UNIVERSITATEA TEHNICĂDIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂȘI CALCULATOARE
SECȚIA CALCULATOARE

Segmentarea regională a imaginilor cu funcție de nivel B-Spline

ÎNDRUMĂTOR ŞTIINŢIFIC:
dr. ing. Szilágyi László

ABSOLVENT
Gábor Bernát

2011

UNIVERSITATEA TEHNICA CLUJ-NAPOCA
FACULTATEA DE AUTOMATICA SI CALCULATOARE
SECTIA CALCULATOARE

VIZAT DECAN — VIZAT SEF CATEDRA
Prof.Dr.Ing. Sergiu NEDEVSCHI

TEMĂ PROIECT DE DIPLOMĂ

Conducătorul temei: — Candidat: **GÁBOR BERNÁT**
conf. dr. SZILÁGYI László — Anul absolvirii: 2011

1. Conţinutul proiectului

a) Tema proiectului de diplomă: ***Segmentarea regională a imaginilor cu funcţie de nivel B-spline***

b) Problemele principale care vor fi tratate în proiect:
- Metodologia segmentării imaginilor
- Tehnici de segmentare prin modele active de contur

c) Desene obligatorii:
- Schemele realizate în cadrul proiectării softului
- Cel puţin patru exemple de imagini segmentate

d) Softuri obligatorii
-Un program care implementează metoda segmentării regionale cu funcţie de nivel de tip B-spline

Bibliografie recomandată:

1. M. Sonka, V. Hlavac, R Boyle – Image Processing: Analysis and Machine Vision, CL-Engineering, 1998
2. T. F. Chan, L. A. Vese, Active contours without edges, IEEE Transactions on Image Processing, vol. 10, no. 2, pp. 266-277, 2001
3. O. Bernard, D. Friboulet, P. Thénevaz, M. Unser, Variational B-spline level-set: a linear filtering approach for fast deformable model evolution, IEEE Transactions on Image Processing, vol. 18, no. 6, pp. 1179-1191, 2009

Termene obligatorii de consultaţii: - săptămânal
Locul practicii: Universitate
Primit la data de: 15. 04. 2010
Termen de predare: 20.06. 2011

Semnătura şefului de catedră — Semnătura îndrumătorului ştiinţific

Semnătura coordonatorului

Semnătura candidatului

Segmentarea regională a imaginilor cu funcție de nivel B-Spline

Extras

În anul 1965 savantul Gordon E. Moore a scris într-o lucrare științifică enunțul care îl cunoaștem azi ca legea lui Moore. Conform acestuia, numărul tranzistorilor ce poate fi integrat fără costuri mari se dublează după un ciclu de aproximativ doi ani. Această rată de evoluție extraordinar de rapidă este probabil depășită doar de nivelul de aplicație a tehnologiei și a numărului utilizatorilor.

În aceste condiții nu ar trebui să surprindă pe nimeni că, odată cu răspândirea sistemelor de formare a imaginilor (ca camerele de fotografie, sonagraful etc.) în cadrul vieții de zi cu zi, este din ce în ce mai importanta procesarea lor într-un timp cât mai scurt și cât mai exact.

Algoritmii care au ca scop principal găsirea obiectelor și marginile acestora pe o imagine digitală sunt de tip segmentare. Numele provine din faptul că, de obicei, la sfârșit, imaginea din start este partiționata în diferite segmente.

În literatura domeniului prelucrării imaginilor găsim nenumărate metode pentru realizarea segmentării. În cadrul acestei lucrări vom prezenta și realiza unul care are ca element central o funcție de nivel interpolată cu ajutorul funcțiilor B-Spline.

La începutul realizării proiectului am parcurs literatura din domeniu pentru a putea încadra algoritmul în cauză în domeniul segmentării imaginilor. În era clasică s-au folosit tehnici ca clasificarea fuzzy a lui Bezdek [1], diviziunea și îmbinarea sau metoda bazinelor hidrografice. De obicei țelul segmentării este să aflăm conturul obiectului căutat. În aceste cazuri putem folosi și algoritmi de căutare a marginilor ca a lui Canny [2], sau transformarea generală Hough [3]. Aceste tehnici de segmentare sunt studiate în detaliu de către Sonka-Hlavac-Boyle [4].

În ultimele decenii sau format un vast număr de modele moderne pe care, din considerație de spațiu, nu le vom detalia aici. În schimb, vom prezenta câteva modele bazate pe variație și ecuații cu derivate parțiale: Mumford-Shah [5] [6] și contururi activi [7].

În cazul primului imaginea este împărțită în regiuni omogene. Pentru asta se definește următorul funcțional pe care îl minimizează:

$$F_{MS}(I,C) = \int_{\Omega} |I - I_0|^2 dx + \mu * \int_{\Omega - C} |\nabla I|^2 dx + \nu \mathcal{H}^{N-1}(C) \qquad (1.)$$

Aceasta este o sarcină destul de greoaie cum putem citi din lucrările lui Chen-Vese [8] și Pock [9]. Mai mult, acest model nu face diferența între regiunile găsite, considerându-le pe toate la fel de importante. În practică, însă, de cele mai multe ori din imaginea prelucrată ne interesează doar o anumită regiune. Al doilea model, cel de contur activ, are capacitatea de a integra aceste informații în algoritmul său.

Din această cauză nu mai realizăm segmentarea totală a imaginii, ci doar una parțiala. Acest lucru se manifestă prin faptul că găsim doar conturul cel mai apropiat față de punctul de inițializare. Primul model de acest tip a fost introdus de Kass-Witkin-Terzopoulos [10] și după denumirea creatorilor săi este cunoscut și ca modelul de șarpe activ. La baza algoritmului stă o curbă a cărei formă este trasată la început de către utilizator.

După trasare, modelul va deforma curba pe baza schimbărilor puternice de intensitate găsite în imagine. În final curba aceasta va reprezenta conturul obiectului căutat. Pentru realizarea acestuia se minimizează un funcțional care este compus din forțe care conduc curba în evoluția sa și impun trăsăturile sale. În figura de mai jos putem vedea un exemplu de segmentare a acestui algoritm:

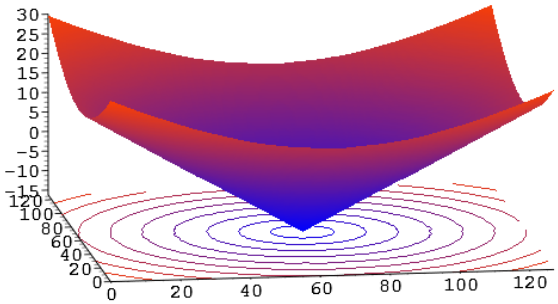
Figură 1. Segmentarea cu un model activ

Observăm că modelul nu este unul flexibil din punct de vedere topologic deoarece, pornind dintr-un singur contur, rezultatul va avea tot un singur contur. Pentru rezolvarea acestui impediment, Osher și Setian au recomandat folosirea modelului de setare a nivelului [11]. Ideea este că rigiditatea topologică a modelelor este dată de tratarea curbei prin forme parametrizate.

Alcătuirea unui model cu parametrii care să permite îmbinarea și împărțirea curbelor este o sarcină foarte grea, dacă nu imposibilă.

Pentru evitarea acestei probleme, modelul de setare a nivelului tratează curba în forma sa implicită. Adică atât conturul inițial cât și cel final le vom trata prin enumerarea totală a pixelilor care îl formează. Mai mult, modelul lucrează cu o funcție de nivel. Pentru fiecare pixel al imaginii prelucrate se determină o valoare care spune exact dacă pixelul face parte din obiectul căutat, din fundal sau dacă este pe conturul căutat. În cea mai simplă formă a sa se folosesc numere negative pentru obiect, pozitive pentru fundal și numărul zero pentru punctele aflate pe contur.

La început, din conturul inițial stabilit de noi, algoritmul inițializează funcția de setare a nivelului, de obicei folosind o funcție de distanță cu semn. După aceea, stabilește un funcțional al cărui minimizare va ghida funcția de setare a nivelului într-o formă în care nivelul de zero va coresponda cu conturul căutat de noi. Minimizarea în literatura clasică se făcea prin folosirea schemelor diferențiale finite, care sunt destul de complicate și greoi de folosit. Pentru mai multe modele active sau o prezentare mai în detaliu citiți capitolul doi.



Figură 2. Funcția de nivel setat ca o funcția de distanță cu semn pornind de la un contur de cerc

Pentru a evita acest inconvenienţă, Olivier Bernard [12] recomanda în 2009 folosirea funcțiilor B-Spline pentru a reprezenta funcția de setare a nivelului. Michael Unser [13] [14] [15] a recomandat pentru manipularea funcțiilor B-Spline folosirea filtrelor digitale (atât de tip IRR cât și de tip FIR). Prin aceste metode, manipularea funcțiilor interpolate de către funcții B-Spline se reduce la niște convoluții care sunt foarte ușor de programat.

Totodată algoritmul poate fi ușor format dintr-o serie de operații de tip "batch" care permite o paralelizare ușoară a acestuia. Așadar, datorită folosirii funcțiilor B-Spline, ne alegem cu un algoritm care poate exploata la maxim multiplele nuclee de procesare prezente în

computerele de azi ca în CPU sau GPU. În capitolul trei găsim descrierea detaliată a modelului recomandat de Olivier Bernard. Pe pagina următoare putem citi o schiță a algoritmului:

Tabelă 1.Schița algoritmului implementat în lucrare

Inițializare	• Setăm parametrii de funcționare a algoritmului (numărul maxim de iterații permis, scalare, precizie dorită, mărimea pasului în cazul metodei de minimizare) • Citirea imaginii (dacă este de tip color convertim în imagine de intensitate) • Stabilim conturul (curba) inițială (de exemplu prin desenare unui cerc) în masca • Extindem imaginea și masca ca dimensiunile lor să fie un număr la puterea a doua • Inițializăm funcția de setare a nivelului folosind masca conform unei funcții de distanță cu semn • Conform valorii de scalare stabilită alegem filtrul folosite pe coeficienții de B-Spline • Stabilirea coeficienților de B-Spline pentru funcția de setare a nivelului • Calculăm μ_{in} și μ_{out} ca $\mu_{in} = \frac{\int_\Omega f(x)H_\epsilon(\phi(x))dx_1 \dots dx_d}{\int_\Omega H_\epsilon(\phi(x))dx_1 \dots dx_d}$ $\mu_{out} = \frac{\int_\Omega f(x)(1-H_\epsilon(\phi(x))dx_1 \dots dx_d}{\int_\Omega (1-H_\epsilon(\phi(x)))dx_1 \dots dx_d}$ • $J\epsilon(\phi, \mu_{in}, \mu_{out}) =$ $\int_\Omega (f(x) - \mu_{in})^2 H_\epsilon(\phi(x))dx_1 \dots dx_d +$ $\int_\Omega (f(x) - \mu_{out})^2 \left(1 - H_\epsilon(\phi(x))\right) dx_1 \dots dx_d +$ $\nu \int_\Omega \| \nabla\phi(x) \| \delta(\phi(x))dx_1 \dots dx_d$ • numărător = 0, numărătorDeStabilitate=0
Funcționare	**Până când** (numărător < NumărMaximDeIterație și numărătorDeStabilitate< 6) • Minimizare ○ $\omega_\epsilon(x) = ((f(x) - \mu_{in})^2 - (f(x) - \mu_{out})^2)\delta\epsilon(\phi(x))$ ○ Normalizăm ω_ϵ ○ $< \nabla_c J > [k] = (\omega_\epsilon * b_h^n)_{\downarrow h}[k]$ ○ Minimizăm cu metoda gradient de ordinul întâi de cinci ori ▪ $c^{(i+1)} = c^{(i)} - \lambda\nabla_c J(c^{(i)})$ ▪ $c^{(i+1)} = \frac{c^{(i+1)}}{\|c^{(i+1)}\|_\infty}$ ▪ Calculăm $\phi(x)$, μ_{in}, μ_{in}, $J\epsilon(\phi, \mu_{in}, \mu_{out})$ ▪ **Dacă** $J\epsilon(\phi, \mu_{in}, \mu_{out})$ scade • Salvează valorile calculate și minimizează în continuare ▪ **Altfel** • Ieșire din minimizare ▪ **Sfârșit Dacă** • Calculăm masca (funcția nivelul de setare>= 0), numărăm pixeli diferiți în masca • **Dacă** schimbare < precizia dorită • numărătorDeStabilitate= numărătorDeStabilitate+1 • **Altfel** • numărătorDeStabilitate = 0 • **Sfârșit Dacă** • numărător= numărător+1 **Sfârșit Până Când**

Ca funcțional pentru energia funcției de setare a nivelului am ales una regională introduse în literatură de către Chen-Vese [8]. Problema modelelor de contur activ bazate pe margini (ca

modelul șarpelui activ) este că sunt sensibile la zgomote în imagini și la contraste mici a acestora, un lucru des întâlnit în procesarea imaginilor medicale. În schimb modele bazate pe funcționale regionale ghidează evoluția curbei inițializate nu prin apariția marginilor, ci prin elemente regionale.

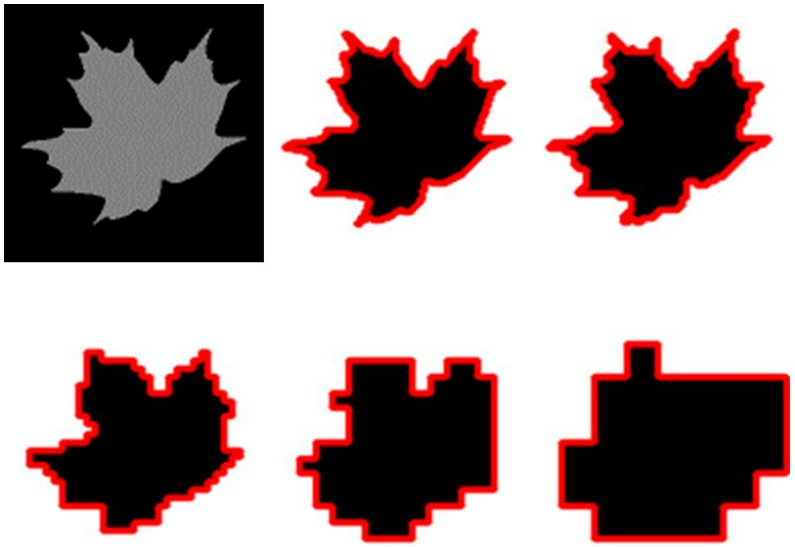
În capitolul patru sunt descrise cerințele sistemului care trebuie realizat în cadrul lucrării. Acesta trebuie să realizeze segmentarea imaginilor bidimensionale. Implementarea algoritmului trebuie să ofere o interfață cât mai ușor de utilizat, de menținut, de extins și un timp cât mai rapid de funcționare. Trebuie să opereze pe o gamă largă de formate de imagini ca png, jpeg sau bmp. Parametrii algoritmului trebuie să fie ușor de modificat. De asemenea este necesara și crearea unui interfețe grafice de utilizare (GUI) care să vizualizeze algoritmul în timpul funcționării sale. S-au realizat diagramele use case și de component ca să observăm funcțiile de implementat și structura interfeței grafice.

Capitolul cinci explică în detaliu problemele soluționate în cursul implementării algoritmului și a GUI-ului. La început am ales funcționalul folosit și algoritmul Expectation-Maximization (EM) pentru minimizarea mai multor parametrii a acestuia. Pentru ca algoritmul să aibă un timp de execuție cât mai scurt s-a folosit limbajul C++. Totodată pentru satisfacerea cerințelor de utilizare, menținere si extindere ușoară, am folosit paradigma de programare orientată pe obiecte. Astfel, datele folosite de algoritm și metodele de procesare a acestuia au fost puse într-o singură clasă denumită *BSplineLevelSet*. Mai mult, acesta a fost plasat într-un spațiu de nume (*LevelSetSegmentation*) care la rândul lui este integrat într-un DLL.

Librăria OpenCV a oferit metodele folosite pentru citirea și salvarea imaginilor. De asemenea, structura de reprezentare a imaginilor din cadrul librăriei (*cv::Mat*) s-a folosit pentru manipularea ușoară a pixelilor din imagini. Pentru crearea interfeței grafice am folosit pachetul Qt. Am realizat și vizualizarea funcției de setare a nivelului în trei dimensiuni. În crearea acestuia a ajutat pachetul de cod sursă denumit QwtPlot3D [16].

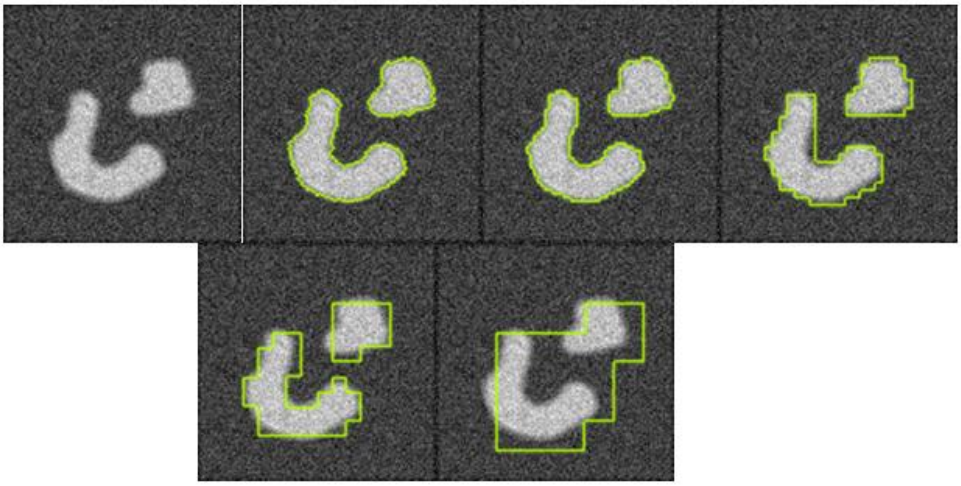
Tot aici poate fi observată diagrama de clasă a implementării și, pentru o mai bună înțelegere a algoritmului, am creat și o diagramă de flux a datelor (Dataflow diagram). Capitolul șase explică cum să utilizăm atât algoritmul realizat pentru integrarea în alte sisteme, cât și cum să folosim interfața grafică pentru vizualizarea și segmentarea imaginilor selectate de către utilizator.

În capitolul șapte observăm rezultatele obținute, măsurăm performanța realizată și vorbim despre eventualele limitări a algoritmului care duc în cazul anumitor imagini la segmentare suboptimală. Pentru început să vedem algoritmul la lucru pe o imagine generată:



Figură 3. Segmentarea frunzei

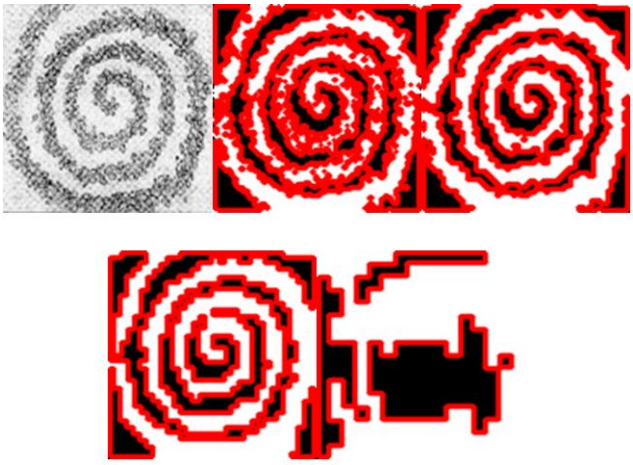
Pe această imagine puteam observa că mărirea parametrului de scalare a algoritmului determină rigiditatea conturului. O valoare de scalare din ce în ce mai mare ține cont tot mai puțin de detalii și se ocupă tot mai mult de localizarea frunzei. Asta se observă și prin timp de execuție mai scurt necesar pentru finalizarea algoritmului.



Figură 4. Segmentarea de două obiecte

Dacă avem mai multe obiecte de segmentat la un anumit nivel de scalare, algoritmul s-ar putea să nu mai poată sa facă diferența intre obiectele prea mici pentru scală și ca urmare le îmbină cum se observă și in imaginea de mai sus. În cazul acestei imagini, am ales să desenez

conturul rezultat direct pe imaginea prelucrată. Atenție! Mărirea valorii de scalar peste un anumit nivel la imagine poate duce și la obținerea rezultatelor fără sens ca în cazul spiralei următoare:



Figură 5. Segmentarea spiralei

Un exemplu mai aproape de lumea în care trăim e segmentarea sculpturilor care au un fundal deschis. Exemplul următor demonstrează un astfel de caz.

Figură 6. Segmentarea sculpturilor fugătoare

Algoritmul are complexitatea influențată în principal de dimensiunile imaginii (înălțime si lățime) și apoi de numărul de iterații necesare pentru minimizarea funcționalului de energie folosit. Performanța algoritmului este una foarte bună în cazul imaginilor de dimensiuni mici si medii (pana la 500 x 500 pixeli): Pentru imagini mai mari e necesară implementarea algoritmului pe un sistem cu mai multe unități de procesare, ca GPU-ul.

Tabela următoare prezintă câteva rezultate ca puncte de orientare. Testele au fost făcute pe un procesor Intel P8700 de 2,53 GHz si memorie de 4GB. Olivier Bernard, în implementarea

sa din 2009 pe un procesor de Intel 1.4 GHZ si 1GB de memorie, a obținut pentru frunza segmentată un timp de aproximativ două secunde. Timpii au fost măsurați în secunde și *N.I.* reprezintă numărul de iterații necesare pentru finalizarea segmentării.

Tabelă 2. Performanța algoritmului

Imagine (dimensiuni)	Mărimea pasului în funcția de B-Spline discretă							
	1		*2*		*4*		*8*	
	Timp(s)	*N.I*	*Timp(s)*	*N.I.*	*Timp(s)*	*N.I.*	*Timp(s)*	*N.I.*
Frunză(128×128)	0.240000	7	0.225003	7	0.240847	8	0.210587	7
Regională (128×128)	0.360040	12	0.265840	10	0.271019	10	0.250874	7
Spirală (128×128)	0.647694	24	0.520128	16	0.304954	11	0.201891	7
Celule (128×128)	0.436916	15	0.273539	10	0.252977	9	0.297395	11
Creier MRI(128×128)	0.247314	8	0.200423	7	0.167227	6	0.180082	6
Echo-cardiograf (387×387)	4.001256	15	3.128002	11	2.209847	8	1.942836	7

Din tabelă putem să deducem că cerințele proiectului au fost realizate cu succes. În folosirea algoritmului trebui să știm și când acesta nu ne va oferi rezultatele dorite. Trebuie să reținem că facem o segmentare regională în care delimităm pe imagine părțile întunecate și cele luminoase. Dacă într-o regiune a unei imagini se găsesc mai multe puncte luminoase decât întunecate, sau invers, întreaga regiune va fi clasificata ca aparținând părții dominante. Acesta este și cazul sculpturilor unde soarele strălucește pe anumite suprafețe ca în situația de jos:

Figură 7. Segmentarea lebadelelor

Puncte de dezvoltare viitoare a proiectului pot fi implementarea paralelizării algoritmului folosind tehnologia OpenMP pentru CPU-uri și Nvidia CUDA ori OpenCL pentru unitățile de procesare grafica. O altă lucrare are fi extinderea algoritmului pe imagini tridimensionale sau folosirea sa pentru segmentare în cazul fluxurilor video. Tot un teritoriu de cercetat este găsirea și realizarea unei optimizări superioare față de cea actuala: optimizarea prin metoda gradient de ordinul întâi cu lungimea pasului modificata prin feedback.

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM

MŰSZAKI és HUMÁNTUDOMÁNYOK KAR
VILLAMOSMÉRNÖKI TANSZÉK
SZÁMÍTÁSTECHNIKA SZAK

Regionális kép szegmentáció B-Spline szintfüggvénnyel

Diplomadolgozat

TÉMAVEZETŐ,
dr. ing. Szilágyi László

VÉGZŐS HALLGATÓ,
Gábor Bernát

2011

KIVONAT

1965-ben Moore egy tudományos írásában megfogalmazta a ma Moore törvényeként ismert kijelentést, mely szerint egy áramkörbe beintegrált alkotó elemek száma duplázódik évente. Az elektronika e látványos fejlődési ritmusát talán csak a felhasználási módszereknek és felhasználók számának sikerül túlszárnyalni.

Nem meglepetés tehát, hogy a digitális kamerák és egyéb képalkotó rendszerek rohamos elterjedésével megnőtt az az igény is, hogy a begyűjtött képekről könnyen és gyorsan nyerjünk ki különböző ismereteket.

A képfeldolgozás azon területét mely formák, alakzatok megtalálásával és kinyerésével foglalkozik, szegmentációnak hívjuk. A szakirodalomban számos olyan módszer van leírva, mellyel egy képből ki lehet nyerni egy tárgy körvonalát. Jelen dolgozatba ezek közül egy B-Spline függvénnyel interpolált szinthalmaz megközelítést mutatok be.

Ennek egyik fő tulajdonsága, hogy a szükséges transzformációkat és deriválásokat szűrők segítségével valósítjuk meg. Emiatt lehetséges gyors, ún. „batch" típusú utasításokkal való megvalósítás létrehozása. Implementálásra a C++ nyelvezetet használtam az OpenCV könyvtár segítségével.

Emellett ugyanakkor elkésztetetem a Qt platform független csomaggal egy grafikus felhasználói felületet. Az algoritmus lehetséges alkalmazási területei közül megemlíteném egy kéz követését, egy képen található szobor szegmentációja, avagy az orvosi képfeldolgozásban alkalmazható úgy mikroszkóp felvételen sejtek hely azonosítására, mint MRI és echókardiográf képeken szegmentációra.

Kulcsszavak: *szegmentáció, szinthalmaz függvény, B-Spline, energia minimalizálása, C++*

ABSTRACT

In 1965 Gordon E. Moore inside one of his scientific papers jots down the expression nowadays known as Moore's law. This states that: "The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years." This sensational evolution rate may be only suppressed by its application methods and the number of people using it on a daily basis.

Under these conditions, it isn't a surprise that alongside the spread of the digital cameras and other image forming systems (like sonographers) the need to process and extract new information from the acquired images increases with every moment passed.

The area of the image processing whose main goal is to find and obtain objects, contours in digital images, by partitioning it into different segments, is segmentation. In the literature there are many known algorithms to solve this problem. Here we will present one that builds on a level set function interpolated by B-Spline functions.

This has the unique and from an implementation point of view important trait, that the required transformations and derivations are achieved using digital filters. Due to this it is possible to use batch operations that can significantly speed up the algorithms running time. For coding down the algorithm into source code snippets I've used the C++ programming language and the OpenCV library.

Beside this I've also made a graphical user interface for demonstration purposes by using the Qt cross-platform package. The areas of where one could use the algorithm presented here are the tasks like of following a hand in a video, segmentation of dark statues from bright backgrounds, or in the field of medical image processing at locating the cells in a microscope image, and for the MRI and echocardiographic image segmentation.

Keywords: *segmentation, level-set function, B-Spline, energy minimization, C++*

TARTALOMJEGYZÉK

KIVONAT **12**

ABSTRACT **13**

TARTALOMJEGYZÉK **14**

ÁBRAJEGYZÉK **16**

TÁBLÁZAT JEGYZÉK **18**

1. ÁLTALÁNOS BEVEZETÉS **19**

2. SZAKIRODALOM ÁTTEKINTÉS **20**

2.1. KLASSZIKUS MODELLEK 20

2.2. VARIÁCIÓS KÉPSZEGMENTÁLÁSI MÓDSZEREK 21

2.3. A SZINTHALMAZ MÓDSZER 23

2.4. RÉGIÓ ALAPÚ AKTÍV KONTÚROK 24

2.5. FORMA ALAPÚ AKTÍV KONTÚROK 25

3. ELMÉLETI ALAPOK **26**

3.1. SZINTFÜGGVÉNY 26

3.2. ENERGIA KRITÉRIUM 27

3.3. SZINTFÜGGVÉNY INTERPOLÁCIÓ 27

3.3.1. Interpoláció *27*

3.3.2. Szintézisfüggvény óhajtott tulajdonságai *29*

3.3.3. B-Spline *30*

3.3.4. B-Spline szintfüggvény modell *31*

3.4. ENERGIA KRITÉRIUM MINIMALIZÁLÁSA 32

3.5. DISZKRÉT FORMA 33

3.5.1. Diszkrét jellemző függvény *33*

3.5.2. Gradiens számítás *34*

3.6. IMPLEMENTÁCIÓS PROBLÉMÁK 34

3.6.1. Korlátos szinthalmaz függvény *34*

3.6.2. Első fokú gradiens módszer *36*

3.7. FUTÁS IDEJŰ VIZUALIZÁCIÓ 36

4. A RENDSZER SPECIFIKÁCIÓJA ÉS ARCHITEKTURÁJA **39**

4.1. SPECIFIKÁCIÓ 39

4.2. ARCHITEKTÚRA 40

4.2.1. Use Case Diagram *40*

4.2.2. Grafikus felület komponens diagram 40

5. MEGVALÓSÍTÁS 42

5.1. HASZNÁLT FÜGGVÉNYEK ÉS PARAMÉTEREK 42

5.2. ALGORITMUS VÁZLAT 45

5.3. HASZNÁLT FEJLESZTÉSI ESZKÖZÖK ÉS METÓDUSOK 46

5.4. DIAGRAMOK 46

5.4.1. Osztálydiagram 46

5.4.2. Adatfolyam 48

5.5. GRAFIKUS FELÜLET MEGVALÓSÍTÁSA 50

6. A RENDSZER FELHASZNÁLÁSA 52

6.1. ALGORITMUS 52

6.2. GRAFIKUS FELÜLET 52

7. SZEGMENTÁCIÓS EREDMÉNYEK 55

7.1. SZIMULÁCIÓS KÉPEK 55

7.2. VALÓ VILÁGI KÉPEK 57

7.3. ORVOSI KÉPEK 59

7.4. FUTÁSIDŐ ÉS ITERÁCIÓ SZÁM 62

7.5. AZ ALGORITMUS HATÁRAI 62

8. KÖVETKEZTETÉS 64

9. IRODALOMJEGYZÉK 66

10. FÜGGELÉK 69

ÁBRAJEGYZÉK

Figură 1. Segmentarea cu un model activ 4
Figură 2. Funcţia de nivel setat ca o funcţia de distanţă cu semn pornind de la un contur de cerc .5
Figură 3. Segmentarea frunzei 8
Figură 4. Segmentarea de două obiecte 8
Figură 5. Segmentarea spiralei 9
Figură 6. Segmentarea sculpturilor fugătoare 9
Figură 7. Segmentarea lebadelelor 10
Ábra 8. A Mumford-Shah funkcionál minimalizálásával kapott szegmentált kép 21
Ábra 9. Szegmentálás aktív kontúr modellel 22
Ábra 10. A kezdeti kontúr, kép, és szinthalmaz függvény 36
Ábra 11. A kontúr, szegmentáció és szinthalmaz függvény 1 iteráció után 37
Ábra 12. A kontúr, szegmentáció és szinthalmaz függvény 4 iteráció után 37
Ábra 13. A kontúr, szegmentáció és szinthalmaz függvény 7 iteráció után 37
Ábra 14. A kontúr, szegmentáció és szinthalmaz függvény 10 iteráció után 37
Ábra 15. A kontúr, szegmentáció és szinthalmaz függvény 14 iteráció után 38
Ábra 16. A szinthalmaz függvény a szegmentáció végén (14 iteráció) 38
Ábra 17. Use-Case Diagram 40
Ábra 18. A grafikus felület komponens diagramja (Fül 1) 41
Ábra 19. A grafikus felület komponens diagramja (Fül 2) 41
Ábra 20. Előjeles távolság függvény egy kör esetében 44
Ábra 21. A BSplineLevelSet osztálydiagramja 48
Ábra 22. Adatfolyam diagram 49
Ábra 23. A Qt Designer-el tervezett grafikus felület 50
Ábra 24. A grafikus felület 53
Ábra 25. A grafikus felületműködés közben 54
Ábra 26. Levél szegmentáció 55
Ábra 27. Változó intenzitás szegmentáció 56
Ábra 28. Két objektum szegmentáció 56
Ábra 29. Spirál szegmentáció 57
Ábra 30. Sas szegmentálása 57
Ábra 31. Repülő szegmentálása 58

Ábra 32. Futó szobrok 58
Ábra 33. Ölelkező körszobor 58
Ábra 34. Fluoreszkáló sejt szegmentáció 59
Ábra 35. Agy MRI szegmentáció 59
Ábra 36. Echókardiográf felvétel szegmentáció 60
Ábra 37. Kezek detektálása 61
Ábra 38. Fog CT hibás szegmentálása 63
Ábra 39. Repülő hattyúk pontatlan szegmentációja 63

TÁBLÁZAT JEGYZÉK

Tabelă 1.Schița algoritmului implementat în lucrare 6
Tabelă 2. Performanța algoritmului 10
Táblázat 3. Az algoritmus vázlata 45
Táblázat 4. A szegmentálás teljesítménye 62

1. ÁLTALÁNOS BEVEZETÉS

Századunk gyors elektronikai fejlődésével számos iparágban, szakterületen egyre gyakrabban bukkan fel a probléma, hogy egy képi anyagon levő tárgyakat, formákat, alakzatokat azonosítani kell. Máskor ezek változását kell nyomon követnünk, hogy majd a gyűjtött adatok szerint valamilyen beavatkozást végezzünk el.

Ez lehet egy konkrét elektronikai gép vezérlése egy ipari gép esetében, avagy egy emberi döntés az orvos részéről, ha echókardiográf képekről beszélünk. Azt a folyamatot, amelynek során, egy képanyagon megállapítjuk, annak homogén régióit (szegmenseit) szegmentációnak nevezzük.

A kép szegmentáció fő célja objektumok és határok megkeresése és azonosítása. A szegmentáció eredménye ugyanakkor felírható úgy is, mint egy kontúr halmaz. Ezáltal a szegmentáció közel áll az él detektáló algoritmusokhoz. Ugyanakkor annál sokkal többet jelent, hiszen míg az él detektáló algoritmus csak éleket határozza meg egy képen, a szegmentáció minden képpontot egyértelműen valamelyik képalkotó szegmenshez rendeli.

E dolgozat célja, hogy egy olyan szegmentációs algoritmust valósítson meg mellyel könnyen alkotó szegmenseire bontsunk képeket, amelyek akár zajosak is lehetnek. A kép szegmentálás egyik fő alkalmazási célterülete az orvosi képfeldolgozás.

E szakterületre jellemző, hogy az alkotott képek főleg intenzitás jellegűek. Ezek esetében fontos, hogy e képeken könnyen azonosítani tudjuk az anatómiai szerveket és elkülönítsük azokat a háttértől. Ez megegyezik a világos és sötét képrégiók szegmentációjával melyekbe gyakori a használt technológia által eredményezett zajok (például a szonográfiás képekre jellemző “speckle” zajok).

Továbbá fontos, hogy a szegmentációs algoritmusunk lehetőleg minél közelebb fusson a valós időhöz, hiszen az orvosnak gyakran azonnal kell az eredmény (lásd operáció) és nem hosszas várakozás után.

Emellett további felhasználási terület lehet például futószalagon történő áru minőség ellenőrzése (penész megjelenése), szobrok szegmentációja és egyéb területek ahol sötét régiókat kell elválasztani világosabb párjaitól.

2. SZAKIRODALOM ÁTTEKINTÉS

Ebben a fejezetben egy gyors képet kaphatunk a képfeldolgozásba használt szegmentációs metódusokról, különös képen ennek a variációs és parciális differenciálegyenletekre épülő modell típusairól. Mivel a fejezet célja jóval meghaladja e modellek teljes áttekintését itt csupán azok általános alakját és működési elvét fogjuk bemutatni. Az érdekelt olvasóra bízzuk, hogy ezek implementációs kihívásaira, részleteire utána kutasson. Ennek ellenére az itt leírt áttekintés segíteni fog, hogy a későbbiekben itt felhasznált és implementált algoritmust el tudjuk helyezni a szegmentálási tudományterületen.

2.1. Klasszikus modellek

A klasszikus szakirodalomban számos módszer létezik a képek homogén részeire való felbontására. Ilyen például egy vágás alkalmazása intenzitás képeknél, Bezdek által leírt fuzzy osztályozás [1], felosztás és egyesítés módszere, avagy a víztárolók módszere. Sajátos esetekben, ha képünk jó minőségű és csupán a szegmentáció eredménye (a kontúr) érdekel, alkalmazhatjuk az élek alapján történő szegmentációt.

Ilyenkor tanulmányozhatjuk a Canny éldetektáló eljárást [2], avagy az általános Hough transzformációt [3] is, hogy csak egy párat említsek. Az érdekelt olvasó ezek alapos áttanulmányozását megfigyelheti Sonka-Hlavac-Boyle könyvében [4].

Képeink habár számunkra diszkrét formában állnak rendelkezésre, egy folytonos világból származnak. Hogy ezeket diszkrét formában tárolni tudjuk, mintavételezésre és kvantifikációra szorulunk. A klasszikus módszereknek egyik fő gondjuk, hogy egy diszkrét modellen alapulnak és így érzékenyek a paramétereik jó megválasztására. Ez gondot okozhat, amikor komplex változó alakzatokat keresünk, és zajos képeken akarjuk őket alkalmazni.

Nemrég új kép szegmentációs modelleket írtak le melyek a variációs megközelítésen alapulnak. Ezek folytonos térben vannak definiálva és így matematikailag jól kitanulmányozattak, mivel a folytonos analízis sokkal jobban ki van dolgozva, mint diszkrét párja. A két legismertebb modell: a Mumford-Shah [5] [6] és az aktív kontúr [7].

2.2. Variációs képszegmentálási módszerek

A Mumford-Shah modell esetében a cél, hogy a képet egymástól független homogén régiókra bontja. Ehhez definiál először is egy funkcionálist:

$$F_{MS}(I,C) = \int_{\Omega} |I - I_0|^2 dx + \mu * \int_{\Omega - C} |\nabla I|^2 dx + \nu \mathcal{H}^{N-1}(C) \qquad (2.)$$

ahol I_0 a kezdeti kép, I az eredmény kép, C egy halmaz mely az I éleit foglalja magába és ennek mérete az N-1 méretű Hausdorff mérték $\mathcal{H}^{N-1}(C)$ adja meg. ν és μ pozitív paraméterek.

Ennek a minimalizálása eredményez majd egy szakaszonként sima megközelítését a képnek. Azaz, ahogy a lenti képen is láthatjuk kapunk egy képet mely homogén régiókból áll és szakaszonként sima. A kép T. Pock, D. Cremers, H. Bischof, és A. Chambolle cikkéből származik [9].

Ábra 8. A Mumford-Shah funkcionál minimalizálásával kapott szegmentált kép

Az algoritmus nem tud textura mintázatokat meghatározni. A szegmenseket csak akkor azonosítja, ha azok egyetlen homogén intenzitás régióból áll. Ugyanakkor a funkcionális minimalizálása egy igencsak nehéz feladat, ahogy azt Chen-Vese [8] és Pock [9] munkáiban kifejtette.

A Mumford-Shah modell nem tesz különbséget a talált részek között. Mindegyiket ugyanolyan fontosnak tartja. A gyakorlatban azonban gyakori (különösen az orvosi képfeldolgozásban), hogy az alkalmazástól függően a kép csak bizonyos régiók érdekelik a felhasználót. E fontossági, prioritási információt az aktív kontúr modell már képes magában foglalni.

Itt már nem teljes szegmentációt valósítunk meg, hanem csak adott objektumot próbálunk azonosítani a képen. Ez abban nyilvánul meg, hogy csupán a kiindulási ponthoz legközelebbi kontúrt detektálja. Kass-Witkin-Terzopoulos [10] vezette be a szakirodalomba és még aktív kígyó modellként is ismert. Az algoritmus lényege, hogy erős intenzitásváltozásokat határoz meg azáltal, hogy egy C görbét deformál a kezdeti alakjából a keresett objektum élei felé.

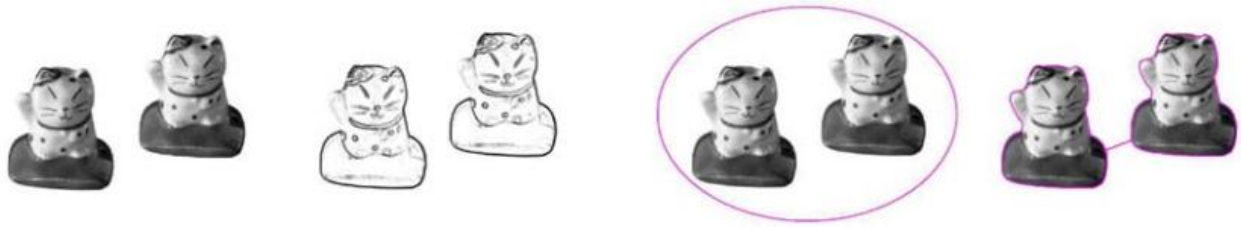
Matematikailag ez alábbi energia funkcionális minimalizálását jelenti:

$$F_{KWT}(C) = \alpha \int_0^1 \left|\frac{\partial C(p)}{\partial p}\right|^2 dp + \beta \int_0^1 \left|\frac{\partial^2 C}{\partial p^2}\right|^2 dp + \alpha \int_0^1 f^2\big(I_0(C)\big)dp \qquad (3.)$$

Az első két tag jelöli a fizikai alapú simasági megkötést a kontúrra, míg az utolsó a külső energia, mely a kezdeti kontúrt tolja a keresett régió határai felé felhasználva egy él detektáló függvényt, mint például:

$$f(I_0) = \frac{1}{1 + \gamma|\nabla(I_0 * G_\sigma)|^2} \qquad (4.)$$

ahol G_σ a σ értékű szórással rendelkező Gauss függvény és $I_0 * G_\sigma$ az eredeti kép egy simított alakja egy γ pozitív konstanssal.



Ábra 9. Szegmentálás aktív kontúr modellel

A fenti képen balról számolva, az első az eredeti kép, a második az él detektáló függvény, a harmadik a kezdeti kontúr és a negyedik az algoritmus végeredményét mutatja be. A képek mind Brensson dolgozatából vannak átvéve [17]. Megfigyelhetjük, hogy nem enged meg topológiai változásokat, hiszen egy kiinduló kontúr esetén az eredményben is egy kontúr lesz. Egy másik hátránya, hogy az eredmény függ a használt paraméterektől, még ha ugyanabból a kezdeti kontúrból is indultunk ki. E paraméteres függőséget Lagrange féle megfogalmazásnak nevezzük.

Az aktív kígyó modell paraméter függőségének megoldására Casselles-Kimmel-Sapiro [18] és Kichenassamy-Kumar-Olver-Tannenbaum-Yezzi [19] egy újabb energia funkcionálist javasolt, melyet geodéziai/geometriai aktív kontúrnak neveztek:

$$E_{GAC}(C) = \int_0^1 f\left(\left|\nabla I_0(C(p))\right|\right)|C_p|dp = \int_0^{L(C)} f\left(\left|\nabla I_0(C(s))\right|\right)ds \qquad (5.)$$

itt dsaz Eukleidészi hosszelem,$L(C)$a keresett görbe hossza és f a (4.)-as egyenletnél felírt él detektáló függvény. Így az energia már nem függ attól, hogy parametrikusan a görbét miképp írjuk fel.

2.3. A Szinthalmaz Módszer

Ez a keresett görbét implicit (Euler) megfogalmazással kezeli, azaz a görbe minden pontját egyértelműen megnevezi, azt nem valamilyen paraméterek formájában téríti vissza. Megalkotói Osher és Sethian [20]. Általános alakja egy görbének $C(p,t): [0,1] \times [0,T) \rightarrow \mathbb{R}^2$, ahol p leírja a görbe geometriáját és t a fejlődő görbecsaládot paraméterezi. E fejlődést az alábbi parciális differenciál egyenletrendszer add meg:

$$\begin{cases} \dfrac{\partial C}{\partial t} = V_{\|}\mathcal{T} + V_{\perp}\mathcal{N} \\ C(t=0) = C_0 \end{cases} \qquad (6.)$$

ahol $\mathcal{T}$ és $\mathcal{N}$ az egység érintőleges és külső normálja C görbének. Ennek megfelelően $V_{\|}$ és $V_{\perp}$ az érintőleges és normál sebességek. A görbe mértanát $V_{\|}$ nem befolyásolja ezért elhagyható, ha csak a görbe mértani alakja érdekel. A módszer lényege, hogy a görbe paraméteres alakját teljesen elhanyagoljuk, hogy megkapjuk annak implicit alakját a szinthalmaz függvény formájában:

$$\begin{cases} \phi(x) > 0, & \forall x \in \Omega_{in} \\ \phi(x) < 0, & \forall x \in \Omega_{out} \\ \phi(x) = 0, & \forall x \in \Gamma \end{cases} \qquad (7.)$$

Ezt felhasználva az (6.) – ös egyenlet átírható a következő alakra:

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = V_{\perp}|\nabla \phi| \\ \phi(t=0) = d(\Gamma_0) = \phi_0 \end{cases} \qquad (8.)$$

A mozgó határfelületet Γ – val jelöltűk, d egy függvény mely megadja a szinthalmaz függvény kezdeti alakját az inicializált kezdeti kontúr alapján. A szinthalmaz fejlődése egy fix koordinátarendszerben van számolva mivel ez egy paraméter szabad megfogalmazás. Ugyanakkor, az algoritmus topológiailag flexibilis, a kezdeti kontúrok szétvallhatnak, összeolvadhatnak, mivel ez nem változtat a szinthalmaz függvény topológiáján.

A hagyományos irodalomban a szinthalmaz függvény mellé rendelt energiafüggvény minimalizálását egy parciális differenciál egyenletre vezetik vissza, amelyeteket véges-differenciál formulákkal oldanak meg [20]. Ezek kezelése nehézkés ezért. ennek elkerülésére érdekében 2009-ben Bernard [12] ajánlott egy B-Spline alapú megközelítést.

Itt e probléma a használt keret implicit tulajdonságai miatt már nem merül fel. Ez lesz továbbá dolgozatunk fő témája és a következő fejezetben részletesen kifejtük majd a módszert. Mielőtt viszont áttérnénk erre, az áttekintés teljesége érdekében még megemlítenék pár további aktív kontúr modellt is.

2.4. Régió alapú aktív kontúrok

Habár a geometriai aktív kontúr modell párosítva a szinthalmaz modellel hatékony numerikus sémáival és flexibilis topológiájával sok képszegmentáló problémát megoldott, ezek még mindig nagyon érzékenyek voltak a képeken megjelenő zajokra és azok esetleges kis kontrasztjára. Hogy e problémát kikerüljék megjelentek olyan aktív kontúr modellek, melyek az energiafüggvénybe olyan elemeket helyeztek el melyek a kontúr fejlődését nem az él detektáló függvénnyel, hanem régió elemekkel irányítják.

Zhu-Yuille [21] egy variációs és statisztikai kép szegmentációs modellt ajánlott, amely az aktív kígyó modellt régiónövesztő elemmel egészíti ki. Azonban mivel a Lagrange féle megfogalmazásra építenek, a kezdeti rendszer topológiája nem változhat. Ennek ellenére ők vezeték be az aktív kígyó modellbe a régió alapú kritériumot. Az általuk használt funkcionális formája:

$$F_{ZY}(C,\{\alpha_i\}) = \sum_{i=1}^{N_R}\{\frac{\mu}{2}\int_{\partial\Omega_i} ds - \int_{\Omega_i} \log P(I_0(x)|\alpha_i)dx\} \qquad (9.)$$

ahol $C = \bigcup_{i=1}^{N_R} \partial\Omega_i$ a szegmentálási határok és P a Gaussi valószínűségi sűrűség eloszlása.

Chan-Vese ajánlotta, hogy az aktív kontúr modell tulajdonságait építsük bele a Mumford-Shah funkcionálisba. Így az (2.) - es egyenletet az ő értelmezésükben felírhatjuk, mint:

$$F_{MS}^{CV}(I,C) = \int_{\Omega} |I - I_0|^2 dx + \mu \int_{\Omega - C} |\nabla I|^2 dx + \nu \int_C ds \qquad (10.)$$

Ezt követően Chan-Vese megalkotta az élek nélküli aktív kontúr modellt [8] alapozva erre az egyszerűsített Mumford-Shah funkcionálisra, amelyben a gradiens alapú elemet is

elhagyták. Ez eredményezi részenként konstans esetét a Mumford-Shah modellnek, melyet a szakirodalomban minimális partíció problémának is neveznek, hiszen az optimális megoldás egy kép melyen a régiók nagyjából azonos intenzitással rendelkeznek. Ez az intenzitás érték ugyanakkor megegyezik a régió átlag intenzitásával.

Habár Chen-Vese megoldotta a teljes partícionálás problémáját, mi itt csak két partícióra tárgyaljuk: egy előtér (objektum) és egy háttér esetében. Ugyanakkor használjuk a szinthalmaz modellt az egyszerűbb számolás és a topológiailag flexibilis eredmény érdekében. E alakra a későbbiekben még részletesen visszatérek, ezért tovább nem részletezem.

$$F_{ACWE}(\phi, c_1, c_2) = \int_\Omega \left|H\epsilon\big(\phi(x)\big)\right| dx + \lambda \int_\Omega \Big(H\epsilon(\phi)\big(c_1 - I_0(x)\big)^2 + H\epsilon(-\phi)\big(c_2 - I_0(x)\big)^2\Big) dx \tag{11.}$$

2.5. Forma alapú aktív kontúrok

A régió alapú szegmentációk gyengesége az erősen zsúfolt háttérrel rendelkező képeknél és a szegmentálandó objektum takarásánál mutatkozik meg. Ilyen esetekben a sikeres szegmentációhoz szükséges, hogy valamilyen priori alak információt is belehelyezünk a modellbe. Mi itt egy pár olyan alkotást említünk meg, melyeknek geometriai alakját előre tudjuk.

Leventon-Grimson-Faugeras [22] voltak az elsők, akik a priori alak információt használtak fel aktív kontúr modell esetében melyet egy szinthalmaz keretben helyeztek el. Az alak modell a fő komponens analízis segítségével készült, mely a tanító halmazba próbálja megtalálni a fő variációkat, eldobva a redundáns információkat.

Ezt Cootes-Taylor [23] parametrikus aktív kontúrokon alkalmazta. Leventon új ötletének alapja, hogy a fő komponens analízist nem a parametrikus geometriai kontúrra, hanem ezek előjeles távolságfüggvényére alkalmazza, melyek implicitek és paraméter függetlenek.

E fejezetben felsorolt modellekből számos továbbfejlesztés készült még az elmúlt évek során. Az érdekelt olvasó ezeknek egy alapos áttekintését megtalálja Brensson 2005-ös doktori tézisében [17].

3. ELMÉLETI ALAPOK

A kép szegmentációban a szinthalmaz modell alatt értjük azon deformálódó modelleket, melyek a célalakzatot úgy kapják meg, hogy egy határfelületet terjesztenek végig egy sima függvény nulla szintjén. Az ilyen függvényeket nevezzük szinthalmaz típusúnak. Azt, hogy a határfelület terjeszkedjen, egy variációs megfogalmazáson keresztül érjük el. Azaz a szegmentáció problémáját egy energiafüggvény minimalizálásaként fogalmazzuk meg, mely magába foglalja a keresett forma tulajdonságait.

E lépést továbbá át lehet írni, mint egy időfüggő parciális differenciálegyenlet. Egy ilyennek a megoldása történhet véges-differenciál módszerrel [11], avagy ahogy ebben a dolgozatban bemutatjuk, használunk egy folytonos megközelítést. Itt a szintfüggvényt ki tudjuk fejezni egy folytonos parametrikus függvényként, melyen a differenciálegyenlet már könnyen megoldható. A felhasznált folytonos függvények, melyekből építkezni fogunk, a B-Spline-ok [12].

3.1. Szintfüggvény

Legyen Ω egy korlátos nyílt részhalmaza $\mathbb{R}^d$–nek és f: $\Omega \rightarrow \mathbb{R}$ egy d dimenziójú kép. A szinthalmaz megfogalmazásban a terjeszkedő $\Gamma \subset \mathbb{R}^d$ határfüggvény nem más, mint a nulla szint Lipschitz-folytonos függvénye egy ϕ–nek, mely d+1 dimenziójú és teljesíti az alábbi megkötéseket:

$$\begin{cases} \phi(x) > 0, & \forall x \in \Omega_{in} \\ \phi(x) < 0, & \forall x \in \Omega_{out} \\ \phi(x) = 0, & \forall x \in \Gamma \end{cases} \qquad (12.)$$

ahol Ω_{in} egy régió melyet a határfelületet határolja ($\Gamma = \partial\Omega_{in}$) és Ω_{out} meg a teljes tér többi része, azaz $\Omega \setminus \Omega_{in}$.

3.2. Energia kritérium

A határfelületet egy kezdeti helyzetből az energia kritérium függvény fogja a kívánt alakzatra vezérelni úgy, hogy közben ennek értéke a lehető legkisebb legyen. E függvénynek az általános alakját a következő alakban írhatjuk fel::

$$J(\phi) = \nu_{in} \int_{\Omega} g_{in}\big(x, \phi(x)\big) H\big(\phi(x)\big) dx_1 \dots dx_d$$

$$+ \nu_{out} \int_{\Omega} g_{out}\big(x, \phi(x)\big)\Big(1 - H\big(\phi(x)\big)\Big) dx_1 \dots dx_d \qquad (13.)$$

$$+ \nu_c \int_{\Omega} g_c\big(x, \phi(x)\big)\delta(\phi(x)) \parallel \nabla\phi(x) \parallel dx_1 \dots dx_d$$

Itt az első két elem a régió tagok, azaz a határfelülethez viszonyított külső és belső régióhoz rendelt energia érték. Az utolsó pedig a határfelülethez (Γ) kapcsolódik és a szakirodalomban kontúr tagként említik. Ennek megfelelően g_{in} és g_{out} leírja az objektumot és a hátteret, míg végül a g_c magát a keresett kontúrt. Továbbá a H a Heaviside (más nevén egységugrás) és δ a Dirac egyváltozós függvényt jelöli. ν_{in}, ν_{out} és ν_c pozitív paramétervektorok és egy-egy tag fontosságát lehet velük szabályozni.

3.3. Szintfüggvény interpoláció

3.3.1. Interpoláció

Arra, hogy mi is az interpoláció, több meghatározás létezik. Sok helyen úgy hallunk róla, mint az ismeretlen egy informált becslése. Ennek egy sokkal pontosabb megfogalmazását adja a következődefiníció: folytonos adat modell alapú visszanyerése diszkrét adatokból egy ismert intervallumon. Tehát az extrapolálásnál úgy becsülünk, hogy nem rendelkezünk ismert adattal azon az intervallumon. Emiatt becslésünk pontosabb lesz a modell közelében és kevésbé pontos attól távol.

Az interpoláció három fő hipotézisen alapszik:

- A háttérben egy folytonos jel van (adat formájában)
- Amennyiben adott egy csoport minta a jelből, belőle kiszámítható a mintavételezett folytonos jel értéke bármely pontban

- A mintavételezési pontokban az interpolálás pontosan a mintavételezett értékeket adja vissza

Ezért interpolációt kell használnunk, amikor egy feldolgozott folytonos jel csupán diszkrét minta sorozatként áll rendelkezésünkre. A digitális világunkban ez az állítás szinte mindig igaz. A képfeldolgozás terén alkalmazási példák: kép nagyítása, kicsinyítése, eltolása, tetszés szerinti szöggel való elforgatása és esetleges koordináta transzformációk.

A szakirodalomban két nagy interpolációs modellt alkalmaznak. Ezek habár másképp vannak megfogalmazva, felcserélhetőek, hiszen átmenet létezik mindkét irányba a kettő között [24]. A klasszikus interpoláció egy lineáris algoritmus és működését a következő formula írja le:

$$f(x) = \sum_{k \in \mathbb{Z}^q} f_k \gamma_{int}(x-k)\ ,(\forall)\ x = (x_1, \dots, x_q) \in \mathbb{R}^q \qquad (14.)$$

ahol f (x) az interpoláció értéke az x pontban az adott q dimenziós térben. Kiszámítása megegyezik a mintavételezett értékek (f_k) egy lineáris kombinációjával ahol egy-egy minta súlyát az $\gamma_{int}\,(x\text{-}k)$ szintézis függvény adja. A klasszikus keretben az γ_{int}-*re* egy jó példa a *sinc* függvény.

$$sinc(x) = \frac{\sin(\pi x)}{\pi x} \qquad (15.)$$

Habár a fenti képlet az egész számhalmazra van felírva, ez a gyakorlatban leegyszerűsödik, hiszen méréseink mennyisége véges. Azon egész pontokban, ahol nem rendelkezünk mintákkal, nullának tekinthetjük őket, vagy tükrözött szimmetria megkötéseket alkalmazhatunk [25]. A harmadik hipotézis ekvivalens az alábbi interpoláció megszorítással:

$$f_{k_0} = \sum_{k \in \mathbb{Z}^q} f_k \gamma_{int}(k_0 - k), \qquad (\forall) k_0 \in \mathbb{Z}^q \qquad (16.)$$

ami egy diszkrét konvolúció:

$$f_{k_0} = \left(f * \gamma_{int}(k)\right)_{k_0}, (\forall) k_0 \in \mathbb{Z}^q \qquad (17.)$$

Az általános interpoláció algoritmusnál a minták (f_k) helyét interpolációs együtthatók (c_k) veszik át, a következőképpen:

$$f(x) = \sum_{k \in \mathbb{Z}^q} c_k \gamma(x-k)\ , \qquad (\forall)\ x \in \mathbb{R}^q \qquad (18.)$$

Ezen átírással egy interpoláció két lépésből áll. Először a mintákból (f_k) meghatározzuk az együtthatókat (c_k), majd kiszámoljuk az f(x) értéket az együtthatók és a γ szintézis függvény

által szolgáltatott súlyzók lineáris kombinációjaként. E forma erőssége, hogy most a szintézis függvény megválasztására sokkal nagyobb szabadságunk van. Hátrány, hogy megjelenik egy extra lépésünk [24].

Az együtthatók meghatározására kiindulunk az interpoláció harmadik hipotéziséből és vegyük csupán az egész pontokban a mintákat:

$$f_{k_0} = \sum_{k \in \mathbb{Z}^q} c_k p_{k_0 - k}, \qquad (\forall) k_0 \in \mathbb{Z}^q \text{ és } p_k = \gamma(k) \tag{19.}$$

Ha adott a szintézis függvény, akkor a p_k értékeit is tudjuk, és az egyenlet átalakul egy lineáris egyenletrendszerré, ahol az ismeretlenek a c_k együtthatók. A fenti kifejezés egyetlen gondja, hogy végtelen ismeretlent és egyenletet tartalmaz, hiszen az összes egész pontot figyelembe vesszük felírásakor. Hogy az egyenletrendszert meg tudjuk oldani, a szintézis függvényt úgy választjuk meg, hogy csupán egy véges intervallumon legyen értelmezve és használjuk azt a tényt, hogy az adott mintaszám (k_0) is véges. Ekkor az együtthatók meghatározása [26]:

$$c = P^{-1} f \tag{20.}$$

A mátrix inverzió egy költséges és komplex feladat. Egy alternatív megoldás, hogy az interpoláció megkötést felírjuk egy konvolúciós szorzat formájában:

$$f_{k_0} = (c * p)_{k_0}, (\forall) k_0 \in \mathbb{Z}^q \tag{21.}$$

ami átalakítható a következő alakra [25]:

$$c_{k_0} = ((p)^{-1} * f)_{k_0}, (\forall) k_0 \in \mathbb{Z}^q \tag{22.}$$

Tehát elmondhatjuk, hogy a diszkrét konvolúció (ami egy digitális szűrőnek felel meg) alternatíva lehet az inverz mátrix kiszámítására az interpolációs együtthatók meghatározásának esetében. Ennek egy hatékony megvalósítását a B-Spline szintézis függvények esetében Unser dolgozta ki [13] [14] és Thévenaz implementálta C-ben [27].

3.3.2. Szintézisfüggvény óhajtott tulajdonságai

Olyan szintézisfüggvényt szeretnénk használni, amely a lehető legpontosabban megközelíti az eredetit. Ezt egy minél tágabb intervallumon definiált szintézis függvény éri el. Ennek az ára viszont több számítási művelet elvégzése a nagyobb figyelembe vett intervallum

miatt. Ezért meg kell, hogy találjunk egy kompromisszum értéket mely még elfogadható pontossággal dolgozik a lehető leggyorsabban.

Egy másik tulajdonság a szétválaszthatóság. Ez fontos az egynél nagyobb dimenziós interpoláció hatékony megvalósítására. Egy 10 egész pontra értelmezett szintézis függvény esetében, ha ez nem szétválasztható, egy 3D pont interpolációja 10^3=1000 műveletet jelent. Ellenkező esetben a magasabb dimenzió lebontható alacsonyabb dimenziók értékeinek a szorzatára, ami 3×10=30 művelet elvégzésével ekvivalens. Láthatjuk, ez lényeges teljesítményjavulást jelent és megengedi, hogy a gyakorlatban ún. "batch" műveletekre bontsuk az interpolációt [24].

$$\gamma_{sep}(x) = \prod_{i=1}^{q} \gamma(x_i)\,, \qquad (\forall) x \in \mathbb{R}^q \tag{23.}$$

Ahhoz, hogy a térbeli relációkat megfelelően megőrizzük, szükséges, hogy a szintézis függvény szimmetrikus legyen:

$$\gamma(x) = \gamma(-x) \tag{24.}$$

Az utolsó óhajtott tulajdonság az konstans minta megőrzése. Tehát ha mintáink mind azonosak, akkor az interpolált folytonos jel is legyen konstans és az egész pontok között ne oszcilláljon [25]. A gyakorlatban az egyik e tulajdonságokkal rendelkező és leggyakrabban használt szintézis függvény a B-Spline.

3.3.3. B-Spline

A Spline függvények nagyon jó szintézis függvények az általános interpolációs keretben. A B-Spline a legegyszerűbb ilyen függvény (polinom) és ezért a neve elején a B betű bázisnak felel meg. A B-Spline foka meghatározza, hogy egy hányad fokú polinommal írjuk le a szintézis függvényt. Egy B-Spline csak a hozzá legközelebb eső n+1 csomópontra építkezik, hiszen csak e pontokban van értelmezve.

Mint korábban említettük, ha a szintézis függvényünk (jelen esetben a B-Spline) nagyobb intervallumon van definiálva (nagyobb a támaszpontjainak száma), pontosabb interpolációt végez. Sajnos a fokszám növekedés fordítottan arányos annak futási gyorsaságával és ezért gyakorlatban a lehető legkisebb fokszámúval dolgozunk.

A 0-ad fokú B-Spline megadható, mint:

$$\beta^0(\mathrm{x}) = \begin{cases} 1, & |\mathrm{x}| < \frac{1}{2} \\ \frac{1}{2}, & |\mathrm{x}| = \frac{1}{2} \\ 0, & |\mathrm{x}| > \frac{1}{2} \end{cases} \tag{25.}$$

Magasabb fokszámnál alkalmazzuk az alábbi konvolúciós szorzatot:

$$\beta^n(x) = \beta^{n-1}(x) * \beta^0 = \beta^0(x) * \beta^0(x) * \ldots * \beta^0(x \tag{26.}$$

Tehát egy n-ed fokszámú B-Spline megegyezik n+1 darab 0-ad fokszámú B-Spline konvolúciós szorzatával. Mivel egy n-ed fokú polinommal végezzük az interpolációt, az így alkotott függvény n alkalommal deriválható. A képfeldolgozás során ritkán van szükség magasabb rendű deriváltra, mint a gradiens. Ezért a gyakorlatban a köbös B-Spline terjedt el mivel ennek még a másodfokú deriváltja is folytonos. Általános alakja [15]:

$$\beta^3(x) = \begin{cases} \frac{2}{3} - |x|^2 + \frac{|x|^3}{2}, & 0 \le |x| < 1 \\ \frac{(2-|x|)^3}{6}, & 1 \le |x| < 2 \\ 0, & 2 \le |x| \end{cases} \tag{27.}$$

Mi is ezt fogjuk használni a szintfüggvény interpolációjára. A B-Spline alkalmazási lehetőségeit és módszereit (főleg hatékony szűrök formájában) Michael Unser [13] [14]és Philipe Thévenez dolgozta ki [27].

3.3.4. B-Spline szintfüggvény modell

Fejezzük ki ϕ(x) – et mint B-Spline bázis függvények lineáris kombinációja [28]:

$$\phi(x) = \sum_{k \in \mathbb{Z}^d} c[k] \beta^n \left(\frac{x}{h} - k \right). \tag{28.}$$

Ahol $\beta^n(\cdot)$ az egységes és szimmetrikus d-dimenziójú n-ed fokú B-Spline. E függvény, habár paraméterként egy vektort kap, könnyen kezelhető, hiszen szétválasztható és ezért felírható, mint n darab egy dimenziós B-Spline szorzat [28]:

$$\beta^{\mathrm{n}}(x) = \prod_{j=1}^{d} \beta^n(x_j) \tag{29.}$$

Ez gyakorlatilag azt jelenti, hogy egy 2D-os B-Spline esetében például a ϕ kiszámítása nem más, mint alkalmazni az egy dimenziós számítási módszereket úgy a sorokra, mint az oszlopokra, egymás után. A B-Spline függvények csomópontjai, ami köré felírjuk őket, legyenek egy háló eloszlásban a Ω felületen, és két szomszédos pont közti a távolság legyen h. A B-Spline együtthatókat a $c[k]$ tárolja.

3.4. Energia kritérium minimalizálása

A hagyományos variációs keretben ezt az energia kritériumot a szinthalmaz függvényre nézve Euler-Lagrange egyenletek [8] avagy Fréchet-Gateux [7] deriváltak felhasználásával végezzük el. Ezzel ellentétben mi a deriválást is a B-Spline keretben végezzük el, ahol kifejezhetjük a minimalizálást is, mint a B-Spline együtthatok optimalizálása úgy, hogy az energia kritérium a lehető legkisebb értéket vegye fel. Bernard [12] bebizonyította, hogy ez nem más, mint (13.) –es egyenlet deriválása a $c[k_0]$ együtthatókra nézve:

$$\frac{\partial J}{\partial c[k_0]} = \int_{\Omega} \omega(x)\beta^n(\frac{x}{h} - k)\, dx_1 \dots dx_d \tag{30.}$$

Ahol ω felírható a következő formában:

$$\begin{aligned} \omega(x) = \nu_{in}&\left(\frac{\partial g_{in}(x,\phi(x))}{\partial \phi(x)} H\big(\phi(x)\big) + g_{in}\big(x,\phi(x)\big)\delta\big(\phi(x)\big)\right) + \\ \nu_{out}&\left(\frac{\partial g_{out}(x,\phi(x))}{\partial \phi(x)}\Big(1 - H\big(\phi(x)\big)\Big) - g_{out}\big(x,\phi(x)\big)\delta\big(\phi(x)\big)\right) + \\ \nu_c&\left(\frac{\partial g_c(x,\phi(x))}{\partial \phi(x)} \parallel \nabla\phi(x) \parallel - div\left(\frac{g_c(x,\phi(x))\nabla\phi(x)}{\parallel\nabla\phi(x)\parallel}\right)\right)\delta\big(\phi(x)\big). \end{aligned} \tag{31.}$$

Mivel ω(x) megadja a szegmentált objektum fő jellemzőit, a szakirodalomban *jellemző függvénynek* nevezik. Ahogy ezt Bernard is leírja, az energia kritérium minimalizálása a B-Spline együtthatókra nézve általánosan nem eredményez egy zárt alakú megoldást (azaz ez nem írható fel, mint véges és ismert függvények kombinációja).

Ezért a lokális minimum meghatározása érdekében az első fokú gradiens módszert alkalmazzuk mely a következő alakhoz vezet:

$$c^{(i+1)} = c^{(i)} - \lambda\nabla_c J(c^{(i)}) \tag{32.}$$

Itt λ a gradiens módszer lépése és ∇_c az energiafüggvény gradiense a B-Spline együtthatókra nézve, ahogy ezt az (30.) és (31.) egyenletek leírják. Az (30.)-es egyenlet egy érdekes értelmezéshez vezet amennyiben az alábbi formában definiáljuk egy n-ed fokú h-val felskálázott B-Spline-ot:

$$\beta_h^n(x) = \beta^n\left(\frac{x}{h}\right) \tag{33.}$$

Így az energia gradiens felírható az alábbi alakban:

$$\nabla_c J = \frac{\partial J}{\partial c[k]} = \int_\Omega \omega(x)\beta_h^n(x - hk)dx_1 \dots dx_d \tag{34.}$$

Ez meg már igencsak hasonlít és meg is egyezik a jellemző függvény és a $\beta_h^n(x)$ konvolúciójával, melyet h periódussal mintavételezünk.

3.5. Diszkrét forma

Képeinket a folytonos világból vesszük, azonban tárolási kapacitásunk végleges, ezért azt diszkrét formába alakítjuk valamilyen mintavételezési séma alapján. Képek szegmentációkor e forma áll rendelkezésünkre, ezért a fenti folytonos egyenleteket a gradiens kiszámolásához diszkrét formára kell hoznunk. E feladat megegyezik a jellemző függvény és a B-Spline bázis diszkrét alakjának felírásával.

3.5.1. Diszkrét jellemző függvény

Ennek megvalósítása a(17.)–os egyenlet diszkrét alakra való hozását jelenti és megegyezik a Heaviside és Dirac függvény megközelítésével. E problémával már több dolgozat is foglalkozott részletesen. Mi most a Chan-Vese [8]által javasolt formát fogjuk használni azáltal, hogy H-nak a Heaviside függvény valamely megközelítését és C^1 szabályosítását vesszük (H_ε). A Delta függvény (δ_ε) meg ennek a deriváltja ($H_\varepsilon`$) lesz, amikor $\varepsilon \to 0$.

E két alakot behelyettesítjük a (13.)–es egyenletbe, hogy megkapjuk az energia kritérium szabályosított formáját. Hasonlóan cselekedve a(31.)–as egyenlet esetében is megkapjuk az ezzel azonosított jellemző függvényt. Jelöljük az így kapott egyenletet $\omega_\varepsilon[u]$ –val, ahol $u \in \mathbb{Z}^d$.

3.5.2. Gradiens számítás

A gradiens diszkrét formájának felírásának érdekében a korábban meghatározott $\omega_\varepsilon[u]$ jellemző függvény mellett a B-Spline-t is átírjuk diszkrét alakra, ahogy azt Unser megfogalmazta [13]. Jelölje a szimmetrikus d dimenziós n-ed fokú diszkrét B-Spline–t a $b^n[u]$. Ezt megkapjuk, ha a mintavételezzük a folytonos $\beta^n(x)$–et az egész pontokban. A folytonosnál érvényes szétválaszthatóság tulajdonság itt is érvényes marad. Ha egész értékű távolságot használunk a B-Spline csomópontoknak ($h \in \mathbb{N}\backslash\{0\}$), az n–ed fokú h–val skálázott diszkrét B-Spline felírható a következő alakban:

$$b_h^n[u] = \beta^n\left(\frac{u}{h}\right) \tag{35.}$$

Ezt a (9.) – es egyenletbe behelyettesítve kapjuk a keresett diszkrét alakot:

$$< \nabla_c J > [k] = < \frac{\partial J}{\partial c[k]} > = \sum_{u \in \mathbb{Z}^d} \omega_\epsilon[u] b_h^n[u - hk]. \tag{36.}$$

Tehát az energia gradiense megfelel a jellemző függvény konvolúciós szorzatának a B-Spline–nal és alul mintavételezve egy h faktorral:

$$\nabla_c J > [k] = (\omega_\epsilon * b_h^n)_{\downarrow h}[k] \tag{37.}$$

E egyenlet egy hatékony eszközt ad mellyel könnyedén és gyorsan kiszámolható a gradiens és így a szinthalmaz függvény fejlődése a(32.)–es egyenlet segítségével.

3.6. Implementációs problémák

3.6.1. Korlátos szinthalmaz függvény

A határfelület terjedése során a szinthalmaz függvényben létrejöhetnek nagyon meredek vagy lapos régiók. Mindkét esetben a gradiens ilyenkor félrevezetheti a szintfüggvényünket, mivel úgy a nagyon nagy értékek, avagy a nullához közeliek numerikus instabilitást vezetnek be módszerünkbe. A klasszikus szinthalmaz keretben ennek elkerülésére időnként módosítják a kialakuló szintfüggvényt úgy, hogy az ismét egy távolság függvényt írjon le a nulla szinthez képest.

E megoldásnak azonban az egyértelmű algoritmus őszszámítási költség megnövekedése mellett egy topológia megkötést is eredményez. Ez úgy nyilvánul meg, hogy a kezdeti határfelülettől távol nem engedi meg, hogy újabb nulla szintek jöjjenek létre. Tehát csak azon kontúrokat kapjuk meg, amelyek már a kezdeti kontúr is tartalmazott valamilyen szinten.

E probléma egy elegáns megoldása a szinthalmaz függvény korlátozását jelenti. Ezt a módszert Bernard már sikeresen használta a radiális bázis függvénnyel felírt szinthalmaz függvény esetében [29]. Mivel a B-Spline alakja egy lineáris kifejezés, ez is könnyen felírható, mint a B-Spline együtthatók normalizálása. Bernard [12] bebizonyította, hogy az így nyert korlátos szinthalmaz függvény megakadályozza a meredek és lapos régiók kialakulását és ez által szükségtelenné válik az időnkénti újra inicializálási lépés a klasszikus numerikus megoldási módszerekkel szemben.

Hogy gyakorlatban is könnyen alkalmazható legyen, a normalizáláshoz a végtelen normát használjuk. Ezt maximum normáként is ismeretes mivel általános formája:

$$\| c \|_\infty = \max(|c_1|, |c_2|, \ldots, |c_n|). \tag{38.}$$

Egy implicit függvényt amennyiben beszorzunk egy nem nulla együtthatóval, az nem változtatja meg annak felületét. Mivel ϕ a B-Spline együtthatókon keresztül van ábrázolva,ϕ szorzása ε – al megfelel a $c[k]$együtthatók szorzásával ugyancsak ε–nal. Bernard a B függelékében [12] bebizonyította, hogy a B-Spline együtthatók végtelen normája ($\|c\|_\infty$) korlátossá teszi a szinthalmaz függvényt:

$$|\phi(x)| < \| c \|_\infty \tag{39.}$$

Tehát a szinthalmaz függvény újranormalizálási lépése (a [-1, 1] intervallumba) nem más, mint, hogy egy-egy első fokú gradiens módszer lépése után elvégezzük a következő műveletet:

$$c^{(i+1)} = \frac{c^{(i+1)}}{\| c^{(i+1)} \|_\infty} \tag{40.}$$

Szükséges megemlíteni, hogy ez egyúttal határokat szab a szinthalmaz függvény gradiens normájára is. Ennek mértékére és annak bizonyítására az érdekelt olvasó Bernard cikkét tanulmányozhatja [12].

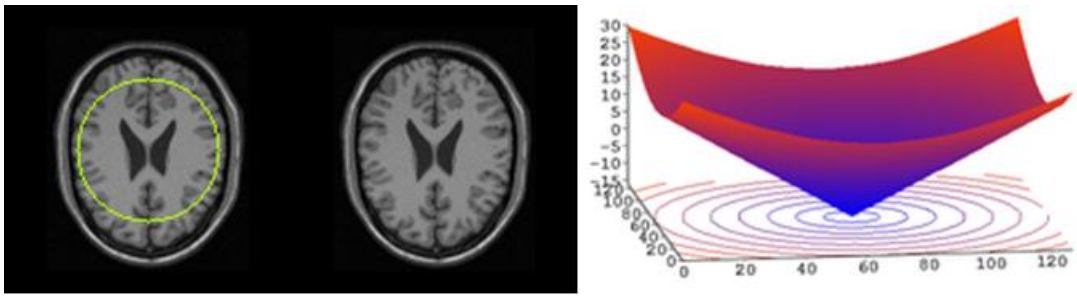
3.6.2. Első fokú gradiens módszer

Az energia függvény minimalizálására egy első fokú gradiens módszert alkalmazunk melynél visszacsatolt lépésváltoztatást használunk. Minden iterációban alkalmazzuk először a(32.)–es, majd a(40.)–es egyenletet, hogy a jelenlegi ismert $c^{(i+1)}$–ből kiszámoljunk egy lehetséges $c^{(i+1)}$–et és ezen együtthatók által eredményezett energia éréket. Ha a korábbi energia értékhez képest kisebb értéket kapunk, akkor a gradiens módszer e lépése sikeres volt és elfogadjuk a kiszámolt értékeket.

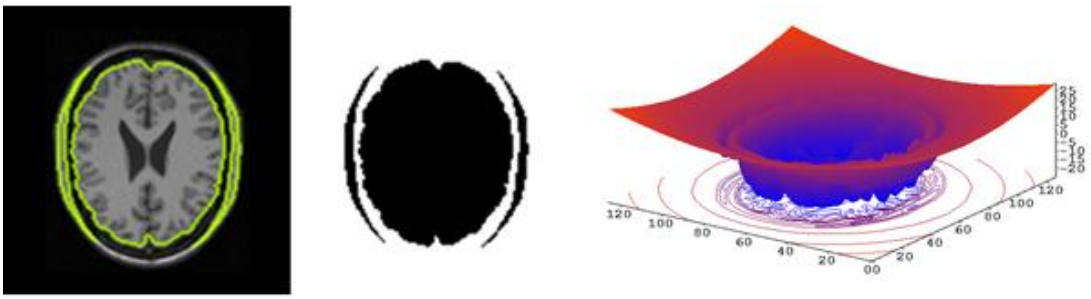
A továbbiakban ezzel számolunk, és a lépés méretét egy $\eta_f \geq 1$ értékkel *beszorozzuk*. Ellenkező esetben eldobjuk a kapott értéket és a lépést egy $\eta_f \geq 1$ értékkel *osztjuk*. Ezután megismételjük az iterációt addig, amíg egy maximális lépésszám határt el nem érünk vagy amíg $c^{(i)}$ egy kívánt küszöbérték alatti mértékben nem változik több egymás utáni iteráció során. E több lépésre azért van szükség, hogy megbizonyosodjunk, hogy a megoldásunk stabil és a felület már biztos nem fog alakulni tovább.

3.7. Futás idejű vizualizáció

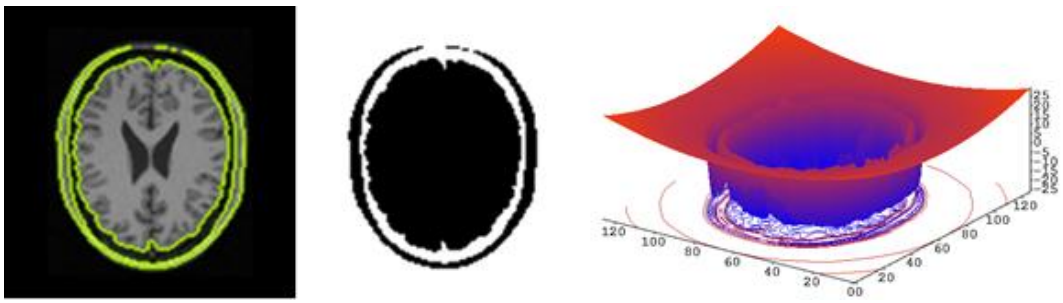
Az algoritmus működését jól megfigyelhetjük, ha egymás mellé rakjuk a szinthalmaz függvény, a szegmentálás és a képen a kontúr fejlődését. A további ábra sorozaton különböző iteráció után figyelhetjük meg mindezek állapotát. Az első ábra szintfüggvényén jól látható az előjeles távolság függvény formája (ami a szinthalmaz függvény kezdeti felülete) és a továbbiakban annak fejlődése az EM algoritmus iterációi során.



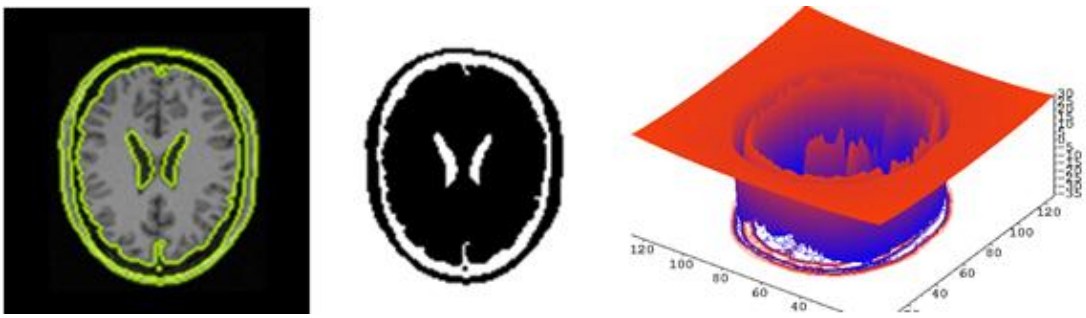
Ábra 10. A kezdeti kontúr, kép, és szinthalmaz függvény



Ábra 11. A kontúr, szegmentáció és szinthalmaz függvény 1 iteráció után

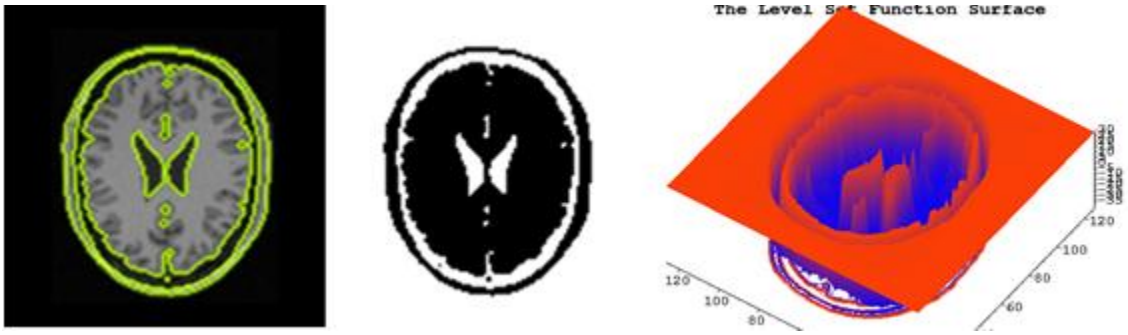


Ábra 12. A kontúr, szegmentáció és szinthalmaz függvény 4 iteráció után



Ábra 13. A kontúr, szegmentáció és szinthalmaz függvény 7 iteráció után

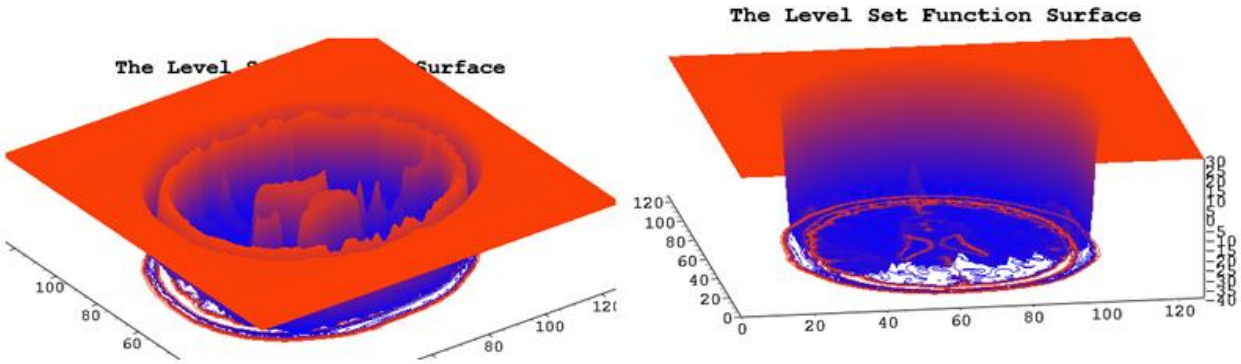
Ábra 14. A kontúr, szegmentáció és szinthalmaz függvény 10 iteráció után



Ábra 15. A kontúr, szegmentáció és szinthalmaz függvény 14 iteráció után

A fenti képsorozat egy agy MRI szegmentációját mutatja be. A kép egy intenzitás kép. Célunk az agyat elválasztani a háttértől (amely sötét színeket vesz fel) és a szegmentálás eredményeként egy kontúr halmazt határozunk meg. A képen e halmaz sárga színnel van feltüntetve.

A szegmentálást egy bináris képen jelenítsük meg. A háttérhez tartozó képpontok fehér színt vesznek fel, míg az agy és koponyához tartozóak feketét. A szegmentáció meghatározására a szintfüggvény használjuk fel. A szintfüggvény azon pontjai ahol pozitív értéket vesz fel, a háttérhez tartozik. A negatív értékek az agy-koponyához. Az alábbi képeken láthatjuk a szintfüggvény 3 dimenziós reprezentációját az algoritmus végén. A keresett kontúr megegyezik azokkal a pontokkal, amelyek a nulla szint értéken helyezkednek el.



Ábra 16. A szinthalmaz függvény a szegmentáció végén (14 iteráció)

4. A RENDSZER SPECIFIKÁCIÓJA ÉS ARCHITEKTURÁJA

4.1. Specifikáció

Célunk az előbbi fejezetben bemutatott elmélet, algoritmus megvalósítása az asztali számítógépekre. Az algoritmus megvalósításában fontos, hogy az majd könnyen használható és alkalmazható legyen különböző konzol és grafikus alapú felhasználó felületek között. Továbbá fontos, hogy a lehető leggyorsabban fusson, és könnyedén továbbfejleszthető legyen.

Összpontosítsunk egyelőre a kétdimenziós képekre. Biztosítsuk, hogy a lehető legtöbb formátumú képanyagot kezelni tudunk, de mindenképp a png, jpeg és bmp formátumokat. A fejlesztett csomagnak könnyű és direkt hozzáférést kell biztosítania a fontos paraméterei futás előtti konfigurálására.

Ugyanakkor, meg kell valósítani egy grafikus felületet mely lehetőleg platform független legyen és lehetővé tegye, hogy a felhasználó könnyedén módosítsa az algoritmus futási paramétereit. Ide értünk olyan műveleteket, mint:

- a szegmentált kép kiválasztása
- a kiindulási kontúr újra rajzolása
- Algoritmus paraméterek változtatása futás előtt:
  - skálázási érték
  - célpontosság
  - maximális iteráció szám

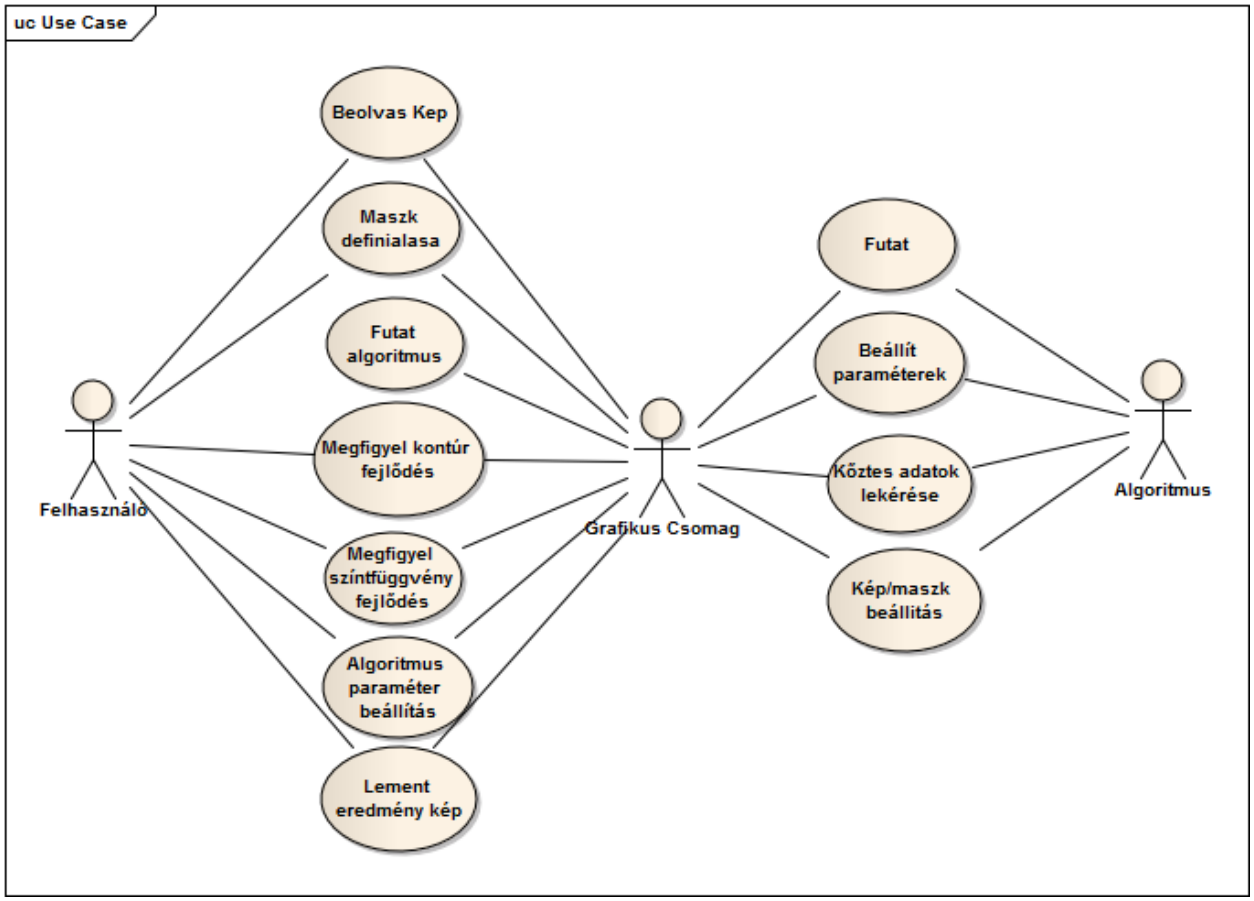
Továbbá, hogy a felhasználó jobban átlássa, az algoritmus működése közben rajzoljuk ki úgy a fejlődő kontúrt, mint a szegmentálás eredményét. Továbbá valósítsuk meg, hogy a felhasználó háromdimenziósan megfigyelhesse a fejlődő szintfüggvény felületét. Fontos, hogy a szegmentáció eredményét könnyedén le lehessen menteni a számítógép merevlemezére.

Mivel a grafikus felület a futás közbeni adatokat is megjelenítheti, tervezzünk egy olyan interfészt, amely időt ad a grafikus felületnek, hogy elvégezze a köztes adatok kirajzolást az algoritmus egy-egy iterációja között.

4.2. Architektúra

4.2.1. Use Case Diagram

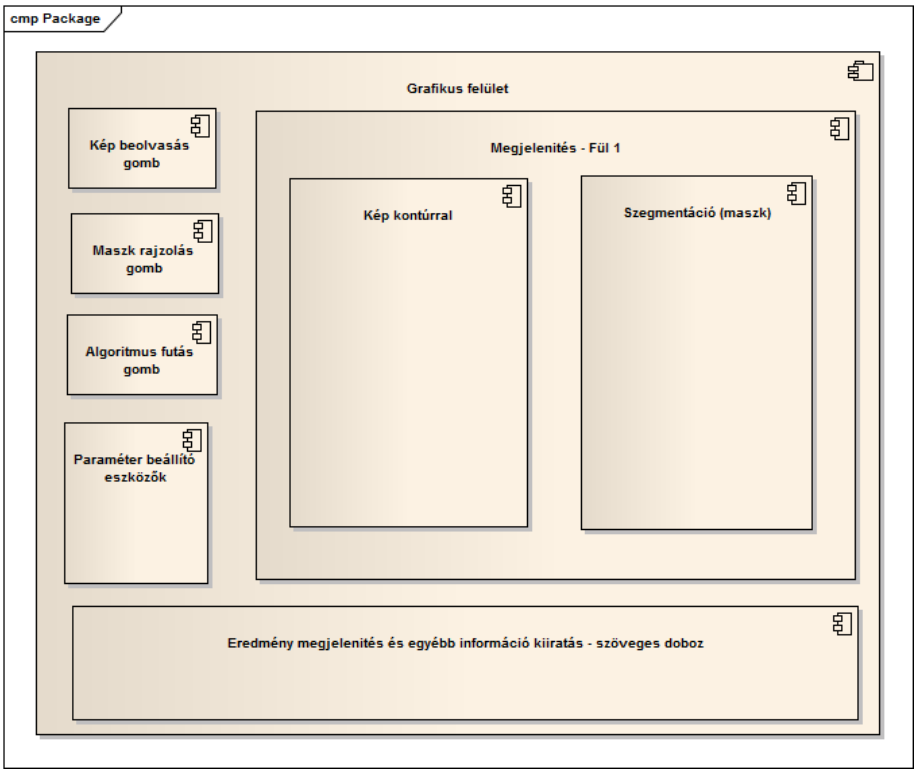
Az Enterprise Architect szoftvercsomag segítségével elkészítettem az alábbi use case diagramot melyben megfigyelhetjük a rendszer fő funkcióit és annak felelőségi körét:



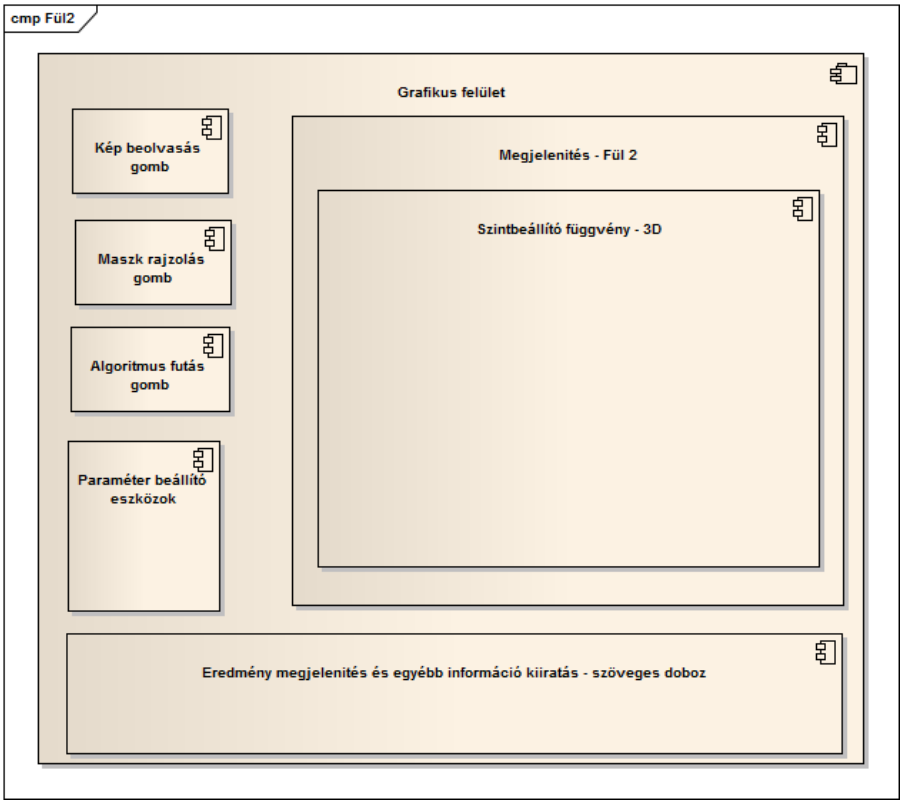
Ábra 17. Use-Case Diagram

4.2.2. Grafikus felület komponens diagram

Most lássuk a fejlesztendő grafikus felület komponens diagramját. Mivel minél jobban ki akarjuk használni a rendelkezésünkre álló képernyő felületet, különválasztjuk a képek (maszk/szegmentáció és a kontúros kép) megjelenítését és a szintfüggvény háromdimenziós kirajzolását. Hogy a kettő között gyorsan tudjunk váltogatni e két elemet a megjelenítési felületen egy füles ablakba helyezzük el. Ezt szemlélteti az alábbi két ábra is:



Ábra 18. A grafikus felület komponens diagramja (Fül 1)



Ábra 19. A grafikus felület komponens diagramja (Fül 2)

5. MEGVALÓSÍTÁS

5.1. Használt függvények és paraméterek

Az algoritmus implementálásához most definiálnunk kell a függvényeket, melyek leírják a keresett alakzat belső, külső, és esetleg a határfelületeit. Legyen ez a Chan-Vese féle funkcionál mely részenként konstans intenzitású régiókra oszt egy képet.

$$
\begin{aligned}
&J\epsilon(\phi, \mu_{in}, \mu_{out}) \\
&\quad = \int_{\Omega} (f(x) - \mu_{in})^2 H_\epsilon\big(\phi(x)\big) dx_1 \dots dx_d \\
&\quad + \int_{\Omega} (f(x) - \mu_{out})^2 \Big(1 - H_\epsilon\big(\phi(x)\big)\Big) dx_1 \dots dx_d \qquad (41.) \\
&\quad + \nu \int_{\Omega} \| \nabla\phi(\mathrm{x}) \| \delta\big(\phi(x)\big) dx_1 \dots dx_d
\end{aligned}
$$

ahol μ_{in} és μ_{out} az energia függvény két paramétere, melyek a jellemző függvény paraméterei is lesznek. Továbbá a ν egy vektorparaméter mely a régiók (az első két integrál) és kontúr energiája (az utolsó integrál) közti viszonyt fejezi ki. E kifejezést deriválva megkapjuk a jellemző függvényt:

$$
\begin{aligned}
\omega_\epsilon(x) = (\,(f(x) - \mu_{in})^2 - (f(x) - \mu_{out})^2 \\
- \nu\, div\left(\frac{\nabla\phi(x)}{\| \nabla\phi(\mathrm{x}) \|}\right))\delta\epsilon(\phi(x)) \qquad (42.)
\end{aligned}
$$

Általában az alakzatot leginkább a két régió tag határozza meg és a harmadik tag inkább egy megkötés, ami ellenszegül a határfelület terjedésének. Mivel a gyakorlatban sokkal gyakoribb, hogy a keresett alakzat és a tárgy egyértelműen meghatározza a határfelületet, elhagyjuk a harmadik tagot. Ezzel lényegesen felgyorsítjuk algoritmusunk futási idejét, hiszen a divergencia és a ∇ operátor számításigényes műveleteket jelent. Ugyanakkor Bernard kísérleti eredményei is alátámasszák e döntést [12].

A fenti egyenletekbe a Heaviside és a Dirac függvény, következő egyváltozós szabályosított alakjait használjuk:

$$\begin{cases} H_\epsilon(x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x}{\epsilon}\right) \\ \delta_\epsilon(x) = \frac{d}{dx} H_\epsilon(x) = \left(\frac{1}{\pi}\right) \frac{\frac{1}{\epsilon}}{1 + \left(\frac{x}{\epsilon}\right)^2} = \frac{1}{\pi\epsilon} \frac{1}{1 + \left(\frac{x}{\epsilon}\right)^2} \end{cases} \quad (43.)$$

E egyenletben ϵ egy valós pozitív konstans és a Dirac meg a Heaviside függvény a végtelen normalizált skálázását adja meg. Ennek elég nagynak kell lennie, hogy a határfelület terjeszkedése kihasson az egész szinthalmaz függvényre és így az algoritmus végén megkapjuk annak globális minimumát [8]. Mivel a szinthalmaz függvényét a maximális normalizálással a [-1,1] intervallumba korlátozzuk, ennek értékét egyre állítjuk be.

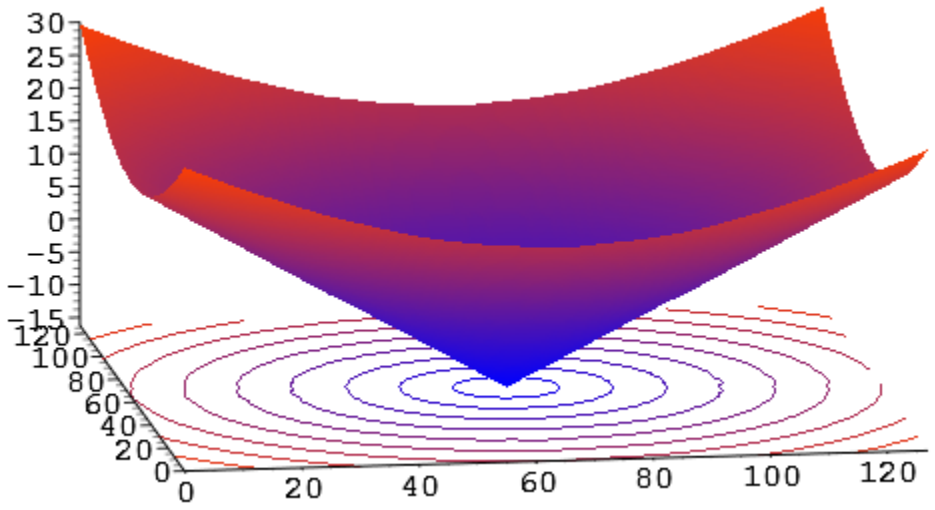
Az μ_{in} és μ_{out} meghatározására az Expectation-Maximization (EM) algoritmust használjuk. Nevét 1977-ben kapta mikor Dempster, Laird és Rubin a Royal Statistic folyóiratban először publikálták. A maximum likelihood becslés iteratív számítására alkalmas főként olyan helyzetekben, amikor hiányzó adataink vannak. Jelen esetben a μ_{in} és μ_{out} azok az értékek melyektől átlagosan a keresett tárgy, illetve a háttér intenzitása legközelebb található. Ezt előre viszont nincs, hogy honnan tudjuk, ezért a jellemzőfüggvény e két paraméterét is frissítjük minden iterációban.

Az energia és jellemző függvény értékét három dolog befolyásolja. A μ_{in} és μ_{out} paraméterek mellett, mint változó megjelennek a B-Spline együtthatók is. Ezért az optimalizálás egy-egy iterációja két lépésből áll. Először lekötjük a B-Spline együtthatókat és elvégezzük az energia minimalizálását a paraméterekre nézve. Ez felírható a következő alakban:

$$\begin{cases} \mu_{in} = \frac{\int_\Omega f(x) H_\epsilon\big(\phi(x)\big) dx_1 \dots dx_d}{\int_\Omega H_\epsilon\big(\phi(x)\big) dx_1 \dots dx_d} \\ \mu_{out} = \frac{\int_\Omega f(x)(1 - H_\epsilon\big(\phi(x)\big) dx_1 \dots dx_d}{\int_\Omega \Big(1 - H_\epsilon\big(\phi(x)\big)\Big) dx_1 \dots dx_d} \end{cases} \quad (44.)$$

Ez diszkrét formában ekvivalens a jelenlegi tárgynak vett képpontok (μ_{in}) és a háttár képpontok átlag intenzitásával (μ_{out}) külön-külön. Az optimalizálás második lépését (az együtthatókra vonatkoztatva) meg a (32.) –es egyenletet felhasználva végezzük el. A gradiens lépését η_f= 1–el szorozzuk siker esetén és η_f' = 1.5 – el osztjuk másképpen.

A kezdeti határfelületnek a kép közepében (vagy akár ezt tehetjük máshova is) a kép átmérőjének negyede sugarú maszkot rajzolunk. A maszk egy bináris kép mely nulla és egyes értékekből áll. A kör belseje megfelel a keresett tárgynak és egyesekkel jelöljük a maszkban.



Ábra 20. Előjeles távolság függvény egy kör esetében

A kezdeti szinthalmaz függvény ennek lesz az előjeles távolság függvénye. E függvény felépítésére kiszámoljuk a maszk és az inverz maszk távolság függvényét. E kettőnek a különbsége meg a maszk mínusz még egy egység összege adja a keresett függvényt. Egy kör esetében például ennek formája a fenti ábrán tekinthető meg:

5.2. Algoritmus vázlat

A következő táblázatban megtekinthetjük az algoritmus pszeudokódba leírt működését.

Táblázat 3. Az algoritmus vázlata

Inicializálás	• Algoritmus paramétereinek beállításai (maximális iteráció szám, skálázás, kívánt pontosság, első fokú gradiens lépésváltozás, stb.) • Kép beolvasás (ha színes kép átalakítani intenzitás képbe) • Kezdeti zárt határfelület (maszk) meghatározássá (például egy kör rajzolása a képbe) • Kép és maszk kiegészítése, hogy méretei kettő többszöröse legyen • A szintfüggvény inicializálása a maszk alapján az előjeles távolságfüggvény szerint • A skála alapján kiválasszuk a futáskor B-Spline együtthatókra alkalmazót szűrő együtthatóit • B-Spline együtthatók meghatározása a kezdeti szintfüggvényből $\mu_{in} = \frac{\int_\Omega f(x) H_\epsilon(\phi(x)) dx_1 \dots dx_d}{\int_\Omega H_\epsilon(\phi(x)) dx_1 \dots dx_d}$ • μ_{in} és μ_{out} meghatározása $\mu_{out} = \frac{\int_\Omega f(x)(1-H_\epsilon(\phi(x)) dx_1 \dots dx_d}{\int_\Omega \left(1-H_\epsilon(\phi(x))\right) dx_1 \dots dx_d}$ • $J\epsilon(\phi, \mu_{in}, \mu_{out}) =$ $\int_\Omega (f(x) - \mu_{in})^2 H_\epsilon(\phi(x)) dx_1 \dots dx_d +$ $\int_\Omega (f(x) - \mu_{out})^2 \left(1 - H_\epsilon(\phi(x))\right) dx_1 \dots dx_d +$ $\nu \int_\Omega \| \nabla \phi(x) \| \delta(\phi(x)) dx_1 \dots dx_d$ • számoló = 0, stabilSzámoló=0
Futás	**Amíg** (számoló <MaxIterációSzám és stabilSzámoló< 6) • Minimalizál ○ $\omega_\epsilon(x) = ((f(x) - \mu_{in})^2 - (f(x) - \mu_{out})^2)\delta\epsilon(\phi(x))$ ○ ω_ϵ normalizálása ○ $< \nabla_c J > [k] = (\omega_\epsilon * b_h^n)_{\downarrow h}[k]$ ○ Első fokú gradiens módszerrel minimalizál 5-ször ▪ $c^{(i+1)} = c^{(i)} - \lambda \nabla_c J(c^{(i)})$ ▪ $c^{(i+1)} = \frac{c^{(i+1)}}{\|c^{(i+1)}\|_\infty}$ ▪ Kiszámol $\phi(x)$, μ_{in}, μ_{in}, $J\epsilon(\phi, \mu_{in}, \mu_{out})$ ▪ **Ha** $J\epsilon(\phi, \mu_{in}, \mu_{out})$ csőken • Elment értékek és folytat minimalizálás ▪ **Másképp** • Kilép minimalizálás ▪ **Vége** Ha • Kiszámol maszk (szintfüggvény>= 0), megszámol változás • **Ha** változás <kívánt pontosság • stabilSzámoló = stabilSzámoló + 1 • **Másképp** • stabilSzámoló = 0 • **Vége** Ha • számoló = számoló +1 **Vége** amíg

5.3. Használt fejlesztési eszközök és metódusok

A program specifikációi megkövetelték, hogy az algoritmus a lehető leggyorsabb legyen. E cél elérése érdekében algoritmusom a C++ programozási nyelvben lett megvalósítva, ugyanis ez a leginkább hardverközeli környezet mellyel a legjobb teljesítményt lehet elérni anélkül, hogy az **O**bjektum **O**rientált **P**rogramozási (OOP) paradigmákról le kellene, hogy mondjunk.

Az OOP paradigma jelenléte azért fontos, mert ezzel könnyen lehet teljesíteni egy következő specifikációs követelményt: a könnyen felhasználhatóságot különböző grafikus környezetben. Ennek érdekében az algoritmus adatait összezárjuk az azokon dolgozó függvényekkel egy osztályba (*BSplineLevelSet*) és, hogy ne kelljen mindezt összekeverni a grafikus interfész projektünkkel, ezt elhelyezzük egy futás időben dinamikusan betöltött könyvtárba (Windows operációs rendszer DLL).

Fejlesztési környezetnek a Microsoft Visual Studio programcsomagot használtam. Továbbá felhasználtam az OpenCV képfeldolgozási könyvtárat mivel ez elegáns és megbízható módon megoldja a kép beolvasási és kiíratási műveleteit, és biztosít egy osztályt, amellyel beolvasás után könnyedén lehet annak képpontjait kezelni. Kiemelném itt, hogy én az új OpenCV 2.0-ban megjelent C++ interfészen dolgoztam.

A grafikus felület megtervezésére a Qt platform független fejlesztőcsomagot használtam fel. Ugyanakkor projektemben szerepet kapott a QwtPlot3D forrásállomány csomag is mely egy pár olyan osztályt valósít meg, amivel háromdimenziós felületek megjelenítése lehetséges felhasználva az OpenGL technológiát [16].

5.4. Diagramok

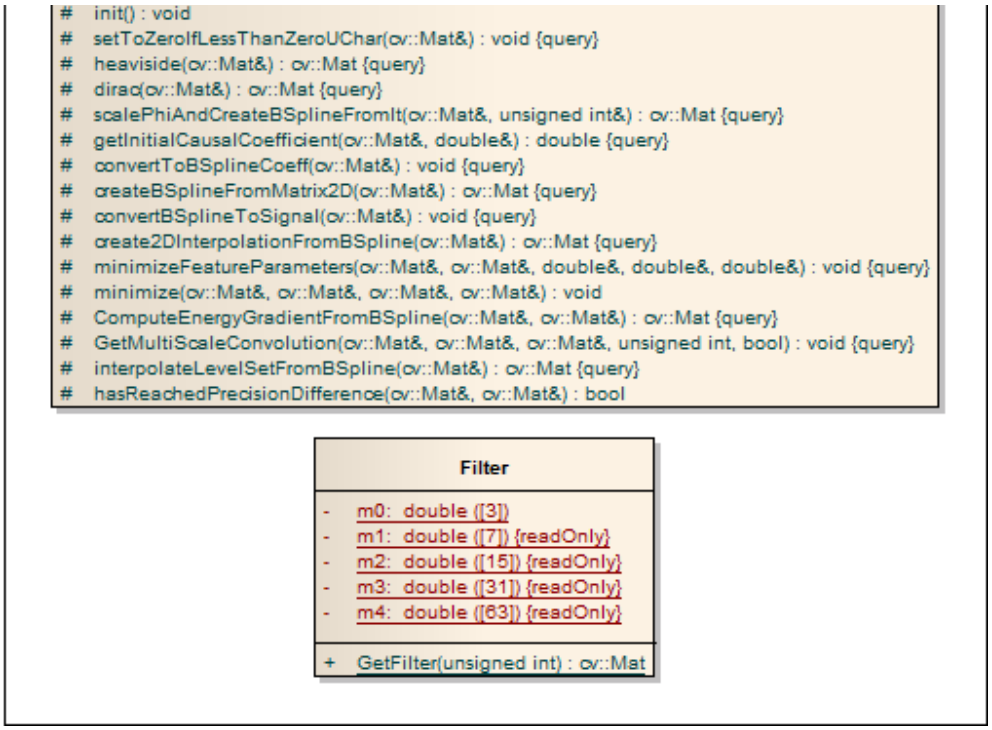
A diagramok szerkesztésére az Enterprise Architect program csomagot használtam.

5.4.1. Osztálydiagram

Az algoritmust egy objektum orientált keretben valósítottam meg. Így az algoritmus be- és kimeneti paraméterei egy osztály attribútum tagjai, melyet a felhasználó beállít futatás előtt felhasználva az erre megalkotott függvényeket (*Pl. Mask()*). Az algoritmus részműveletei az

osztályba elrejtett (privát vagy védett) metódusokban található meg. Így elkerüljük az osztály helytelen felhasználását.

Az osztály használatára csupán be kell állítani a bemeneti képet és a kezdeti maszkot (amely meghatározza, hogy kezdeti kontúrból indul az algoritmus) és meghívjuk ennek a *Run()*publikus eljárását amennyiben az alapértelmezett algoritmus paraméterekkel szeretnénk dolgozni. A *Run()* minden egyes hívásra elvégez egy iterációt, lehetőséget hagyva, hogy a köztes adatokat is meg tudjuk jeleníteni.

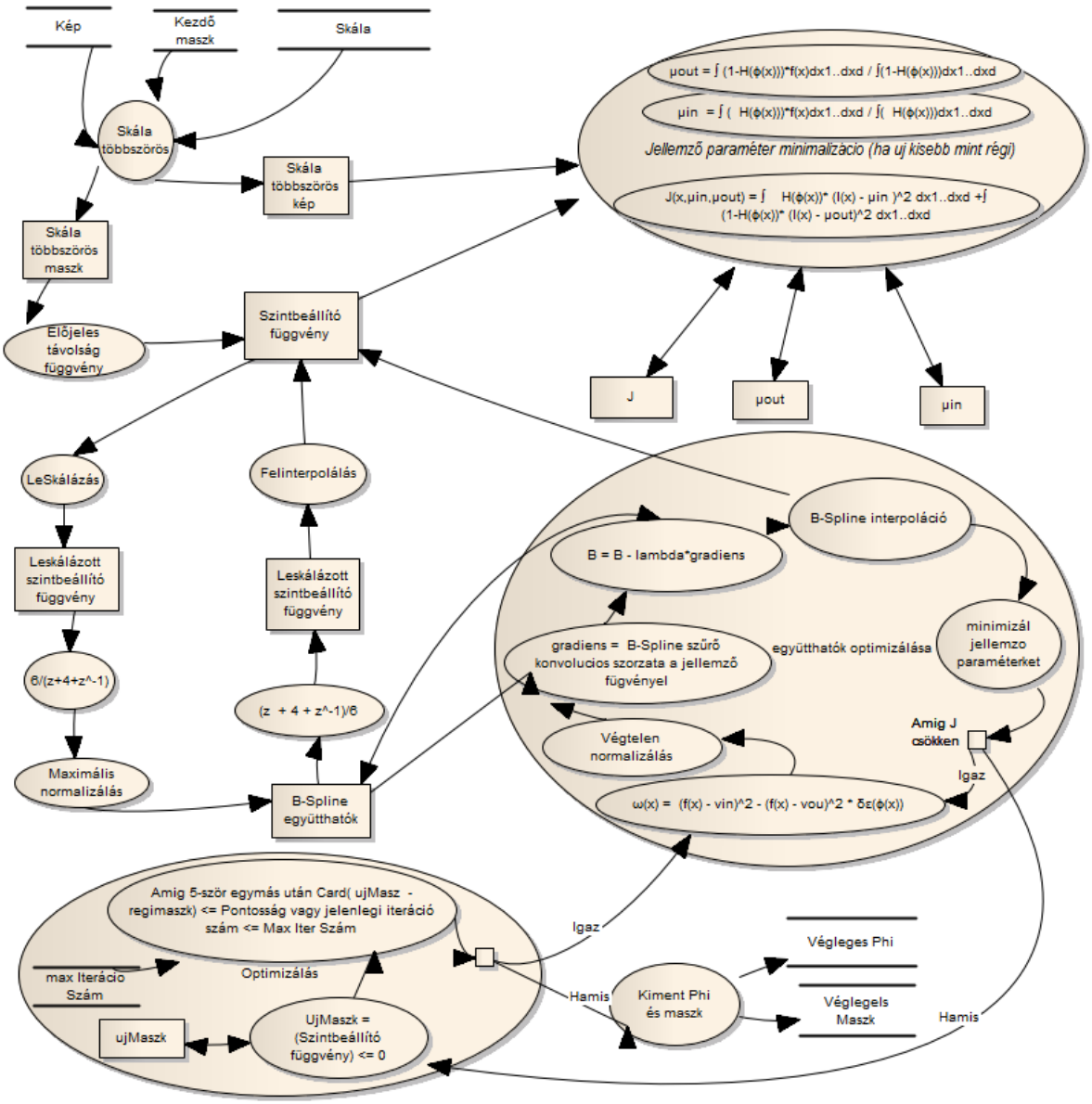


Ábra 21. A BSplineLevelSet osztálydiagramja

E függvény visszatérítési értéke igaz lesz mindaddig, amíg az algoritmust ismét meg lehet hívni (mert egyik befejezési feltétel sem teljesült). Ez után a *drawContour()* függvénnyel megkaphatjuk az algoritmus során előállítót szinthalmaz függvény által meghatározott kontúrt egy kép formájában. A Filter egy belső segítő osztály.

5.4.2. Adatfolyam

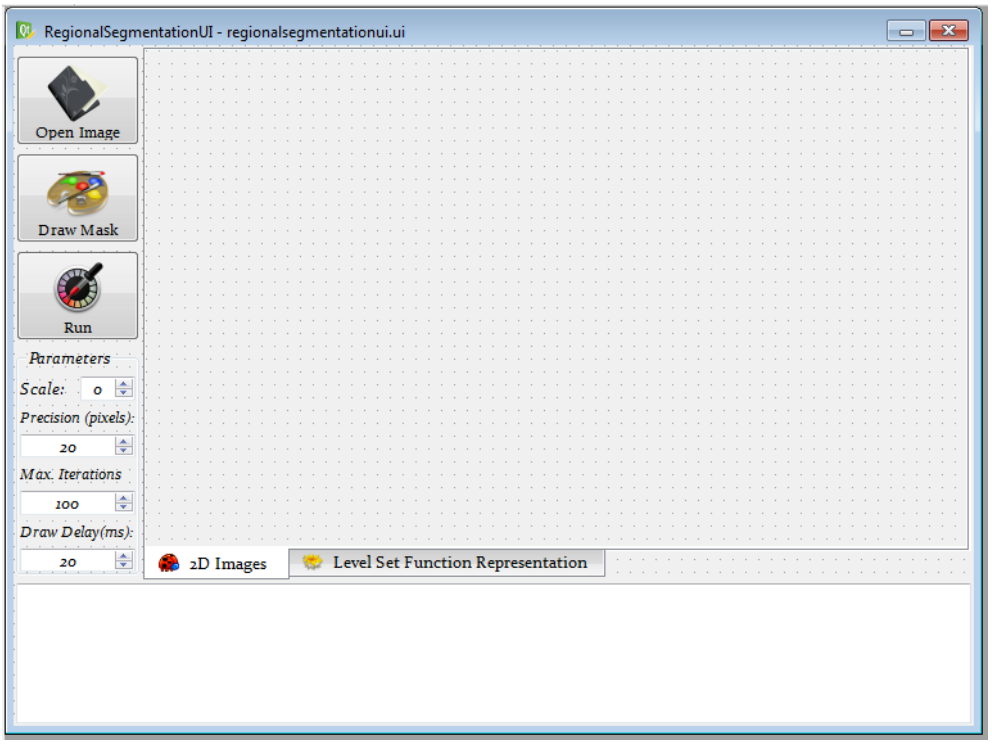
Az itt megtalálható ábrán megfigyelhetjük, hogy az adatok milyen átalakulásokon mennek keresztül, amíg megkapjuk a keresett tárgy alakzatát. Fontos megjegyezni, hogy ezen ábra nem tartalmaz minden apró részletet. Ennek ellenére jól kiemeli azon részfeladatokat, amelyeket meg kell oldanunk az implementációs fázis során: bemeneti kép kiegészítése kettőnek a többszöröséhez, a kezdeti előjeles távolságfüggvény alapján a szintfüggvény létrehozása felhasználva a bemeneti maszkot, meg az EM és a minimalizálási algoritmus.



Ábra 22. Adatfolyam diagram

5.5. Grafikus felület megvalósítása

A grafikus felület elkészítése a Qt Designer alkalmazással kezdődött. Ebben létrehoztam a következő dialógus sablont.



Ábra 23. A Qt Designer-el tervezett grafikus felület

A Qt signal/slot mechanizmusával az alkalmazás egy-egy függvényéhez csatoltam a gombokat. Továbbá felhasználtam az OpenCV Qt specifikus megjelenítését (a highgui forráskódjaiból kiemelve), azon kisebb módosításokat végeztem és beültetem az én alkalmazásomba. A képek beolvasására ismét az OpenCV könyvtárat használtam. Azok megjelenítésére a módosított Qt jellegű highgui-it.

Az algoritmus főleg nagyobb és komplexebb képeken jelenlegi implementációjában igencsak hosszas ideig futhat. Ez alkalmatlanná teszi, hogy direkt felhasználjuk alkalmazásunkban, hiszen hosszabb ideig az válaszképtelenné válna. Ennek elkerülése érdekében az algoritmus objektumát egy külön szálon példányosítjuk és futatjuk. A köztes eredmények megjelenítésére szinkronizálnunk kell, a grafikus felületet működtető és az algoritmust futató szállakat. E célra egy mutex-et használók.

Az algoritmust futató száll egy-egy iteráció között egy előre adott időre alszik, hogy ez idő alatt biztos kiütemeződjön és a grafikus szál el tudja végezni a kirajzolást és esetleges más

közben érkező üzeneteket. Ez az érték milliszekundumba kifejezve van a "Draw Delay" mezőben tárolva. Ezt felelősségteljesen módosítsuk, hiszen kis időtartam esetén a grafikus interfész válaszképtelenné válhat, amíg az algoritmus le nem fut teljesen.

A szinthalmaz függvény háromdimenziós megjelenítésére a Qwt3DPlot csomagot használom. Az algoritmus futása végén egy OpenCV képmátrix formájában áll rendelkezésre a szinthalmaz függvény. Ezen értékeket át kell, hogy adjuk a Qwt3DPlot *SurfacePlot* típusú osztálynak. Hogy a kirajzolási formán is tudjak elvégezni módosításokat származtattam ebből az osztályból egy újat (*Plot2D*) mely konstruktőrében elvégeztem a szükséges beállításokat és az *update* metódusában lekezeltem, hogy az éppen használt szintfüggvény függvényében méretezze újra magát.

Mindemelet a *Function* osztályból származtatva (melyet a *SurfacePlot)* paraméterként kap, létrehoztam egy saját osztályt (*LevelSetFunctionSurface*) mely elvégzi az adat átalakítást az OpenCV mátrixból a Qwt3DPlot számára ismertté. A *Plot2D* osztályt az alkalmazás indításakor példányosítóm és elhelyezem a második fül belsejében.

A maszk újrarajzolására egy szűrőt alkalmaztam az alkalmazás üzenet rendszerén. Figyelem a bal klikket a kontúr kép belsejébe a funkció aktiválása után, és újabb kezdeti kör kontúrt lehet megadni a kör és sugarának meghatározása során. A felhasználóval a program aljábba elhelyezett szöveges dobozzal kommunikálok.

6. A RENDSZER FELHASZNÁLÁSA

6.1. Algoritmus

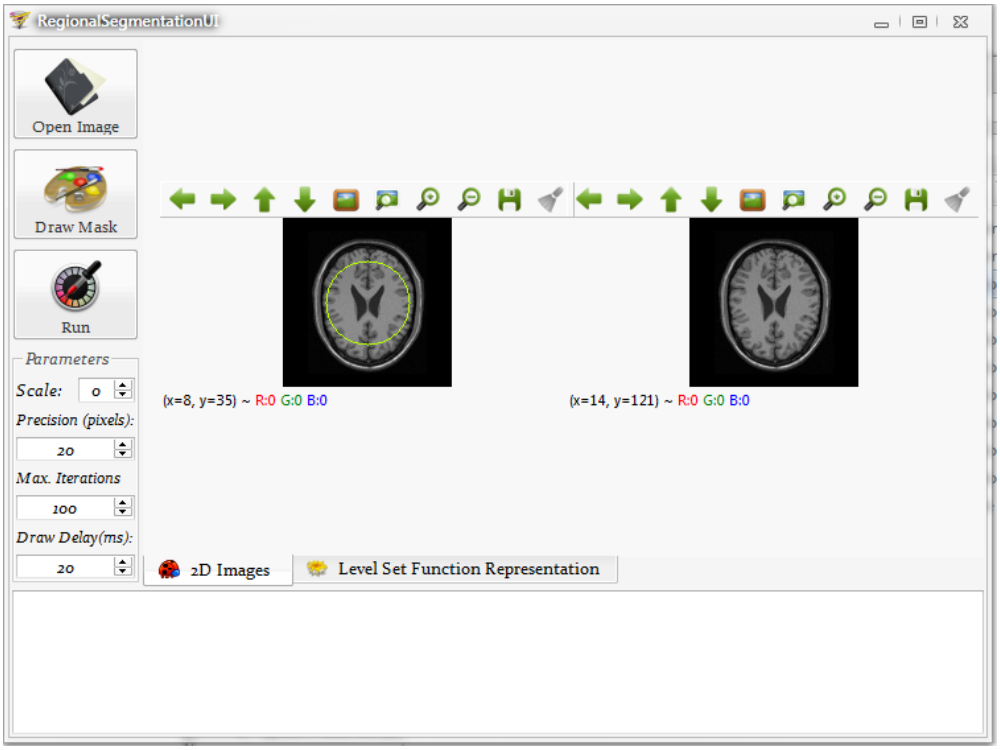
A megvalósított algoritmus felhasználási útmutatóját olvashatjuk a következőkben. Mint korábban megemlítettük, az algoritmus megvalósításához felhasznált nyelvezet a C++ volt, ezért a programcsomag felhasználása is ebben a környezetben a legegyszerűbb. De ugyanakkor lehetséges a programcsomagot .Net környezetből is felhasználni, vagy egyéb olyan programozási nyelvben, mely képes egy könyvtárat betölteni és annak függvényeit felhasználni. Az algoritmus futatásához a következő lépéseket kell, hogy elvégezzük:

- Az algoritmus a *BSplineLevelSet* osztályba van elhelyezve, amely a *LevelSetSegmentation* névtérbe található. Ezért a névtéren keresztül érhetjük el a függvényeket és az algoritmus osztályát.
- Hozzunk létre egy példányt a *BSplineLevelSet* osztályból. Pl: *BSplineLevelSet l;*
- Az osztály az OpenCV kép objektumaival dolgozik (*cv::Mat*), ilyen struktúrában fogadja el a bemeneti adatait. Olvasunk be egy ilyen objektumba egy képet (legyen neve *I*).
- Hozzunk létre az előbbi pontba beolvasott képpel megegyező fekete (nullaértékű) intenzitás képet (legyen neve *M*) és töltsük fel a maszk helyét (a kezdeti kontúr belsejét) egyes értékekkel
- Adjuk át az algoritmusnak a beolvasott képet az *Im()* függvénnyel: *l.Im() = I;*
- Adjuk át az algoritmusnak a maszkot a *Mask()* függvénnyel: *l.Mask() = M;*
- Állíthatjuk az algoritmus paramétereit
    - Skála: *l.Scale() = 0;*
    - Célpontosság: *l.Precision() = 20;*
    - Maximális iteráció szám: *l.MaxIterationNr() = 100;*
- Futassuk az algoritmust. A *Run()* műveletet addig hajtjuk végre,amíg az hamis értéket nem küld vissza. Például felhasználva egy "while" ciklust: *while(l.Run());*
- Futás alatt az iterációk között a jelenlegi szegmentálási eredményt a *tempMask()* függvénnyel kaphatjuk meg. Itt az előtérhez tartozó elemek 1-el, a többi 0-val van kódolva a visszatérített OpenCV kép objektumon belül.
- Futás után a szegmentálási eredményre felrajzolt kontúrt a *drawContourAndMask (megfordít eredmény)* függvény téríti vissza.
- Egy képre az eredmény kontúrt a *drawContour(mire, megfordít eredmény)* függvénnyel kapjuk meg.

6.2. Grafikus felület

A grafikus felület elkészítésekor az irányelv a könnyű felhasználhatóság és a képernyő lehető legjobb kihasználása volt. Az első felét a nagy képpel ellátható gombok és a futási paraméterek állandó megjelenítése biztosítja. A végtermék formáját ugyanakkor a már korábban

bemutatott komponens diagramok is befolyásolták. Végső formáját az alábbi ábrán szemléltetjük:



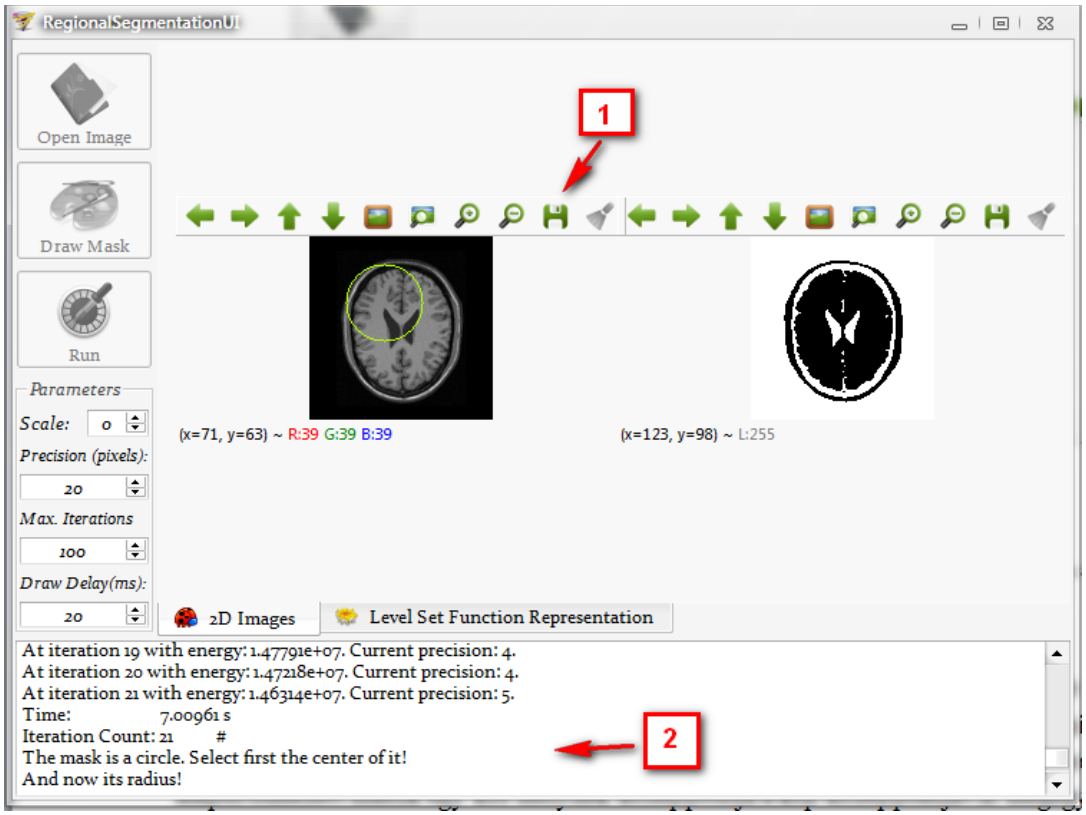
Ábra 24. A grafikus felület

Ha egy feladat a jelenlegi helyzetben nem értelmezett, akkor az a gomb kikapcsolódik (szürke árnyalatúvá válik), hogy elkerüljük a felület helytelen használatát. Emiatt a program indításakor csak a kép beolvasó gomb aktív, azaz használható. A kép beolvasásakor az alapértelmezett maszk egy kör melynek középpontja a kép középpontjával megegyezik és sugara a kép hosszúság és szélessége közüli kisebb érték negyede.

A maszk beállításakor a második fülben a szinthalmaz függvény kezdeti alakja (a maszknak megfelelő elem előjeles távolságfüggvénye) automatikusan kiszámolódik es megjelenítődik. Nagy képeknél ez hosszabb időt is tarthat ezért a felület egy rövid ideig válaszképtelené válhat.

A második gombbal az alapértelmezett maszkot újra rajzolhatjuk egy újabb körrel helyettesítve azt. Miután megnyomtuk a gombot, helyezzük a kurzort (ami egy kereszté alakul át a maszk képen) a kívánt kör középpontjába. Ezután tartsuk lenyomva az egér bal gombját és e ponttól mérve meghatározzuk a kör sugarát. A végleges sugár a bal gomb felengedésekor lesz elmentve, újra számolva a kezdeti szinthalmaz függvényt is.

Ezután módosíthatunk az alapértelmezett értékeken és elindíthatjuk az algoritmust. Míg ez be nem fejeződik, újabb parancsot már nem adhatunk és a gombok és paraméterek inaktív helyzetbe kerülnek. A grafikus felület alsó részében olvashatunk adatokat az algoritmus fejlődéséről, illetve a két fül belsejében a "képes" visszajelzést is figyelemmel követhetjük.



Ábra 25. A grafikus felületműködés közben

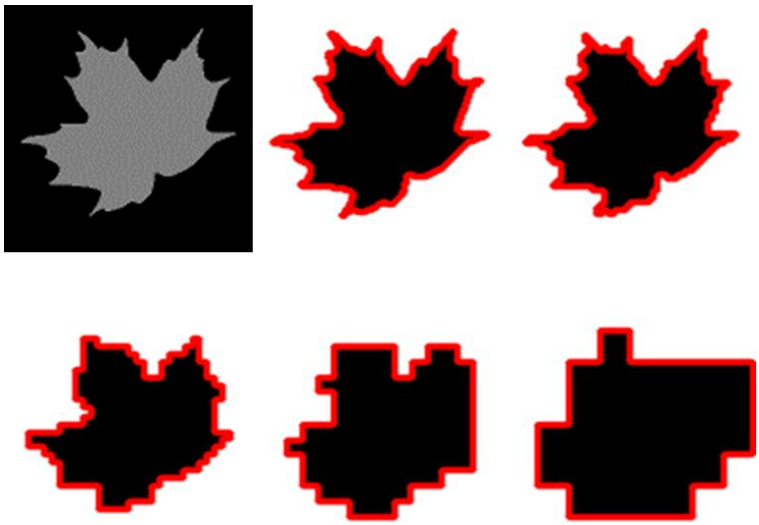
A fenti ábrán az 1 – es számú nyíl által mutatott gomb teszi lehetővé, hogy az épp ablakba levő képet kimentsük a merevlemezre. A gomb megnyomása után egy ablak jelenik meg melyben kiválaszthatjuk a hova, milyen formátumba és milyen név alá kérdésekre a választ. A 2. –es nyíl demonstrálja, a felhasználóval közölt információkat melyek lehetnek úgy az algoritmus eredményéről, teljesítményéről szólóak, mint utasítások annak. Itt a maszk rajzolása közben figyelhetjük meg az interfészt.

7. SZEGMENTÁCIÓS EREDMÉNYEK

A tesztelés során megfigyeltük az algoritmus teljesítményét (futási időt mérve), iterációs lépés számát és ugyanakkor megfigyeltük, hogy a diszkrét B-Spline – nak más-más mértéket véve (0, 2, 4 és 8-as lépésekkel), hogyan befolyásolja a kialakuló szintfüggvényt. A szinthalmaz függvény alapján létrehoztuk a bináris képeket a következő színezési szabállyal: ha a szinthalmaz függvény értéke nagyobb vagy egyenlő, mint zéró akkor 255 és másképp nulla.

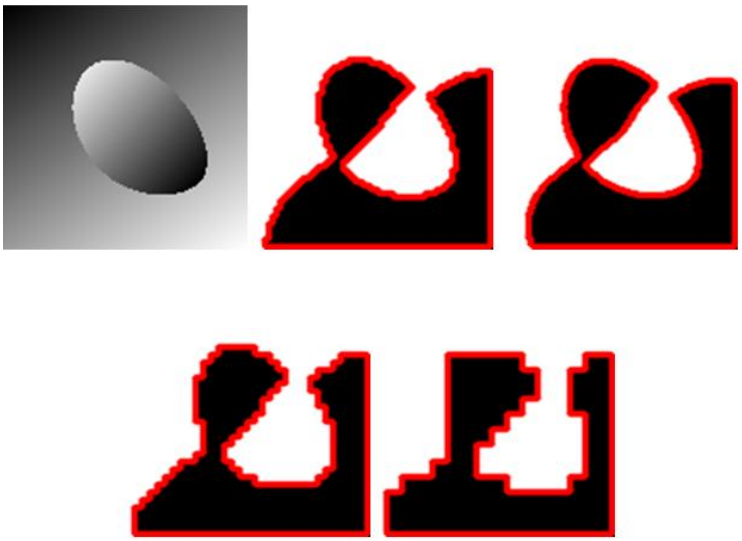
A bináris képeken ezután meghívtuk az OpenCV bináris képen való kontúrkereső függvényét és a kapott eredményt felrajzoltuk a bináris képnek egy színes másolatára. E fejezet következő alegységeiben az olvasó ezek eredményét követheti nyomon. A felhasznált képek nagy részét a CREASEG szegmentációs MATLAB szoftver tesztállományai jelentik, mivel ezek tartalmazzák a szakirodalomban használt számos képanyagot [30]

7.1. Szimulációs képek



Ábra 26. Levél szegmentáció

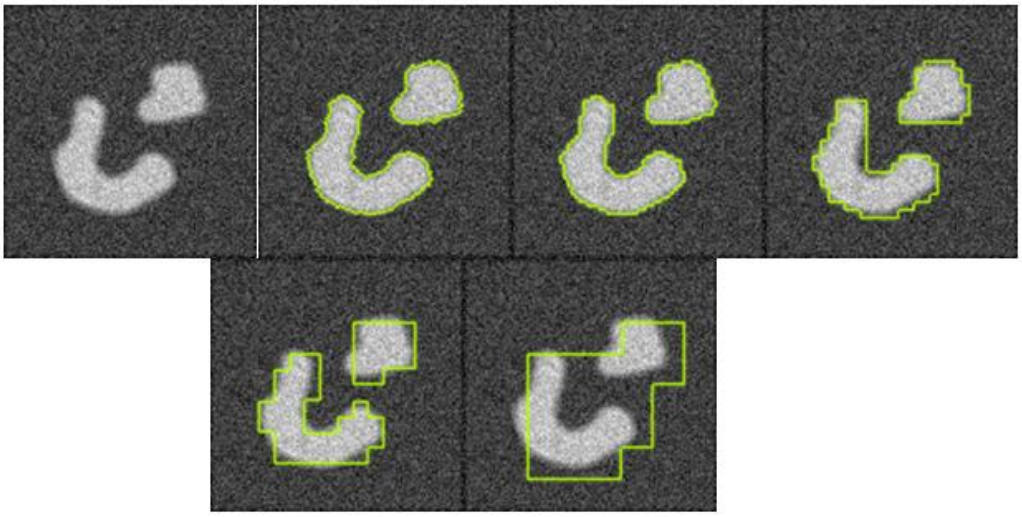
A fenti ábrán egy levél szimulált formáját keressük. A bal felső sarokban található az eredeti kép majd ezt követi rendre az 1, 2, 4, 8 és 16 lépésméretes diszkrét B-Spline által eredményezett szegmentáció. A két felület között a kontúr eltérő színnel van kiemelve. Észrevesszük, hogy a növekedő lépésszám egyre kevésbé képes visszaadni az apró részleteket és durvább határfelületet eredményez.



Ábra 27. Változó intenzitás szegmentáció

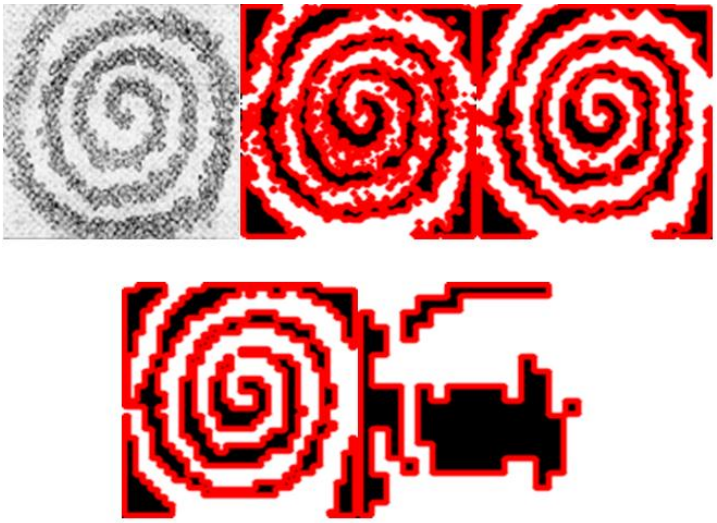
A 27-es ábra egy jó példa arra, hogy a szegmentáció homogén régiókat keresve a képpontokat sötét és világos osztályba sorolva határozza meg, hogy az illető pont a tárgyon belül vagy kívül van. Megfigyelhető, hogy az összes világos képpont a kapott kontúr belsejében található. Ugyanakkor elhagytuk a 16 lépéses szegmentációt mivel ez már annyira nagy lépésekbe halad, hogy ilyen kis képeknél az így vett minták már nem elégségesek egy elfogadható interpoláció elvégzésére. A továbbiakban is hasonlóan fogunk cselekedni.

Most figyeljük meg az alábbi zajos képen a szegmentációt, de most az eredményezett kontúrt egyenesen felrajzolva a bemeneti képre:



Ábra 28. Két objektum szegmentáció

7.2. Való világi képek



Ábra 29. Spirál szegmentáció

A növekvő lépés a zajokat egyre jobban kiszűri, de ha igen magas lépésszámot használunk, a B-Spline már nem rendelkezik elég képi információval és az azonosítás meghiúsul. Ez és korábbi képek egyértelműen bebizonyítják, hogy a diszkrét B-Spline csomópontjai közötti lépes a kontúr simaságát befolyásolja és használható a határfelület által eredményezett megkötés helyett, ahogyan ezt Bernard kísérleti eredményei is igazolták [12].

Figyeljünk meg egy sas és egy repülő szegmentációját csupán a kontúr megjelenítésével:

Ábra 30. Sas szegmentálása

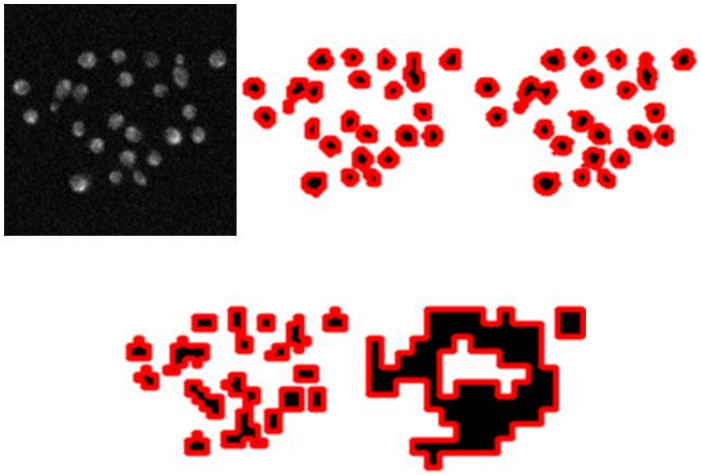
Ábra 31. Repülő szegmentálása

A következő képek az Osló-i Vigeland szoborparkban készültek egy felhős napon. Algoritmusunk a sötét szobrokat jól elválassza a világosabb háttértől.

Ábra 32. Futó szobrok

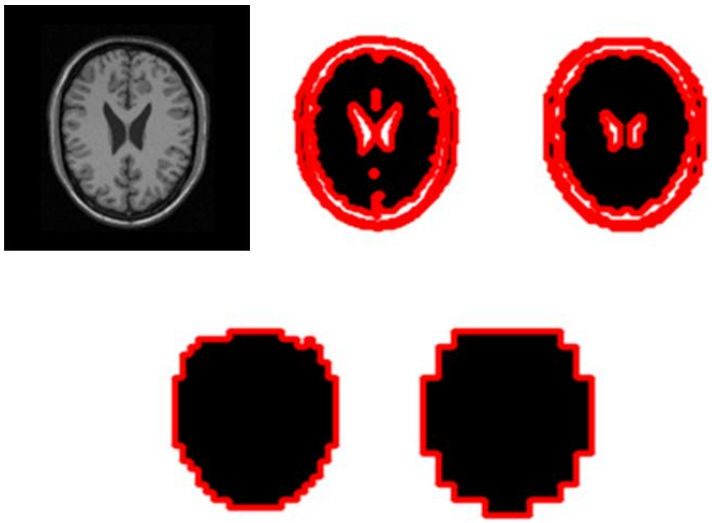
Ábra 33. Ölelkező körszobor

7.3. Orvosi képek



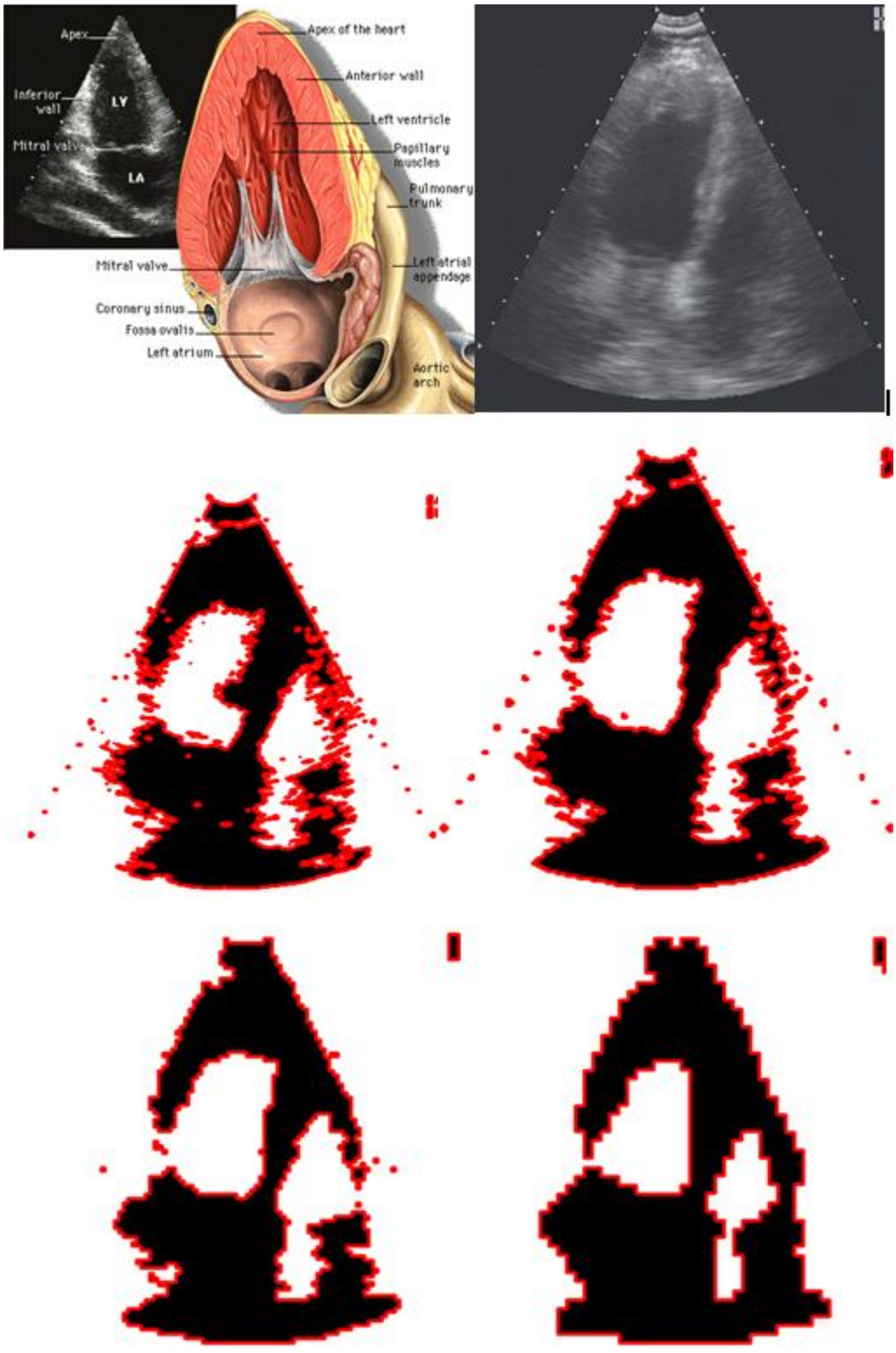
Ábra 34. Fluoreszkáló sejt szegmentáció

A 34-es ábrán egy mikroszkóp fluoreszkált felvételét láthatjuk. Az érdekelt sejtek a kezelés hatására a háttérhez képest világosabbak lesznek. E képen a sejtek megkeresése könnyen elvégezhető programunkkal, hiszen az topológiailag rugalmas és képes a terjeszkedés során osztódni. Tehát a kezdeti határfelületen kívül újak létrehozása nem jelent gondot.



Ábra 35. Agy MRI szegmentáció

Az MRI felvétel érdekessége, hogy a lépés növelésével a központi fekete régiót zajként van kezelve és a 4-es lépésmérték esetén már ki is szűri azt. Ezáltal csak a koponya keretével maradunk, mint a szegmentáció eredménye.

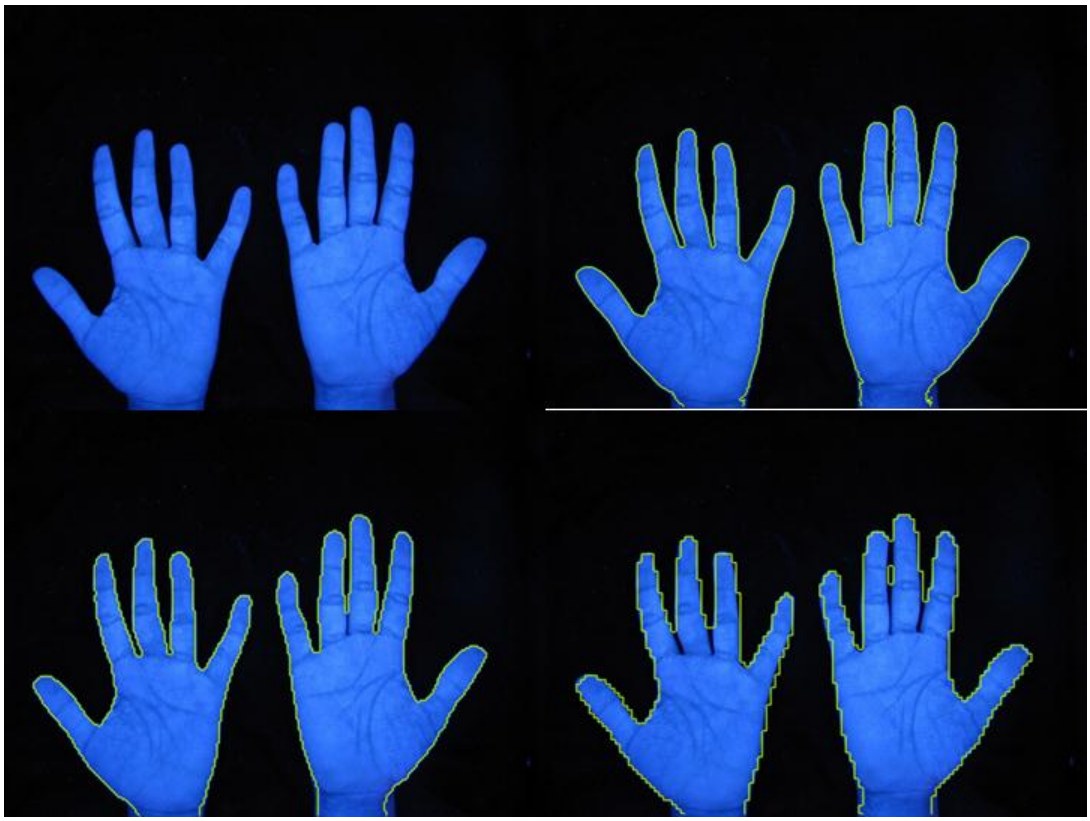


Ábra 36. Echókardiográf felvétel szegmentáció

Az echókardiográf felvételek több nézetből készíthetők. Az itt szegmentált kép egy csúcsi kétkamrás nézet. E nézet fő jellemzőit láthatjuk a fenti ábra bal felső sarkában. A képen a két sötétebb régió megfelel a bal kamrának és a bal pitvarnak (ilyen nagysági sorrendben). A szegmentációra használt felvétel a jobb felső sarokban található. Ezek után megfigyelhetjük a pitvar és a kamra azonosítását a diszkrét B-Spline különböző lépései esetén. Az elért pontosság

elég meggyőző, de ez akár tovább javítható, amennyiben hajlandók vagyunk hosszabb ideig futatni az algoritmust.

A következő kép esetében az eredmény helyességének könnyebb megvizsgálása érdekében eltekintünk a szegmentációs képtől és a szegmentálás eredményét (a kontúrt) direkt a feldolgozott képre rajzoljuk. Itt egy kézről láthatunk egy képet ahol a kutatók az emberi kézmosás teljeségét vizsgálták. A nem megmosott területek sötétebb kék színnel jelennek meg. Láthatjuk, hogy itt algoritmusunk nagyon jól alkalmazható a kéz megkeresésére:



Ábra 37. Kezek detektálása

7.4. Futásidő és iteráció szám

E tesztekre az algoritmust addig engedtük dolgozni, amíg öt iteráción keresztül a képpontok számának kevesebb, mint 10 százalékát változtatta meg előtér/háttér pontból az ellenkező csoportban. Ez megegyezik a határfelület terjeszkedése által történő változásnak. Minden képre az algoritmust háromszor végeztük el és a táblázatba e három időnek az átlaga van feltüntetve másodpercben mérve. A táblázatba az *I.Sz.* Iteráció számot jelöl. A tesztelésre felhasznált rendszer egy Intel P8700-as processzoron történt, 4GB DDR2 memóriával egy Microsoft Windows 7-es operációs rendszerbe, ahol a végrehajtható állomány a Microsoft Visual Studio 2010-el készítettük el.

Táblázat 4. A szegmentálás teljesítménye

Kép (méret)	Diszkrét B-Spline pontok közötti lépés							
	1		*2*		*4*		*8*	
	Idő(s)	*I.Sz.*	*Idő(s)*	*I.Sz.*	*Idő(s)*	*I.Sz.*	*Idő(s)*	*I.Sz.*
Levél (128×128)	0.240000	7	0.225003	7	0.240847	8	0.210587	7
Változó (128×128)	0.360040	12	0.265840	10	0.271019	10	0.250874	7
Spirál (128×128)	0.647694	24	0.520128	16	0.304954	11	0.201891	7
Sejtek (128×128)	0.436916	15	0.273539	10	0.252977	9	0.297395	11
Agy MRI(128×128)	0.247314	8	0.200423	7	0.167227	6	0.180082	6
Echókardiográf (387×387)	4.001256	15	3.128002	11	2.209847	8	1.942836	7

A táblázat alapján összesítésként elmondható, hogy a lépésszám növekedésével csökken a futási idő és a szükséges iteráció szám, hogy az EM algoritmus egy stabil energia értéket találjon a minimalizálás során. Az algoritmus futási ideje ugyanakkor elfogadható figyelembe véve, hogy egyelőre még nincs párhuzamosítva és a batch műveleteket szekvenciálisan hajtódnak végre egyetlen processzor magon.

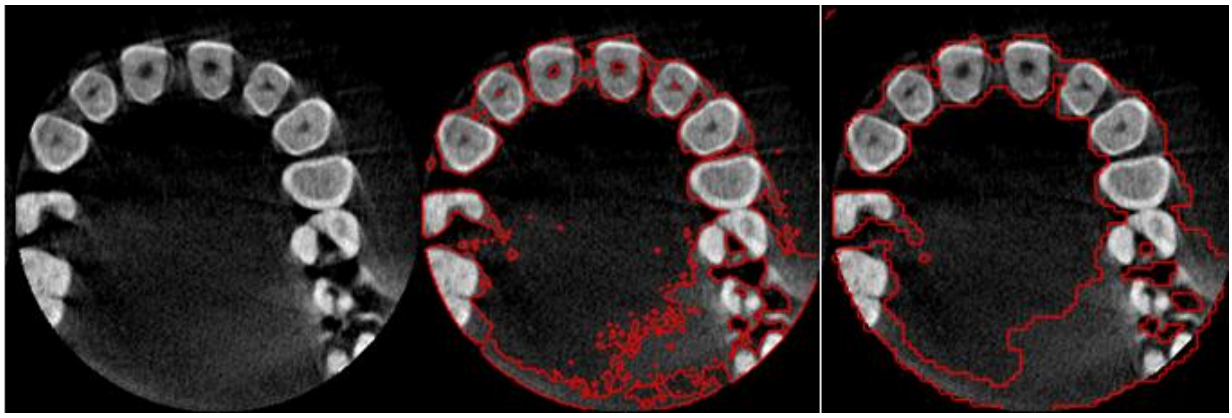
7.5. Az algoritmus határai

Ha eddig azzal foglalkoztunk, hogy milyen körülmények között működik megbízhatóan az algoritmus, most figyeljük meg annak jelenlegi formájában a határait is. Először is mi is a legkisebb még objektumnak detektált felület melyet képes szegmentálni. A válasz, hogy környezetfüggő, és teljes mértékben meghatározza az algoritmussal éppen használt skála érték. A természetben is fontos, hogy milyen skálával dolgozunk, ugyanis ez határozza meg, hogy mit tekintünk még fontosnak és mi az, ami már elhanyagolható.

Például ha egy tisztásról figyelünk egy erdőt, akkor mi a fontos abból, amit látunk? Ha a mértékegységünk centiméter, akkor a fa levele, ha viszont méter, akkor a fa meghatározó, ha

meg a kilométer, akkor minden bizonnyal az egész erdő a legkisebb egység. Az itt bemutatott szinthalmaz szegmentáló algoritmus érzékenységét a használt skála értéke hasonlóan befolyásolja. Annak növekedésével egyre inkább eldobja a részleteket és csak a nagy elemeket veszi észre, azt is pontatlanul, hiszen nem a levél érdekli őt, hanem csak maga a fa létezése.

Továbbá azt, hogy egy pont az előtérhez vagy a háttérhez kerül, nagymértékben meghatározza, hogy a környezetében levő elemek többsége mely részhez tartozik. Erre egy jó példa az alábbi szegmentáció ahol a fog CT egyik fele (alsó) annyira zajos, hogy a környezetben levő háttér képpontok kisebbségbe kerülnek és ez által az objektumhoz lesznek szegmentálva:



Ábra 38. Fog CT hibás szegmentálása

Ugyanakkor a szobrok szegmentációjánál az erős nap visszaverődhet a fényes felületen, ez által létrehozva egy olyan világos régiót, amely megint intenzitásban inkább közelebb van a háttér átlag intenzitásához mintsem az objektumhoz. Ennek példáját az alábbi ábra mutatja be:

Ábra 39. Repülő hattyúk pontatlan szegmentációja

Ezen hibák kiiktatására megoldás lehetne, ha az objektum és a háttér átlagintenzitására megkötéseket tennénk, és nem engednénk, hogy azok az algoritmus során egy adott érték alá vagy felé változzon.

8. KÖVETKEZTETÉS

A dolgozat főtémáját adó regionális szegmentációs algoritmust sikeresen megvalósítottam a C++ nyelvezetben, objektum orientált ideológiákat követve. Továbbá annak működését leteszteltem különböző környezetű és minőségű kép esetében. Megfigyelhető volt, hogy az algoritmus gyorsan és könnyedén detektál alakzatokat, amelyeknek előzetesen nem ismerjük mértani formáját. Még akkor is igaz ez, ha ez nem egyetlen objektumot alkot, hanem több kis tárgy formájában található szétszóródva a képen, ahogy a fluoreszkált sejtek esetében látható.

A variációs modell, melyre építünk, társítva a B-Spline interpolációs szintézisfüggvénnyel lehetővé teszi, hogy a határfelületet globális szinten terjesszük tovább az EM algoritmus iterációi során. Az energia függvény minimalizálását a szinthalmaz felületet leíró B-Spline együtthatókra nézve a változó léptékű, első fokú gradiens módszer segítségével oldottuk meg.

Az alkalmazott módszer előnye, hogy amikor a szinthalmaz függvényünk közel van a megoldáshoz, csupán pár iteráció szükséges, hogy a képhez rendelt energia függvény minimális legyen. Ez hasznos lehet egy forma követés feladatában, hiszen két képkocka között többnyire kis különbség merül fel, amelyet az algoritmusunk hamar korrigálni tud. Ugyanakkor az alkalmazott módszer eredménye egy szinthalmaz függvény lesz. Ez jól leírja, hogy egy-egy képpont milyen mértékben tartozik az energia kritérium függvény által leírt tárgyhoz, háttérhez vagy amennyiben közel van a nulla szinthez, magához a tárgy határfelületéhez.

A zajos rendszerek esetében, vagy ha csökkenteni akarjuk a cél határfelület simaságát (akár zajkiszűrés miatt) jól alkalmazható a B-Spline diszkrét alakjának skálázása. A B-Spline csomópontok közti távolság növelésével a kapott határfelület egyre durvább lesz, ahogy ezt megfigyelhettük a tesztelés során. Láthattuk azon körülményeket is melyek e szegmentálási algoritmus helytelen vagy nem elég pontos működéséhez vezethet.

Olivier Bernard a cikk írásakor megírt implementációja a szegmentált levél esetében megközelítőleg 2 másodperc alatt futott le egy 1.4 GHz-es Intel processzor magon párosítva 1GB memóriával. Ezzel szembe a mi implementációnk, habár igaz, hogy szinte egy kétszer erősebb gépen fut, de mégis maximális futási ideje 0,25 másodperc alatt marad.

A rendszer továbbfejlesztési lehetőségei közé tartozik ennek implementációja és adaptációja egy képsorozat anyagra ahol egy bizonyos tárgyat próbálunk időben követni. Ilyen például egy echókardiográf felvételen a bal szívkamra azonosítása és követése. Ennek sikere után fontos információ lenne az orvos számára, ha ennek a kerületét és területére vonatkozó információkat kiszámolnánk és esetleg egy grafikonon megjelenítenénk valós időben. Ennek segítségével könnyedén detektálható például az esetleges szívkamra zavarok jelenléte és azok mértéke egy beteg echókardiográf felvételén.

Továbbá az algoritmus sok-sok helyen használ úgynevezett batch műveleteket. Ezek könnyen párhuzamosíthatóak és lehetőséget adnának az algoritmus valós idejű alkalmazására (esetleg kis holtidő jelenlétében). Első lépésben ez történhet a központi processzor magra az OpenMP segítségével majd egy még látványosabb gyorsulásért a grafikai processzor magon felhasználva az nVIDIA CUDA architektúráját, avagy az OpenCL keretet.

Mint az előbbi paragrafusokban láthattuk, e dolgozatban bemutatott algoritmus annak ellenére, hogy még gyerekcipőben jár, nagyon ígéretes. Kevesebb, mint két éve alkották meg és alkalmazási lehetőségei, a területek ahol bevethető, csak a fejlesztő fantáziáján múlnak, hiszen egy topológiailag rugalmas és nagyon párhuzamosítható módszert ad homogén régiók szegmentálására. A modern orvosi tudományban rengeteg ilyen helyzet merül fel, legyen szó echókardiográf, röntgen, mikroszkóp, avagy 2D, esetleg 3D MRI kép és video felvételről.

9. IRODALOMJEGYZÉK

[1] James C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. MA, USA: Kluwer Academic Publishers Norwell, 1981.

[2] John Canny, "A Computational Approach to Edge Detection," *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, November 1986.

[3] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, September 1981.

[4] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image processing, analysis, and machine vision*, 1st ed. London, UK: Chapman & Hall Computing, 1993.

[5] D. Mumford and J. Shah, "Boundary Detection by Minimizing Functionals," *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 22-26, 1985.

[6] D. Mumford and J. Shah, "Optimal Approximations of Piecewise Smooth Functions and Associated," *Communications on Pure and Applied Mathematics*, vol. 42, pp. 577-685, 1989.

[7] Gilles Aubert, Michel Barlaud, Olivier Faugeras, and Stéphanie Jehan-Besson, "Image segmentation using active contours: calculus of variations or shape gradients?," *SIAM Appl. Math.*, vol. 63, no. 6, pp. 2128-2154, 2003.

[8] Tony F. Chan and Luminita A. Vese, "Active Contours Without Edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266-277, February 2001.

[9] Thomas Pock, Daniel Cremers, Bischof Horst, and Antonin Chambolle, "An algorithm for minimizing the Mumford-Shah functional," *Proceedings of ICCV*, pp. 1133-1140 , 2009.

[10] Michael Kass, Andrew Witkin, and Demetri Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321-31, 1988.

[11] Stanley Osher and Ron Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. New York, USA: Springer-Verlag, 2002.

[12] Olivier Bernard, Denis Friboulet, Philippe Thévenaz, and Michael Unser, "Variational B-Spline Level-Set: A Linear Filtering Approach for Fast Deformable Model Evolution," *IEEE Transcations On Image Processing*, vol. 18, no. 6, pp. 1179-1191, June 2009.

[13] Michael Unser, Akram Aldroubi, and Murray Eden, "B-Spline Signal Processing: Part I - Theory," *IEEE Transactions On Sigal Processing*, vol. 41, no. 2, pp. 821-832, February 1993.

[14] Michael Unser, Akram Aldroubi, and Murray Eden, "B-Spline Signal Processing: Part II - Efficient Design and Applications," *IEEE Transactions On Signal Processing*, vol. 41, no. 2, pp. 834-848, February 1993.

[15] Michael Unser, "Splines: A perfect fit for signal/image processing," Swiss Federal Institute of Technology, Lausanne, to appear in IEEE Signal Processing Magazine August 16, 1999.

[16] krischnamurti@users.sourceforge.net. (2007, June) QwtPlot3D. [Online]. http://qwtplot3d.sourceforge.net/

[17] Xavier Bresson, "Image segmentation with variational active contours," École polytechnique fédérale de Lausanne, Lausanne, Phd Thesis no. 3283, 2005.

[18] V. Cassels, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61-79, 1997.

[19] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A.J. Yezzi, "Gradient Flows and Geometric Active Contour Models," *International Conference on Computer Vision*, pp. 810-815, 1995.

[20] S. Osher and J.A. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *ournal of Computational Physics*, vol. 79, no. 1, pp. 12-49, 1988.

[21] S. C Zhu and A. Yuille, "Region Competition: Unifying Snakes, Region Growing, and

Bayes/MDL for Multiband Image Segmentation," *IEEE Transactions on Pattern Analysis*, vol. 18, no. 9, pp. 884-900, 1996.

[22] M. Leventon, "Statistical Models for Medical Image Analysis," PhD Thesis 2000.

[23] T. Cootes and C. Taylor, "Statistical Models of Appearance For Computer Vision," University of Manchester, Manchester, Technical Report 1999.

[24] Philippe Thévenaz and Michael Unser, "Interpolation Revisited," *IEEE Transactions on medical imaging*, vol. 19, no. 7, pp. 739-758, July 2000.

[25] Philippe Thévenaz and Michael Unser, *Image Interpolation and Resampling*. Orlando, Florida, USA: Academic Press, 2000, ISBN:0-12-077790-8.

[26] M. L. Liou, "Spline Fit Made Easy," *IEEE Transactions on Computers*, vol. C-25, pp. 522–527, May 1976.

[27] Philippe Thévenaz. (2009, January 5) Spline Interpolation. [Online]. http://bigwww.epfl.ch/thevenaz/interpolation/

[28] Michael Unser, "Splines: A perfect fit for signal and image processing," *IEEE Transactions on Signal Processing* , vol. 10, no. 2, pp. 22-38, November 1999.

[29] Olivier Bernard, "Segmentation in echocardiographic imaging using parametric level set model," Institut National des Sciences Appliquees, Lyon, PhD Thesis in Computer Science 2006-ISAL-0096, 2006 December.

[30] Olivier Bernard. (2010) CREASEG: a level-set platform. [Online]. http://www.creatis.insa-lyon.fr/~bernard/creaseg/

10. FÜGGELÉK

Itt található meg a forráskódot és dokumentációt tartalmazó CD merevlemez beragasztva:

UNIVERSITATEA TEHNICA CLUJ-NAPOCA
FACULTATEA DE AUTOMATIZARE SI CALCULATOARE
SECTIA CALCULATOARE

Vizat decan

Vizat şef catedră