást használtunk és magának a mintát tároló edény falának is van egy bizonyos vastagsága, a felvételek többségén az egyes klasztereknek olyan mintha „árnyéka” lenne. Ez a klaszterezési eljárásoknál figyelembe kell venni, ugyanis hibát eredményezhet, ronthatja az eredményt.
2. Bár emberi szem számára a legtöbb kép élesnek tűnik, azonban az egyes képek élessége nem tökéletes. Sok esetben a kép bemozdult, mivel a felvételt készítő szerkezet – elhelyezéséből és felépítéséből adódóan – érzékeny a külső fizikai behatásokra.
3. Legfőbbképpen a TNF mintákról készült felvételeken (3. ábra) látszik, hogy a szétterült sejtklaszterek határvonala, illetve a sejtklaszterek által növesztett állabak nem élesek, az egyes állabak sokszor – avatatlan szem számára – nem azonosíthatók. Ez a hiba abból adódik, hogy a használt objektív fókuszpontjának és az érzékelőnek a helyzete nem optimális.

A képeket feldolgozó program MATLAB-ban készült, mivel a felvételt készítését is MATLAB vezérli. Az elkészült felvételeket nem a készítéskor, hanem az összes felvétel elkészülte után szeretnék elemezni, ezért nem elsődleges elvárás a program minél gyorsabb futása.

A program az előállított adatok későbbi – más programban történő feldolgozása végett – mentse ki a kinyert adatokat, illetve jelenítse meg az eredményt grafikon formájában.

3. Előfeldolgozás

Az előfeldolgozó eljárások célja, hogy a képet olyan optimális állapotba hozzák, mellyel az egyes későbbi algoritmusok jobb eredményt produkálnak.

Az egyik legfontosabb eljárás ebben az esetben az RGB (színes) kép szürkeskálássá alakítása, olyan kép előállítása, mely csak intenzitás információkat tartalmaz. Erre több féle megoldás lehetséges:

1. **Súlyozott összeg:** Ekkor az RGB kép egyes csatornáin található értékeket valamilyen súlyok alapján összeadjuk, így áll elő – képpontonként – egy intenzitásérték. A MATLAB `rgb2gray` függvénye [URL1] e metodika alapján dolgozik, a következő súlyokkal: $I = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$, ahol R, G és B rendre a vörös, zöld és kék színcsatornák.
2. **Zöld komponens:** A legtöbb képalkotó berendezésben használt érzékelővel használnak egy CFA-t (color filter array) [URL2], ami azt szabályozza, hogy az adott érzékelő pontra az oda érkező fény mely hullámhosszú komponense haladjon át. Az emberi szem képalkotásáról tett megfigyelések alapján, az emberi szemben közel kétszer akkora mennyiségű a zöld hullámhossz tartományú fényt érzékelő pálcikasejt. Ezt követve a legelterjedtebb Bayer CFA-ban kétszer több a zöld komponens. Ennél fogva egy RGB kép G komponense több, pontosabb intenzitás információt tartalmaz, így legtöbbször jól használható intenzitásképként (ezért is van nagyobb súllyal az 1. eljárásban).
3. **Színterek:** Az RGB színmodellből át lehet térni más színmodellbe (HSI, YCbCr, YIQ - NTSC) melyek egyes komponensei sokszor jobban megfelelnek az egyes algoritmusok számára.

Jelen feladatban nagyon fontos, hogy a későbbi szegmentáló jó eredményt tudjon elérni, hogy egy olyan intenzitásképet állítson elő a program, mely minél kontrasztosabb, a sejtklaszterek határvonalai és az állabak minél jobban elkülönüljenek a homogén háttértől. A fenti eljárások közül a súlyozott összeg használata a kézenfekvő, ugyanis itt a súlyok változtatásával változtatható a kép kontrasztja. Ebben az eljárásban a cél egy olyan súlyvektor megtalálása, mellyel előállított intenzitásképre az `imgcontrast` függvény (10.1.2 fejezet) értéke maximális. Az optimális súlyvektor megtalálására egy – a témavezetőm, Dr. Hajdu András és Antal Bálint által leírt – szimulált hűtés algoritmus [1] egy változatát használja a program.

A szimulált hűtés algoritmus lényege, hogy egy E energia függvény – melynek értelmezési tartománya a keresési halmaz – minimumát keressük. Ezt úgy tesszük, hogy egy kiinduló értéktől – a keresési halmazból – lépünk egy „véletlenül választott szomszédra”, ha annak energiája kisebb, mint a jelenlegi. Ellenkező esetben véletlenszerűen „ugrunk” a keresési halmazban, mely kiugrást akkor fogadjuk el – azaz onnan folytatjuk az algoritmust, – ha annak elfogadási valószínűsége nagyobb, mint egy véletlen választott valószínűség. Ennek eredményeképp az

algoritmus – megfelelő paraméterek mellett – nem reked meg egy lokális minimumban, hanem nagy valószínűséggel megtalálja a globális minimumot.

Jelen problémánál a keresési halmaz:

$$S = \{(s_1, s_2, s_3) | (s_1, s_2, s_3) \in [0,1]^3 \text{ és } \sum_{i=1}^3 s_i = 1\}$$

$$s_0 = (0.2989, 0.587, 0.114)$$

Az energia függvény:

$$E: S \rightarrow \mathbb{R}, E(s) = -imgcontrast(rgb2gray2(I, s)),$$

ahol az `rgb2gray2` függvény egy I RGB képből készít egy szürkeskálás képet, az s súlyok felhasználásával. Mivel minimumot keresünk, ezért az `imgcontrast` által visszaadott érték ellentettje szerepel az energia függvényben.

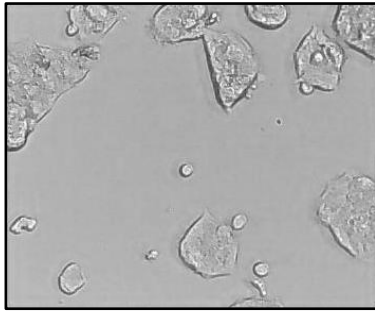
Az elfogadó függvény:

$$simann_accept(e, e_i, T, y) = \begin{cases} true, ha \exp\left(\frac{e - e_i}{T}\right) > y \\ false, egyébként \end{cases}$$

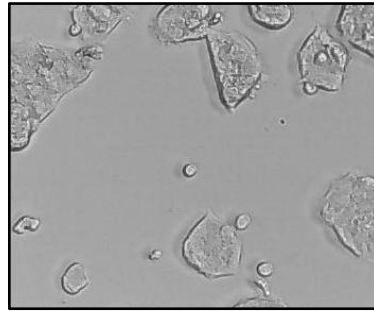
A teljes `simann_rgb2gray` függvény forrása a 10.1.1 fejezetben található.

A 6-7. ábrán látható a három fent említett eljárás alkalmazása egy – a kontroll mintából vett – képre.

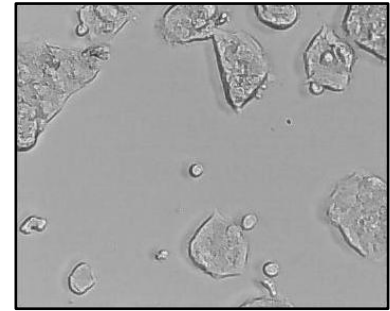
A három eljárást összehasonlító készítettem egy script-et, amely mind a három mintából kiválaszt 100 képet, majd ezekre alkalmazza a három eljárást és megméri az eredmény képek kontrasztját. Az így előálló $3 \times 3 \times 100$ eredményből pedig átlagot számol, így mérhető a három eljárás hatékonysága, hogy melyik eredményez kontrasztosabb képet. Az eljárás eredménye az 9. ábrán látható táblázatban olvasható.



6. ábra - kontroll minta - zöld (G) csatorna



7. ábra - kontroll minta - rgb2gray függvény



8. ábra - kontroll minta - simann_rgb2gray függvény

Az eredményekből következik, hogy a beépített **rgb2gray** eljárás nem megfelelő, hiszen jóval rosszabb eredményt produkált mind a másik két eljárás. Mint látható a szimulált hűtéssel kombinált súlyozott összeg eljárás jó eredményt adott. Az is látható, hogy ez az eredmény függ a kép típusától (a mintától) is, ugyanis, míg az IFN minta esetében a **simann_rgb2gray** jobb eredményt produkált, a TNF minta esetében nem.

Az eredmény fényében a **simann_rgb2gray** eljárás alkalmazásakor célszerű az eredmény szürkeskálás kép kontrasztját megvizsgálni és szükség esetén inkább a zöld csatornás képet használni a továbbiakban.

	zöld (G) csatorna	rgb2gray	simann_rgb2gray
kontroll minta	0,3292	0,2290	0,3292
IFN minta	0,2385	0,1870	0,2852
TNF minta	0,3295	0,2655	0,3188

9. ábra - előfeldolgozó eljárások összehasonlítása

A klasszikus előfeldolgozók közül a mediánszűrő alkalmazása ebben az esetben nem célszerű, ugyanis bár eltünteti a kisebb zajokat a képekről, de az élességet is rontja, ezzel a szegmentálás eredményét.

4. Háttér és klaszterek szegmentálása

4.1. Szegmentációs technikák

A legfontosabb lépés az algoritmusban a háttér és a sejt klaszterek különválasztása, azaz a kép szegmentálása [URL3]. Ez technikailag egy olyan maszk meghatározását jelenti, mely a sejt klaszterek pixeleit jelöli.

Többféle szegmentációs technika létezik, ezek közül néhány fontosabb:

1. **Küszöbölés** (thresholding): A legegyszerűbb technika. Intenzitásképek esetében azon képpontok, melyek intenzitása nagyobb, mint egy megadott T küszöb, értéke 1 lesz, a többi pedig 0. Az eredmény képet ebben az esetben bináris képnek nevezzük, ahol az 1 értékek az előtér, a 0 értékek pedig a háttér képpontjai. Szegmentálás esetén az 1 értékek a szegmentált objektum képpontjai, a 0 értékek pedig a felesleges képpontok. Ezt a technikát főleg olyan képek esetén alkalmazzuk, amikor a háttér és a szegmentálandó objektum jelentősen elkülönül. A küszöbérték meghatározására több eljárás is van, ezek közül a legismertebb az Otsu módszer [5] vagy használható még a k-means [3] osztályozó eljárás is erre a célra.
2. **Régió növelés** (region growing): Egy vagy több régióból, legtöbbször 1-1 kiindulási pontból (seed points) párhuzamosan, valamilyen feltétel alapján növeljük az egyes régiókat. A régió növelés úgy történik, hogy vesszük az adott régióval szomszédos képpontokat, és ha megfelel a konkrét módszer feltételének, akkor hozzávesszük a régióhoz. Régió növelési módszerek például:
 - a. **Globális küszöb alapú**: ha a régió határával szomszédos képpont még nem tartozik egy régióhoz sem, és küszöbértéken (T) belül van, akkor hozzá vesszük a régióhoz, különben nem.
 - b. **Tűrés alapú**: ha a régió határával szomszédos képpont még nem tartozik egy régióhoz sem, és „nincs nagy különbség” a vizsgált képpont és a vele szomszédos, régióhoz tartozó képpont intenzitása között (vagyis az intenzitáskülönbség nem halad meg egy t küszöbértéket), akkor hozzá vesszük a régióhoz, különben nem.
 - c. **Statisztikai alapú**: Kiszámoljuk a régió belüli képpont-intenzitások átlagát és szórását. Ha egy régió határával szomszédos képpont intenzitása és az

átlagintenzitás különbsége kisebb, mint a szórás k -szorosa, akkor vesszük hozzá a képpontot a régióhoz.

Mivel a képeken a háttér és az előtér (sejtklaszterek) intenzitásai között nincs nagy különbség, nem lehet egy küszöbértéket meghatározni, melyre jól szegmentálna a küszöböléses eljárás, így ez nem megfelelő.

A régiónövelő eljárás ideálisnak tűnik, mivel a háttér eléggé homogén, illetve a sejtklaszterek határvonalai, kontúrjai eléggé eltérnek a háttértől ahhoz, hogy magát a háttér egyetlen régióval le tudja fedni az algoritmus könnyen és viszonylag pontosan. Ezután a régió komplementere lesz a sejtklaszterek régiója, amit már csak darabolni kell – ha szükséges – a következő fejezetben ismertetett eljárással.

Azonban ez az eljárás nagyon érzékeny az említett „árnyékokra” a képen. Illetve mivel nem elég élesek (például a TNF mintákról készült) képek, a határátmenetek elmosódnak és az algoritmus könnyen „átsiklik” felettük, nem veszi ezeket figyelembe.

Mindkét problémán lehetne javítani: míg egy medián-szűrő elmosná az árnyékokat, a sejtklaszterek határával is ugyanezt tenné, így helytelen eredményt kapnánk; illetve az élesítés – mellyel láthatóbbá tehetnénk az elmosott éleket, állábakat – élesítené az árnyékokat is.

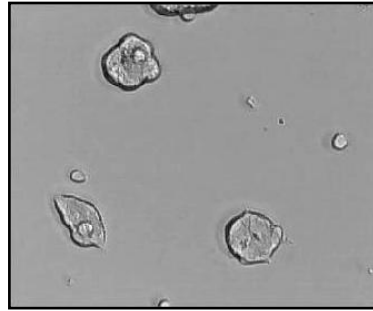
4.2. Fuzzy C-means osztályozás

A Fuzzy C-means osztályozási eljárás [4] egy olyan statisztikai osztályozást tesz lehetővé, mely esetében egy pont több osztályhoz is tartozhat. Ez azt jelenti, hogy az algoritmus eredménye egy vektor, melyben valószínűség értékek vannak, melyek azt mondják meg, hogy az adott pont milyen valószínűséggel tartozik az adott osztályhoz.

Mint minden statisztikai osztályozó eljárásnak, ennek is minden egyes pontra több féle adatot is meg lehet adni, több jellemző (feature) alapján tudja osztályozni a képpontokat.

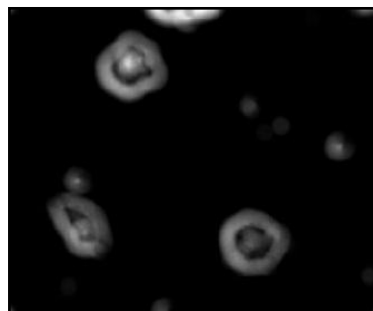
A megfelelő jellemzők megválasztásával kiküszöbölhetőek az előző szegmentáló eljárások hátrányai. A lehetséges jellemzők a tesztelés folyamán a következőket választottam:

1. **Intenzitás kép:** A kép szürkeskálás (`simann_rgb2gray` által előállított illetve a zöld csatorna közül a nagyobb kontrasztú) változatának $[0..255]$ intervallumra skálázott változata. Ez azért fontos, mert így nagy szerepet játszik az osztályozásban az eredeti kép, hisz ennek a képpontjait szeretnénk osztályozni. (10. ábra)



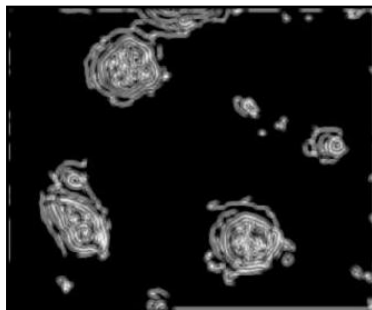
10. ábra - IFN minta, szürkeskálás jellemző

2. **Megvilágítás kiegyenlítés (illumination equalization):** Legyen I_{eq} az eredmény kép, ekkor: $I_{eq}(r, c) = I(r, c) + m - A(r, c)$, ahol $I(r, c)$ a szürkeskálás kép r . sorban és c . oszlopban levő képpontja, az m a várt átlagintenzitás, az $A(r, c)$ pedig a szürkeskálás kép (r, c) képpontjának $N \times N$ -es környezetének vett átlagintenzitása. Az eredmény képből küszöböléssel egy bináris képet állítunk elő, melyen a kép legkontrasztosabb részei – jelen esetben a sejtklaszterek szélei – lesznek. Ezt egy 8 sugarú körlap alakú szűrővel konvolálunk, így elmoszuk azt, nagyobb „teret engedve” az FCM osztályozónak. Az eredmény képet szintén skálázzuk. (11. ábra)



11. ábra - IFN minta, illumination equalization jellemző

3. **Él kép:** A 1. pontban előállított (még nem skálázott) képen egy canny [2] éldetektálást végzünk, majd ezt a 2. ponthoz hasonlóan egy 3 sugarú körlap alakú szűrővel konvoláljuk, elmoszuk azt, illetve skálázzuk. (12. ábra)



12. ábra - élkép jellemző

A MATLAB-ban az `fcm` [URL4] függvény implementálja ezt az eljárást.

Mivel az egymást követő felvételek között nem nagy a differencia, egy videóba rendezhetők. Ebből kifolyólag minden egyes felvételnél (kivéve az elsőt) az előző szegmentálás eredménye felhasználható a szegmentáció eredményének pontosítására.

Ekkor az FCM szegmentáló által felhasznált jellemzővektor kiegészül az előző szegmentálás eredményének bináris képpé alakított változatával.

4.1. FCM szegmentálás

A `cell_cluster_segmentation` függvény forráskódja a 10.1.3. fejezetben található.

A függvény a bemeneti RGB képnek előállítja a szegmentált címkéképét (az opcionális jellemzőkép felhasználásával).

Input: `I` – RGB kép (13. ábra), `Def` – opcionális, jellemzőkép az előző szegmentációból.

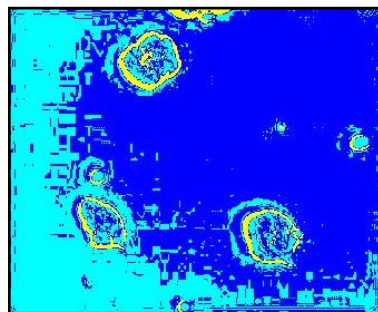
Output: `L` – az eredmény szegmentált címkékép, `n` – a címkéképen levő szegmensek száma.

1. Az `I` Fuzzy C-means szegmentálása, az opcionális `Def` paraméter felhasználásával 3 osztályra (14. ábra).
2. Megvilágítás kiegyenlítés és küszöbölés segítségével egy él kép meghatározása.

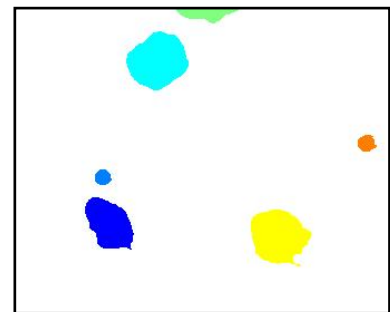
3. Minden szegmens összehasonlítása az élképpel, és amelyikkel a szegmensnek a legtöbb közös része van az lesz az értékes szegmentációs osztály.
4. Bináris kép előállítása úgy, hogy az értékes osztály képpontjai az előtér képpontok (1), a többi pedig a háttér képpont (0).
5. Az így előálló bináris kép dilataciója egy 5 képpont sugarú, kör alakú struktúra elemmel, majd az eredmény bináris képen a lyukak feltöltése az `imfill` MATLAB függvénnyel, illetve az így kapott bináris kép erodálása az előző struktúra elemmel.
6. A javított bináris képből a zaj szegmensek (túl kicsi) eltávolítása, illetve a kisebb, „lyukas” szegmensek feltöltése.
7. A bináris képből egy címkékép előállítása (15. ábra).



13. ábra - szegmentálandó IFN minta



14. ábra - FCM szegmentálás utáni címkékép (3 osztály)



15. ábra - a címkékép feldolgozás utáni, eredmény címkékép

5. Klaszterek szegmentálása

Az elsődleges szegmentálás eredménye nem minden esetben felel meg a valóságnak, ugyanis előfordulhat olyan eset, amikor az egyes sejt-klaszterek átfedik egymást, vagy annyira közel vannak egymáshoz, hogy az előző szegmentációs lépés utófeldolgozásakor „egybeolvadnak”.

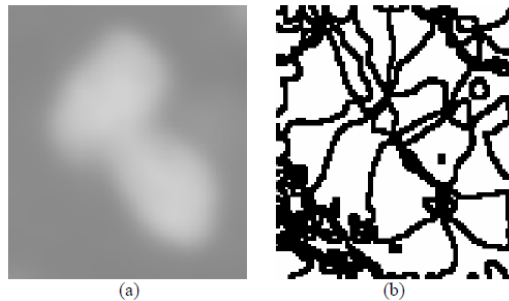
Több technika, megoldás létezik hasonló problémákra, ezek közül egy a **felület felosztás és összevonás** [6] (Surface splitting and surface merging) eljárás.

Az eljárás egy intenzitáskép gradiens vektorain, a felosztandó kép (képrészlet) képpontjainak gradiens alapú osztályozásán, majd az egyes osztályok összevonásán alapul.

1. Először az $I_{i,j}$ szürkeskálás kép (16.(a) ábra) minden egyes képpontjában meghatározzuk a gradiens értékeket. A $\nabla I(x,y)$ gradienst a következőképpen közelíthetjük: $\frac{\partial I_{i,j}}{\partial x} \approx \frac{I_{i+1,j} - I_{i-1,j}}{2}$ és $\frac{\partial I_{i,j}}{\partial y} \approx \frac{I_{i,j-1} - I_{i,j+1}}{2}$.
2. Az előző lépésben előálló $(g_{i,j}, h_{i,j})$ gradiens értékekből kiszámoljuk az adott gradiens vektor x-tengellyel bezárt szögét: $\theta_{i,j} = \text{atan2}(g_{i,j}, h_{i,j}), \theta_{i,j} \in [0, \frac{\pi}{2}]$.
3. Ezután az egyes szögértékeket át kell váltanunk a $[0, 2\pi]$ intervallumra, hogy megfelelően tudjuk osztályozni a képpontokat a gradiensvektor szöge alapján. Ehhez meg kell határoznunk, hogy az egyes gradiensvektorok a koordináta rendszer melyik negyedében vannak ($Q_{i,j}$) és eszerint korrigáljuk a szögértéket. Ez alapján:

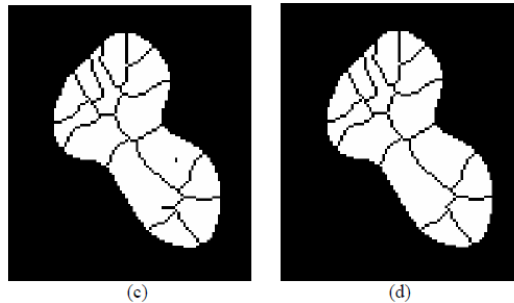
$$\omega_{i,j} = \begin{cases} \text{abs}(\theta_{i,j}), & \text{ha } Q_{i,j} = 1, \\ \text{abs}(\theta_{i,j}) + \frac{\pi}{2}, & \text{ha } Q_{i,j} = 2, \\ \text{abs}(\theta_{i,j}) + \pi, & \text{ha } Q_{i,j} = 3, \\ \text{abs}(\theta_{i,j}) + \frac{3\pi}{2}, & \text{ha } Q_{i,j} = 4, \\ \frac{3\pi}{2}, & \text{ha } \text{sign}(g_{i,j}) = 0 \text{ és } \text{sign}(h_{i,j}) < 0, \\ \pi, & \text{ha } \text{sign}(g_{i,j}) = 0 \text{ és } \text{sign}(h_{i,j}) > 0, \\ \frac{\pi}{2}, & \text{ha } \text{sign}(g_{i,j}) < 0 \text{ és } \text{sign}(h_{i,j}) = 0, \\ 0, & \text{ha } \text{sign}(g_{i,j}) > 0 \text{ és } \text{sign}(h_{i,j}) = 0 \end{cases}$$

4. Az így előálló $\omega_{i,j}$ értékeket osztályozzuk (16.(b) ábra) – akár egy k-means algoritmussal vagy magával az FCM algoritmussal.



16. ábra - (a) felosztandó felület, (b) képpontok osztályozása, forrás: [6]

5. Az így előálló bináris képről kimaszkoljuk a háttérrel, hogy csak a számunkra „értékes” régiókkal dolgozhassunk (16.(c) ábra). Ezután erről eltávolítjuk a zajokat és a kiálló éleket egy szűrés segítségével. (16.(d) ábra)



16. ábra - (c) háttér-eltávolítás utáni osztályok, (d) zajeltávolítás utáni osztályozás, forrás: [6]

6. Minden egyes szegmensnek kiszámoljuk az adott régiót reprezentáló gradiens vektort. Ezt akár az adott régió vektorainak átlagával is meghatározhatjuk.
7. Páronként minden régióra megvizsgáljuk, hogy azok gradiens vektora konvergens-e. Ezt úgy határozhatjuk meg, ha vesszük mindkét vektorhoz az azt tartalmazó egyenest, és megvizsgáljuk, hogy a két egyenes metszéspontja milyen helyzetben van a két vektorral. Ha mind a két vektor iránya a metszéspont felé mutat, akkor a két vizsgált szegmens egy sejthez tartozik, egyébként pedig külön sejtről beszélünk.
8. Az összetartozó régiókat összevonjuk.

Ez az eljárás sajnos meg megfelelő a feladatban vizsgált mintákra, ugyanis az egyes sejtklaszterek felülete nem homogén, sőt teljesen egyenetlen (17. ábra). Ebből adódóan ez az eljárás nagyon sok kis sejtre osztaná a sejtklasztereinket.



17. ábra - kontroll minta, kiemelt klaszter

6. Információ kinyerés

A klaszterek szegmentálása után előáll egy címkézett kép. Ezen a címkézett képen a különböző címkék (számok) különböző szegmenseket, sejtklasztereket jelentenek. Ezen szegmensek különböző tulajdonságait kinyerni a MATLAB `regionprops` [URL5] függvényével lehet, mely többek között a következő – a feladat szempontjából is fontos – tulajdonságokat képes mérni, meghatározni:

1. **Terület (Area):** Egy szám, mely az adott régióhoz (szegmenshez) tartozó képpontok számát jelenti. Ez az egyik tulajdonság, amit mérni szeretnénk.
2. **Középpont (Centroid):** Az adott régió konvex burkának középpontját jelenti. Erre a tulajdonságra az egyes régiók – képek közötti – azonosításához van szükség.
3. **Kerület (Perimeter):** Egy szám, mely az adott régió kerületén levő képpontok számát jelenti. Ezt a tulajdonságot is mérni szeretnénk.
4. **Képpont értékek (PixelValues):** Egy $p \times 1$ -es vektor, mely az adott régióba tartozó intenzitásértékeket tartalmazza ($p = \text{Area}$). Erre a tulajdonságra, illetve az eredmény összegére az egyes sejtklaszterek „súlyának” a méréséhez van szükség.

Azonban, mivel a szegmentálás eredményében nem mindig ugyanazt a címkét kapják az egyes régiók, ezért azok azonosítására is szükség van. Ez úgy történik, hogy megjegyezzük mindig az előző felvétel régióinak a középpontját (**Centroid**). Sorra vesszük az aktuális régiókat, kikeressük az előző felvétel régiói közül azt, amelyik középpontja a legközelebb helyezkedik el az aktuális régió középpontjához. Erre azért van szükség, mert a sejtklaszterek „vándorlása” és a méret változása is nagymértékben befolyásolja a középpont helyzetét.

7. Információk feldolgozása

A kinyert információkból a MATLAB `plot` függvényének felhasználásával állít elő a program egy grafikont, melyből láthatóak az időbeli változások. Illetve a kinyert eredményeket a program el is menti – a MATLAB által használt – (`.mat`) formátumba, a későbbi felhasználás céljából.

7.1. Információ kinyerés és feldolgozás

A `cell_statistics` függvény – mely az információ kinyerést és feldolgozást végzi – forrása a 10.1.4. fejezetben található.

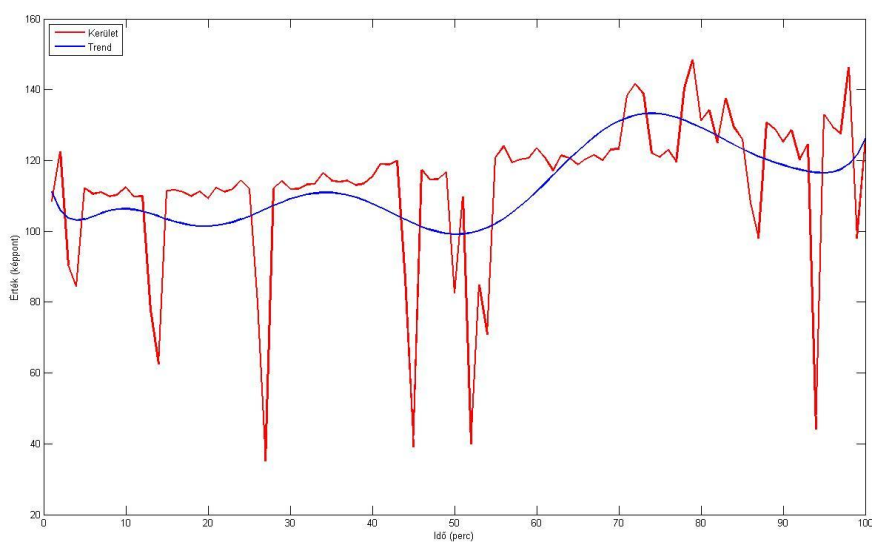
Input: $\mathbf{I}$ – a felvételek (RGB), $\mathbf{L}$ – a szegmentált cimkeképek, n – a címkék száma az összes felvételen külön-külön.

Output: `stats` – az egyes felvételekről készült statisztikai adatok (Terület, Kerület, Tömeg, Kerület/Terület).

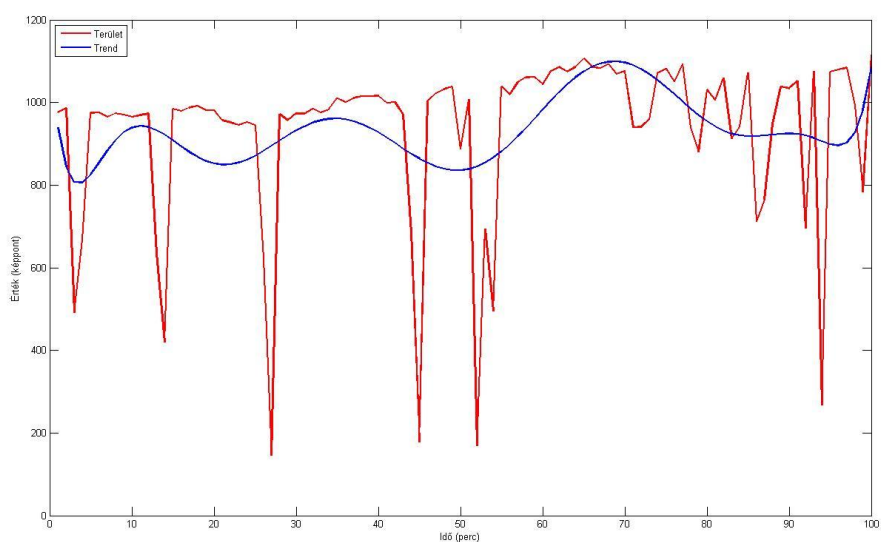
1. Minden egyes cimkeképre:
 - a. Az adott cimkekép statisztikai adatainak meghatározása az adott felvétel segítségével.
 - b. Az adott cimkekép minden egyes címkéjére:
 - i. Ha ez a sor első cimkeképe, akkor a középpontok beállítása és az információk tárolása.
 - ii. Egyébként az előző címkék közül megkeresni azt, amelyik középpontja a legközelebb van a jelenleg vizsgálthoz és az információ tárolása, végül a középpont beállítása.

Az előálló adatok utófeldolgozásaként eltávolítjuk a kiugró (hibás) értékeket majd a hiányzó értékeket interpoláljuk.

A 18-19. ábrán látható egy 100 elemű IFN minta feldolgozása után előálló statisztika (kerület és terület) grafikai ábrázolása. Az ábrákon a piros adatsor a minta átlagos (klaszterek közötti) kerület/terület értékeit, míg a kék adatsor, az erre illesztett poligont (trendet) mutatja.



18. ábra - IFN minta, 100 elem, átlagos kerület és trend



19. ábra - IFN minta, 100 elem, átlagos terület és trend

8. Összefoglalás

A feladat (probléma) az, hogy sejtklaszterekről – egy erre a célra készített szerkezettel – rögzített felvételeken történő időbeni változásokat, ezek dinamikáját tudjuk számszerűsíteni. Ez a feladat

az egyes sejtklaszterek detektálásából, ez előtti előfeldolgozásból és a szegmentált régiók adott információinak kinyeréséből és feldolgozásából áll.

Az **előfeldolgozásnál** sikerült egy olyan eljárást konstruálni, mellyel a szegmentálásnak optimális intenzitás képet állít elő. Nem sikerült olyan előfeldolgozó eljárást találni, ami korrigálni tudná a felvételek hibáit – úgymint a mintaedényen levő kosz szemcsék vagy az árnyék – úgy, hogy az a szegmentálás eredményét ne rontaná.

A **szegmentálásnál** találtam olyan eljárást, mely majdnem optimális pontossággal határozza meg az egyes szegmenseket. Az eljáráshoz sikerült kiválasztani olyan **jellemzőket**, mellyel jó eredmény érhető el, azonban a jellemzők finomításán, vagy más jellemzők felhasználásával javítani lehetne az eredményen.

Az **információk kinyerése és feldolgozása** a MATLAB függvényei és lehetőségei segítségével valósítottam meg.

Összességében a megvalósított eljárás kisebb, jobb minőségű mintára használható. Mindenképpen szükséges az eljárások továbbgondolása, jobb szegmentáló algoritmus, esetleg jobb jellemzők megtalálása. Ez azért is fontos, mert a 18. ábrán jól látszik, hogy sok helyen az adatok nem pontosak.

A képek és a szegmentáló algoritmus korlátai miatt az egyes sejtklaszterek állábainak detektálását nem sikerült megvalósítani.

9. Irodalomjegyzék

- [1] Bálint Antal, András Hajdu: A stochastic approach to improve macula detection in retinal images, *Acta Cybernetica* **20** (2011) pp. 5-15.
- [2] Canny, John Francis: Finding Edges and Lines in Images, *Massachusetts Institute of Technology* (1983)
- [3] J. A. Hartigan and M. A. Wong: Algorithm AS 136: A K-Means Clustering Algorithm, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* Vol. 28, No. 1 (1979) pp. 100-108.
- [4] Keh-Shih Chuang, Hong-Long Tzeng, Sharon Chen, Jay Wu, Tzong-Jer Chen: Fuzzyc-means clustering with spatial information for image segmentation, *Computerized Medical Imaging and Graphics, Volume 30, Issue 1* (January 2006) pp. 9–15.
- [5] Nobuyuki Otsu: A Threshold Selection Method from Gray-Level, *IEEE Transactions on systems, man and cybernetics, Vol SMC-9, No.1* (January 1979) pp. 62-66.
- [6] Yunyong Surut, Pornchai Phukpattaranont: Overlapping Cell Image Segmentation using Surface Splitting and Surface Merging Algorithms, *Proceedings of the Second APSIPA Annual Summit and Conference*, pp. 662–666.

- [URL1] Convert RGB image or colormap to grayscale – MATLAB
<http://www.mathworks.com/help/toolbox/images/ref/rgb2gray.html>
[Letöltve: 2012. április]
- [URL2] Color filter array - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Color_filter_array
[Letöltve: 2012. április]
- [URL3] Segmentation (image processing) - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Image_segmentation
[Letöltve: 2012. április]
- [URL4] Fuzzy c-means clustering – MATLAB
<http://www.mathworks.com/help/toolbox/fuzzy/fcm.html>
[Letöltve: 2012. április]

[URL5] Measure properties of image regions – MATLAB

<http://www.mathworks.com/help/toolbox/images/ref/regionprops.html>

[Letöltve: 2012. április]

10. Függelék

Itt olvashatóak a főbb – általam írt – függvények forrásai. A program teljes forrása és az egyes minták a dolgozat CD mellékletén is megtalálhatóak.

10.1. Fontosabb forráskódok

10.1.1. Színes kép intenzitásképpé konvertálása

```
function [Iret] = simann_rgb2gray(I)
    x0 = [0.2989 0.587 0.1140];
    Ig = rgb2gray2(I,x0);
    c0 = -imgcontrast(Ig);
    T = 0.9; Tmin = 0.001;
    q = 0.9; rws = [0.15 0.85];
    while T > Tmin
        xi = simann_randommap(x0,rws);
        Igi = rgb2gray2(I,xi);
        ci = -imgcontrast(Igi);
        if ci < c0
            x0 = xi; c0 = ci; T = T*q;
        else
            y = rand(1);
            if simann_accept(c0,ci,T,y)
                x0 = xi; c0 = ci; T = T*q;
            end
        end
    end
    Iret = rgb2gray2(I,xi);
end
```

10.1.2. Kép kontrasztjának mérése

```
function [c] = imgcontrast(I)
    glcm = graycomatrix(I);
    stat = graycoprops(glcm,'Contrast');
    c = stat.Contrast;
end
```

10.1.3. FCM szegmentálás

```
function [L, n] = cell_cluster_segmentation(I,varargin)
    se5 = strel('disk',5);
    m = 3;
    if size(varargin,1) == 1
        Def = varargin{1};
    else
```

```

        Def = zeros([size(I,1) size(I,2)]);
    end
    F = fcm_segment(I,m,Def);
    maxn = 0; maxi = 0;
    Ie = lesson_8_1(I,40,40);
    Ieb = ~im2bw(Ie,graythresh(Ie));
    for j=1:m
        s = sum(sum(F == j & Ieb));
        if s > maxn
            maxi = j;
            maxn = s;
        end
    end
    F(F ~= maxi) = 0;
    F(F == maxi) = 1;
    F = imdilate(F,se5);
    F = imfill(F,'holes');
    F = imerode(F,se5);
    Ibw = logical(F);

    [L,n] = bwlabel(Ibw);
    H = fspecial('disk',7);
    If = imfilter(double(Ibw),H);
    STATS = regionprops(L,'Centroid','Area','EquivDiameter');
    s = size(Ibw);
    for i=1:n
        c = uint16(STATS(i).Centroid);
        if STATS(i).EquivDiameter < 8
            Ibw(L == i) = 0;
        elseif STATS(i).EquivDiameter < 20 && If(c(2),c(1)) >= 0.3
            if c(2)<8 || c(1)<8 || c(2) > s(1)-7 || c(1) > s(2)-7
                continue;
            end;
            Ibw((c(2)-7):(c(2)+7),(c(1)-7):(c(1)+7)) = Ibw((c(2)-
7):(c(2)+7),(c(1)-7):(c(1)+7)) | logical(H);
        end
    end
    [L,n] = bwlabel(Ibw);
end

```

10.1.4. Adatok kinyerése és feldolgozása

```

function [stats] = cell_statistics(I,L,n)
    for j=1:size(n,1)
        Sts =
regionprops(L(:, :, j), I(:, :, 2), 'Area', 'Centroid', 'Perimeter', 'PixelVa
lues');
        if j == 1
            stats = zeros([size(n,1) 4 n(j)]);
            ctrs = zeros([n(j) 2]);
        end
        for k=1:n(j)
            if j == 1
                stats(j,1,k) = Sts(k).Area;
                stats(j,2,k) = Sts(k).Perimeter;
                stats(j,3,k) = sum(Sts(k).PixelValues);
                stats(j,4,k) = Sts(k).Perimeter / Sts(k).Area;
            else
                if n(j) > size(ctrs,1)
                    ctrs((size(ctrs,1)+1):n(j), :) = zeros([n(j)-
size(ctrs,1) 2]);
                    stats(:, :, (size(ctrs,1)+1):n(j)) =
zeros([size(n,1) 4 n(j)-size(ctrs,1)]);

```

```

        end
        mind = 1e10; minl = 0;
        D = dist([Sts(k).Centroid; ctrs]);
        for l=1:n(j)
            if D(1,l+1) < mind
                mind = D(1,l+1); minl = l;
            end
        end
        stats(j,1,minl) = Sts(k).Area;
        stats(j,2,minl) = Sts(k).Perimeter;
        stats(j,3,minl) = sum(Sts(k).PixelValues);
        stats(j,4,minl) = Sts(k).Perimeter / Sts(k).Area;
    end
    ctrs(k,:) = Sts(k).Centroid;
end
end
end

```

10.1.5. Információk megjelenítése

```

function [] = show_stats(stats)
    figure();
    colors = {'r','g','b','c','m','y'};
    titles = {'Area','Perimeter','Weight','Perimeter/Area'};

    for j=1:3
        subplot(2,2,j);
        hold on;
        for i=1:6
            plot(1:100,stats(:,j,i),['-' colors{i}]);
        end
        title(titles{j});
        hold off;
    end

    subplot(2,2,4);
    hold on;
    for i=1:6
        plot(1:100,stats(:,2,i)./stats(:,1,i),['-' colors{i}]);
    end
    title(titles{4});
    hold off;
end

```

10.1.6. Mintavideoó feldolgozása

```

function [] = process_cell_video(videoname)
    aviObj = VideoReader(videoname);
    last = zeros([aviObj.Height-2 aviObj.Width-2],'uint8');
    lower = 1;
    upper = aviObj.NumberOfFrames;
    L = zeros([size(last) upper-lower+1],'uint8');
    n = zeros([upper-lower+1 1],'uint8');
    for j=lower:upper
        I = read(aviObj,j);
        I = I(2:(end-1),2:(end-1),:);
        [L(:,:,j-lower+1) n(j-lower+1)] =
        cell_claster_segmentation(I,last);
        last = zeros(size(last),'uint8');
        last(L(:,:,j-lower+1) > 0) = 1;
    end
end

```

```

        stats = cell_statistics(I,L,n);
        save('stats.mat',stats);
        show_stats(stats);
    end

```

11. Saját munka

Saját munkám az egyes szegmentáló, előfeldolgozó eljárások kiválasztása, ha szükséges implementálása. Az egyes eljárások kiválasztásakor szükség volt sok eljárás tesztelésére.

A szegmentációs eljárásnak először egy tűrés alapú régiónövelő eljárást teszteltem, amivel a háttért szegmentáltam, mivel ez elég homogén ahhoz, hogy jó eredményt adjon a szegmentálás, majd a szegmentált régió ellentettje a sejtklaszterek régiója. Azonban ez az eljárás az állabakat egyáltalán nem, illetve több sejtklaszterből is csak a kontrasztos széleket szegmentálta.

Az általam készített függvények, scriptek:

- **cell_claster_segmentation:** Egy I RGB kép szegmentálását végzi.
- **cell_statistics:** Cimkeképekből statisztikai adatokat állít elő.
- **drawframe:** Az I szürkeskálás kép szélére egy megadott vastagságú keretet „rajzol” egy bizonyos értékből.
- **fcm_cluster:** Az fcm függvény által előállított osztályozási valószínűségek alapján előállít egy osztályozást (cimkeképet).
- **fcm_features:** Egy I RGB képből jellemzőket állít elő a Fuzzy C-means osztályozónak.
- **fcm_segment:** Egy I RGB kép Fuzzy C-means osztályozását végzi.
- **features2row:** Az fcm_features által előállított jellemzők átalakítását végzi az fcm függvény számára.
- **filecmp:** Fájlnemek összehasonlítását végzi, rendezéshez szükséges. Pl: I2.jpg < I11.jpg.
- **filesort:** Fájlnemek rendezését végzi el a filecmp használatával.
- **illuminationeq:** Megvilágítás kiegyenlítést végez az I RGB képen.
- **imgcontrast:** Egy szürkeárnyaltos kép kontrasztját számítja ki.
- **imnorm:** Egy szürkeárnyaltos kép skálázását végzi a [0..255] tartományra.
- **make_move:** Egy script ami a minta képekből videót készít.
- **make_pictures:** Egy script amely a 3. fejezetben található statisztikát készíti el.

- **process_cell_video:** Egy videó folyam feldolgozását végzi, szegmentálja az egyes képeket, majd előállítja a statisztikát és el is menti azt.
- **rgb2gray2:** Egy I RGB képből súlyozott összeggel előállít egy szürkeárnyaltos képet.
- **show_stats:** Statisztika megjelenítésére szolgál, a statisztikát a cell_statistics állítja elő.
- **simann_accept:** A szimulált hűtés elfogadó függvénye.
- **simann_randommap:** A szimulált hűtéshez egy random súlyvektort állít elő a régi ismeretében.
- **simann_rgb2gray:** Szimulált hűtéssel állít elő egy maximálisan kontrasztos szürkeskálás képet az I RGB képből.
- **test_movie:** A process_cell_video függvény tesztelésére szolgáló script.

A scripteket MATLAB 7.11.0 (R2010b) próbaváltozattal készítettem. A program és a scriptek egy Asus EEE Pc 904HA (1.6GHz, 1GB RAM) futottak.

12.Köszönetnyilvánítás

Köszönettel tartozom témavezetőmnek, Dr. Hajdu Andrásnak emberi és szakmai támogatásáért, Dr. Bene Lászlónak és Forgács Attilának a sejtbiológiai ismeretekért, Nagy Brigitta PhD hallgatónak ötleteiért.

A TDK dolgozat elkészítését a TÁMOP-4.2.2/B-10/1-2010-0024 számú projekt támogatta.

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.



SZÉCHENYI TERV