DIPLOMAMUNKA

Harangozó Roland

Debrecen
2010

Debreceni Egyetem
Informatikai Kar

Digitális szemfenék képek gépi tanuláson alapuló érhálózat-szegmentálásának gyorsítása mintavételezéssel

Témavezető:
Dr. Hajdu András
egyetemi docens

Külső konzulens:
Dr. Török Zsolt
kutatási igazgató
Astrid Research Kft.

Készítette:
Harangozó Roland
Programtervező
matematikus

Debrecen
2010

Tartalomjegyzék

Bevezetés

A fejlett országokban a cukorbetegség a látás elvesztésének leggyakoribb oka. Ennek hátterében az alapbetegséget kísérő számos szemészeti szövődmény áll. Ezen szövődmények kialakulása korai felismerés esetén, időben alkalmazott kezeléssel megakadályozható, illetve a kialakult szövődmények kezelhetőek. 1997-ben a felmérések szerint 124 millió cukorbeteg élt világszerte. Ez a szám becslések szerint napjainkra elérte a 221 milliót. A cukorbetegek 40%-ánál élete során jelentkezik a diabéteszes retinopátia, mint szemészeti szövődmény. A betegek 5%-ában a látásélesség romlása olyan súlyos formában jelentkezik, amely a látás teljes elvesztéséhez is vezethet.

A digitális képek értékelését végző szakemberek gyakran klinikai leolvasó centrumokba tömörülnek, ahol a számos szemészeti vizsgálóhelyről érkező digitális képeket analizálják. Az eredményt néhány nap elteltével küldik vissza a vizsgálat kérőjének.

A klinikai leolvasó centrumok hatékonyságának növelése érdekében, felmerült az igény az automatikus előszűrő rendszerek kifejlesztésére, melynek részfeladata a pontosan felismert érhálózat.

Cél megfogalmazása

Az későbbiekben bemutatásra kerülő érrendszer detektáló algoritmus végrehajtási ideje – az implementáció részleteitől eltekintve – függ az őt betanító minta méretétől. A tanító minta méretének csökkenésével jelentősen romolhat a detektált érhálózat pontossága, így alkalmatlanná válhat az érrendszer elváltozásainak felismeréséhez, és a további elemzéshez. Célunk találni olyan irányított mintavételező eljárást, amely a minta elemszámának csökkenésére – a lehetőségekhez képest – a legjobb eredményt szolgáltatja.

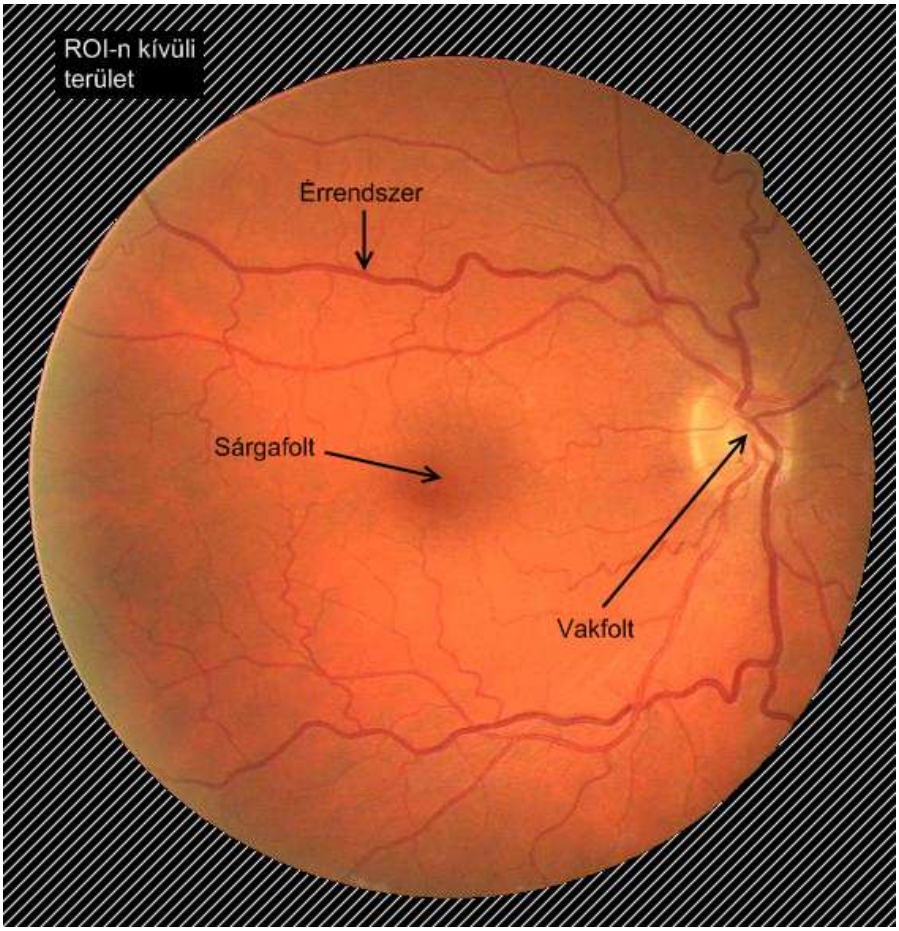
Szemfenéki struktúrák

A retina felvételek automatikus elemzésének első lépése lehet, a következő három szemfenéki struktúra felismerése (1. ábra):

- sárgafolt (macula): Az éleslátás helye a retinán. Nevét a nagy mennyiségben jelenlevő sárga pigment miatt kapta.

- érrendszer (vascular system): A szem vérellátását biztosítja.
- vakfolt (papilla): A retina azon területe, mely nem érzékeli a fényt. Itt lép a szembe a retina egyik legfontosabb artériája, és lép ki az ennek megfelelő véna. Idegrostokat tartalmaz, ezért látóidegfőnek is nevezzük.

A retináról készített felvételeken a szemfenék lényegi információt hordozó területeit érdemes elválasztani a kép hátterétől. Ekkor a minket érdeklő területeket külön vizsgálhatjuk, kiküszöbölve a háttér zavaró hatását. Ezt a területet ROI-nak nevezzük (a rövidítés a Region Of Interest kifejezésből származik [1]). Ha külön nem jelezzük, háttérként a ROI-nak az érhálózatot nem tartalmazó részhalmazát értjük.



1. ábra Szemfenéki struktúrák

Érrendszer detektálás fontossága

Több cukorbetegséget előrejelző elváltozás az érrendszeren felismerhető, ezért a későbbi érrendszer helyes elemzéséhez nélkülözhetetlen az érhálózat lehető legpontosabb szegmentálása. Ilyen elváltozások például a vénás hurok vagy a gyöngyfüzér, amelyek közvetlenül az ér alakjából kerülnek felismerésre. Továbbá a cukorbetegség a mikroaneurizmák kialakulását is eredményezheti, amelyek az érintett érfal kiboltosulásaként jelentkeznek.

Egyéb szemfenéki struktúrákat lokalizáló eljárások feltételezik a szegmentált érrendszert. Például a látóidegfőnek kihasználják azt a tulajdonságát, hogy az erek ezen a helyen lépnek be a szembe, így a retina felvételeken a legszélesebb erek – egy egészséges képen – itt találhatók. Az algoritmus alapja az erek fuzzy konvergenciája, amely egy helytelenül felismert érrendszer esetén természetesen eltér a látóidegfő középpontjától [2]. Egy másik módszer egyszerre ismeri fel az érrendszert, a sárgafoltot, és a látóidegfőt. A struktúráknak azt a jellemzőjét ragadják meg, hogy a sárgafolt körül félkörben helyezkednek el, a látóidegfőből kilépő legszélesebb erek [3].

A fentiekből tisztán látszik, hogy az érrendszer helyes felismerése kritikus pont a retina felvételek automatikus előszűrését célzó szoftverekben.

A diplomamunkámban közölt eredmények egy rangos nemzetközi képfeldolgozó konferencián [4], valamint hazai szakmai fórumokon is bemutatásra kerültek [5].

Alkalmazott módszer

Az osztályozó rendszerek rövid bemutatása

Az osztályozó rendszerek feladata az előre nem megfigyelhető, ismeretlen változók, attribútumok értékének előrejelzése más ismert, megfigyelhető változók, attribútumok ismeretében. Egy másik megfogalmazásban az osztályozásnál egy kitüntetett attribútum értékét kell megjósolnunk a többi attribútum értéke alapján [6]. Az osztályozás során szám *n*-esekkel dolgozunk, amelyeket sajátságvektornak nevezünk. A sajátságvektor elemeit sajátságoknak vagy attribútumoknak nevezzük. A tanítás alapú osztályozó eljárások két lépésre bonthatók:

- Betanítás: A betanítás során egy tanító minta meghatározása a cél. A tanító minta az alapja a későbbi osztályozásnak. Egyes osztályozok a betanítás alapján modelleket építenek (például SVM), más osztályozók – esetleg egy transzformáció után – a tanító mintát használják. A betanításhoz olyan tanító minta szükséges, amelyben minden egyes sajátságvektorhoz egyértelműen meg van adva, hogy melyik osztályba tartozik.
- Osztályozás: Ennek során egy ismeretlen objektumot szeretnénk osztályba sorolni. Az ismeretlen objektumot jellemezzük a tanító minta előállítása során használt rögzített attribútumokkal, és előállítjuk az objektum sajátságvektorát. Az osztályozó feladata a sajátságvektorok alapján a vizsgált objektumok osztályba sorolása.

A *k*-legközelebbi szomszéd módszere (*k*NN)

A *k*NN (*k*-nearest neighbor) osztályozó egy „laza” osztályozó, mert nem épít modellt a tanító mintából [6]. Alapelgondolása, hogy a hasonló attribútumokkal rendelkező objektumok hasonló tulajdonságokkal rendelkeznek. A hasonlóság méréséhez távolságfüggvényt alkalmaz. A távolságfüggvény alapértelmezett esetben a vektorok euklideszi távolságát jelenti. Egy adott vektor osztályba sorolása úgy történik, hogy megkeressük a vektorhoz *k* legközelebbi tanító vektort, és ezen vektorok többségi szavazata alapján megfelelő osztályba soroljuk az osztályozandó objektumot. Ha speciálisan csak két osztályt különítünk el, egyszerűen módosíthatjuk az osztályozás folyamatát úgy, hogy a betanításkor az egyik osztályba tartozó vektorokat 0 címkével, másik osztályba tartozókat 1-es címkével címkézzük, majd többségi szavazás helyett a *k* legközelebbi vektor címkéinek átlagát

számoljuk, amely az 1-es címkéjű osztályba „tartozás” valószínűségét adja. A legközelebbi szomszéd módszer egyik hátránya hogy érzékeny a független attribútumokra. Másik hátránya hogy az attribútumok skálázásra is érzékeny [6]. Utóbbinak részleges megoldása lehet az, ha a tanító mintában szereplő attribútumok várható értékét és szórását kiszámoljuk, majd meghatározunk egy transzformációt (standardizálás), amely alkalmazása után előálló sajátságok várható értéke 0, szórása 1 lesz. Más szóval, az egyes sajátság attribútumokat, mint valószínűségi változókat, standardizáljuk.

Alkalmazott algoritmus leírása

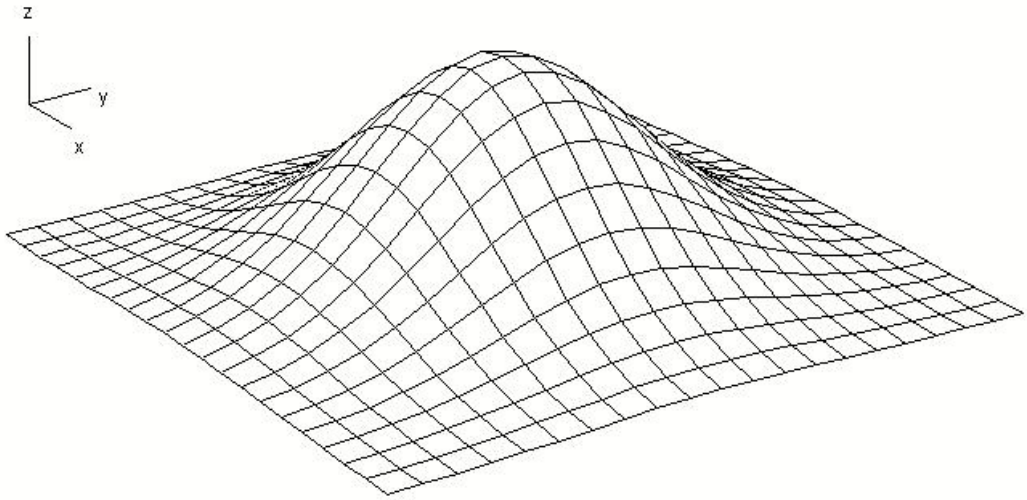
Egy megfelelő érrendszer detektáló algoritmus kiválasztása kulcsfontosságú egy helyesen szegmentált érhálózat meghatározásához. Több ilyen algoritmus is létezik, melyek közül a képpont osztályozás alapú megközelítés a [7] publikációban a legjobb minősítést érte el. Az algoritmus leírását szintén ebben a cikkben közölték.

A módszer alapgondolata, hogy az érrendszer különböző pontjainak környezetei hasonlóak, továbbá a háttérpontok környezetei is hasonlóságot mutat egymással, de az érrendszer pontoknak és a háttérpontoknak a környezetei jól elkülöníthetőek. Ahhoz, hogy ezeknek a környezeteknek a különbözősége alapján el tudjuk dönteni, hogy a vizsgált pont melyik osztályba tartozik, szükséges a környezetek kvantitatív jellemzése. A vizsgált pontnak és környezetének számszerű leírója a sajátságvektor. Minden környezetet azonos módon előállított attribútumokkal látunk el, így sajátságvektoruk elemszáma azonos. A tulajdonságok meghatározásánál fenti algoritmust publikáló cikkben meghatározott módszereket követtük [7].

Az attribútumok kiszámítása – egy kivételtől eltekintve – különböző értékekkel skálázott (s = 1, 2, 4, 8, 16) Gauss szűrők, és ezek elsőrendű és másodrendű deriváltjainak és a szemfenék kép zöld színcsatornájából vett intenzitás értékek konvolúciós szorzatának eredményéből adódnak. Egy kivétel van, ami maga a zöld színcsatorna intenzitás értéke. Zöld színcsatorna kiemelésének mindkét esetben az az oka, hogy ebben a színcsatornában jobban elkülönül az érrendszer.

Az említett Gauss szűrők a következő módon számolhatóak:

- $G(x,y) = \frac{1}{2\pi s^2} e^{-\frac{x^2+y^2}{2s^2}}$
- $G_x^{'}(x,y) = -\frac{x}{2\pi s^4} e^{-\frac{x^2+y^2}{2s^2}}$
- $G_y^{'}(x,y) = -\frac{y}{2\pi s^4} e^{-\frac{x^2+y^2}{2s^2}}$
- $G_x^{''}(x,y) = \frac{x^2-s^2}{2\pi s^6} e^{-\frac{x^2+y^2}{2s^2}}$
- $G_y^{''}(x,y) = \frac{y^2-s^2}{2\pi s^6} e^{-\frac{x^2+y^2}{2s^2}}$
- $G_{xy}^{''}(x,y) = \frac{xy}{2\pi s^6} e^{-\frac{x^2+y^2}{2s^2}}$



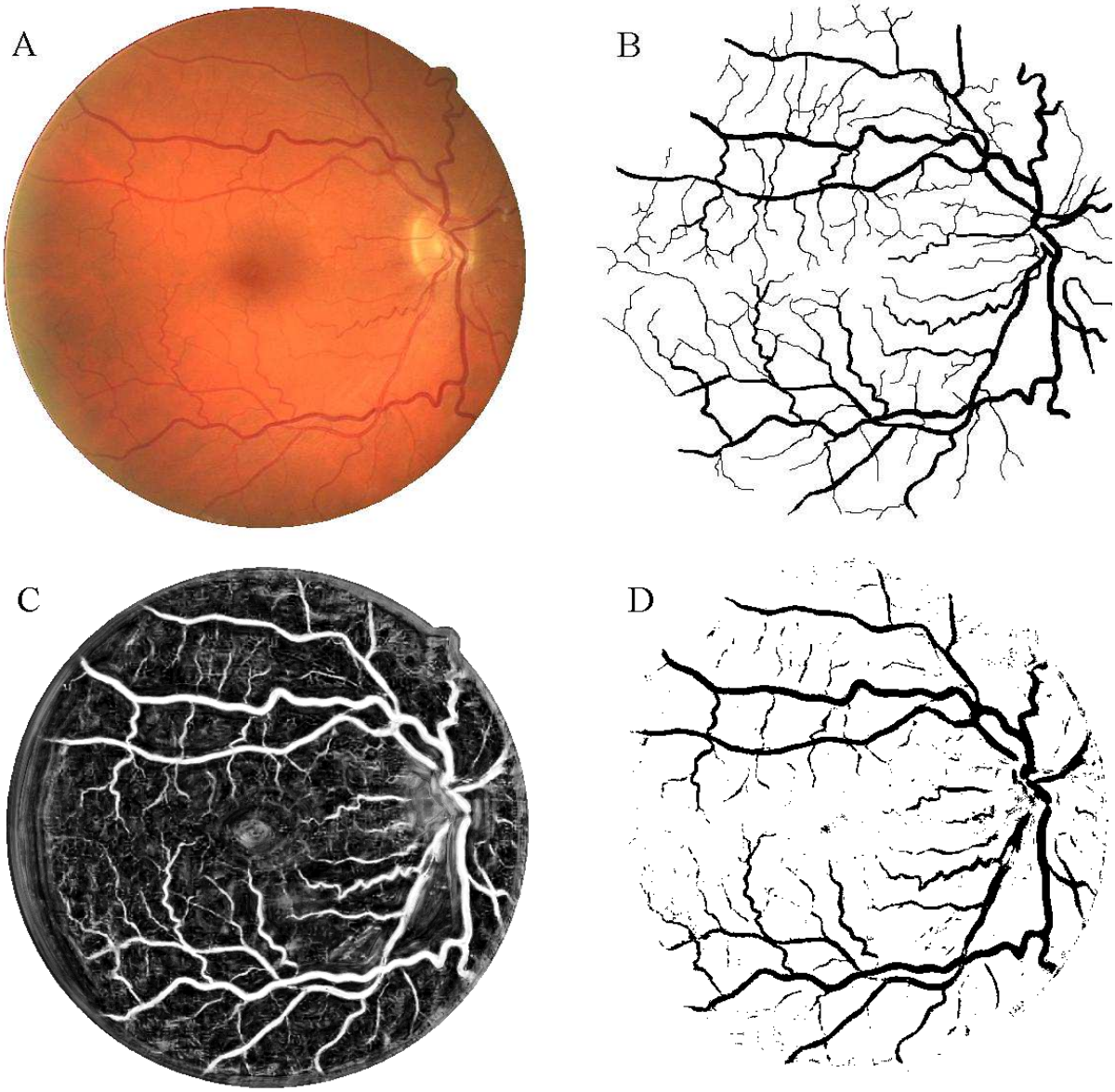
2. ábra Gauss-féle szűrő

Így összesen $5 \times 6 + 1 = 31$ attribútumot határozunk meg minden vizsgált képponthoz, azaz a sajátságvektorok tere 31 dimenziós.

A sajátságvektorok osztályba sorolásához a [7]-ben közölt módszer *k*NN osztályozót használ. A tanító mintát úgy állítjuk össze, hogy az osztályozás eredménye egy valószínűségi érték legyen, melyeknek értéke az adott képpont érhálózathoz tartozásának valószínűségét jelenti. Vagyis az érrendszer pontokból származó sajátságvektorokat 1-essel címkéztük, míg a háttérpontokat 0-val. Így amikor az osztályozó megkeresi a vizsgálandó sajátságvektorhoz *k*

legközelebb elhelyezkedő vektor, az ezekhez kapcsolódó címkék átlagát számolja, pontosan a kívánt valószínűségi érték áll elő.

Miután az összes számunkra lényeges – ROI-n belüli – képponthoz megkonstruáltunk az előbb leírt valószínűségi értéket, egy egyszerű küszöböléssel lehetőségünk nyílik az érhálózat pontjainak kiválasztására, úgy hogy egy bizonyos küszöbérték feletti valószínűségi értékekhez tartozó pontokat soroljuk az érrendszerhez, az alatta levőket pedig a háttérhez (lásd 3. ábra).



3. ábra Szemfenék kép (A), manuálisan szegmentált érrendszer (B), érrendszerhez való tartozás valószínűségi értékei (C), szegmentált érrendszer (D).

Az alkalmazott algoritmus továbbfejlesztése

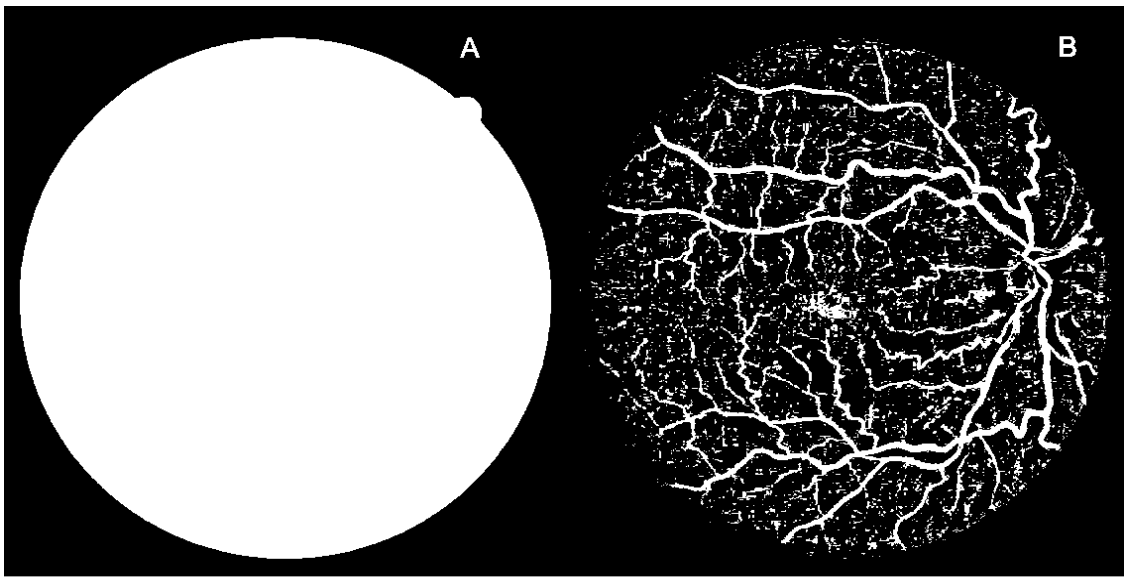
A fentebb bemutatott algoritmus a ROI összes pontjában kiszámolja a sajátságvektort, majd egy osztályozást követően egy valószínűségi értéket rendel hozzá. A végrehajtási időt jelentős mértékben csökkenthetjük, ha ROI-ban szereplő pontok közül eltávolítjuk azokat, amelyek nagy valószínűséggel nem tartozhatnak az érrendszerhez.

Olyan előszűrő eljárást alkalmaztunk, amely a lehetőségekhez képest az érrendszer egészét lefedi, így a vizsgált pontok számossága az eredetinek a töredékére csökken (4. ábra). Az alkalmazott előszűrő eljárás alapjául szolgáló algoritmust, az érrendszer egészének eltávolítására fejlesztették ki [8].

Alkalmazott előszűrő eljárás:

- Álljon I_{green} a szemfenék kép zöld színcsatorna intenzitásértékeiből
- Legyen I_{bg} az I_{green}-en végrehajtott 25x25 –ös mediánszűrés eredménye
- Legyen $I_{roi} = \begin{cases} 1, & I_{green} - I_{bg} < z \\ 0, & egyébként \end{cases}$, ahol z empirikus paraméter. (z = 30)

A további feldolgozásban csak azok a pixelek vesznek részt amelyhez tartozó I_{roi} érték nem nulla. Az előszűrő alkalmazása opcionális. Ahol éltünk ezzel a lehetőséggel, ott ezt külön megemlítjük.



4. ábra ROI (A), előszűrés eredménye (B)

Szegmentált érrendszer jóságának mérése

A tanító mintáról közvetlenül nem tudjuk mérni, hogy az adott mintával betanított *k*NN osztályozó mennyire pontosan határozza meg az érrendszert. Ezért úgy lehetséges a betanító minta jóságának mérése, hogy lefuttatjuk a minta által betanított algoritmust, és az eredményképp létrejött A érrendszerhalmazt összevetjük a manuálisan szegmentált B érrendszerhalmazzal (lásd 3. ábra). A hasonlítás alapja az A és B halmaz S szimmetrikus differenciája.

$$S(A,B) = \frac{|A \backslash B| + |B \backslash A|}{|A \cup B|}$$

Az szimmetrikus differencia 0 értéket ad, ha a paraméterül kapott halmazok minden pontja megegyezik, illetve 1-et, ha nem létezik közös pontjuk.

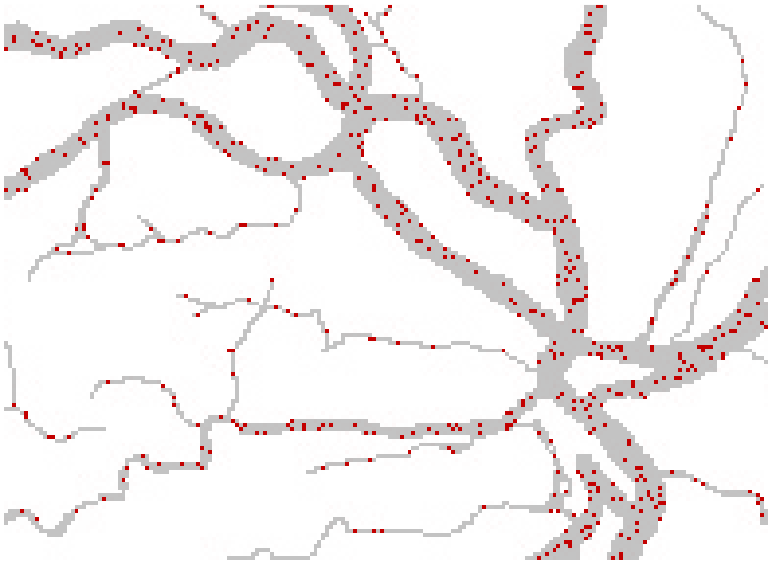
Mintavételezés

Ahhoz, hogy a fent leírt algoritmus teljes legyen, szükséges egy tanító minta meghatározása is. A betanítási folyamathoz és a teszteléshez egy nyilvános adatbázist használtunk [9], amely egyaránt tartalmazza a szemfenékről készült felvételeket, és a manuálisan szegmentált érrendszert. A betanításhoz 10 darab képet használtunk. A megfelelő betanításhoz két halmazt különítünk el: az érrendszer pontokat, és a háttérpontokat. Ezek a halmazok – nagy méretük miatt – alkalmatlanok a betanításra. Szükségessé válik olyan részhalmazok meghatározása, amelyek a gyakorlatban is hatékonyan alkalmazhatók, továbbá a lehető legkevesebb hibát generálják. Az osztályozó rendszereket használó publikációk jelenleg viszont egyáltalán nem, vagy csak felületesen foglalkoznak az osztályozás jóságához létfontosságú tanító minták kiválasztásáról, és általában beérik a véletlen mintavételezéssel, ami értelemszerűen nagy hibához is vezethet.

Véletlenszerű mintavételezés

A véletlenszerű mintavételezés talán az egyik legegyszerűbb módja a mintavételezésnek. Ebben az esetben a mintavételezett halmazból véletlenszerűen választunk ki pontokat, amelyek a tanító mintát adják. A minta elemszámának összhangban kell lennie a mintavételezés fokával. Szélsőséges esetben előfordulhat, hogy a közel azonos sajátságokkal rendelkező részhalmazból származik a minta jelentős része, így kiragadhatja a halmaz elemeinek egyes tulajdonságait. Mivel az érhálózat eltérő részein (például középen vagy szélén) elhelyezkedő pontjai eltérő sajátságvektorokat generálnak, így véletlen mintavételezéskor nagyobb hiba léphet fel. Más szóval, a véletlen mintavételezés jobban szórhat az átlagos hiba körül.

Az érhálózat egy véletlen mintavételezésére az 5. ábrán láthatunk példát.



5. ábra Az érhálózat véletlen mintavételezése. A mintavételezési pontok véletlenszerűen helyezkednek el az érhálózaton.

Központi Voronoi felbontás (CVT) alapú mintavételezés

Mivel a véletlen mintavételezés nagy (szórású) hibát adhat, bemutatunk más mintavételező stratégiákat, amelyek homogénebb módon töltik ki az alaphalmazt. A tekintett mintavételező megközelítések a mintavételezni kívánt halmaz Központi Voronoi felbontáson alapulnak.

Legyen $|.|$ Euklideszi norma R^n–ben, továbbá egy $\Omega \in R^n$ halmaz, K pozitív egészszám és $\{V_k\}_{k=1}^K \subset \Omega$ részhalmazok úgy hogy, $V_k \subset \Omega, V_k \cap V_l = \emptyset \; k \neq l, \bigcup_{k=1}^K V_k = \Omega$ teljesüljön. Azaz a V_k halmazrendszer olyan felosztása az Ω halmaznak, amelyben a halmazrendszer elemei páronként diszjunktak, és az összes részhalmaz uniója megegyezik az Ω halmazzal. Legyen $\{z_k\}_{k=1}^K \subset \Omega$ halmaz, amelyre minden k esetén igaz, hogy

$$V_k = \{x \in \Omega: |x - z_k| < |x - z_l| \; minden \; l = 1, \dots, k - ra \; ha \; l \neq k\}$$

$\{V_k\}_{k=1}^K$ az Ω halmaz egy Voronoi felosztása. $\{z_k\}_{k=1}^K$ a Voronoi felosztás generátorai. Minden V_k egy Voronoi cella a z_k-nak megfelelően.

A homogén mintavételezést speciálisan Voronoi felbontással érhetjük el, ahol a $\{z_k\}_{k=1}^K$ generátorok egyúttal a cellák súlypontjai is.

Adott az Ω halmazon egy nem negatív és mindenhol folytonos sűrűségfüggvény $p(x)$. A $\{V_k\}_{k=1}^K$ cella controidja (súlypontja) ekkor a következő módon számolható:

$$\bar{z}_k = \frac{\int_{V_k} xp(x)dx}{\int_{V_k} p(x)dx}$$

Bemutatunk egy iteratív algoritmus, amely egy adott halmaz központi Voronoi felbontását készíti el. Az algoritmus a következő lépésekből áll: elkészítünk egy Voronoi felbontást a kiindulási generátorok halmazának segítségével, majd a kapott Voronoi cellák centroidjaiból létrehozzuk az új generátorokat. A következő iterációban már az új generátorokkal készítjük el a Voronoi felbontást. Ezt az eljárást ciklikusan ismételjük (6. ábra).

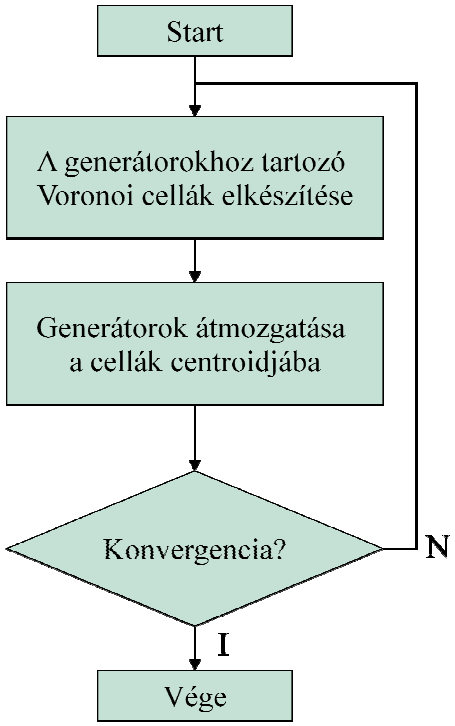
Az eljárást megvalósító (Lloyd) algoritmus [10]:

- Bemeneti paraméterek:
  - Ω
  - *K*, generátorok száma
  - *p*, Ω halmazon értelmezett sűrűségfüggvény
  - $\{z_k\}_{k=1}^K$, generátorok kezdeti halmaza (például véletlen mintavételezéssel)
- Kimenet:
  - $\{V_k\}_{k=1}^K$, egy központi Voronoi felbontás *K* darab generátorral $\{z_k\}_{k=1}^K$
- Az iteráció lépései:
  1. Konstruáljuk meg az Ω halmaz $\{V_k\}_{k=1}^K$ Voronoi felosztását a $\{z_k\}_{k=1}^K$ generátorokkal.
  2. Készítsük el a generátorok új halmazát $\{z_k\}_{k=1}^K$ a $\{V_k\}_{k=1}^K$ centroidjaiból.
  3. Ismételjük meg az 1. és 2. lépést, amíg valamelyik kilépési feltétel teljesül.

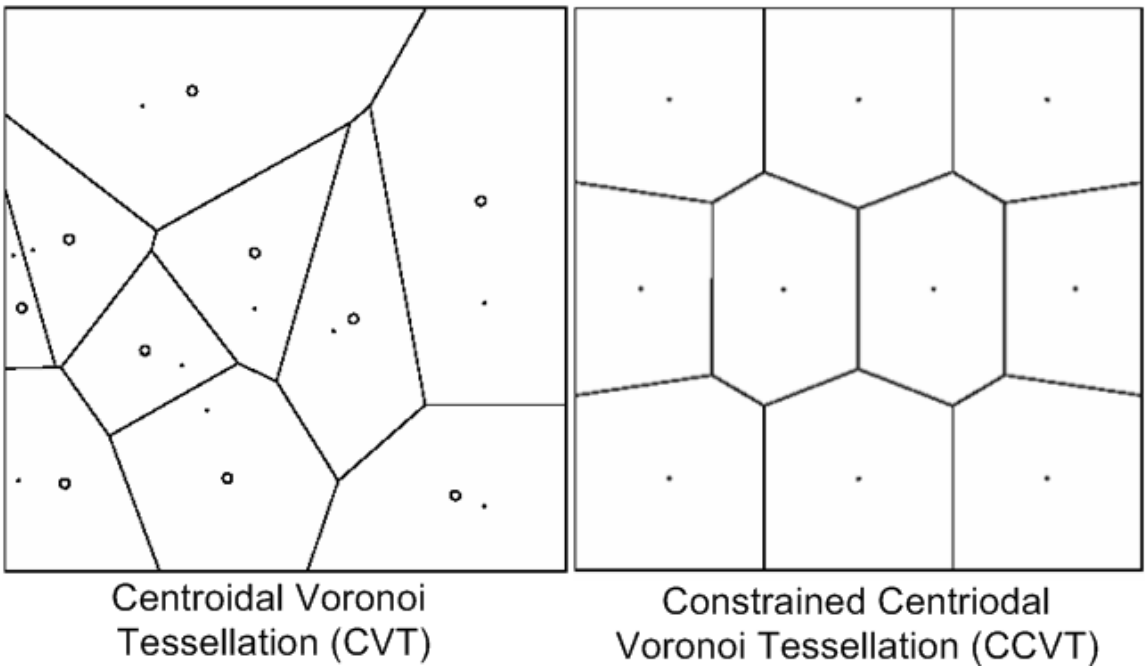
Speciálisan, az érrendszer pontjaira alkalmazva az eljárást, az Ω halmaz az érháló pontjait tartalmazza ($\Omega \subset R^2$), amelyből a $\{z_k\}_{k=1}^K$ generátorok *K* elemszámú kezdeti halmazát véletlenszerűen választottuk ki. A *K* értékét a mintavételezés foka határozza meg $K = \left\lfloor |\Omega| * \frac{mintavételezés\ foka(\%)}{100} \right\rfloor$.
Véges halmazokon a centroidot diszkrét módon kiszámolhatjuk, a következő képletet használva: $z_k = \frac{\sum\{p|p \in V_k\}}{|V_k|}$.

A végső mintavételezett pontokat az algoritmus által produkált $\{z_k\}_{k=1}^K$ generátorok adják, amelyek a Voronoi felbontás centroidjai. Ezek a pontok az érhálózat olyan mintavételét adják,

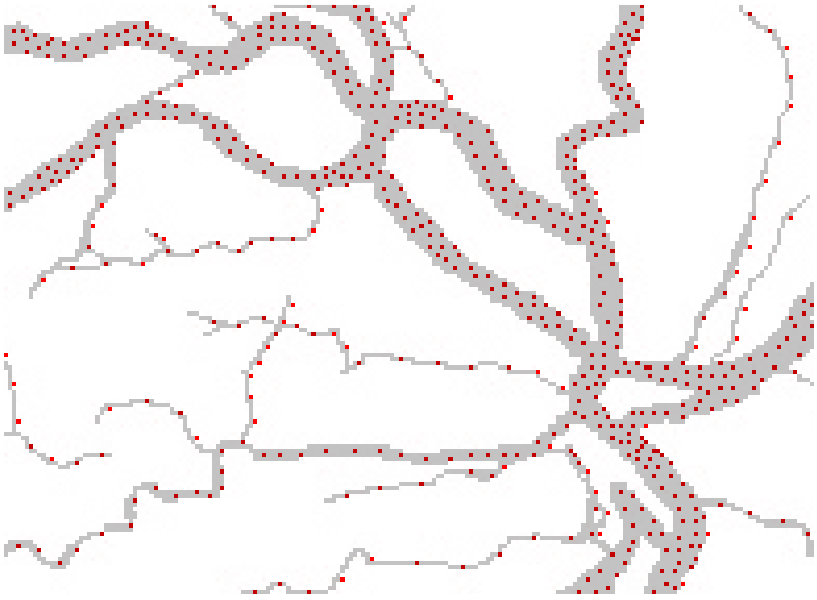
amiben a minta pontjai egyenletesen oszlanak el az érrendszeren, de nem jutnak ki az érrendszer széleire (7. és 8. ábra).



6. ábra A CVT algoritmus lépései



7. ábra Voronoi felosztás (balra), Központi Voronoi felosztás (jobbra)



8. ábra Az érhálózat központi Voronoi felbontás alapú mintavételezése. A mintavételezési pontok homogén módon töltik ki az érrendszert.

Feltételes központi Voronoi felbontás (CCVT) alapú mintavételezés

A központi Voronoi felbontás hátránya, hogy az *n* mintavételezés nem vezet ki a halmaz határára. Ez kompenzálható oly módon, hogy a halmazt mesterségesen felfújjuk az iteráció során, és a halmazon kívülre eső pontokat visszavetítjük a halmaz határára [11, 12].

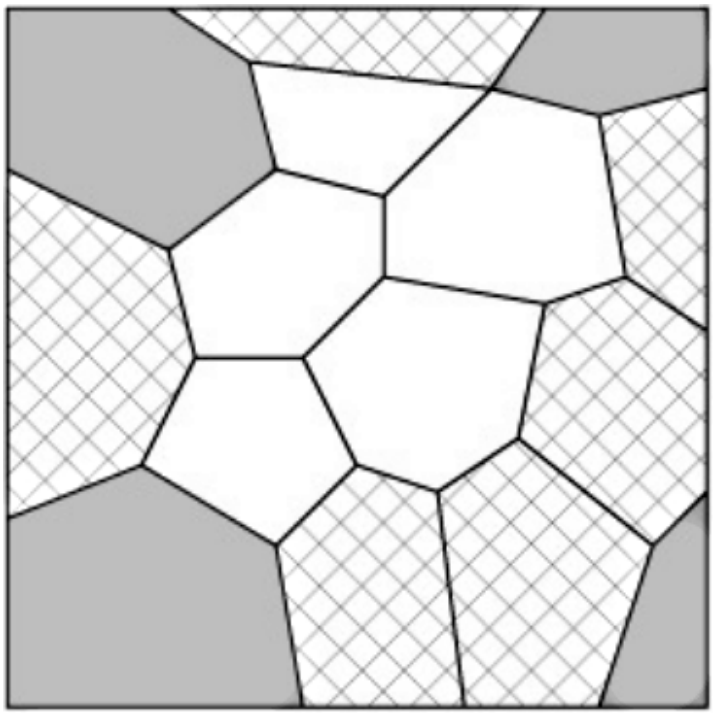
Az eljárás megadásához a központi Voronoi felbontásnál használt jelöléseket alkalmazzuk.

Lloyd módszerének módosított változata a CCVT mintavételezéshez [11]:

- Bemeneti paraméterek
  - Ω
  - K, generátorok száma
  - p, Ω halmazon értelmezett sűrűségfüggvény
- Kimenet
  - $\{V_k\}_{k=1}^{K}$, egy feltételes központi Voronoi felbontás K darab generátorral $\{z_k\}_{k=1}^{K}$

- Algoritmus lépései:
    1. válasszunk ki *K* kezdeti generátort $\{z_k\}_{k=1}^{K} \subset \Omega$
    2. Konstruáljuk meg az Ω halmaz $\{V_k\}_{k=1}^{K}$ Voronoi felosztását a $\{z_k\}_{k=1}^{K}$ generátorokkal.
    3. Határozzuk meg a Voronoi cellák $\{V_k\}_{k=1}^{K}$ centroidjait
    4. Mozgassuk a generátor pontokat $\{z_k\}_{k=1}^{K}$ a hozzá tartozó Voronoi cellák centroidjaiba
    5. Minden Voronoi cellához rendeljünk egy jellemzőt, hogy az adott cella határ- vagy sarok régió. (lásd 9. ábra)
    6. Ha V_k határ- vagy sarokrégió, akkor vetítsük a hozzá tartozó z_k generátort a halmaz határára.
    7. Ha az új pontok megfelelnek a feltételeknek, akkor vége az algoritmusnak. Egyébként hajtsuk végre ismét a lépéseket a 2. ponttól.

A mintavételezés helyét az algoritmus által produkált $\{z_k\}_{k=1}^{K}$ generátorok adják, amelyek a Voronoi felbontás feltételes centroidjai. Ezek a pontok az érhálózat olyan mintavételét adják, amiben a minta pontjai az érrendszer határára (érfal) is eljutnak (10. ábra).



9. ábra Négyzet felbontásának a sarok régiói (árnyékolt), határrégiói (négyzetrácsos), és a közbenső régiói (fehér).

10. ábra Az érhálózat feltételes központi Voronoi felbontás alapú mintavételezése. A mintavételezési pontok homogén módon töltik ki az érrendszert miközben az érrendszer határáról is mintavételhez jutunk.

Kombinált mintavételezés

A kombinált mintavételezéshez felhasználjuk a CVT és a CCVT felbontás eredményeit. A mintavételezés fokának felével elvégezzük mind a CVT mind a CCVT felbontást, és ezeknek a mintáknak az uniója adja a kombinált mintát. A megközelítés célja, hogy mind a halmaz belsejéből, mind annak határáról mintapontokhoz jussunk (11. ábra).

11. ábra Az érhálózat kombinált mintavételezése. Ötvözi a CVT és a CCVT tulajdonságait.

Betanítás

Betanítás alatt egy tanító minta előállítását értjük. Nyilvánvalóan a fent bemutatott algoritmus hatékony működésének feltétele egy megfelelően előállított tanító minta. A tanító minta előállításában fontos szerepet játszik az adott objektumot leíró számszerű attribútumok kiválasztása. Mi ez utóbbi meghatározásában [7]-ban közölt algoritmusra hagyatkoztunk, azaz a 30 Gauss szűrők konvolúciós maszkválaszainak eredményére zöld színcsatornán, továbbá a zöld színcsatornán mért intenzitás értékekre. Ilyen feltételek mellett a sajátság vektor dimenziója 31, továbbá a tanító mintának tartalmaznia kell minden sajátság vektorhoz egy címkét.

A tanító minta előállításához nem elegendő az attribútumok rögzítése, szükség van az adott osztályokból – érrendszerpontokból és háttérpontokból – vett mintára, mivel a teljes halmazok akkora méretű tanító mintát eredményeznek, amelynek kezelése technikailag problematikus és lassú. A mintavételezést többféleképp is elvégezhetjük, mi a fent bemutatott véletlenszerű, központi Voronoi felbontáson alapuló, feltételes központi Voronoi felbontáson alapuló, és az utóbbi két módszer ötvözeteként létrehozott kombinált mintavételező módszereket teszteltük.

Továbbá szükséges meghatározni a tanító mintának az egyszerűsítési fokát, vagyis hogy a mintavételezés az eredeti halmaz pontjainak hány százalékát tartsa meg. Ha az egyszerűsítési fokot túl kicsinek választjuk, azaz az eredeti halmaz kis része vesz részt a betanítási folyamatban, akkor várhatóan az osztályozó algoritmus az ismeretlen sajátságvektorokat hibás osztályba fogja sorolni. Ellenben ha az egyszerűsítési fokot túl nagynak választjuk, akkor a minta mérete és a túltanulási hatás miatt a végrehajtás költséges és pontatlan lesz.

Ahhoz, hogy tanító mintákat hozhassunk létre szükséges rögzíteni egy adatbázist, amely a szemfenék képeken felül a manuálisan szegmentált érrendszert és a háttérpontok halmazát is tartalmazza, minden egyes szemfenék képhez. Található olyan publikus adatbázis az interneten, amely a feltételeknek eleget tesz, és szabadon felhasználható [9]. Mi a betanítási folyamathoz tíz képet és a hozzájuk tartozó leírókat választottuk ki.

Összefoglalva, a betanítás négy komponensből tevődik össze, amelyek a következők:

- sajátságok
- mintavételező metódus

- mintavételező egyszerűsítési foka
- adatbázis

A betanítást minden esetben ugyanazon az adatbázison hajtottuk végre, és az attribútumokon sem változtattunk. A betanítással kapcsolatban két fajta jóság értéket definiáltunk: tanító minta jósága és a mintavételező jósága.

A tanító minta jósága alatt azt a jóság értéket értjük, amit a szemfenék érrendszer szegmentáló algoritmus által szegmentált érrendszer és a manuálisan meghatározott érrendszer szimmetrikus differenciájával mérünk, és ez a mérőszám kizárólag a mintavételező metódustól és a mintavételezés fokától függ. Azaz rögzített környezet mellett csak a mintavételező metóduson és a mintavételezés fokán változtathatunk.

A mintavételező jóságát hasonlóan definiáljuk, mint a tanító minta jóságát, azzal a különbséggel, hogy a jóság értékéke nem függhet a mintavételezés fokától. Vagyis rögzített környezet mellett kizárólag a mintavételező metóduson változtathatunk.

Tesztelés

A tesztelés elvégzéséhez egy teszt adatbázist kellett létrehozni, amelyben minden szemfenék kép –a hozzá tartozó leírókkal együtt – különbözik a betanításhoz használt adatbázistól. A teszt adatbázis négy szemfenék képet, a hozzájuk tartozó manuálisan kijelölt érrendszer pontokat és háttérpontokat tartalmazta. Ezek a képek azonos mérettel rendelkeztek, és szintén a nyilvánosan elérhető DRIVE adatbázisból származnak.

Az érrendszer detektálásához használt osztályozó algoritmus nem szolgáltathat azonos tanító mintával betanítva, azonos képen végrehajtva különböző jóság értékeket, mert ekkor már nem rögzített környezeten hajtanánk végre a tesztelést. Ezért fontos, hogy mindig azonos algoritmussal és a hozzá tartozó rögzített konfigurációval történjen a teszteles. A rögzített konfiguráció a bemutatott detektáló *k*NN algoritmusnál egy *k* érték meghatározásában merül ki, amelyet – igazodva a publikációhoz – harmincnak választottunk. Vagyis egy sajátságvektor osztályba sorolásánál a hozzá legközelebb eső harminc - tanító mintában szereplő - vektor vesz rész.

Minden egyes teszt adatbázisból származó szemfenék képen mértük az érrendszer szegmentálásának sebességét. Ahhoz hogy a különböző teljesítménnyel rendelkező számítógépek közti különbséget kizárjuk, minden tesztet egy - erre a célra rendszeresített – számítógépen hajtottuk végre. Feltételeztük, hogy az azonos elemszámmal rendelkező tanító mintákkal betanított algoritmusok futási ideje közel azonos lesz, így ezeket a mintákat csoportba soroltuk, és hogy az esetleges – operációs rendszer működése miatti - eltéréseket minimalizáljuk, a mért sebességek átlagát számítottuk.

A betanítás jóságának mérése az egész tesztelési folyamat során objektíven zajlott, mivel a teszteléshez fix környezetet tekintettünk, azaz rögzítettünk egy adatbázist, amelynek minden eleme részt vett a betanítási folyamatban, nem változtatunk a sajátságvektor előállításán (attribútumain), azonos osztályozó algoritmust meghatározott konfigurációval használtunk, továbbá minden tesztesetet egy erre a célra létrehozott teszt adatbázison futattuk végig. Így a jóságértékek amit eredményül kaptunk, kizárólag a definícióban szereplő tulajdonságoktól függhetnek (mintavételezés foka, mintavételezés módja). Egy betanításhoz tartozó betanítási jóság értékének meghatározásához a teszt adatbázis minden elemén végrehajtott érrendszer szegmentálás (szimmetrikus differencia) jóságának átlagát számítottuk. Ezzel a jóság értékkel lehetőségünk nyílik arra, hogy egy adott sebességkorláton belül maradva megadjuk azt a mintavételező módszert, és a hozzá tartozó mintavételezés fokát, amellyel a szegmentálás a vizsgált feltételek mellett a lehető legpontosabb lesz.

A mintavételezés jóságának méréséhez szorosabb feltétel kell szabni, azaz a mintavételezés fokának azonosnak kell lennie. Így azt érjük el, hogy az eredményképp előálló jóság érték kizárólag a mintavételezési stratégiától függ. Ennek a feltételnek az elérése érdekében előre rögzítettük néhányat a lehetséges mintavételezési fokok közül. Ezek százalékban megadva a következőek: 0,05%, 0,5%, 1%, 5%, 10%. Ezeket az értékeket kombináltuk a négy lehetséges mintavételezési stratégiával. Így létrejött 4*5=20 különböző tanító minta. Az összes tanító mintával betanított szegmentáló algoritmust lefutattuk a teszt adatbázis valamennyi elemén, és a szimmetrikus differencia segítségével meghatároztuk a pontosságot (jóságot).

Az előző tesztelési eljárás egyik hátránya, hogy kisméretű tanító adatbázissal rendelkezik. Hogy rávilágítsunk a különböző mintavételezési stratégiák által előálló tanítási jóságértékek különbségeire, és hogy ez említett hiányosságon javítsunk egy nagyobb tesztadatbázis került

rögzítésre. Ez az adatbázis már húsz képet és a hozzá tartozó leírókat foglalta magában, és szintén a DRIVE adatbázisból származott.

További különbség, hogy ezekben a tesztekben bekapcsolásra került az előszűrő eljárás, ezért a végrehajtási sebesség nem került mérésre. Továbbá az előző tesztelés eredményeit felhasználva, már csak a véletlenszerű és a CCVT mintavételező stratégia alkalmazására korlátoztuk a betanító folyamatot.

Itt jegyeznénk meg, hogy mind a betanítás alatt, mind a tesztelés során használt adatbázisokban megtalálható képek 565*584 felbontással rendelkeznek, illetve jpg fájlformátumban kerültek tárolásra. A jpg formátum talán nem a legalkalmasabb erre a célra, de mivel a témában megjelent publikáció közül a legtöbb a DRIVE adatbázisból származó képeket használja - azaz a jpg formátumot – így elhanyagolhatónak véltük az ebből adódó pontatlanságot. Az azonos képméret fontos, mert a *k*NN osztályozó érzékeny az attribútumok skálázására.

Implementáció

A program fejlesztéséhez olyan platformot kerestünk, amely könnyen és gyorsan programozható, továbbá fejlett hibakeresési eszközrendszerrel rendelkezik és léteznek hozzá olyan keretrendszerek, programkönyvtárak, amelyek a gépi tanulás, osztályozás és a képfeldolgozási algoritmusok implementálásában segítenek, megvalósítják azt. Választásunk a Java-ra esett, amely egy magas szintű platform független objektum orientált programozási nyelv és környezet. Egy Java programot egy virtuális gép futtat, amelyet minden jelentős operációs rendszerhez megvalósítottak. A modern virtuális gépek teljesítménye, így a java nyelven írt programok futási ideje már felveszi a lépést a natív programkódok teljesítményével. Java nyelven a megjelenése óta számos keretrendszert és programkönyvtár készült, melyek használatával általában a fejlesztési idő csökken és a megbízhatóság nő.

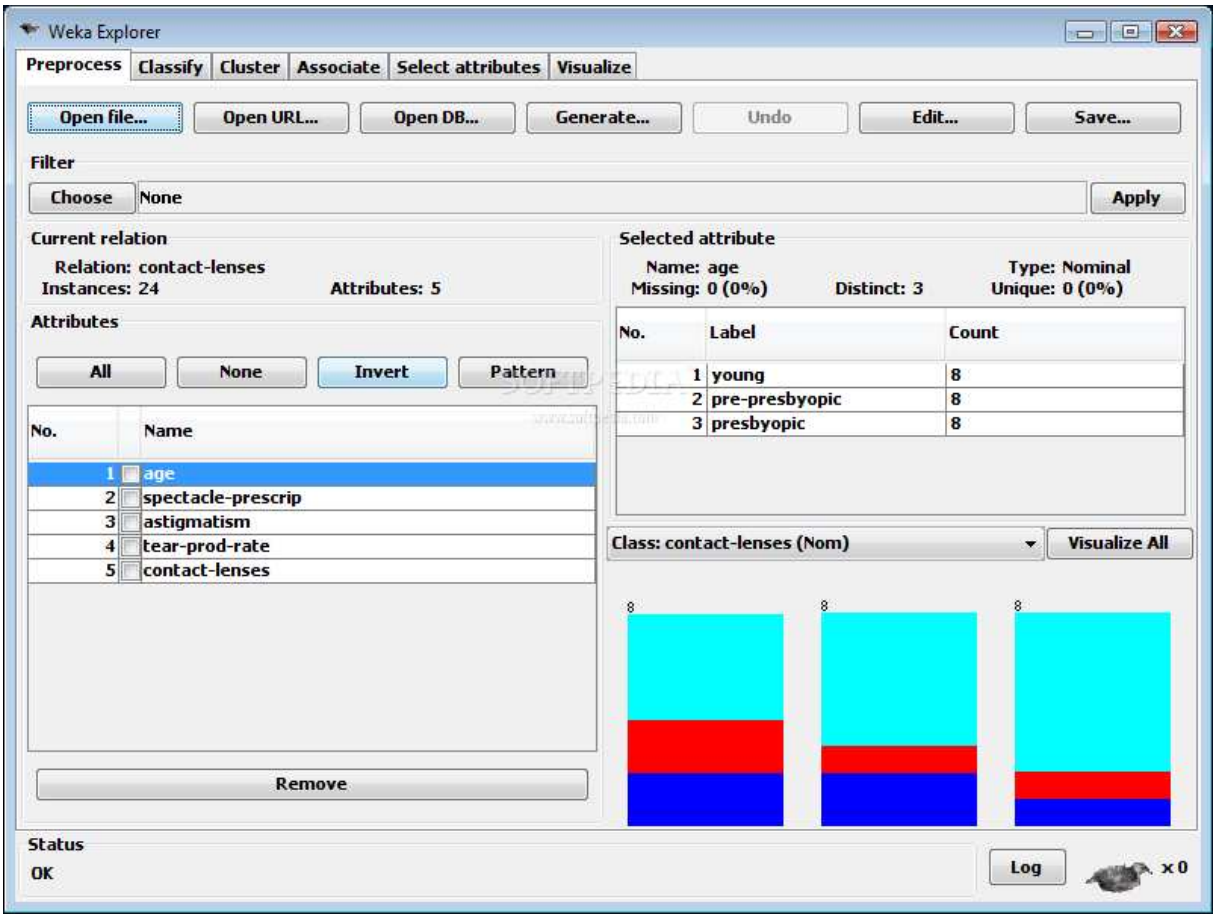
Felhasznált eszközök

A képfeldolgozási algoritmusok implementálásáról, a képek háttértárról történő betöltéséről és lementéséről a JAI (Java Advanced Imaging [13]) keretrendszer gondoskodott. A JAI használatával objektumorientált eszközrendszeren keresztül vagyunk képesek képfeldolgozási algoritmusok végrehajtására. Több általános célú képfeldolgozó algoritmust tartalmaz. Valamennyi modern képformátumot képes kezelni, így levéve a programozó válláról az ilyen kódolási eljárások megvalósításának terhét. A JAI tervezésénél nagy figyelmet fordítottak arra, hogy a platformfüggetlenséget megőrizzék, amellett hogy magas teljesítményt érjenek el. Így elkészült egy tisztán Java nyelven írt változata is, illetve egy natív részeket tartalmazó platform specifikus verzióját is kiadták, amely kihasználja a natív programkódok adta lehetőségeket feladva a platformfüggetlenséget. Objektumközpontú fejlesztés és tervezés révén könnyen kiterjeszthető programcsomaggá vált.

A program fejlesztéséhez a JAI 1.3-as verzióját használtuk, a teszteléshez pedig a platformfüggő natív megvalósítását.

Az osztályozó algoritmusokhoz és működéséhez szükséges tanító minták kezeléséről a Weka [14] programkönyvtár került felhasználásra. A Weka adatbányászati algortimusokat, eszközöket tartalmaz, amelyek a Java-ban írt forráskóddal együtt bárki számára szabadon

hozzáférhetőek. Széles körben elterjedéséhez hozzájárulhatott, hogy kezelésére könnyen kezelhető grafikus felület áll rendelkezésre (12. ábra).



12. ábra Képernyőkép a Weka-ról

A program felépítése

A program felépítése alapvetően két részre tagolható. Egyik részben az alkalmazott algoritmus és a működéséhez szükséges osztályok kaptak helyet, magában foglalva az adatbázis bejáráshoz szükséges eszközöket és a hibaméréshez használt függvényeket. Ez egy különálló programban került megvalósításra. A másik részben a betanításhoz szükséges eszközök, osztályok szerepeltek. Ez utóbbi program felhasználta az előbbi osztályait.

A tanító mintát előállító program megvalósítása

A tanító mintát előállító program feladata, hogy egy Arff [15] fájlformátumú fájlt hozzon létre (tartalmazva a teljes tanító mintát), amely a későbbi tesztelés folyamán kerül felhasználásra.

Az Arff a Weka saját formátuma. Alapvetően ASCII karaktereket tartalmazó szöveges fájl, amely két részre bontható egy fejlécre és egy adatrészre. A fejlécben az attribútumok és a hozzájuk kapcsolódó típusok kerülnek felsorolásra, az adatrészben pedig a sajátságvektorok szerepelnek. Egy sorban egy vektor és a vektor elemei vesszővel vannak elválasztva.

Az általunk használt Arff fájlformátumra egy további megszorítást kötöttünk ki, miszerint az attribútum nevének az attribútumot előállító szűrő szöveges reprezentációjának kell lennie. Ezzel az egyszerű megoldással fenntarthatjuk a későbbi bővítés lehetőségét.

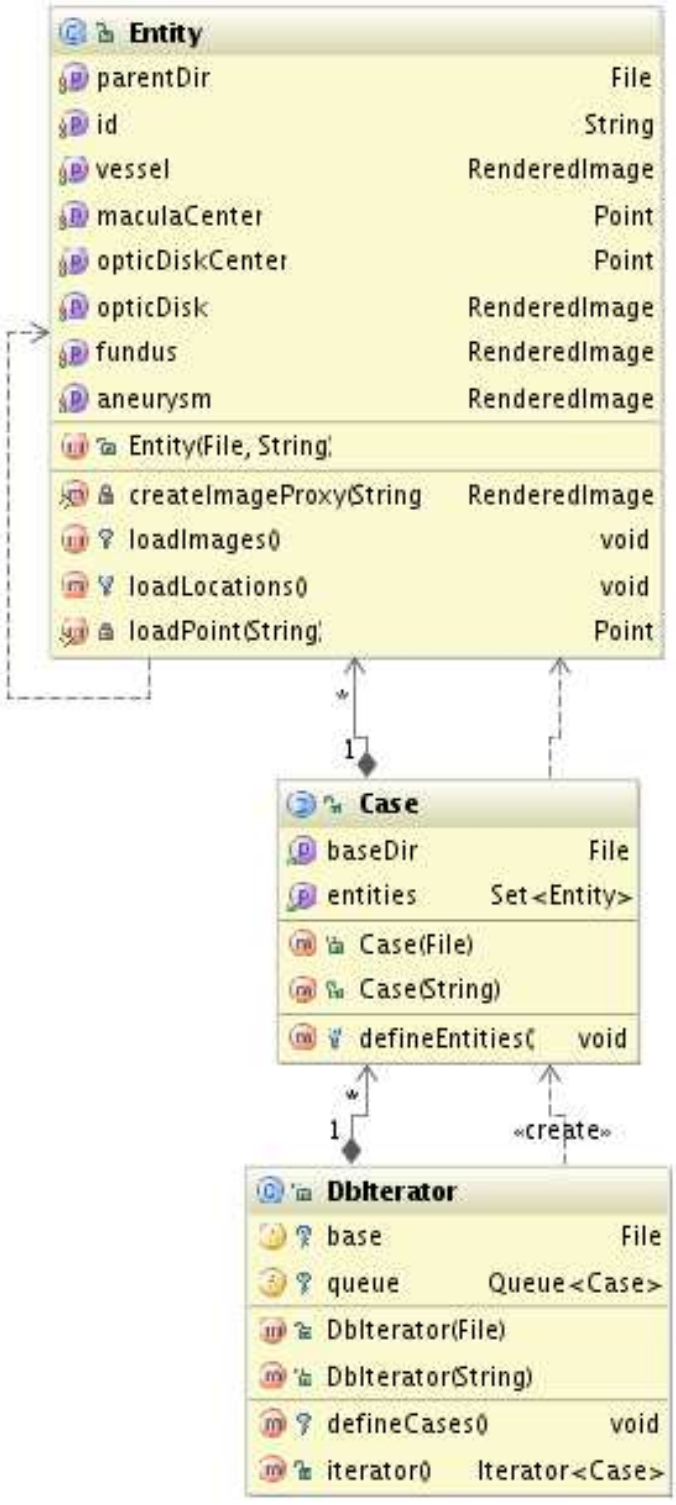
A program funkciói:

- Adatbázis bejárása
- Mintavételezés pontok kinyerése
- Sajátságvektorok előállítása
- Arff fájl összeállítása

Az adatbázis bejárása mind a tesztelés mind a betanítás folyamán fel lett használva. Feladata, hogy egy olyan könnyen kezelhető eszközt adjon a programozó kezébe, amellyel egy előre meghatározott fájlstruktúrába rendezett szemfenék képekhez és a hozzá tartozó leírókhoz kényelmesen lehessen hozzáférni, bejárni. Az adatbázis bejárását a következő osztályok végzik (13. ábra):

- *Entity*: Egy konkrét szemfenék képet, és a hozzá tartozó leírókat reprezentáló adathozzáférést lehetővé tevő osztály. Későbbi felhasználás miatt a dolgozatban szereplő érrendszer vizsgálathoz szükséges leírókon felül további metaadatok is megvalósításra kerültek.
- *Case*: Egy pacienshez több szemfenék kép is kapcsolódhat. Elkülöníthetünk bal illetve jobb szemről készült, továbbá sárgafoltra és éleslátás helyére fókuszált képeket. Az ilyen jellegű megkülönböztetések a későbbi fejlesztés folyamán juthatnak szerephez, a jelenlegi vizsgálatot nem befolyásolják. Ezen osztály példánya segítségével nyerhetjük ki az *Entity* osztály példányait.
- *DbIterator*: Magát az adatbázis bejárót reprezentálja. Ha az adatbázis megfelelő struktúrával rendelkezik, akkor az adatbázis útvonalának megadása után az adatbázisban szereplő valamennyi szemfenék képhez és hozzá tartozó leíróhoz

hozzáférhetünk. Ezen osztályon keresztül férhetünk hozzá a *Case* osztályok példányaihoz.



13. ábra Adatbázis bejárást megvalósító osztályok szerkezete

A képtéren történő mintavételezési pontok kinyeréséhez egy külső programot alkalmaztunk. Ez a program a következő bemeneteket kapta meg:

- Mintavételezendő bináris kép útvonala
- Mintavételezési stratégia (CVT, CCVT)
- Mintavételezés mértéke az előtérpontok százalékában megadva
- Végrehajtandó iterációk száma. Ha ez nulla, akkor a véletlenszerű mintavételezéshez jutunk.

Kimenetként egy bináris képet készít el, amelyik csak a mintavételezési pontokat hagyta meg előtérpontnak a bemenetként kapott bináris képből.

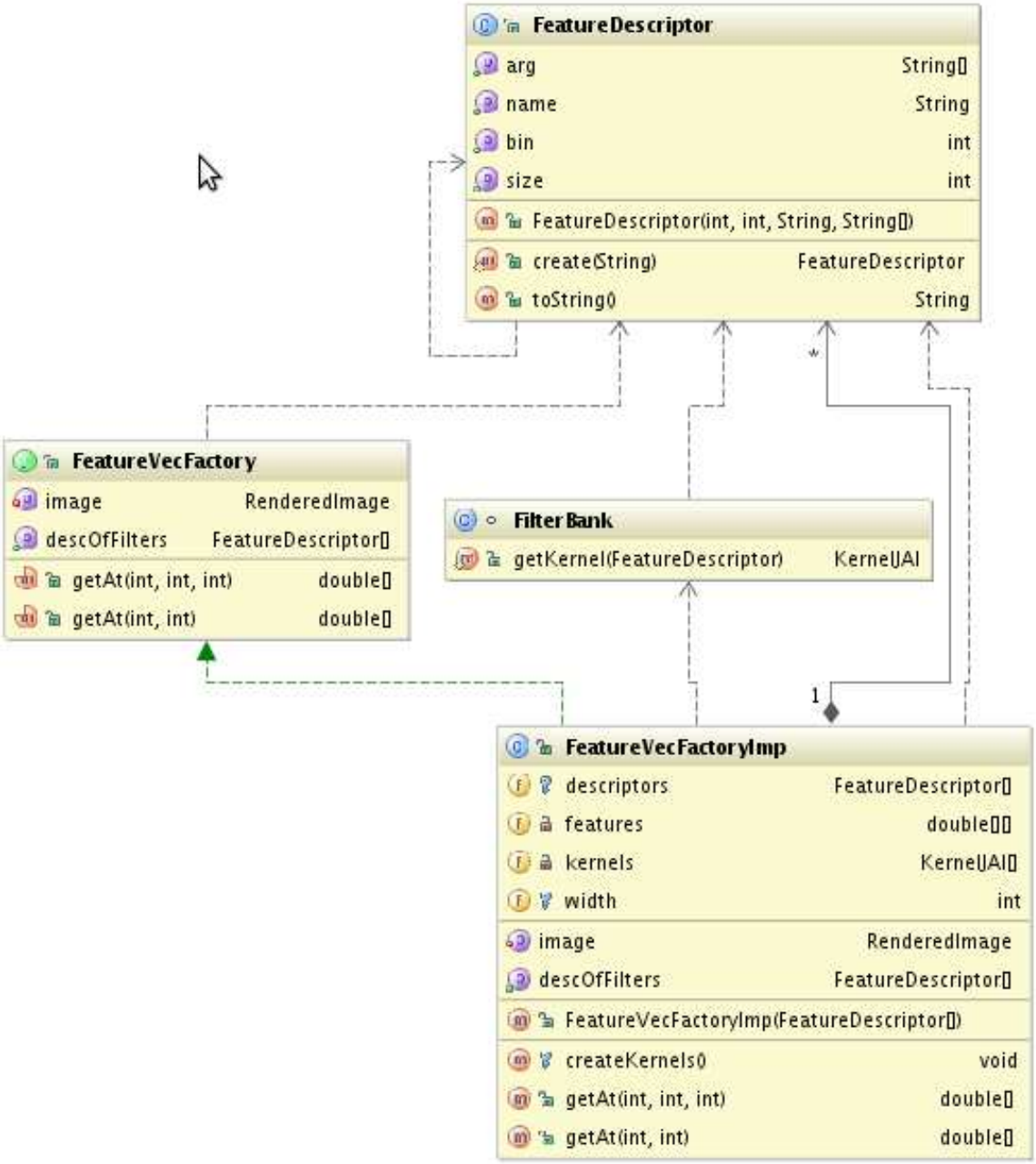
Ezt a külső hívást egy Java metódusba ágyaztuk. A mintavételező lefutása után, a mintavételezési pontokat tartalmazó képet betöltöttük, majd az előtérpontok helyzetéből hozzájutottunk a mintavételezési pontok koordinátájához. Itt mindig külön kezelendő az érrendszerről történő mintavételezés és a háttéren (ROI érrendszeren kívül eső részén) történő mintavételezés. Utóbbi esetben mindig véletlenszerű mintavételezésről van szó.

A mintavételezési pontok meghatározása után ki kell nyerni a szemfenék képből a megfelelő pontokban vett sajátságvektorokat. A megvalósított osztályok egyik követelménye, hogy képes legyen egy szűrő lista, egy szemfenék kép és egy pozicció megadása után a pozicciához tartozó sajátságvektor előállítására, tovább rendelkezzen olyan gyártó függvénnyel, amely egy szűrő szöveges reprezentációjából képes elkészíteni a konkrét szűrőt. Utóbbi a tesztelés során jut szerephez, mert az Arff fájl attribútum neve alapján képes a betanítás során alkalmazott szűrőket rekonstruálni.

A sajátságvektorok előállításában résztvevő osztályok (14. ábra):

- *FeatureDescriptor*: A szűrőt reprezentáló osztály. A szűrők megfelelő szöveges reprezentációjából készíthető egy ilyen osztály példánya, illetve egy ilyen példányból kényelmesen készíthető a hozzá tartozó szűrő megfelelő szöveges reprezentációja. Segítségével egy tetszőleges szűrő leírható.
- *FeatureVecFactory*: Interface amely definiálja, hogy egy képből sajátságvektorokat gyártó osztálynak milyen metódusokat kell megvalósítania.

- *FeatureVecFactoryImp*: *FeatureVecFactory* egy implementációja. *FeatureDescriptor* példányokat tartalmazó lista és egy kép helyes megadása után, a kép egy tetszőleges pontjához képes elkészíteni az adott ponthoz tartozó sajátságvektort.
- *FilterBank*: Egyke, amely egy *FeatureDescriptor* példányból képes létrehozni a neki megfelelő szűrőt.



14. ábra sajátságvektorok előállításában résztvevő osztályok felépítése

Miután a tanító mintát előállítottuk, szükségessé válik a háttértárolókra történő mentése. Kézenfekvő választás az Arff fájlformátum volt, amely az említett konvenciók mellett teljes körűen kielégíti a támasztott követelményeket, továbbá a Weka csomag részét képezik az Arff fájlokat kezelő osztályok.

Összefoglalva a tanító mintának elkészítéséhez létrehoztunk egy rögzített struktúrájú adatbázis bejárását lehetővé tevő eszközöket, amelyek segítségével például a DRIVE adatbázis tanításra és tesztelésre is alkalmas részhalmazainak tartalmát programozási környezetben is könnyen elérhetővé tettük. Az adatbázis entitásain lépkedve minden szemfenék képhez tartozó érrendszerből és a hátteréből is mintapontokhoz jutottunk, amely egy külső programhívás alkalmazásával valósult meg. Miután a mintapontok rendelkezésre állnak kinyerjük az adott pontokhoz tartozó sajátságvektorokat és megfelelő módon felcímkézzük őket. Miután a tanító adatbázis összes entitását feldolgoztuk a memóriában tárolt sajátságvektorok halmazát Arff fájlformátumban lemezre mentjük. A programot úgy fejlesztettük ki, hogy a különböző tanító minták előállításához elegendő legyen a program paramétereinek módosítása.

A tesztelést végrehajtó program megvalósítása

A tesztelést végrehajtó program feladata, hogy kimérje az adott tanító mintával betanított érrendszer detektáló algoritmus jóságértékét.

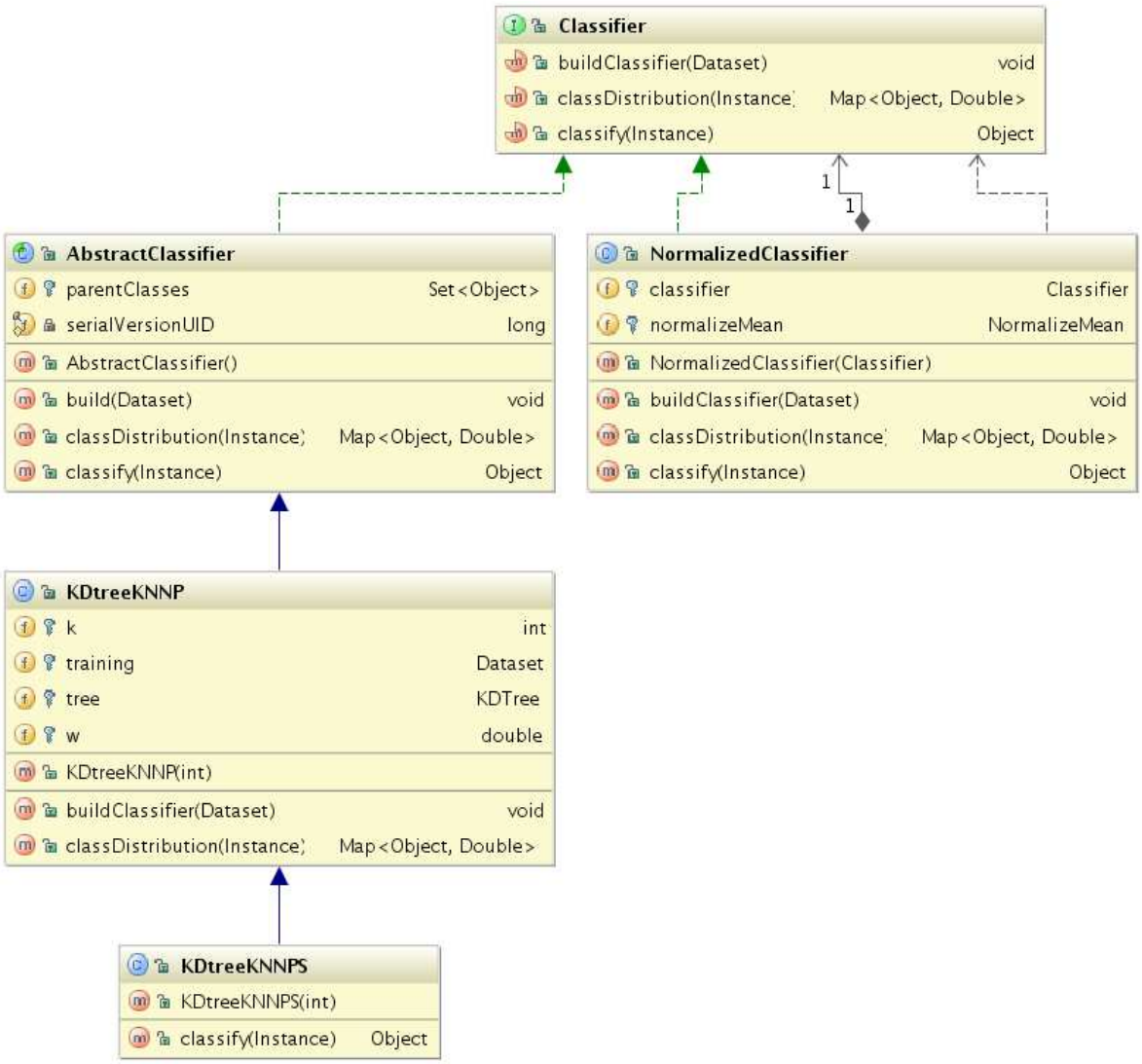
A program végrehajtásának lépései:

1. Arff fájl betöltése, megfelelő *FeatureVecFactory* előállítása
2. Osztályozó inicializálása
3. Teszt adatbázis elemeinek bejárása
4. Detektáló algoritmus végrehajtása a teszt adatbázis elemeire
5. Hibamérés

Az Arff formátumú fájlban tárolt tanító minta betöltéséhez a Weka *ArffLoader* osztályát használtuk fel. Ez az osztály az Arff fájlok tartalmát képzi le objektumok hierarchiájává. Miután a tanító mintát a memóriába töltöttük, az attribútumok neveit kell kinyernünk, és a nekik megfelelő – pontosan a minta létrehozásakor felhasznált – szűrőket kell létrehoznunk. Az attribútumok neveiből egy *FeautreDescriptor* szűrő leíró példányt készítünk el, majd

ezekkel a leírókkal inicializáljuk a *FeatureVecFactory*-t megvalósító *FeatureVecFactoryImp* egy példányát. Ez a példány a *FilterBank* osztálytól a szűrő leírójának megfelelő konkrét szűrőt (*KernelJAI*) állít elő. Vagyis az Arff létrehozásakor használt szűrőknek pontosan megfelelő szűrőket állítottuk elő az osztályozáshoz.

A *FeatureVecFactory* példánya képes a mintához illeszkedő sajátságvektorok előállítására egy szemfenék kép hozzáadása után. Az algoritmusnak megfelelően ezeket a sajátságvektorokat osztályozni kell. Korábban említésre került, hogy a tanító mintát a *k*NN osztályozó jellemzői alapján standardizálni érdemes. Ehhez a Weka absztrakt osztályozójából származó csomagoló osztályt hoztunk létre (*NormalizedClassifier*), amely a paraméterül kapott osztályozó algoritmust használja (15. ábra). Mielőtt tanító mintát a belső osztályozó felé delegálná, meghatározza a transzformáció paramétereit, és végrehajtja a normalizálást. Ha egy ismeretlen vektor kerül osztályozásra, ugyanezen transzformáció végrehajtása után történik a delegálás a belső osztályozó felé. Így ezzel a csomagoló osztály segítségével elfedhetjük a normalizálás részleteit.



15. ábra Osztályozó osztályok szerkezete

KDtreeKNNP osztály a *k*NN algoritmust implementálja illeszkedve a Weka osztályhierarchiájába. A megvalósítás annyiban tér el a Weka-ban használt *k*NN osztályozótól, hogy hatékonyabbá teszi az osztályozás eredményeként előálló valószínűségi értékek meghatározását. Utóbbi osztálynak egy specializációja a *KDTreeKNNPS* osztály, amely már kifejezetten az érrendszerből származó sajátság vektorok osztályozásához készült, tovább gyorsítva az osztályozást. Feladata a kapott vektor érrendszerhez tartozás valószínűségének meghatározása, a kapott tanító minta alapján.

A teszt adatbázis bejárásához a már bemutatott *DbIterator* osztályt használtuk fel. Az adatbázis elérésének megadása után, minden egyes elemet érintve bejártuk a teljes adatbázist.

A szemfenék képekre – vagyis az adatbázis elemeire - egyenként lefutattuk az érrendszer detektáló algoritmust.

Az algoritmus osztályozóját már létrehoztuk, a normalizált mintát felhasználva képes az újonnan érkező vektorok osztályozására. A *FeatureVecFactory* egy szemfenék képet és egy hozzá tartozó ROI-t megkapva képes a ROI előtérpontjai által jelzett helyen a sajátságvektorok előállítására (ez a ROI az előszűrő alkalmazása mellett nem a szemfenék kép számunkra lényeges részét tartalmazza, hanem ennek részhalmazát az érrendszer detektálása szempontjából fontos pontokat). A valószínűségi értékek küszöbölése után, egy bináris képet állít elő. Ez a bináris kép az algoritmus kimenete, a detektált érrendszert tartalmazza.

Miután az algoritmus létrehozta a kimenetét mértük a detektálás hibáját. Az *Entity* osztály segítségével hozzáfértünk az adatbázisban szereplő kézzel szegmentált érrendszerhez, amely szintén bináris kép. Bináris JAI operátorokat használva a szimmetrikus differencia számszerű értéke a standard kimenetre kiírásra került a vizsgált eset (szemfenék kép, hozzá tartozó érrendszer, ROI) azonosítójával együtt. A kimenetet egy fájlba irányítva, majd Exel táblázatban feldolgozva meghatároztuk a tanító mintákra jellemző jóságértékeket.

A teszt adatbázis és az Arff fájl elérésének megadása után a program az adatbázis elemein végiglépkedve méri az adott tanító mintához tartozó jóságértéket, minden egyes szemfenék képre vonatkozóan. A program argumentumain keresztül jut hozzá ezekhez a beállításokhoz.

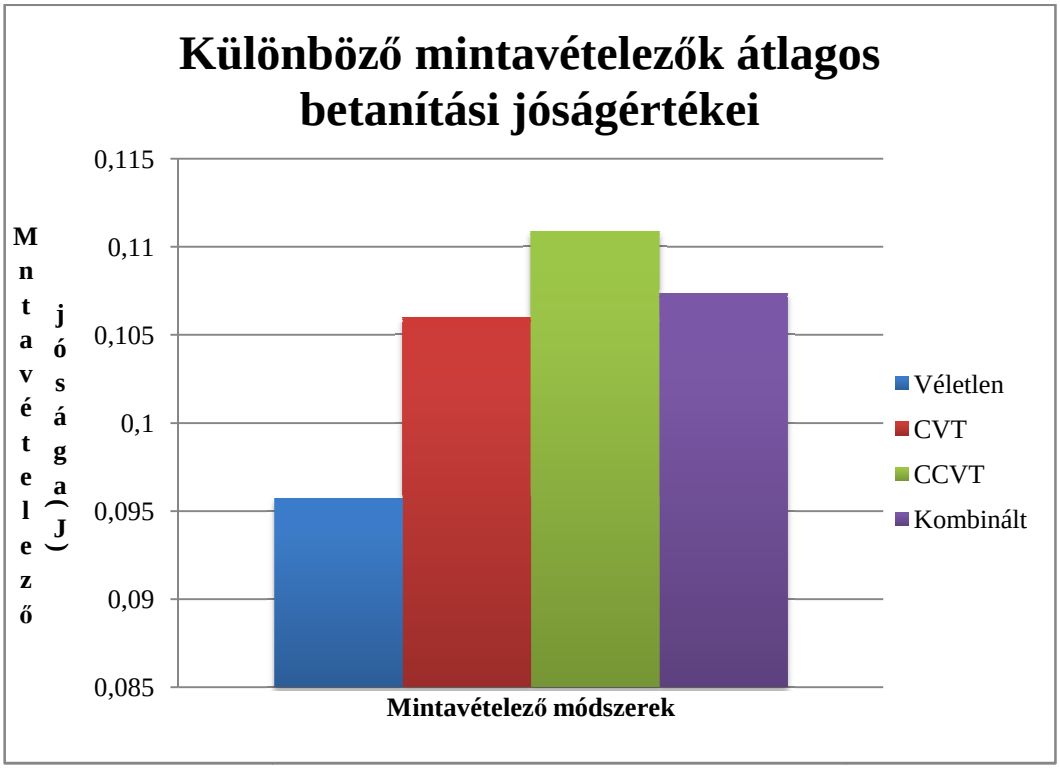
Eredmények

A bemutatott tesztelési eljárás eredményét a 1. táblázatban foglaltuk össze. Megtévesztő lehet, hogy a jóság mérésére használt *S* szimmetrikus differencia értékek 0-hoz közeledve jelzik a növekvő jóságot. Természetesebbnek tűnik a növekvő számértékekhez növekvő jóságértéket kapcsolni, így a későbbi diagramok megalkotásához a *J=1-S* képletet alkalmaztuk.

1. táblázat Teszteredmények

Tanító minta konstrukciója		**Teszt eredmények**				
Módszer	**Mintavételezés foka (%)**	**Teszt képek jóság értékei**				**Jóság átlag (S)**
		1.	**2.**	**3.**	**4.**	
Véletlen mintavételezés	10,000%	0,885	0,88	0,872	0,852	**0,87225**
	5,000%	0,894	0,885	0,878	0,874	**0,88275**
	1,000%	0,911	0,9	0,895	0,89	**0,899**
	0,500%	0,921	0,911	0,904	0,906	**0,9105**
	0,005%	0,963	0,955	0,958	0,951	**0,95675**
	összesített	**0,9148**	**0,9062**	**0,9014**	**0,8946**	**0,90425**
CVT mintavételezés	10,000%	0,895	0,886	0,879	0,871	**0,88275**
	5,000%	0,906	0,897	0,889	0,883	**0,89375**
	1,000%	0,906	0,899	0,892	0,887	**0,896**
	0,500%	0,905	0,905	0,9	0,894	**0,901**
	0,005%	0,894	0,888	0,894	0,91	**0,8965**
	összesített	**0,9012**	**0,895**	**0,8908**	**0,889**	**0,894**
CCVT mintavételezés	10,000%	0,878	0,867	0,86	0,852	**0,86425**
	5,000%	0,832	0,875	0,867	0,859	**0,85825**
	1,000%	0,895	0,889	0,882	0,878	**0,886**
	0,500%	0,904	0,893	0,893	0,879	**0,89225**
	0,005%	0,956	0,942	0,944	0,937	**0,94475**
	összesített	**0,893**	**0,8932**	**0,8892**	**0,881**	**0,8891**
Kombinált mintavételezés	10,000%	0,884	0,875	0,868	0,861	**0,872**
	5,000%	0,891	0,882	0,874	0,865	**0,878**
	1,000%	0,899	0,891	0,887	0,881	**0,8895**
	0,500%	0,903	0,895	0,89	0,887	**0,89375**
	0,005%	0,937	0,928	0,929	0,926	**0,93**
	összesített	**0,9028**	**0,8942**	**0,8896**	**0,884**	**0,89265**

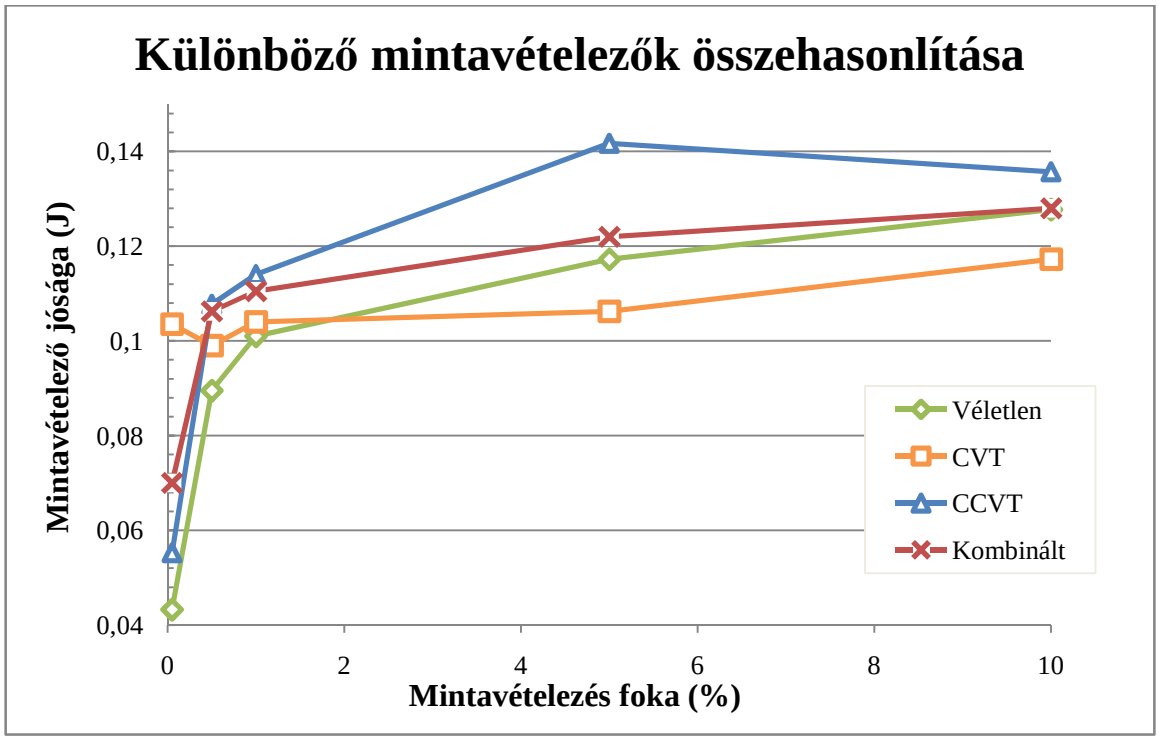
A 16. ábrán a mintavételezők jóságát hasonlítjuk össze úgy, hogy egy mintavételező módszer által szegmentált érhálózatra kapott jóság értékek átlagát számítjuk. A legjobb eredményt a CCVT mintavételező alkalmazásával értük el. A CVT és a kombinált mintavételező módszer jóságértéke megközelítőleg azonos, továbbá – a várakozásunknak megfelelően – a legrosszabb eredményt a véletlenszerű mintavételezés érte el.



16. ábra Különböző mintavételezők átlagos betanítási jóságértékei

A 17. ábrán a jóságértéket, a mintavételezés fokát és a különböző mintavételezőket hasonlítjuk össze. Az általunk vizsgált legkisebb tanító minta vizsgálatakor a CVT mintavételező kiemelkedően jó eredménnyel szerepelt, de a mintavételezés fokának növekedésével a CCVT kivétel nélkül a legjobb szegmentálást eredményezte. A CCVT által előálló tanító minta jóságértékének görbéjét vizsgálva szembetűnik, hogy a 10%-os mintavételezési fokkal történő mintavételezéskor rosszabb eredményt kaptunk, mint az 5%-os esetben tapasztaljuk. Ennek az lehet az oka, hogy az osztályozónál túltanulás lépett fel. Mivel egyetlen másik mintavételező használatakor sem tapasztalható ez a hatás, illetve hogy a CCVT algoritmus szinte az összes vizsgált elemszámú tanító minta esetén a legjobb eredményt adta, arra a hipotézisre jutottunk, hogy a CCVT által előálló tanító minta reprezentatívabb a tesztben szereplő többi mintavételezővel produkált tanító mintával

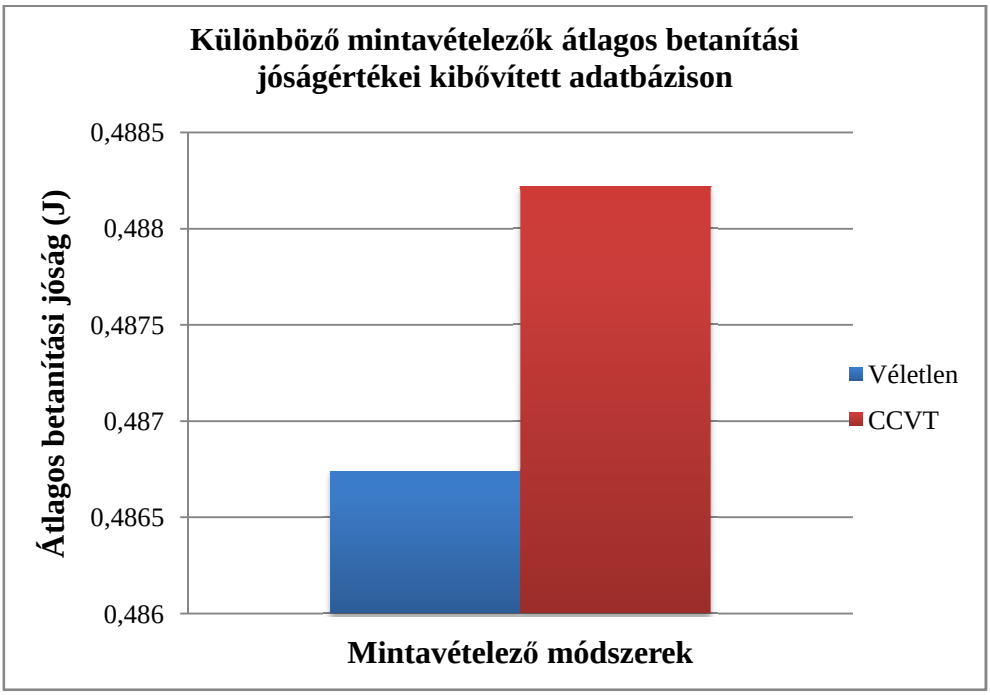
szemben [6]. A különböző mintavételezési fokokkal betanított algoritmus átlagos futási eredményei a 18. ábrán láthatóak.



17. ábra Különböző mintavételezők összehasonlítása

18. ábra A futási idő csökkenése

A kibővített adatbázisokon történő tesztelésre kapott jóságértékek a 19. ábrán láthatóak. Ebben az esetben a sebességmérést az előszűrő bevezetése miatt mellőztük. Jól látható hogy a CCVT mintavételező stratégia több tesztképen vizsgálva is jobb eredményt ért el a véletlenszerű mintavételezőhöz képest.

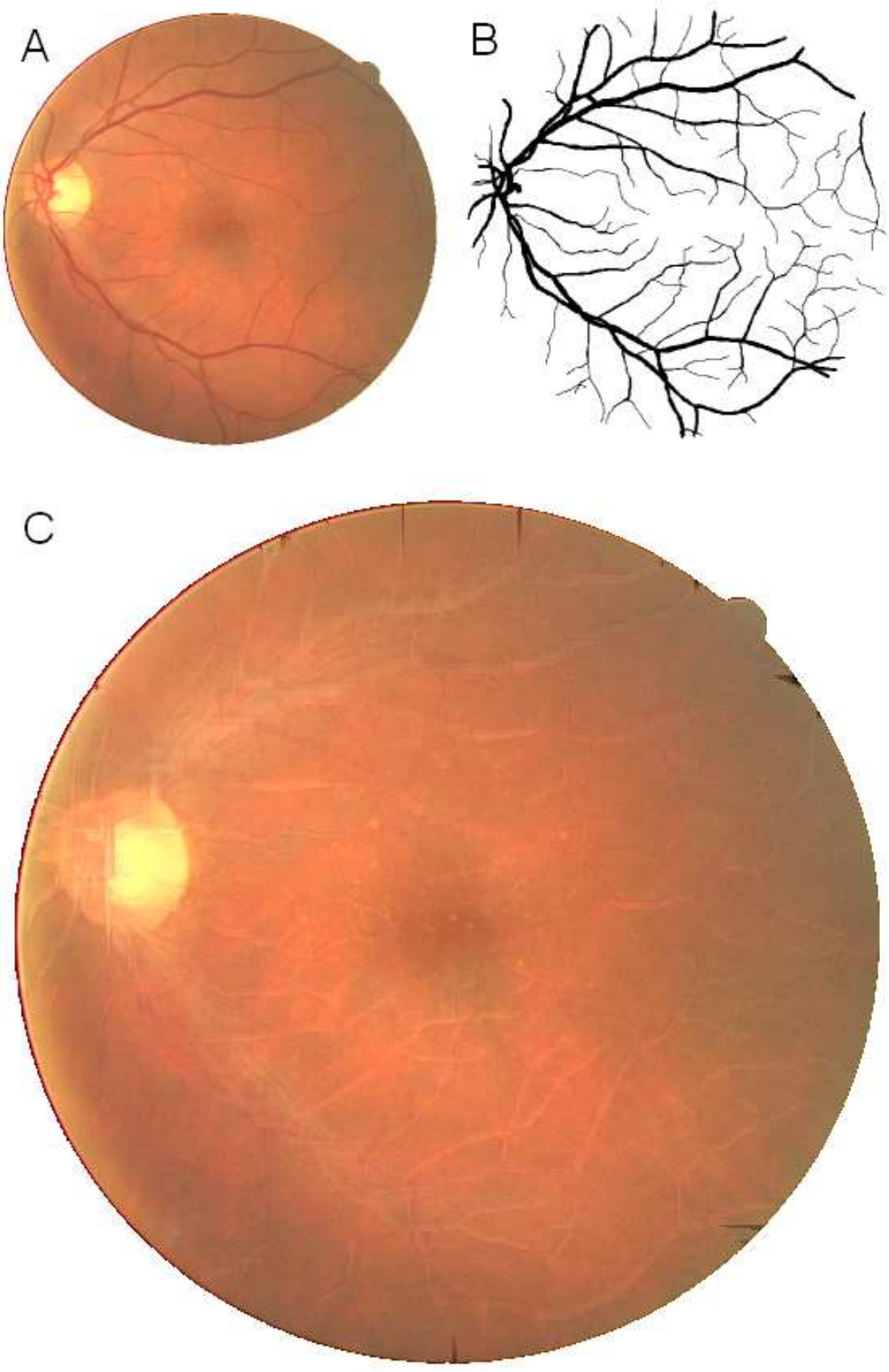


19. ábra Különböző mintavételezők átlagos betanítási jóságértékei kibővített adatbázison

Eredmények hasznosítása

Eredményeink egy szemszövődmények elváltozásait detektáló előszűrő rendszer elkészítése során több részfeladat megoldása esetén is felhasználható.

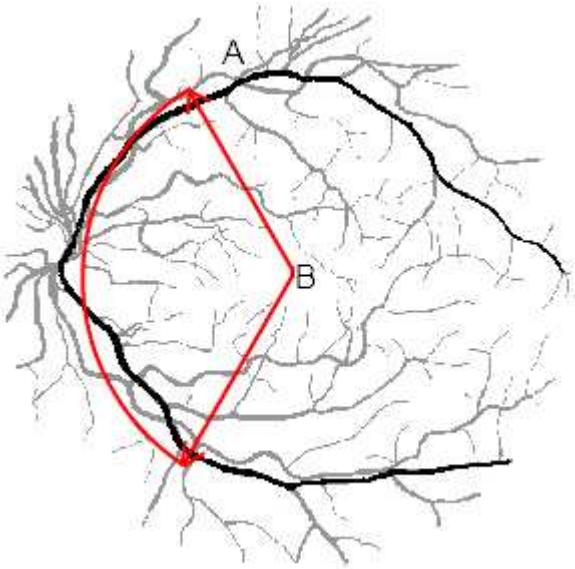
A mikroaneurizmák pontos szegmentálása kiemelt szerepet kap egy ilyen előszűrő rendszerben. A mikroaneurizmákhoz nagy hasonlóságot mutató képrészletek előfordulhatnak az érrendszeren, amelyek elkülönítése a tényleges mikroaneurizmáktól nagy nehézséget jelent a detektáló algoritmusnak. Gyakran az érrendszeren elhelyezkedő mikroaneurizmát jeleznek. A szemfenék kép érrendszerének eltüntetésével, azaz az érrendszer interpolálásával várhatóan a mikroaneurizma kereső módszerek javíthatóak (20. ábra).



20. ábra Szemfenék kép (A), érrendszer (B), szemfenék kép az érrendszer interpolációja után (C)

A mikroaneurizmák és egyéb elváltozások megjelenését a sárgafolton kiemelten kerülnek kezelésre, mivel a páciens látását veszélyeztetik. Így a sárgafolt pontos felismerése

elengedhetetlen. A szemfenék képek jellemzője a sárgafolt és a vakfolt távolsága, így a vakfolt ismerete mellett a sárgafoltot kereső algoritmusok szűkíthetik a keresési területet, továbbá a temporális érív megközelítőleg azonos távolságra esik a sárgafolttól (21. ábra). Az ilyen összefüggések kihasználásával várhatóan pontosabb eredményt érnek el a sárgafolt helyét meghatározó módszerek. A temporális érív a sárgafolt mellett a vakfolt detektálásában is szerepet kaphat. Egy rosszul felismert érrendszer a temporális érív meghatározását pontatlanná, szélsőséges esetbe használhatatlanná teheti, így halmozott hibát idézve elő ezeknek az anatómiai képletek felismerésében.



21. ábra Temporális érív (A) és a sárgafolt középpontja (B) megközelítőleg azonos távolságra esik egymástól

Irodalomjegyzék

[1] http://en.wikipedia.org/wiki/Region_of_interest

[2] A. Hoover and M. Goldbaum: Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels, *IEEE Transactions on Medical Imaging* **22/8**, 951-958, 2003.

[3] M. Niemeijer, M.D. Abramoff and B. van Ginneken: Segmentation of the Optic Disc, Macula and Vascular Arch in Fundus Photographs, *IEEE Transactions on Medical Imaging* **26/1**, 116-127, 2007.

[4] R. Harangozo, P. Veres, A. Hajdu: Subsampling strategies to improve learning-based retina vessel segmentation, *IEEE 16th International Conference on Image Processing (ICIP2009)*, 7 – 11 November 2009

[5] András Hajdu, Péter Veres, Attila Tanács, Tünde Pető, Roland Harangozó, Julianna Hüvely, Zsolt Török, Attila Biró, Adrienne Csutak: Model-based subsampling: Applications, *KÉPAF 2009*

[6] Dr. Bodon Ferenc: Adatbányászati algoritmusok, http://www.cs.bme.hu/~bodon/magyar/adatbanyaszat/tanulmany/index.html, 2008.

[7] M. Niemeijer, J. Staal, B. van Ginneken, M. Loog and M.D. Abramoff, „Comparative study of retinal vessel segmentation methods on a new publicly available database”, in *SPIE Medical Imaging*, Editor(s): J. Michael Fitzpatrick, M. Sonka, SPIE, 2004, vol. 5370, pp. 648-656.

[8] M. Niemeijer, B. van Ginneken, J. Staal, Maria S. A. Suttorp-Schulten and M.D. Abramoff: Automatic Detection of Red Lesions in Digital Color Fundus Photographs, *IEEE Transactions on Medical Imaging* **24/5**, 584-592, 2005.

[9] http://www.isi.uu.nl/Research/Databases/

[10] Qiang Du, Maria Emelianenko, and Lili Ju: Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations, *SIAM J Numer. Anal.* **44/1**, 102-119, 2006.

[11] Hoa Nguyen: Centroidal Voronoi Tessellations for Mesh Generation: From Uniform to Anisotropic Adaptive Triangulations. Ph.D. Thesis, Florida State University (FSU), 2008.

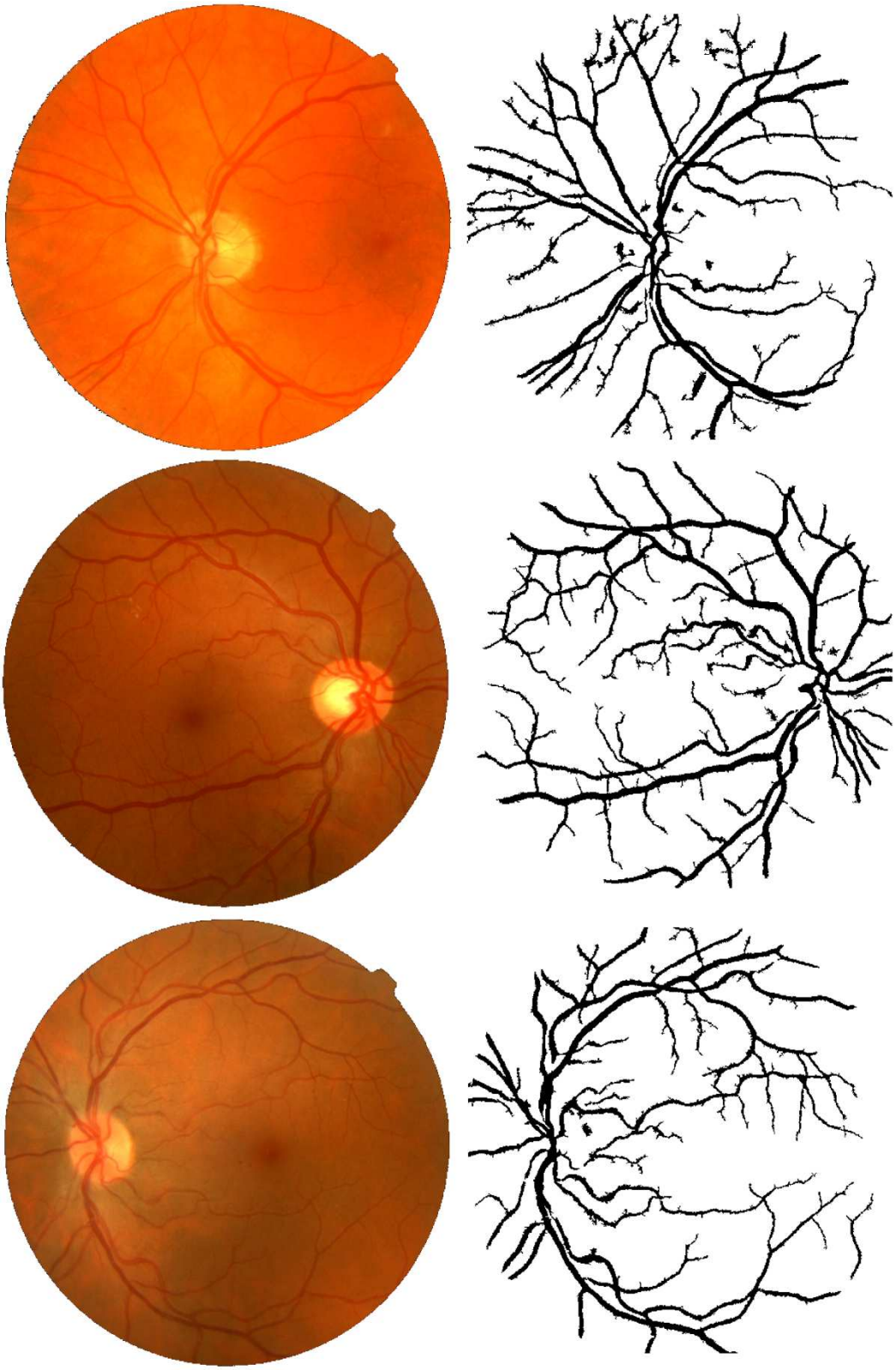
[12] András Hajdu and Ioannis Pitas: Optimal approach for fast object-template matching, *IEEE Trans. on Image Processing* **16/8**, 2048-2057, 2007.

[13] http://java.sun.com/javase/technologies/desktop/media/jai/

[14] http://www.cs.waikato.ac.nz/~ml/weka/

[15] http://www.cs.waikato.ac.nz/~ml/weka/arff.html

Függelék



22. ábra Szemfenék kép (baloldali oszlop), detektált érrendszer (jobboldali oszlop)

Köszönetnyilvánítás

Köszönettel tartozom témavezetőmnek, Dr. Hajdu Andrásnak segítőkészségéért, szakmai hozzáértéséért és tanácsaiért.

Köszönöm külső konzulensemnek, Dr. Török Zsoltnak hogy orvosi szaktudásával elősegítette a diplomamunkán elkészítését.

A dolgozat elkészítését a DRSCREEN: A cukorbetegség szemszövődményeinek szűrésére alkalmas képfeldolgozó rendszer kifejlesztése című NKTH-TECH08-2 (szerződés sz.: OM-00194/2008, OM-00195/2008, OM-00196/2008) projekt támogatta.