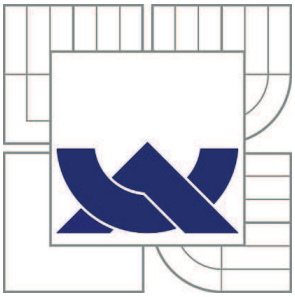
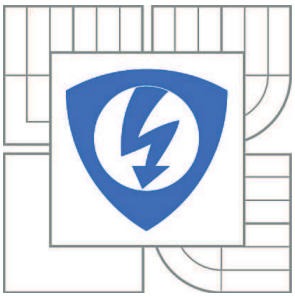


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

METODY SEGMENTACE BIOMEDICÍNSKÝCH OBRAZOVÝCH SIGNÁLŮ V JAVĚ

METHODS FOR BIOMEDICAL IMAGE SIGNAL SEGMENTATION IN JAVA

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

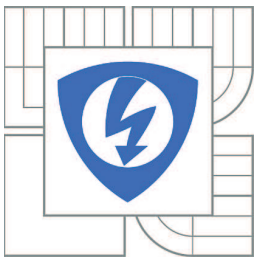
AUTOR PRÁCE — Bc. JAKUB ROMÁNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ ŠMIRG

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jakub Románek	***ID:***	98563
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Metody segmentace biomedicínských obrazových signálů v Javě

POKYNY PRO VYPRACOVÁNÍ:

Student vytvoří java modul pro segmentaci biomedicínských obrazů pomocí metody Level Set. Využije k tomu pouze OpenSource knihovny. Může se inspirovat v Java pluginu projektu Fiji, ale výsledný modul bude univerzální třídou použitelnou pro obecné projekty se vstupem ve formátu ImagePlus. Bude vytvořeno jednoduché GUI pro ukázku výsledků metody.

DOPORUČENÁ LITERATURA:

[1] NIXON, Mark; AGUADO, Alberto. Feature Extraction & Image Processing. Second edition. Great Britain : Academic press, 2008. 406 s. ISBN 978-0-1237-2538-7.
[2] PARKER, J.R. Algorithms for image processing and computer vision. USA : WILEY COMPUTER PUBLISHING, 1997. 416 s. ISBN 0-471-14056-2.
[3] Osher, Stanley J.; Fedkiw, Ronald P. (2002). Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag. ISBN 0-387-95482-1.

Termín zadání:	6.2.2012	***Termín odevzdání:***	24.5.2012

Vedoucí práce: Ing. Ondřej Šmirg
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se skládá ze dvou hlavních částí, části teoretické a realizační. V teoretické části jsou popsány jednotlivé segmentační metody. Zejména jde o metodu Level Set. V realizační části bylo za úkol vytvořit java modul pro segmentaci biomedicínských obrazů pomocí metody Level set. Práce řeší jednoduchého Gui pro ukázku dosažených výsledků.

KLÍČOVÁ SLOVA

Level set, Java, ImageJ, Segmentace, Java Swing, Eclipse

ABSTRACT

This thesis contains two main parts, theoretical and implementation. In the theoretical part there are described the different segmentation methods. Mainly it is about description method Level Set. The aim of practical part was to create a java module for segmentation of biomedical images using Level Set methods. The work solves example of GUI for the display of results.

KEYWORDS

Level set, Java, ImageJ, Segmentation, Java Swing, Eclipse

Bibliografická citace mé práce

Románek, Jakub. Metody segmentace biomedicínských obrazových signálů v Javě. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2012. 52 s., 1 s. příloh. Diplomové práce. Vedoucí práce: Ing. Ondřej Šmirg.

PROHLÁŠENÍ

Jako autor diplomové práce na téma „Metody segmentace biomedicínských obrazových signálu v Jave" dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorskápráva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorskýchpráv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 anásledujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.
40/2009 Sb.

V Brně dne 24.5. 2012 .....

(podpis autora)

PODĚKOVÁNÍ

Děkuji konzultantovi diplomové práce Ing. Ondřeni Šmirgovi za účinnou metodickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 24.5.2012 .....

(podpis autora)

OBSAH

\

SEZNAM OBRÁZKŮ

ÚVOD

Segmentace obrazu je jedním z nejdůležitějších kroků automatického zpracování obrazu. Jde tedy o metodu, nebo spíš o skupinu metod postavených na různých principech digitálního zpracování obrazu, které se využívají k rozdělení daného obrazu na oblasti se společnými vlastnostmi, které mají smysluplný význam.

Segmentace se například využívá ve zpracování lékařských obrazových dat (Medical Imaging) nebo počítačovém vidění. V medicíně se segmentace používá pro zdůraznění odlišných částí v obraze. Například u rentgenového snímku dokážeme pomocí segmentace určit tvrdé a měkké tkáně nebo dutiny. Kosti jsou na snímcích zobrazeny bílou a světle šedou barvou, svaly šedou až tmavě šedou barvou a tělní dutiny jsou zobrazeny téměř černě. V praxi je snaha pomocí těchto metod rozšiřovat teoretické základy medicíny a tím co nejúčinněji pomáhat člověku.

V posledních letech se pro účely segmentace často využívá přístup označován jako aktivní modely, které jsou založeny na minimalizaci vhodně zvoleného energetického funkcionálu [1]. Pro samotnou výpočetní realizaci implicitních aktivních modelů se v současnosti využívá metoda zvaná Level Set, která představuje robustní přístup založený na řešení parciálních diferenciálních rovnic. Hlavní negativní stránkou je její vysoká výpočetní náročnost, která tuto metodu činí prakticky nepoužitelnou při segmentaci obrazu v reálném čase.

Cílem diplomové práce bylo vytvořit java modul pro segmentaci biomedicínských obrazových signálů pomocí metody Level Set. K tomuto účelu měly být využity pouze OpenSource knihovny. Výsledný modul měl být univerzální třídou použitelnou pro obecné projekty se vstupem ve fromátu ImagePlus. Práce by měla obsahova jednoduché GUI pro ukázku dosažených výsledků.

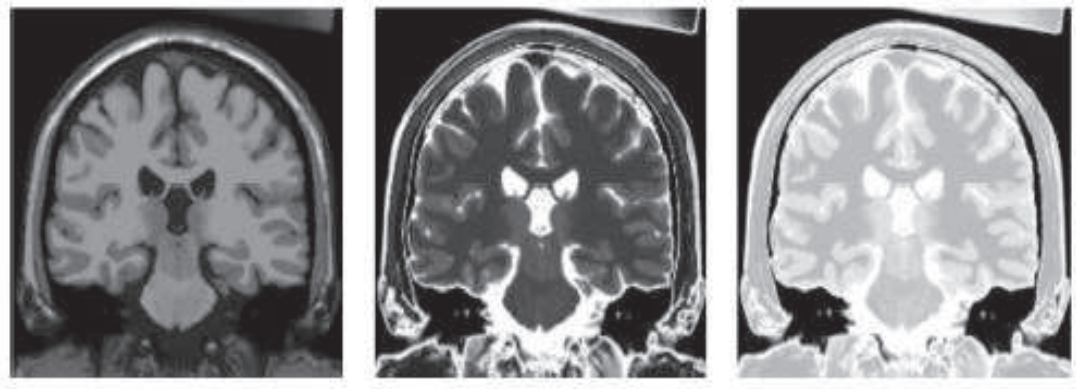
1 MAGNETICKÁ REZONANCE

Magnetická rezonance (MRI) z anglického jazyka „magnetic response imaging“ je neinvazivní zobrazovací technika používající se v lékařství k zobrazení vnitřních částí lidského těla. Využívá elektromagnetické vlnění v radiofrekvenčním spektru (3-100 MHz)[2]. Dosahuje mnohem detailnějších informací o struktuře lidských tkání oproti jiným zobrazovacím metodám. Jde o metodu, která nemá negativní účinky na organismus jako je tomu například u rentgenového záření, kde se jedná o formu ionizujícího záření.

MRI využívá magnetická pole k získání signálu od rezonujících jader. Nejčastěji jsou jimi jádra vodíku, méně často jádra jiných prvků. Signály od protonů v molekulách vody u MRI jsou ovlivněny kromě koncentrace molekul vody i jejími chemickými vazbami, pohybem molekul a průtokem. Reprezentují tedy nejen fyzikální, ale i chemické vlastnosti snímané scény. [2,3]

Při pořizování obrazu dochází k modulaci (váhování) obrazu následujícími parametry: hustotou protonových jader (vzniká tzv. PD-weighted image), relaxační dobou T1, dávající vznik T1-váhovanému obrazu (T1-weighted image), relaxační dobou T2, jež vede k vytvoření T2-váhovaného obrazu (T2-weighted image), a průtokem protonů. Ukázka různě váhovaných obrazů je na obr. 1.1 [3].

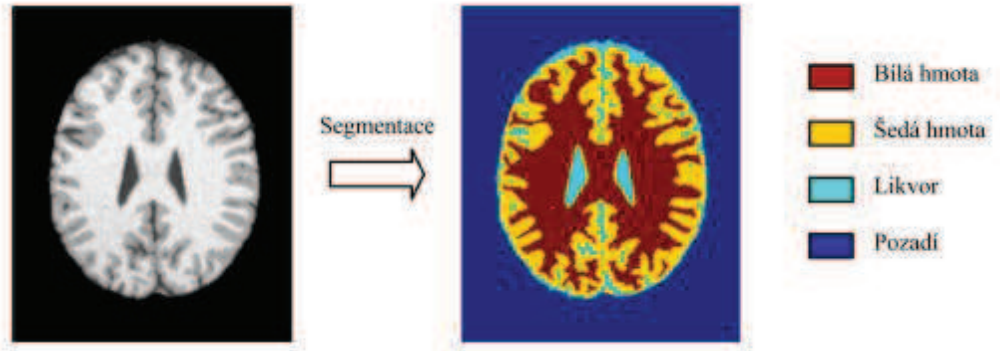
a) b) c)



Obr. 1 Koronární řezy různě váhovaných MRI obrazů a) T1, b) T2, c) PD [3]

2 SEGMENTACE OBRAZU

Segmentace medicínských obrazů představuje proces, při kterém se snažíme oddělit pixely námi zkoumaného objektu v popředí od pixelů spadajících do pozadí obrazu. Princip jakým se to provádí je vidět na obr. 2.1. Je zde znázorněna segmentace mozku z MRI, který představuje rozdělení obrazu mozku na jednotlivé objekty, tedy na šedou hmotu (GM), bílou hmotu (WM), likvor a pozadí.



Obr. 2 Rozdělení mozku pomocí segmentace [3]

Definice: Segmentace obrazu f (x, y) je jeho dělení na jednotlivé objekty R_1, $R_{2,\ldots,}$ R_n tak, že tyto objekty splňují následující kritéria:

$$\cup_{i=0}^{n} R_i = f(x, y) \tag{1}$$

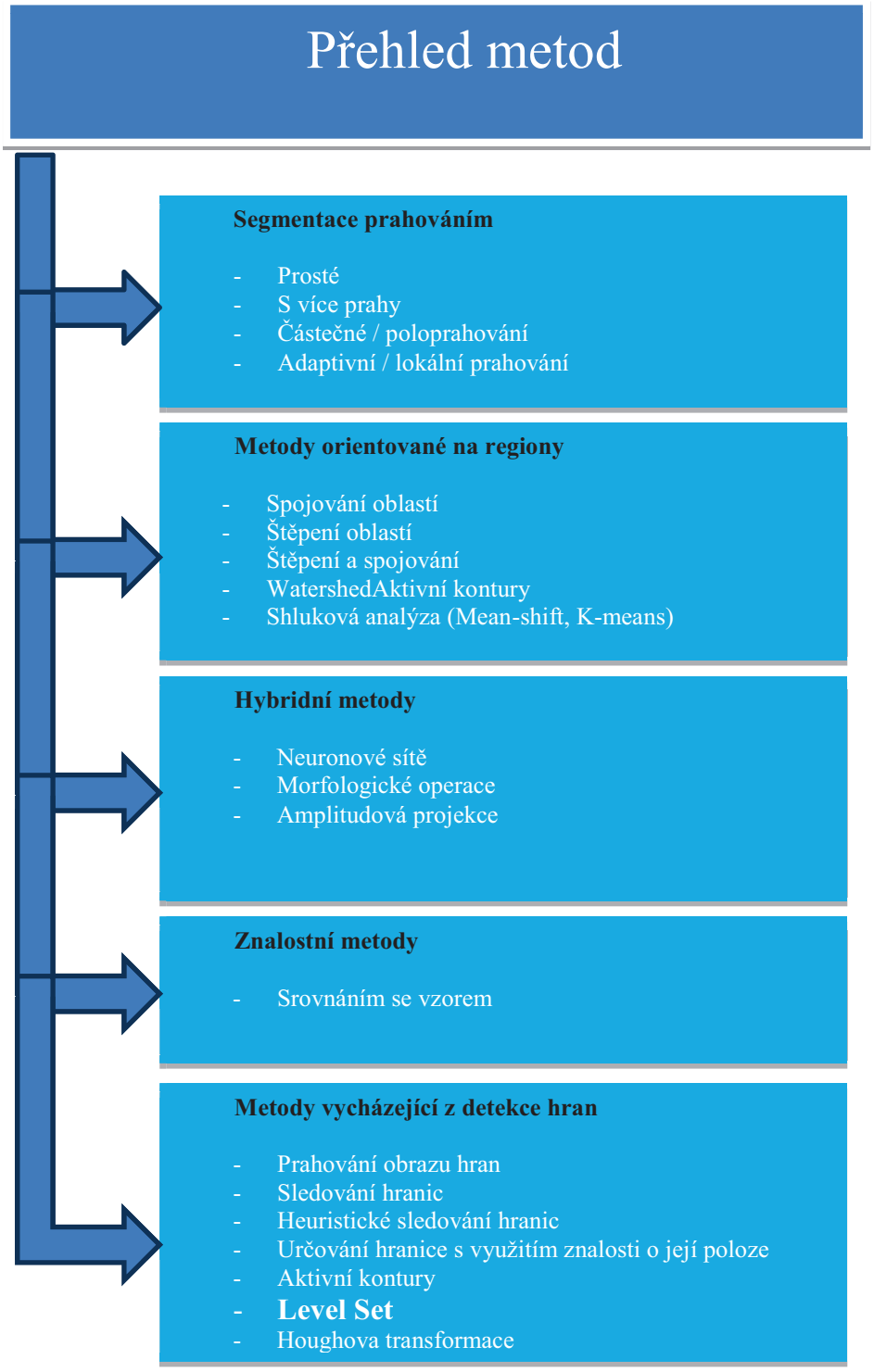
$$R_i \cap R_j = 0,\ i \neq j \tag{2}$$

Každá pod obraz musí splnit jedno z následujících tvrzení, popřípadě množinu tvrzení.

- Všechny pixely v podobraze R_i mají stejnou úroveň šedi.
- Všechny pixely v podobraze R_i se neliší v úrovni šedi více než o předepsanou hodnotu.
- Standardní odchylka úrovni šedi všech pixelů objektu R_i je dostatečně malá.

.

3 SEGMENTAČNÍ METODY



4 AKTIVNÍ KONTURY (SNAKES)

4.1 Základní definice

Aktivní kontury našly největší uplatnění při segmentaci medicínských obrazů, které často bývají poškozeny šumem či vzorkovacím artefakty, které mohou způsobit selhání segmentačních technik, ať už jde například o detekci hran nebo metodu prahování.

Aktivní kontury se často také označují jako „hadi“ neboli snakes. Jde tedy o definovanou křivku uvnitř obrazu, která je deformována pod vlivem obrazových sil, vnějších respektive vnitřních sil. Ukázku definice si můžeme prohlédnout na obr. 4.1. Vnitřní síly dohlížejí nad udržením hladkostí průběhu, obrazové síly se zabývají tvarováním (deformací) kontur směrem ke hraně objektu a vnější (externí) síly jsou výsledkem původního umístění kontury, jsou zadány uživatelem. Můžeme je definovat jako spline s minimální energii, řízený vnějšími omezujícími silami a ovlivněný silami obrazu, které ho táhnou směrem k vyznačeným rysům, jako jsou hrany a vrcholy. První algoritmus aktivních kontur představil v roce 1988 Kass v práci [4].

Aktivní kontura je reprezentována diskrétní sadou bodů dle definice:

$$v_n = [x_n, y_n], \text{ pro } n = 0, 1, \dots N, \tag{3}$$

kde v … je bod křivky,

x …je x-ová souřadnice bodu v,

y … je y-ová souřadnice bodu v.

Nebo může být definována jako parametrická křivka:

$$v(s) = [x(s), y(s)], \; s \in [0,1] \tag{4}$$

Energetická funkce je výsledná pozice aktivní kontury dle definice: $\mathrm{E_{snake}} =$

$$= \int_0^1 \mathrm{E_{snake}}(\mathrm{v}(s))\mathrm{ds} = \int_0^1 [\mathrm{E_{internal}}(\mathrm{v}(s)) + \mathrm{E}_{image}(\mathrm{v}(s)) + \mathrm{E_{con}}(\mathrm{v}(s))]\mathrm{ds} \quad (5)$$

- v(s) je parametrická křivka,
- s je délka křivky,
- $\mathrm{E_{internal}}$ je vnitřní energie křivky,
- $\mathrm{E_{image}}$ je vnější energie křivky,
- $\mathrm{E_{con}}$ je vnější omezení jako aproximaci křivky její druhou derivací.

$\mathrm{E_{internal}}$ je vnitřní energie, vzhledem k zakřivení křivky v první a druhé derivace. Jde o součet elastické energie $\mathrm{E_{elastic}}$ a ohýbací energie $\mathrm{E_{bending}}$,

$$\mathrm{E}_{internal} = \frac{1}{2}\int_s (\alpha(s)|\mathrm{v}_s|^2 + \beta(s)|\mathrm{v}_{ss}|^2)\mathrm{ds} \quad (6)$$

Elastická energie je odpovědná za smršťování kontury, kde váhová funkce $\alpha(t)$ definuje hladkost kontury podle vztahu:

$$\mathrm{E}_{elastic} = \frac{1}{2}\int_s \alpha(s)|\mathrm{v}_s|^2\mathrm{ds} \text{ kde } \mathrm{v}_s = \frac{d\,\mathrm{v}(s)}{ds} \quad (7)$$

Ohýbací energie má váhovou funkci $\beta(t)$, která definuje pružnost kontury podle vztahu:

$$\mathrm{E}_{bending} = \frac{1}{2}\int_s \beta(s)|\mathrm{v}_{ss}|^2\mathrm{ds} \quad (8)$$

$\mathrm{E_{image}}$ je vnější energie která se skládá ze tří složek, jejichž energie jsou označovány jako $\mathrm{E_{line}}$, $\mathrm{E_{edge}}$ a $\mathrm{E_{term}}$.

$$\mathrm{E}_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \quad (9)$$

$\mathrm{E_{line}}$ znázorňuje přitažlivost kontury ke světlým, respektivě tmavým částem kontury obrazu podle příslušného váhového faktoru $\mathrm{w_{line}}$.

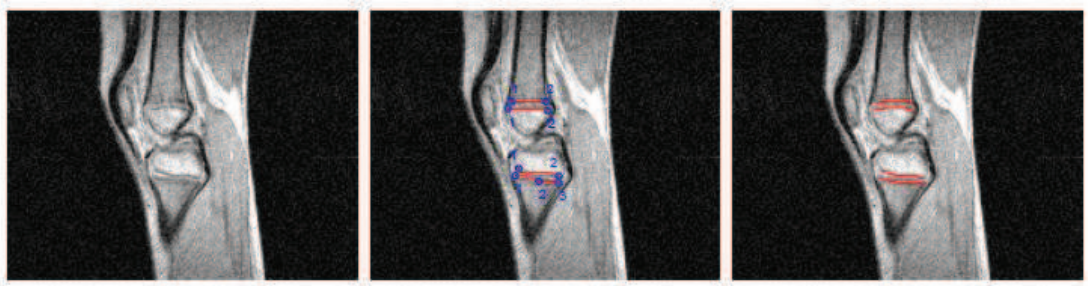
$$\mathrm{E}_{line} = f(\mathrm{v}(s)) \quad (10)$$

$\mathrm{E_{edge}}$ snaží se přitahovat kontury směrem k hranám, tj. k místům s vysokou hodnotou gradientu.

$$\mathrm{E}_{edge} = -|\nabla f(\mathrm{v}(s))|^2 \quad (11)$$

E_{term} spadá mezi nejsložitější složku, která má za úkol detekovat ostré rohy a konce hran pomocí zkoumání křivosti. Daná křivost se počítá na rozmazaném obrazu kvůli předpokládanému šumu. Zpravidla hodnota E_{term} bývá malá, jestliže kontura se pohy buje po rovné, hladké hraně.

E_{con} spadá mezi externí síly, které jsou definovány uživatelem a mohou znázornit jeho interaktivní požadavky.



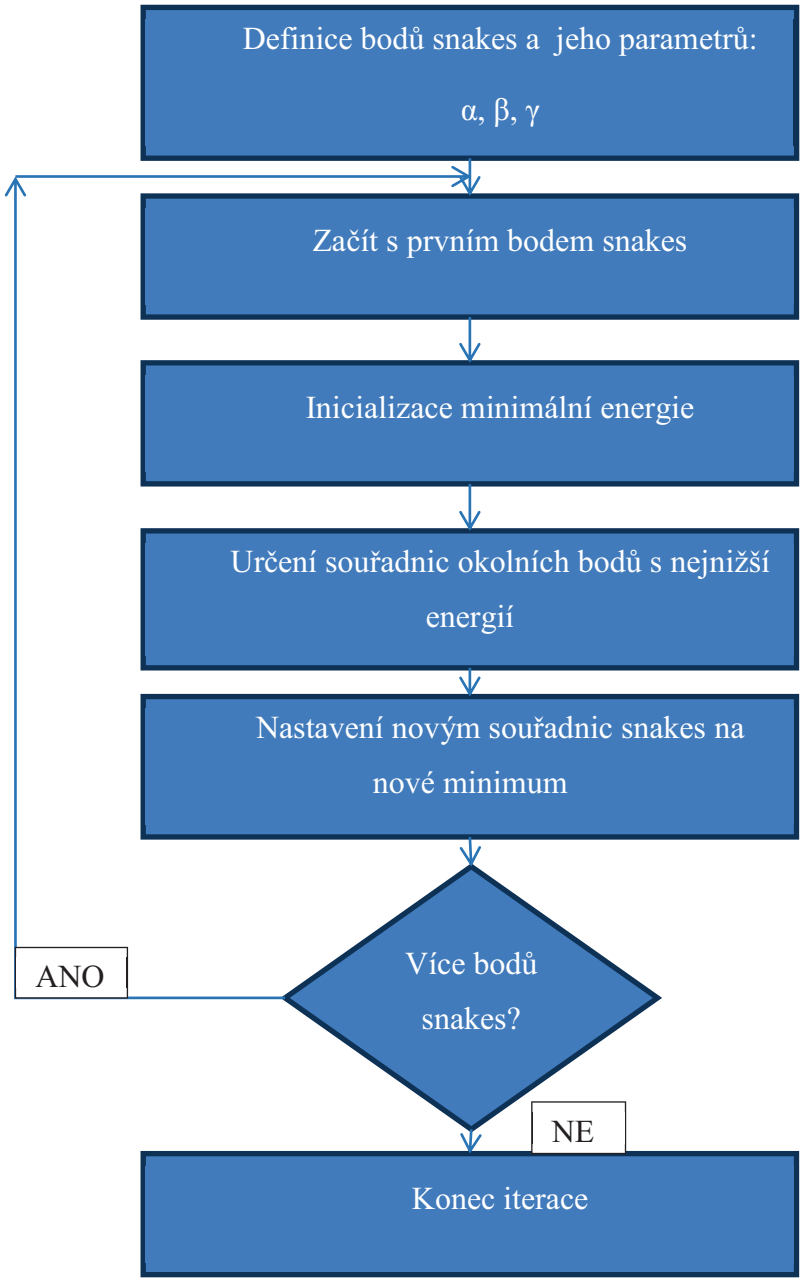
(a) (b) (c)

Obr. 3 Obázek ploténky na MR obrazech. a) MR obraz b) definice aktivni kontury c) výsledek aktivni kontury po segmentaci [15]

4.2 Greedyho algoritmus

Greedyho (nenasytný) algoritmus představil Wiliams a Shas 1992. Jejich postup se však lišil od původního Kass et al. snakes a balónkového snakes (viz další kapitola) tím, že počítá pohyb bodů (pixelů) snakes zcela diskrétním způsobem. To znamená, že propočítává pohyb každého pixelu snakes narozdíl od výpočtu celého snakes najednou Kass at. al algoritmu.

Pohyb každého pixelu snakes je vypočítán ze sousedních pixelů (z pohledu na sousední pixely kolem bodů snakes), kde se pixel snakes přesune na danou pozici v sousedství, což minimalizuje energii term E_{term}. Název Greedyho algoritmus je tedy odvozen od způsobu, jakým si snakes vybírá své nové pozice. Průběh Greedyho algoritmu znázorňuje obrázek 4.



Obr. 4 Průběh Greedyho algoritmu

Nastavení bodů snakes vychází z parametrické rovnice (4). Minimální energie pro každý bod snakes můžeme vypočítat podle rovnice:

$$\mathrm{E_{snake}}(s) = \mathrm{E_{internal}}(\mathrm{v}_s) + \mathrm{E_{image}}(\mathrm{v}_s) \tag{12}$$

Přesněji se dá vyjádřit:

$$\mathrm{E}_{snake} = \alpha(s)\left|\frac{\mathrm{d}\mathbf{v}_s}{\mathrm{d}s}\right|^2 + \beta(s)\left|\frac{\mathrm{d}^2\mathbf{v}_s}{ds^2}\right|^2 + \gamma(s)E_{img}(\mathbf{v_s}), \tag{13}$$

kde rozdíly diferenciálu prvního a druhého řádu jsou aproximovány pro každé okolí bodu hledané křivky. Váhové parametry α, β a γ jsou všechny funkce pro křivku. Proto každý bod křivky má příslušné hodnoty pro α, β a γ. Diferenciál prvního řádu je aproximován jako modul z rozdílu mezi průměrnou vzdáleností bodů křivky (jako Euklidova vzdálenost mezi nimi) a Euklidovou vzdálesnosti mezi aktuálně vybraným obrazovým bodem v_s, a dalším bodem křivky. Je zřejmé, že diferenciál prvního řádu podle rovnice (14) klesne na nulu, když je křivka rovnoměrně rozložena dle potřeby.[13]

$$\left|\frac{\mathrm{d}\mathbf{v}_s}{\mathrm{d}_s}\right|^2 = \left|\sum_{i'=0}^{S-1} \frac{\|\mathbf{v}_\mathrm{i} - \mathbf{v}_{\mathrm{i}+1}\|}{S} - \|\mathbf{v}_\mathrm{i} - \mathbf{v}_{\mathrm{i}+1}\|\right| =$$

$$= \left|\sum_{i'=0}^{S-1} \frac{\sqrt{(\boldsymbol{x_i} - \boldsymbol{x_{i+1}})^2 + (\boldsymbol{y_i} - \boldsymbol{y_{i+1}})^2}}{S} - \sqrt{(\boldsymbol{x}_\mathrm{i} - \boldsymbol{x}_{\mathrm{i}+1})^2 + (\boldsymbol{y}_i - \boldsymbol{y}_{\mathrm{i}+1})^2}\right| \qquad (14)$$

Druhá derivace může být vyjádřena jako odhad zakřivení, Mezi předchozím a následujícím bodem křivky $\mathbf{v}_{s+1}$ a $\mathbf{v}_{s-1}$ a bodem v lokálním okolí bodu $\mathbf{v}_s$. Následující rovnice představuje její výpočet.

$$\left|\frac{\mathrm{d}^2\mathbf{v}_s}{\mathrm{d}_s}\right|^2 = (\boldsymbol{x_{i-1}} - 2\boldsymbol{x_i} + \boldsymbol{x_{i+1}})^2 + (\boldsymbol{y_{i-1}} - 2\boldsymbol{y_i} + \boldsymbol{y_{i+1}})^2 \qquad (15)$$

Poslední částí rovnice je enrgie daného obrazu E_{img}, která se vypočíta podle následujcící rovnice

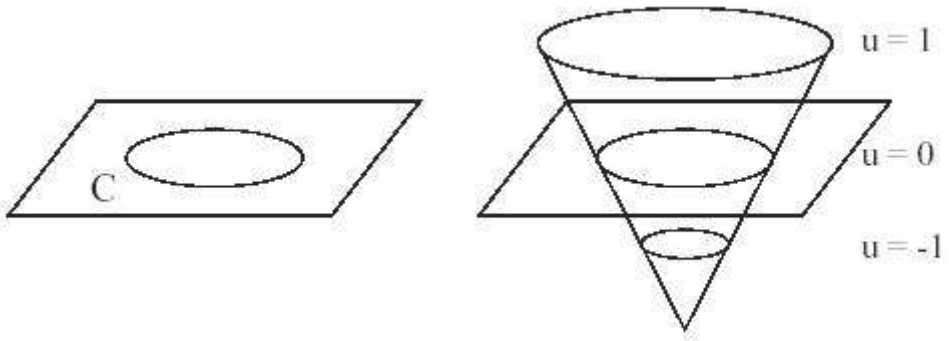
$$E_{img} = -\|\nabla[G_\sigma(x,y) * I(x,y)]\|^2, \qquad (16)$$

kde $G_\sigma$je gausova funkce (Gausovo rozmazání obrazu) a I je obraz.

5 LEVEL SET

Level set metody jsou velmi podobné segmentačním metodám aktiviních kontur. Je zde také zapotřebí manuálně inicializovat jednotlivé body kontury. Používají se při hledání složitých tvarů v obraze, kde kontura *f*(x,y) je reprezentována řezem v rovině xy v tzv. nulové hladině pomocí multidimenzionální funkce, která se nazývá level set funkcí.

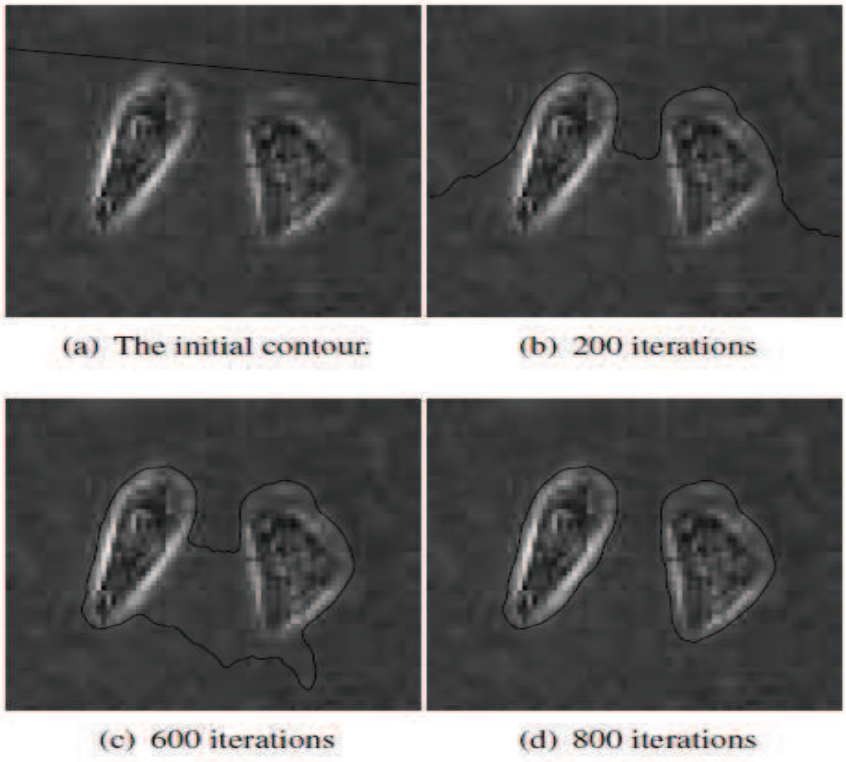
Tato funkce každému bodu v rovině xy přiřazuje jeho výšku *u* nad nulovou hladinou (obr. 5.1). Základním rozdílem level set metod oproti metodám aktivních kontur je ten, že tvar křivky neměníme přímo, ale prostřednictvím leve set funkce. [5] [7]



Obr. 5 Level set funkce (vpravo) pro uzavřenou 2D konturu C [5]

Podstatnou výhodou Level set funkcí je, že jsou výpočetně rychlé, paměťově nenáročné a dají se použít pro detekci parametricky nepopsatelných objektů. Není problém segmentovat více objektů najednou (obrázek 6).

Díky těmto přednostem se tyto metody využívají v aplikacích pro segmentovaní obrazových dat z MRI, CT popřípadě při segmentování videa, kde objekty v obraze mění topologii mezi jednotlivými snímky. Metody leve setů jsou výslekem mnohaleté práce a bádní v oblastech segmentace. [7] [5]



Obr. 6 Level set Evolution Without Re-initialization [7]

5.1 Metoda Level Set vývoje bez Re-inicializaci

V tradiční level set metodě je zapotřebí, aby se rozvíjející se level set funkce držela v blízkosti znaménkové vzdálenpsti funkce, tzv. Signed distance function. Proto se používá tzv. technika Re-Inicializace v průběhu evoluce.

5.1.1 Tradiční Leve Set metoda

Level Set metodu můžeme definovat, jako vyvíjející se konturu nebo povrch implicitně, pomocí funkce vyšších dimenzí. Tato funkce bývá zpravidla označována jako level set funkce $\phi(x,y)$. Rozvíjející se konturu (povrch), lze získat z nulové úrovně (zero level set) podle následující rovnice: [7]

$$C(t) = \{(x,y)| \phi\ (t,x,y,) = 0\} \tag{17}$$

Průběh rovnice Level Set ϕ můžeme napsat v následující podobě:

$$\frac{\partial \phi}{\partial t} + \mathrm{F}|\nabla \phi| = 0, \tag{18}$$

kde funkce F znázorňuje rychlost funkce, která závisí na nastavené funkční úrovni ϕ.

5.1.2 Kompenzace spojená bez Re-Inicializací.

` Re-inicializace se velmi často používá, jak numerická pomoc při tradičních metodách level set [8]. Tradiční metoda Re-Inicializace a její obecné řešení vyjadřuje následující rovnice:

$$\frac{\partial \phi}{\partial t} = \mathrm{sign}(\phi_0)(1 - |\nabla \phi|), \tag{19}$$

kde, ϕ_0 je funkce, která má být re-inicializována a sign(ϕ_0) je znaménková funkce (signed distance function). Pokud ϕ_0 není hladké nebo je strmější na jedné, než na druhé straně, může být zero level set funkce posunuta nesprávně od původní funkce. Z praktického hlediska může být tento proces re-inicializace velmi komplikovaný, náročný s vedlejšími efekty.[7]

5.1.3 Hlavní formulace Level Set metody s penalizací energie

Podstatou vlastností level set metody je udržet rozvíjející se křivku funkce level set, jako aproximační znaménkovou funkci v čase vývoje. Tuto vlastnost označujeme jako vzdálenost funkce, která musí splňovat $|\nabla \phi| = 1$. Obecně navrhujeme následující integrál:

$$\mathrm{P}(\phi) = \int_{\Omega} \frac{1}{2}(|\nabla \phi| - 1)^2 dxdy, \tag{20}$$

jako míru charakterizující blízkost funkce ϕ od označené funkce vzdálenosti . Tato míra hraje klíčovou roli v Level Set formulaci. Společně s výše uvedeným vyjádřením $\mathrm{P}(\phi)$, předpokládáme následnou Level Set formulaci:

$$\mathcal{E}(\phi) = \mu \mathrm{P}(\phi) + \mathcal{E}_m(\phi), \tag{21}$$

kde, μ > 0 je kontrolní parametr penalizace efektu ϕ z označnené funkce vzdálenosti a $\mathcal{E}_m(\phi)$ je určitá energie pohybující se od nulové úrovně ϕ. V této části označujeme $\frac{\partial \phi}{\partial t}$ Getauxovu derivaci funkcí Ɛ a následný vývoj rovnice

$$\frac{\partial \phi}{\partial t} = -\frac{\partial \mathcal{E}}{\partial \phi} \tag{22}$$

je gradientní tok, který minimalizuje funkci Ɛ. Pro partikulární funkci Ɛ(ϕ) definujeme explicitně podmínky ϕ, kde Getauxova derivace může být počítána právě z podmínek funkce ϕ a její následných derivací.[7]

5.1.4 Variační formulace Level Set aktivní kontury bez Re-inicializace

V obrazové segmentaci aktivní kontury jsou dynamické křivky pohybující se k zájmu objektu. K dosažení tohoto cíle je potřeba explicitně definovat vnější energetickou sílu, která posunuje nulovou hodnotu křivky směrem k objektu zájmu. Nechť *I* je obraz a *g* je okrajová indikační funkce definována:

$$\mathrm{g} = \frac{1}{1+|\nabla G_\sigma * I|^2}, \tag{23}$$

kde G_σ je gausova funkce (Gaussian kernel) se směrodatnou odchylkou σ.

Definujeme vnější energii jako funkci *ϕ(x,y)* uvedenou níže.

$$\mathcal{E}_{g\lambda v}(\phi) = \lambda \mathcal{L}_g(\phi) + v A_g(\phi), \tag{24}$$

kde λ>0 a *v* jsou konstanty a členy $\mathcal{L}_g(\phi)$ a $A_g(\phi)$ jsou definovány:

$$\mathcal{L}_g(\phi) = \int_\Omega g\delta(\phi)\,|\nabla\phi| dxdy, \tag{24}$$

$$A_g(\phi) = \int_\Omega gH(-\phi)\,dxdy, \tag{26}$$

navzájem, kde δ je jednorozměrná Diracova funkce a H je Heavisideova funkce. Nyní definujeme celkovou funkční eregii:

$$\mathcal{E}(\phi) = \mu P(\phi) + \mathcal{E}_{g,\lambda,v}(\phi). \tag{27}$$

Vnější energie $\mathcal{E}_{g,\lambda,v}(\phi)$ řídí nastavení nulové úrovně (zero level set) směrem k objektu zájmu, zatím co vniřní energie $\mu P(\phi)$ penalizuje odchylku ϕ znaménkové funkce v průběhu svého vývoje. Pokud počáteční kontura je umístěna ve vnitřku objektu zájmu, koeficient *v* musí nabývat záporných hodnot k urychlení rozšíření kontury.

U variačního počtu Getauxuovu derivační funkci Ɛ v rovnici (27) můžeme zapsat jako,

$$\frac{\partial \mathcal{E}}{\partial \phi} = -\mu\left[\Delta\phi - \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)\right] - \lambda\delta(\phi)\operatorname{div}\left(g\frac{\nabla\phi}{|\nabla\phi|}\right) - vg\delta(\phi), \quad (28)$$

kde Δ je Laplacov operator. Z tohoto důvodu funkce ϕ minimalizuje funkci splňující Euler-Lagrange rovnici $\frac{\partial \mathcal{E}}{\partial \phi} = 0$. Proces nejstrmějšího sestupu pro minimalizaci funkce Ɛ je v následujícím gradientním toku:

$$\frac{\partial \phi}{\partial t} = \mu\left[\Delta\phi - \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)\right] + \lambda\delta(\phi)\operatorname{div}\left(g\frac{\nabla\phi}{|\nabla\phi|}\right) + vg\delta(\phi), \quad (29)$$

Tento gradientní tok představuje vývoj rovnice Level Set funkce v navržené metodě. Druhý a třetí člen na pravé straně rovnice (26) odpovídá gradientnímu toku energetické funkce $\mathcal{L}_g(\phi)$ a $A_g(\phi)$, které jsou důležiné při řídění zero level set urovně směrem k objektu zájmu. Pro vysvětlení účinku prvního člena, který je spojený s vnitřní energií $\mu P(\phi)$ si můžeme všimnout že, gradientní tok z rovnice (27)

$$\Delta\phi - \operatorname{div}(\frac{\nabla\phi}{|\nabla\phi|}) = \operatorname{div}[(1 - \frac{1}{|\nabla\phi|})\nabla\phi] \quad (30)$$

má faktor $1 - \frac{1}{|\nabla\phi|}$,jako difuzní rychlost. Když $|\nabla\phi| > 1$ vytváří se větší plocha ϕ a snižuje se gradient $|\nabla\phi|$. Když $|\nabla\phi| < 1$, člen má vliv zpětného rozptylu a tak zvyšuje gradient. [7]

5.1.5 Numerické schéma

V praxi je Dirakova funkce δ(x) v rovnici (28) mírně vyhlazená, jako následující funkce δƐ(x) definována:

$$\delta\mathcal{E}(x) = \begin{cases} 0, & |x| > \varepsilon \\ \frac{1}{2\varepsilon}[1 + \cos\frac{\pi x}{\varepsilon}], & |x| \le \varepsilon \end{cases} \quad (31)$$

Aproximaci z rovnice (28) s uvedeným diferenčním schématem můžeme napsat jako,

$$\frac{\phi_{i,j}^{k+1}-\phi_{i,j}^{k}}{\tau} = L(\phi_{i,j}^{k}) \tag{32}$$

kde, $L(\phi_{i,j}^{k})$ je přiblížení k pravé straně v rovnici (28). Diferenční rovnici (32) upravíme na následující iteraci

$$\phi_{i,j}^{k+1} = \phi_{i,j}^{k} + \tau L(\phi_{i,j}^{k}). \tag{34}$$

5.1.6 Vývoj časového kroku

Základní otázkou je jaký rozsah časového kroku, kde je iterace v rovnici (34) ještě stabilní. Z experimentů bylo zjištěno, že časový krok τ a koeficient μ musí splňovat $\tau\mu < 1/4$ v diferenčním shématu popsaným v kapitole 5.1.5, aby byla udržena stabilní křivka level set funkce. [7]

Použitím většího časového kroku můžeme zrychlit vývoj level set funkce, ale můžou vzniknout značné chyby v oblasti zájmu. Volíme tedy kompromis mezi velikostí časového kroku a přesným určením oblasti zájmu. Většinou se používá τ≤10.

5.1.7 Flexibilní inicializace Level Set funkce

V této části navrhneme následující inicializační funkci ϕ_0. Nechť Ω_0 je podmnožina obrazové domény Ω a $\delta\Omega_0$ představuje body zájmu z Ω_0, které můžou být efektivně identifikovatelné morfologickými operátory. Inicializační funkci ϕ_0 definujeme jako,

$$\phi_0(x,y) = \begin{cases} -p, & (x,y) \in \Omega_0 - \delta\Omega_0 \\ 0 & (x,y) \in \delta\Omega_0 \\ p & \Omega_0 - \delta\Omega_0 0 \end{cases} \tag{35}$$

kde p>0 je konstanta. Navrhujeme tuto konstantu většinou jako 2ε, kde ε je šířka dirakovy funkce v rovnici (28).

5.2 Metoda Level Set – Region Scalable Fitting Energy

Následujcí kapitola čerpá z literatury [17].

Jde o metodu založenou na rozpoznávání oblasti zájmu. Respektivě jde o oblastně založené modely, které inklinují ke spolehlivosti v intenzite homogenity každé oblasti, která má být segmentována. V této kapitole si ukážeme oblastně založenou, jako aktivní konturu ve variační formulaci level set.

Nejprve definujeme region-scalable fitting (RSF), jako energetickou funkci v podmínkách obrazové intenzity ze dvou stran kontury. Tato energie je potom zahrnuta do variačního úrovňového vyjádření s level set regulovatelnými podmínkami. Tím se tedy vyhneme proceduře zvanou reinicializace.

5.2.1 Modely oblastí založené na aktivních konturách

Nechť $\Omega \in R^2$ obrazové doméně a I: $\Omega \rightarrow R$ dostaneme obraz v úrovni šedi. Mumford and Shah formulovali problém segmentace obrazu následovně, kde navrhli následující energetickou funcki:

$$\mathcal{F}^{MC}(u, C) = \int_{\Omega} (u - I)^2 dx + \int_{\Omega/C} |\nabla u|^2 \, dx \; + vC \tag{36}$$

Kde *C* je délka kontury, *I* predstavuje originální obraz a *u* je aproximovaný obraz originálního obrazu *I*. V praxi je obtížné tuto rovnici minimalizovat kvůli nekonvexnosti funkce.

Chan a Vese navrhovali aktivní kontru, která se přiblíží Mumford-Shah problému, kde obraz v rovnici (36) je po částech hladká fukce. Pro obraz *I(x,y)* v obrazové doméně Ω navrhli:

$$\mathcal{F}^{CV}(C, c_1, c_2) = \lambda_1 \int_{outside(C)} |I(x) - c_1|^2 dx + \lambda_1 \int_{inside(C)} |I(x) - c_2|^2 dx + v|C|$$

(37)

Kde outside (C) a inside (C) reprezentují oblast vnějšku a vnitřku kontury C v tomto pořadí a c_1, c_2 jsou dvě konstanty aproximované outside (C) a inside (C). První dva členy v rovnici (37) nazýváme *the global fitting energy*.

5.2.2 Model oblastní – měřitelné uprvy (Region-Scalable Fitting Model)

Oblastně založené modely využívají informace o intenzitě v lokálních oblastech regulovanými váhami.

Nejprve si představíme Kernelovu funkci K: $R^n \rightarrow [0,\infty]$ s následujícími vlastnostmi:

1) *K(-u)=K(u)*
2) *K(u)≥ K(v),* kde $|u| < |v|$ *a* $\lim_{u\rightarrow\infty} K(u) = 0$
3) $\int K(x)dx = 1$

Druhou vlastnost nazýváme lokalizační (*localization property*) kernel funkci K, která hraje klíčovou úlohu v této metodě.

Nechť *C* je uzavřená křivka v obrazové doméně **Ω**, která odděluje **Ω** do dvou regionů Ω_1 = vnější (C) a Ω_2 = vnitřní (C). Pro x ∈ **Ω** definujeme následující energetickou lokálně měřitelnou fukci (*local intensity fitting enegry*):

$$\mathcal{E}_x^{Fit}(C, f_1(x), f_2(x)) = \sum_{i=2}^{2} \lambda_i \int_{\Omega_i} |K(x-y)I(y) - f_i(x)|^2 dy \qquad (38)$$

Kde λ_1, λ_2 jsou pozitivní konstatny, f_1 (x) a f_2(x) jsou dve hodnoty, které aproximují obrazové intenzity v Ω_1, Ω_2. Intenzity *I(y)* jsou v lokálních oblastech koncentrovány okolo bodu *x*, který je kontrolován kernelovou funkci K. Proto nazýváme energii lokální intenzity v rovnici (34) jako *region-scalable fitting RSF* .

V této metodě je Gaussovo vyjadření definováno podle následující rovnice:

$$K_\sigma(u) = \frac{1}{(2\pi)^{n/2}\sigma^n} e^{-|u|^2/2\sigma^2} \qquad (39)$$

S parametrem $\sigma > 0$. Pro vysvětlení rozvedeme význam rovnice (38) , kde $\mathcal{E}_x^{Fit}$ je váhová středně kvadratická odchylka aproximačních obrazových intenzit ve vnitřní a vnější kontuře *C* s vhodnými hodnotami $f_1(x)$ *a* $f_2(x)$. Zvláštností je, že Gaussian kernel *K(x-y)* se velmi přibližuje k nule, když *y* nabývá hodnot velmi rozdílných od *x.*

Následně provedeme vyhlazení kontory *C* pomocí následující funkce,

$$\mathcal{E}(C, f_1(x), f_2(x)) = \int \mathcal{E}_x^{Fit}(C, f_1(x), f_2(x))\, dx + v|C| \tag{40}$$

kde topologické změny konvertujeme na formulaci level set.

Formulace level set

V této formulaci kontura $C \in \Omega$ je reprezentována nulovou úrovňovou monožinou (level set) od Lipschitz funkce ϕ:Ω.

Tuto funkci označujeme ϕ, která veme kladné a záporné hodnoty z vnější a vnitřiní kontury *C* vzájemně. Nech H je Heavisaidova funkce, potom energetickou funkční monožinu vyjádříme,

$$\mathcal{E}_x^{Fit}(\phi, f_1(x), f_2(x)) = \sum_{i=2}^{2} \lambda_i \int_{\Omega_i} |K(x-y)I(y) - f_i(x)|^2 M_i(\phi(y))dy \tag{41}$$

kde $M_1(\phi) = H(\phi)\ a\ M_2(\phi) = 1 - H(\phi)$

V praxi je Heavisideova funkce H uvdená v rovnici (41) aproximována pomocí hladké funkce H_e podle následující rovnice.

$$H_e = \frac{1}{2}[1 + \frac{2}{\pi} arctan(\frac{x}{e})] \tag{42}$$

Derivací H_e dostaneme

$$\delta_e(x) = H_e'(x) = \frac{1}{\pi} \frac{e}{e^2 + x^2} \tag{43}$$

Pro uchování pravidelnosti funkce level set, která je nutná pro správný výpočet a stabilitu vývoje je nutné si představit level set regulatization členy ve variační formulaci podle rovnice (19), která charakterizuje znaménkovou funkci vzdálenosti. Proto je navrhovaná funkce na minimalizovane energii následující podle rovnice (44).

$$\mathcal{F}(\phi, f_1, f_2) = \mathcal{F}(\phi, f_1, f_2) + \mu P(\phi) \tag{44}$$

Kde μ je pozitivni konstanta. Minimalizace této energie jeho gradientem je v následujícím textu.

Minimalizace energie

Pro minimalizivaní energetické funkce použijeme gradientní sklon v rovnici (44) pomocí následující Euler-Lagrange rovnice.

$$\int K_\sigma(x-y)M_i^e(\phi(y))(I(y)-(f_i(x))dy=0,\ i=1,2. \tag{45}$$

Poté dostaneme následující rovnici (46) pro $i = 1,2$.

$$f_i(x)=\frac{K_\sigma(x)*[M_i^e(\phi(x))I(x)]}{K_\sigma(x)*M_i^e(\phi(x))}$$

Kde $f_1(x)$ a $f_2(x)$ jsou váhové průměry v okolí x, kde velikost je úměrná váhovému parametru σ.

Pro uderžení stálého f_1 a f_2, minimalizujeme energetickou funkci s ohledem na ϕ a řešíme gradientní rovnici (*gradient flow equation*) následovně:

$$\frac{\delta\phi}{\delta t}=-\delta_e(\phi)(\lambda_1 e_1-\lambda_2 e_2)+v\delta_e(\phi)div(\frac{\nabla\phi}{|\nabla\phi|})+\mu(\nabla^2\phi-div(\frac{\nabla\phi}{|\nabla\phi|})) \tag{47}$$

Kde δ_e je Diracova delta funkce dána z rovnice (39) a e_1 a e_2 jsou funkce

$$e_i(x)=\int K_\sigma(x-y)|(I(y)-f_i(x))|^2dy, i=1,2. \tag{48}$$

Kde f_1 a f_2 váhové průměry z rovnice (41).

Rovnice (42) je vývojová rovnice navrhované metody. Člen $=-\delta_e(\phi)(\lambda_1 e_1-\lambda_2 e_2)$ je derivovaný z údajů upravené energie a tak ho označujeme jako *data fitting term*. Tento člen hraje klíčovou úlohu v tomto modelu. Kde popisuje jak se má aktivní kontura chovat směrem k hranicím. Druhý člen $v\delta_e(\phi)div(\frac{\nabla\phi}{|\nabla\phi|})$ ma charakter zkrácené délky, popřípadě vyhlazovací efekt na kontuře nulové urnovnove mnoziny(*zero level set*), která je důležitá k udržení pravidelnosti kontury. Třetí člen udržuje celkovou funkci level set.[17]

6 REALIZACE PROGRAMU

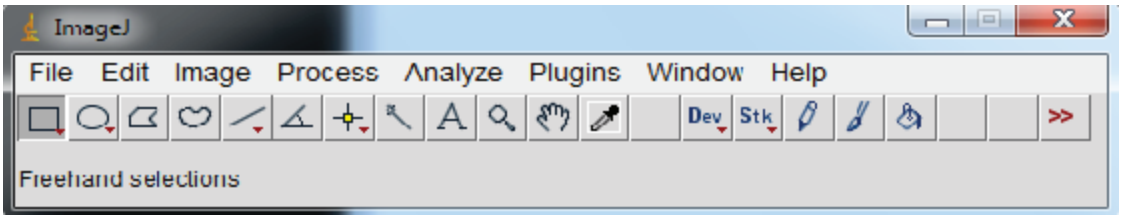
Tato kapital bude zaměřena na vývoj programu pro segmentaci biomedicínských obrazových signálů. Tento program bude implementovat java modul metody Level Set, která bude vytvořena pomocí OpenSource knihoven. Výsledný modul bude univerzální třídou použitelnou pro obecné projekty se vstupem ve formátu ImagePlus. Dále zde budou zobrazeny výsledky pomocí jednoduchého grafického rozhraní GUI. Vlastní program se bude skládat ze 4 tříd, kdy každá třída bude řešit určitý problém. Celkový program bude napsát pod Windows 7 x64.

6.1 Software

Pro vytvoření modulu v jazyce JAVA byl vybrán program ImageJ a prostředí Eclipse pro vývoj kódu. Eclipse pro RCP/Plug-in Developers s verzi build id: 20080619-0625..

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems. Java je nejpopulárnější programovací jazyk. Díky své přenositelnosti je využívána pro programy, které mohou pracovat na různých systémech (Windows, Mac OS, Linux, OS X). Hlavní nevýhodou tohoto programovacího jazyka je pomalejší start programů, protože se program musí nejprve přeložit a až poté spustí. Pro tento projekt byla zvolena JRE systemova knihovna JavaSE-1.7.

ImageJ je volně dostupný program na zpracování obrazů, který je napsaný v programovacím jazyce JAVA. ImageJ umožňuje upravovat, analyzovat různé druhy formátů např. JPEG, BMP, GIF, TIFF, DICOM. Výhodou programu ImageJ je jeho rozšiřitelnosít pomocí tzv. Java pluginů. Pluginy je možné vytvořit použitím vestavěného editoru a kompilátoru jazyka Java. Tato skutečnost umožňuje uživateli vyřešit jakýkoliv problém z oblasti zpracování obrazů. ImageJ obsahuje různé příklady pluginů. Prostředí ImageJ obsahuje čtyři základní panely: Menu, panel nástrojů, panel stavů a panel činností [11]. Pro spuštění ImageJ je nutné mít v počítači nainstalován Java Development Kit (JDK), který je volně přístupný na stánkách java.sun.com/javase/downloads/ . Pro tento projekt by vybrán verze ImageJ 1.45.



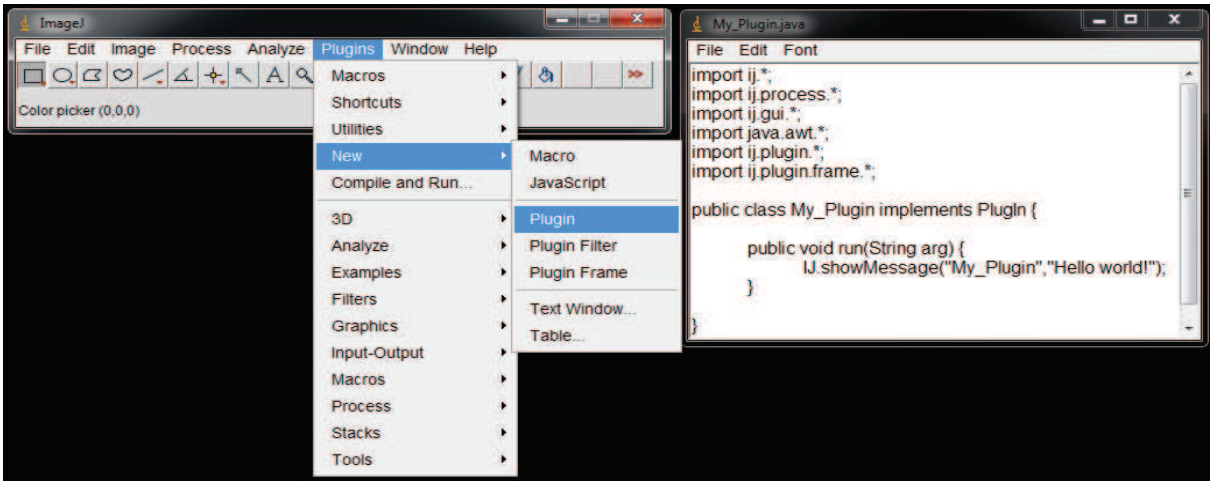
Obr. 7 ImageJ

V horní části okna se nachází menu, pod kterým je panel nástrojů, který obsahuje nástroje pro označení části obrázků. V našem případě budeme potřebovat nástroj zvaný polygon na vytvoření pozice kontury, který budeme volat do námi vytvořeného pluginu.

Stavový panel je na začátku práce neaktivní, aktivuje se až po stlačení nějakého nástroje z panelu nástrojů. Zobrazuje souřadnice aktuální pozice kurzoru a jeho následnou hodnotu.

6.1.1 Implementace pluginu

Jak už bylo zmíněno v kapitole 6.1.1 program ImageJ umožňuje využit přímo vloženou funkcionalitu na vytvoření pluginu, který se dá dále programovat (obrázek 8). Tento postup je však velmi neefektivní kvůli následnému psaní kódu a obtížný na vyhledávání vzniklých chyb během programování. Proto v rámci práce bylo zvoleno prostředí Eclipse do kterého bylo potřeba vložit program ImageJ, aby bylo jednodušší testování (debugování) vyvíjeného pluginu.[12]



Obr. 8 Příklad vytvoření pluginu v ImageJ

6.2 Třída LevelSetMain.java

Ukázka třídy:

```
package levelsets.method;

public class LevelSetMain_ {

	public static void main(String[] args) {
		LevelSetGui_ gui = new LevelSetGui_();
		gui.setVisible(true);
	}

}
```

Jedná se o spustitelnou třídu, pomocí které se spuští výsledný program.

`LevelSetGui_ gui = new LevelSetGui_();` zde představuje kotstruktor pro spustení třídy GUI. Tedy spustění grafického rozhraní.

6.3 Třída Point.java

Ukázka třídy:

```
package levelsets.method;

class Point {
	private int x; // radek
	private int y; // sloupec

	public Point(int x, int y) {
		this.x = x;
		this.y = y;
	}

	public int getX() {
		return x;
	}

	public int getY() {
		return y;
	}


}
```

Tato třída třída představuje vstupní a výstupní segmentační body pro metodu Level Set. Obsahuje pouze verejné (public) metody, k vůli přístupu z ostatatních tříd.

6.4 Třída LevelSet.java

Tato třída obsahuje celkový algoritmus k provedení segmentace metodou Levelset. Jde tedy o hlavní jádro pro vývoj java modulu. Obsahuje proměnné a různé parametry, vektory a pole, které budou nastavovány v GUI kvůli správnému fungování Level Set metody pomocí metody:

```
    public void setParameters(int fm_iter, double fm_alpha, int
ls_inner_iter,int ls_outer_iter, double ls_delta_t, int
ls_bandwidth,double ls_epsilon, double ls_beta);
```

Ukázka tříy:

```
public class LevelSet_ {
      private Roi roi;
      private ImagePlus imp;
      private ImageProcessor ip;

      private static final double BIG_NUMBER = Double.MAX_VALUE;
      private static final double SMALL_NUMBER = Integer.MIN_VALUE;
      // pocet radku (vyska) obrazu ImagePlus
      private static int Num_Row = 0;
      // pocet sloupcu (sirka) obrazu ImagePlus
      private static int Num_Col = 0;
      // vypocet matic a vectoru pro fast marching
      private double[][] T = null; // T value
      // vypocet delky kroku (unit is 1)
      private static final int DELTA_X = 1;
      private static final int DELTA_Y = 1;
      // parametry k fungování fast marching
      public static int FMSteps = 3000; // pocet iteraci fast marching
      public static double ALPHA = 70; // alfa pro exponenciální term
      // nastaveni parametrů pro level set
      // iterací krog pro vnitřní smyšku
      public static int iter_inner = 5;
      // iterační krok pro vnější smičku
      public static int iter_outer = 0;
      public static double DELTA_T = 0.05; // časový kro
      public static double bandWidth = 5.0; //  band width
      // zakřivení křivky
      public static double EPSILON = 0.045;
      public static double BETA = 0.51; // attraction force term
      public static double FA = 2.0; // advection force term
      // vector pro nastaveni alive
      public Vector alive = new Vector();
      // vector pro nastaveni close
      public Vector close = new Vector();
      // vector pro nastaveni farAway
      public Vector farAway = new Vector();
      // vector pro nastaveni neighbour
      public Vector neighbour = new Vector();
      // vypocet matic a vectoru pro level set
      private double[][] phi = null; // level set Phi hodnota
```

```
// level set Phi předešlá hodnoty
private double[][] phiOld = null;
// rozmazane hodnoty pixelu obrazu - Gussovo rozmazani
private double[][] Io = null;
private double[][] I = null; // puvodni hodtony pixelu obrazu
// hodnoty KI gradientu pro level set
private double[][] KI = null;
// hodnoty KI_FM gradientu pro Fast Marching
private double[][] KI_FM = null;
```

Hlavní strukturu této třídy tvoří 3 public metody

- `public void readImagePlus(ImagePlus imp);`
- `public void startFastMarch(Vector seedPoints);`
- `public void starLevelSet();`

6.4.1 Metoda public void readImagePlus(ImagePlus imp)

Tato metoda se věnuje načtení a výpočtu hodnot z obarzu, který je určen k segmentaci. Jelikož, cílem projetku je pracovat s obrazem se vstupním formátem ImagePlus je zapotřebí se zaměřit na třídu ImagePlus.

ImagePlus imp v této metodě je objekt, který představuje příslušný obraz. Je založený na třídě ImageProcessor, která pracuje s hodnotami jednotlivých pixelů a dělá skutečnou práci nad obrázkem. Typ ImageProcessor bývá použit v závislosti na typu obrazu. Tyto typy obrazů jsou zastoupeny pomocí konstant deklarovaných v ImagePlus.[16]

V projetku jsou hodnoty obrazu ukládány do pole Io [][] a poté jsou rozmazány Gaussovým filtrem z důvodu, že by byl segmentovaný obraz zatížen velkým šumem.

Ukázka třídy:

```
public void readImagePlus(ImagePlus imp) {

            ip = imp.getProcessor();

            // nastaveni poctu radku a sloupcu
            Num_Col = imp.getWidth();
            Num_Row = imp.getHeight();
            System.out.println(Num_Col + "  " + Num_Row);


            phi = new double[Num_Row][Num_Col];
            phiOld = new double[Num_Row][Num_Col];
            I = new double[Num_Row][Num_Col];
            Io = new double[Num_Row][Num_Col];
            KI = new double[Num_Row][Num_Col];
            KI_FM = new double[Num_Row][Num_Col];
```

```
T = new double[Num_Row][Num_Col];

// nacteni hodnot pixelu v originalnim obraze (ImagePlus)
for (int i = 0; i < Num_Row; i++) {
      for (int j = 0; j < Num_Col; j++) {
            Io[i][j] = ip.getPixelValue(i, j);
      }
}
```

6.4.2 Metoda public void startFastMarch(Vector seedPoints)

Představuje segmemtační metodu Fast Marching, která určí hrubý obrys segmentované části obrazu. Tato metoda vede k jednodušší formulaci Eikonal ronvnice:

$$|\nabla u(x,y)|F = 1 \tag{49}$$

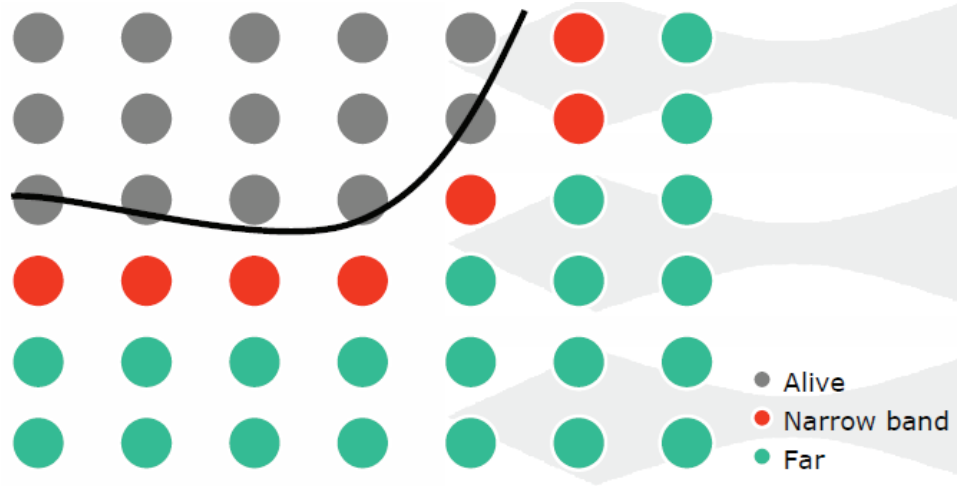
V kombinaci s optimálním tříděním této techniky, vede rovnice $\Phi(x,y,t = t_0) = 0$ k rychlému řešení.

Fast Marching algoritmus:

LOOP {

1. Nechť Trial je bod v Narrow band (uzkém másmu) s nejmenšní hodnotou *u*
2. Přesň Tial z Narrow band do Allive
3. Přesuň všech Neighbors (sousední body) z Trial do Narrow band
4. Přepočítej *u* pro všechny Neighbors z Trial

}



Obr. 9 Algoritmus Fast Marchnig

Tato metoda je tvořena dvěma privátními metodami

```
private void initializeFastMarch(Vector seedPoints);
```

Zde jde o inicializaci hodnot pro metodu Fast Marching. Nejprve zde dochází k vynulování vše hodnot jednotlivých vektorů alive, close, farAway a neighbour. Poté se projdou všechny pixelu segmentovaného obrazu a jejich hodonoty se nastaví do farAway. Dalším nezbytným krokem je inicializace seedPoints, tedy bodů které budou nastavovány uživatelem v GUI. Jde tedy o počáteční umístě kontury, respektivě jednotlivých bodů ze kterých se bude vyvíjet segmentovaný obraz.

```
private void march();
```

Tato metoda představuje hlavní část průběhu implementace algoritmu metody Fast Marching. První část je tvořena for cyklem, který nám zajistí počet iteračních kroků, které budou stanoveny uživatelm v GUI. Jde tedy o průběh algoritmu který je znázornění v záčátku kapitoly 6.4.2.

Pro správné fungování této metody jsou pužity metody `private boolean isAlivePoint(Point p)` a `private boolean isClosePoint(Point p)`. Ty zajištují zda určité body jsou Alive a obsaženy v close. Zajišťují, zda dané vlákno stále běží. Pokud ano, vrátí metoda booleovskou hodnotu true. V opačném případě vrátí hodnotu false. Používají se tedy ke zjištění zdar dceřiný proces stále běží.

```
private boolean isAlivePoint(Point p) {
            int numAlivePoints = alive.size();

            for (int i = 0; i < numAlivePoints; i++) {
                  if (p.equals((Point) alive.elementAt(i)))
                        return true;
            }

            return false;
      }

private boolean isClosePoint(Point p) {
            int numClosePoints = close.size();

            for (int i = 0; i < numClosePoints; i++) {
                  if (p.equals((Point) close.elementAt(i)))
                        return true;
            }

            return false;
      }
```

Ukázka třídy:

```
//
*************************************************************************
*******
      // Fast Marching funkce
      //
*************************************************************************
*******

public void startFastMarch(Vector seedPoints) {

            initializeFastMarch(seedPoints);
            march();

      }

      /***
       * prubeh inicializace hodnot pro fast marching
       ***/
      private void initializeFastMarch(Vector seedPoints) {
            // vynulovani vsech vektoru
            alive.removeAllElements();
            close.removeAllElements();
            farAway.removeAllElements();
            neighbour.removeAllElements();

            // nastaveni vsech bodu do farAway
            for (int i = 0; i < Num_Row; i++) {
                  for (int j = 0; j < Num_Col; j++) {
                        farAway.addElement(new Point(j, i));
                        T[i][j] = Integer.MAX_VALUE;
                  }
            }

            // inicializace seedPoints - body z roi
            int numFrontPoints = seedPoints.size();

            for (int i = 0; i < numFrontPoints; i++) {
                  Point p = (Point) seedPoints.elementAt(i);
                  initialFMAux(p.getX(), p.getY());
            }
      }
```

6.4.3 Metoda public void startLevelSet()

Jde o veřejnou metodu která spouští celkový algoritmus Level Set. V první části bylo potřeba inicializovat hodnotu `phi[i][j] = 0.0;` pro všechny segmentované pixely obrazu. Poté bylo potřeba stanovit inicializaci všechno ostatních vektorů , což bylo provedeno pomocí metod `initializeFront() a initializeNonFront()`.

V další části došlo ke zkontrolování phi hodnot a následnému obnovení Narrow band. Tato funkcionalita baly provedena pomocí metody `private void setNarrowBand()`.

Poté byl vypočítán počet iteračních kroků, kde počet vnějších a vnitřních kroků určil sám uživatel pomocí rozhrani GUI. Následně bylo provedeno uložení phi hodnto pro reinicializaci. Tedy výpočet `phiOld[i][j] = phi[i][j]`,což bylo provedeno metodou `private void storePhi()`. V další části bylo potřeba zkontrolovat všechny body a přiřadit vzniklé nové body do vektoru frontLine. Tuto funkci realizuje metoda `private void setFront()`. Všechny body, které tvoří výsledný segment daného obrazu jsou obsaženy právě v daném vektoru fronLine. V poslední části došlo opět ke zkotrolovaní phi hodnot a následnému obnovení Narrow band.

Ukázka metody:

```
/***
       * start level set funkce
       ***/
      public void startLevelSet() {
            // restart všech vektrů
            frontLine.removeAllElements();
            narrowBand.removeAllElements();
            nbBoundary.removeAllElements();
            testNB.removeAllElements();

            // opětovná inicializace všech vektorů
            initializePhi();
            initializeFront();
            initializeNonFront();
            setNarrowBand();

            // vnější iteraci
            for (int iter = 0; iter < iter_outer; iter++) {
                  int m = 0;

                  // vnitřní iterace
                  while (m < iter_inner) {
                        updatePhi();
                        m++;
                  } // while

                  // uložení phi hodnot pro reinicializaci

                  storePhi();
                  setFront();
                  setNonFront();
                  setNarrowBand();
            } // konec cyklu

      }
```

6.5 Třída LevelSetGui.java

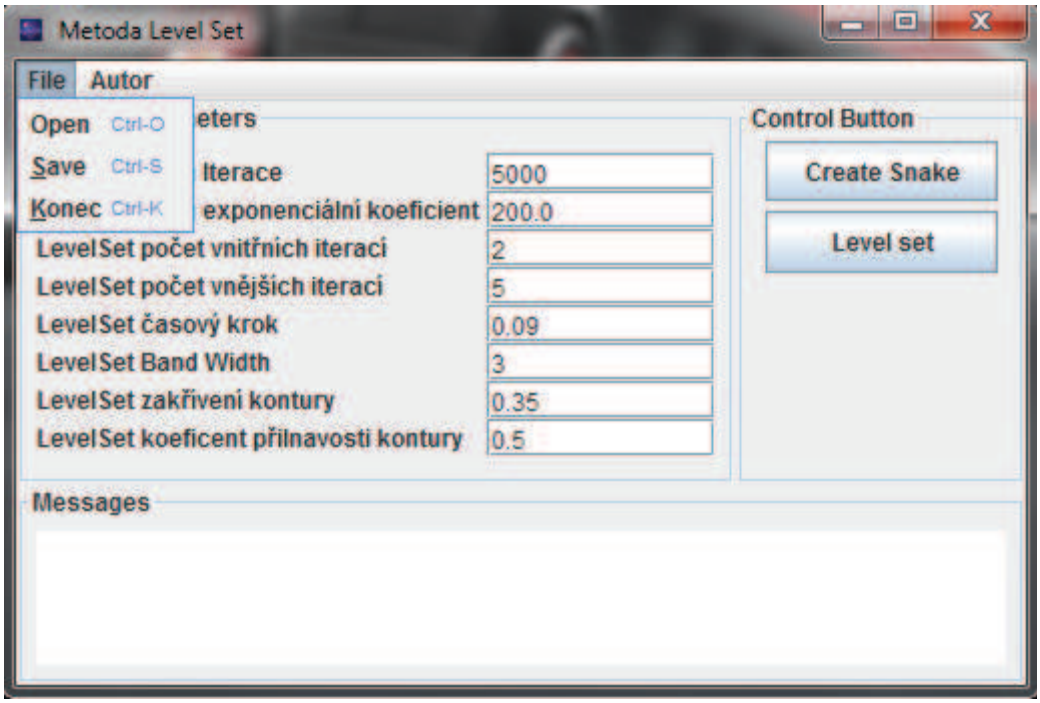
Ukázka struktury:

```
public class LevelSetGui_ extends JFrame {

	private static final long serialVersionUID = 1L;
	private JMenuBar menuJMenuBar = null;
	private JMenu fileJMenu, autorJMenu = null;
	private JPanel containerjPanel, ControljPanel, FunctionljPanel,
			windowjPanel, parameterjPanel, labeljPanel = null;
	private JScrollPane scrolljPanel = null;
	private JMenuItem KonecjMenuItem, openJMenuItem, saveJMenuItem =
null;
	private JTextArea textjArea = null;
	private Roi roi = null;
	private ImagePlus imp = null;
	private ImageProcessor ip = null;
	private BufferedImage image = null;
	private JButton snakejButton, LevelSetjButton = null;
	private LevelSet_ levelset = new LevelSet_();
	private Vector seedPoints;
	private int nPoints = 0;

	private Point p = null;
	private int x_p, y_p;
	private int[] x_roi, y_roi;
	private int[] x, y = null;


	// vsup řídicí parametrů - control param
	private JTextField FM_Iter;
	private JTextField FM_ALPHA;
	private JTextField LS_Inner_Iter;
	private JTextField LS_Outer_Iter;
	private JTextField LS_DELTA_T;
	private JTextField LS_BANDWIDTH;
	private JTextField LS_EPSILON;
	private JTextField LS_BETA;

	// počáteční hodnoty řídicích parametrů
	private String fm_iter_initial = "5000";
	private String ls_inner_iter_initial = "2";
	private String ls_outer_iter_initial = "5";
	private String ls_bandwidth_initial = "3";
	private String fm_alpha_initial = "200.0";
	private String ls_delta_t_initial = "0.09";
	private String ls_epsilon_initial = "0.35";
	private String ls_beta_initial = "0.5";

	public LevelSetGui_() {
		super();
		initialize();

	}
```

Tato třída se stará o celkové grafické rozhraní programu. Pro zpracování byla především využita knihovna Java Swing. Vzhled aplikace byl volen, podle standardních grafických aplikací a je intuitivně řazen do k sobě patřících skupin. Z praktického řešení lze o grafickém rozhraní říc, že jej dokáže ovládat pouze osoba která se o danou problematiku zajímá. Protože pak by docházelo ke špatnému nastavení určitých parametrů, které jsou nezbytné k správnému fungování Level Set metody.

Celkově grafické rozhraní podle obrázku (10) se da rozdělit do do čtyř skupin, které řídí metody:

- `private JMenuBar getmenuJMenuBar()`
- `private JPanel getWindowjPane()`
- `private JPanel getControljPanel()`
- `private JPanel getFunctionljPanel()`



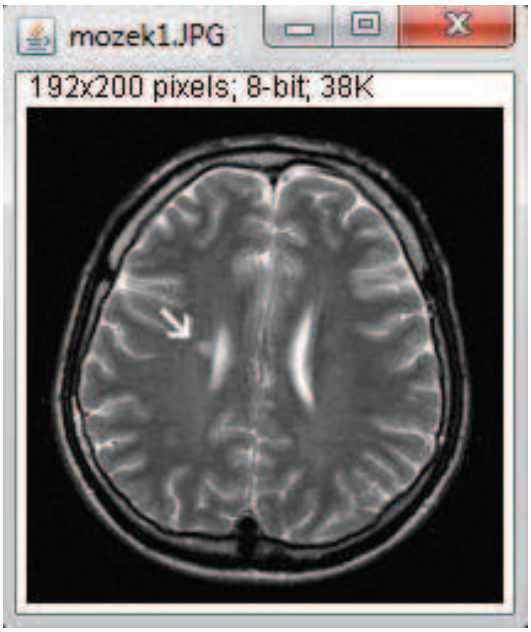
Obr. 10 Grafické rozhraní programu

6.5.1 Metoda private JMenuBar getmenuJMenuBar()

Tato metoda zajišťuje přístup do submenu File a Autor. Kde složka File se stará o vytvoření sady parametru k načtení, uložení a ukončení programu. K načtení obrázku se volá metoda:

```
private JMenuItem getOpenJMenuItemu()
```

Pomocí této metody se spouští obrázek se vstupním formátem ImagePlus příkazem `IJ.openImage();` Teto příkaz je obsaže v programu ImageJ.



Obr. 11 Načtený obrázek mozku.jpg se vstupním formátem ImagePlus

Pro uložení obrázku se volá privátní metoda:

```
private JMenuItem getSaveJMenuItemu()
```

Tato metoda uloží obrázek v počítači s cestou C:/..... a název ponechá původního obrázku.

6.5.2 Metoda private JPanel getWindowjPane()

V této metodě se zajištuje zpětná vazba s uživatelem. Informuje jej o načtení obrazků, pořípadě o nenastavení seedPoints v příslušném obrazu.

6.5.3 Metoda private JPanel getControljPanel()

Tato metoda implementuje rozložení dvou jediných tlačítek v celém programu. Jde o tlačítka Create Snake, a Level Set. Tlačítko Create mělo vytvořit seedPoints (pouze jede bod) v segmentovaném obrazu. Pomocí talčítka Level Set se volá třída LevelSet.java pro vykonání algoritmu, zárovně se nastavuje počáteční hodnoty paramentrů pro metodu Lelevel Set a Fast Marching.

6.5.4 Metoda private JPanel getFunctionljPanel()

Principem této metody je rozdělení umístění názvů controlních parametrů a jejich příslušných hodnot. Tuto funkcionatiltu zajištují metody:

- `private JPanel getparameterjPanel()`
- `private JPanel getlabeljPanel()`

Kde metoda `getparameterjPanel()`, zajišťuje změnu hodnot u všech paramterů, jako je například: Fast Marching počet iteračních kroků, Level Set počet vniřních a vnější iterací atd. A metoda `getlabeljPanel()`, nastavuje názvy příslušných paramterů.

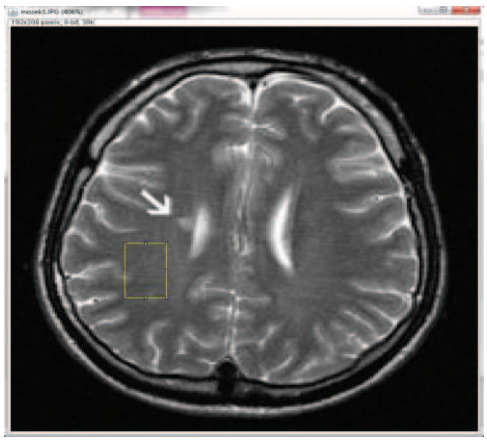
7 DOSAŽENÉ VÝSLEDKY

Cílem práce vytvořit modul v jazyce java, který má implementovat metodu Level Set. Tetno cíl byl splněn s tím že do modulu bala vložena další metoda Fast Marching z důvodu získání hrubého obrysu pro následnou segmentaci Level Set.

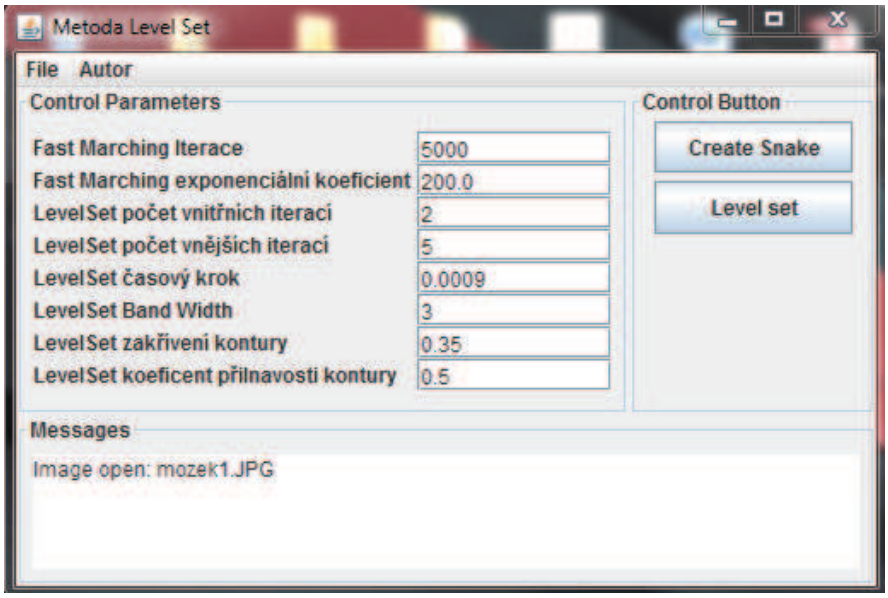
Dále měl být vytovřený modul navržen tak,aby mohl pracovat se vstupním formátem ImagePlus, což se v této prací taktžé podařilo dosáhnout.

Avšak výsledné GUI, které mělo ukázat celkové segmentační výsledky není zcela vyhovující. V programu nastal problém, kdy se do obrazu umísťují počáteční body pro segmentaci v práci ozančované jako seedPoints. Třída LevelSet.java je napsaná tak, aby pro vkládnání seedpoints byl vložen jen jeden bod, který má být vložen do segmentované části. Tohoto jevu se však v práci nepodařilo docílit. Jelikož po načtení obrázku je pomocí ImageJ defaultně nastaven *IJ.setTool(0),* což je rectangular pomocí něhož se vybírají 4 seedPoints. V tomto případě bylo potřeba nastavit *IJ.setTool(7),* který by vybral funkci points. Ze segmentovaného obrazu (14) jsou viditelné chyby, které jsou způsobny, že segmenované body nejsou zpravně propojeny mezi sebou. Propojení bodu znázorňuje žlutá čára.

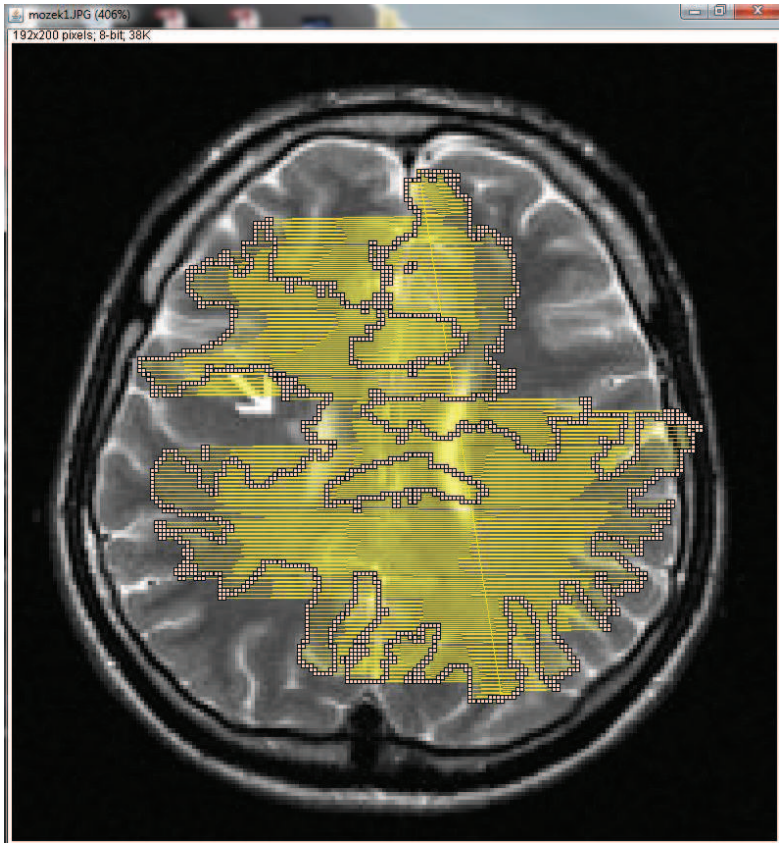
7.1 Segmentace obrazu s počáteční umístění snakes uvnitř objektu zájmu



Obr. 12 Umístění počáteční snakes (kontury) - uvnitř



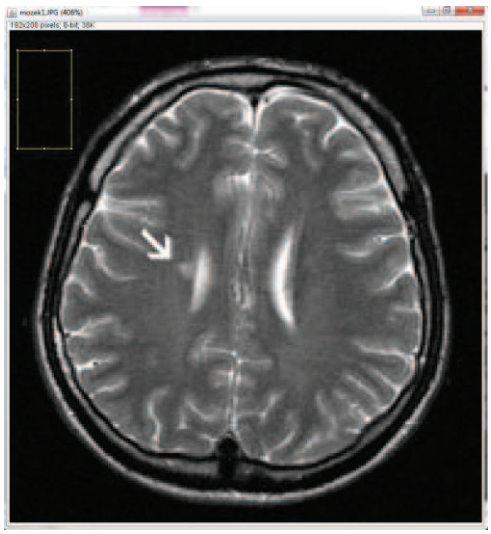
Obr. 13 Natavené paramtery pro segmentaci – uvnitř



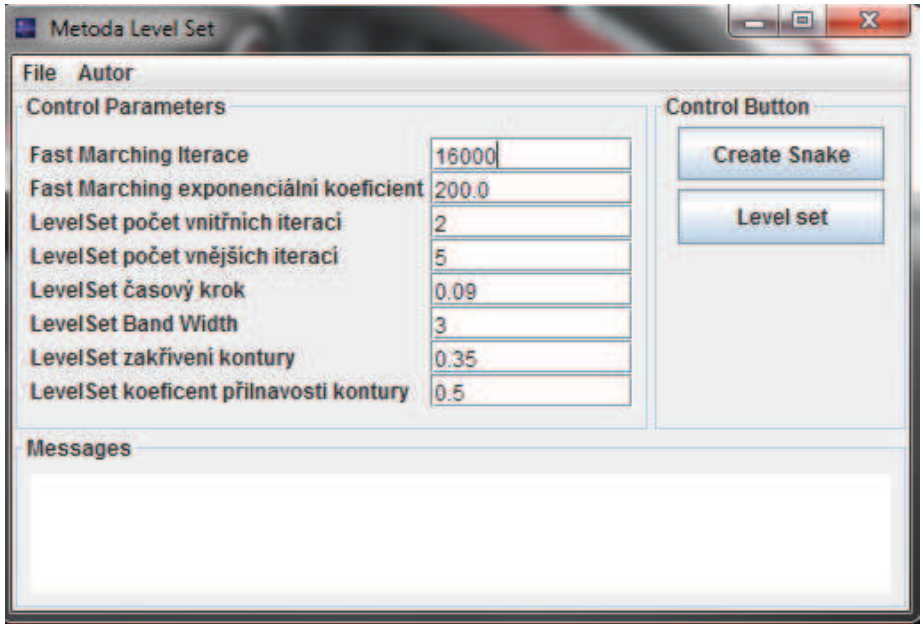
Obr. 14 Výsledná část segmentace pro dané nastavení parametrů. -uvnitř

Z obrázku (14) je patrné jak bylo popsáno v úvodu že pro segmentační metodu Level Set není problém oddělovat jednotlivé segmentované části.

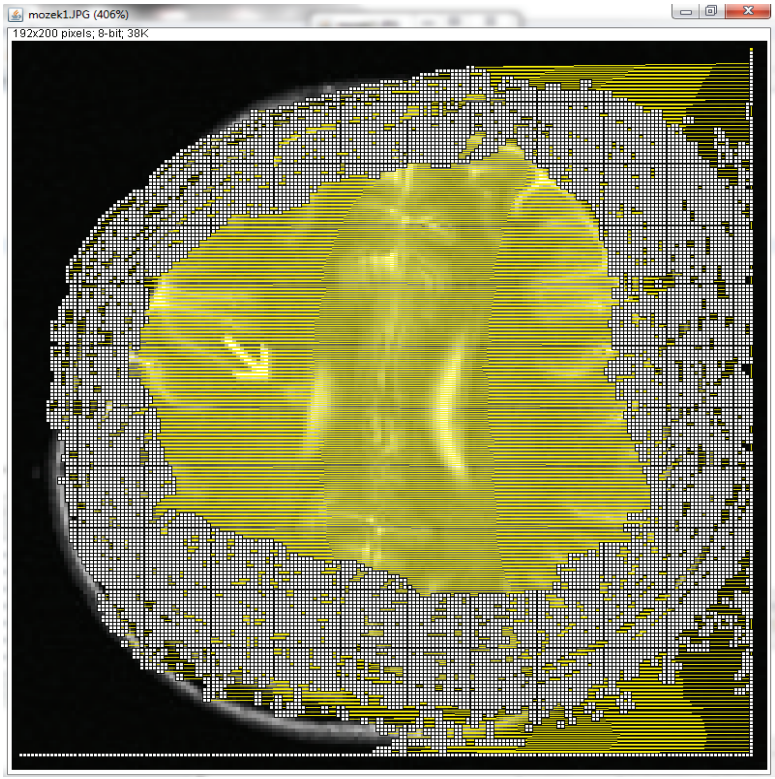
7.2 Segmentace obrazu s počáteční umístění snakes vně objektu zájmu



Obr. 15 Umístění počáteční snakes (kontury) – vně



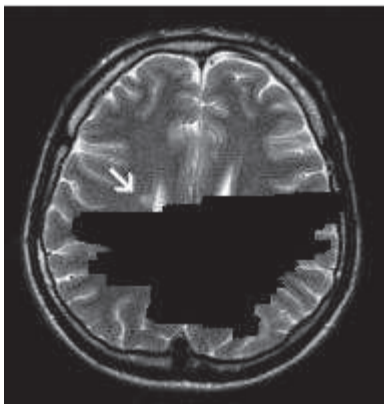
Obr. 16 Natavené paramtery pro segmentaci – vně



Obr. 17 Obr. 18 Výsledná část segmentace pro dané nastavení parametrů. – vně

Z obrázku (17) je patrné že metoda se snaží vysegmetovat pozadí. Segmentace zde na obrázku není úplná kvůli ne zcela ideálně nastaveným kontrolních paramterů.

7.3 Uložení obrázku



Obr. 19 Uložený obrázek s chybným propojením segmentačních bodů.

7.4 Segmentace metodou Level set - Matlab

Jelikož se Level set metodou zabývá celá řada různých autorů, byl vybrán v této čáti práce skripty[9][10] pro program Matlab. Teoreticky je rozebrán tento skript v kapitole 5.1. Původní skript publikoval autor jménem Chumming Li, který vytvořil novou variační formulaci geometrických aktivních kontur, mezi které se právě řadí level set funkce.

V této formulaci není třeba uvažovat o re-inicializaci level set funkce, která právě udržovala tradiční level set funkci viz.kapitola 5.1.1 v blízkosti vzdálenosti funkce a bez nichž by metoda přestala fungovat.

7.4.1 Segmentace šedé kůry mozkové

Nejprve se nahrál obraz, který chceme segmentova. Dalším krokem bylo potřeba vyhladit tento obraz případným filtrem. V našem případě byl použít Gausův filtr. Poté se vytvoří obraz gradientu. Dále bylo zapotřebí inicializovat level set funkci (uzavřenou konturu), která se měla stahovat, neboť šlo o konturu z vnějšku obrazu mozku. K zastavení vyvíjející se kontury dojde, až při dosažení určité hrany, která byla stanovena funkcí indikace hrany. Jinými slovy algoritmus evoluce zajišťuje vývoj level set funkce a metoda končí po uplynutí stanoveného počtu iterací, viz.Obrázek 9.

Proměnné, které je potřeba definovat v m-filu:

g....funkce indikace hrany,

epsilon .... parametry vyhlazené Diracovy funkce,

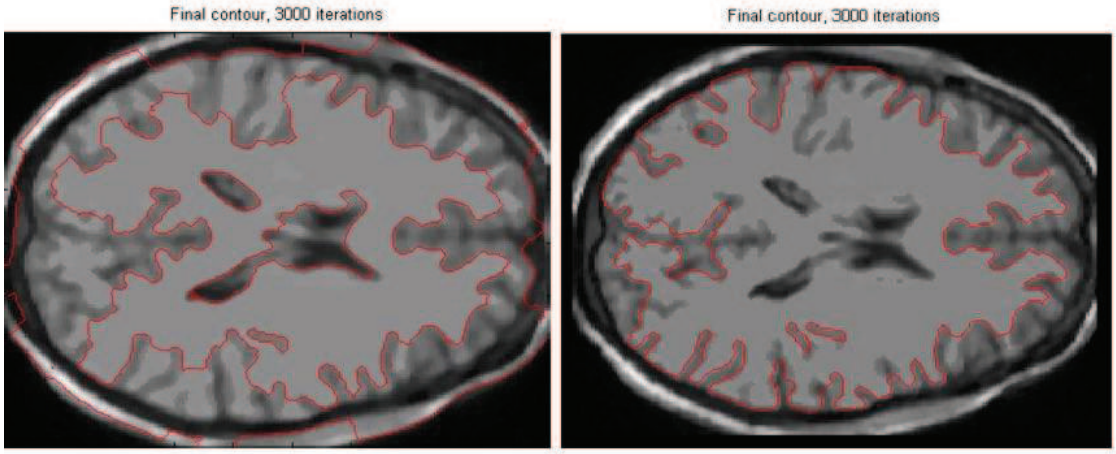
mu ..... koeficient vnitřní energie.

lambda .... koeficient členu vážené délky Lg

alf ... koeficient členu váženého prostoru Ag

delt ... časový krok iterace

numIter ... počet iterací



Obr. 20. Segmentace metodou Level set. Šedá kůra mozková (vlevo), Bíla kůra mozková (vpravo)

7.4.2 Segmentace bílé kůry mozkové:

Obdobný postup jako u detekce šedé kůry mozkové. Rozdíl je jen v inicializaci level set funkce, která se má chovat opačným způsobem než při detekci šedé kůry mozkové.

8 ZÁVĚR

Tato diplomová práce je rozdělen na část teoretickou a realizační. Teoretická část se zabývá segmentačními technikami Aktivních kontru (Snakes) a technikou Level set.

První část práce je věnována obecným poznatkům o segmetaci biomedicínských obrazů. A to zejmnéna obrazům z magenetické rezonance, které jsou v rámci této diplomové práce použity k účelům ověření si funkcí segmentačních techniky Level Set.

Další část práce se zabývá metodami Aktivních kontru (Snakes), které se využávají hlavně při selhávání jiných segmentačních metod. Zejména se jedná o techniky detekce hran či prahování. Je zde navržené průběh Greedyho algoritmu, který popisuje, jakým způsobem se vyvýjí aktivní kontura.

Poslední teoretická část se věnuje segmentační technice Level set, kde se popisují její všeobecné vlastosti.

První část realizce se zabývá vytvořením modulů v jazyce java v programu ImageJ. Potupně popisuje, jak program pracuje a postup, jak vytoří spustitelné pluginy určené ke zpracování obrazu. Dále se zde uvádí důvod, proč vytvářet pluginy pomocí prostředí Eclipse a nikoliv pomocí vestavěné funkce programu ImageJ.

Jelikož ze zadání jasně vyplívá, že má být vytvořené jednoduché GUI. Proto bylo potřeba vložit do programu knihovnu ImageJ. Z tohoto důvodu nebude potřeba před spuštěním GUI spustit samostatný program ImageJ.

Další část práce se věnuje samostatnému java kódu, který implementuje modul pro metodu Level Set. Jous zde popsáný jednotlivé třídy a k nim příslušné metody.

Důležité je se zmínit o metodě pro uložení obrázku, která je s napsána sice správně, ale výsledek pro uložení obrazku je zcela degradující kvůli problému, který byl podrobněji rozebírán v kapitole 7.

V poslední části se práce věnuje segmentační metodě Level set v programu Matlab. Především je zaměřena na segmentaci Šedné a bílé kůry mozkové. Z metody je patrné, že velmi záleží na nastavení parametrů a počtu iterací k dosažení celkové segmentace obrazu.

LITERATURA

[1] BALANIS, A. C. *Antenna Theory: Analysis and Design*, 2/E. New York: J. Wiley & Sons, 1996.

[2] Drastich, A. (ed.) 2004, *Tomografické zobrazovací systémy*, VUT Brno

[3] Janoušová, E. *Statické metody segmentace v MRI obrazech mozku:* bakalářská práce. Brno: Masarykova univerzita, 2008

[4] M. Kass, A. Witkin, D. Terzopoulos. *Snakes: Active contour models*. International Journal of Computer Vision, 1988

[5] Španěl, M. a Beran, V. *Obrazové segmentační techniky*, [online cit. 1.11.2011].

Dostupné na: < http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769325>

[6] Tiilikainen, P, N. *A Comparative Study of Active Contour Snakes*, 2007

[7] Chumming Li, Chenyang Xu, Changfeng Gui, Martin D.Fox. *Level Set Evolution Without Re-Initialization: A New Variational Formulation,* [online cit. 10.11.2011]. Dostupné na: < http://www.engr.uconn.edu/~cmli/paper/levelset_cvpr05.pdf>

[8] Vese, L.; Chan, T. *A multiphase level set framework for image segmentation using the Mumford and Shah model*. 2002.

[9] Chumming Li, *Publication Level Set Method*, [online cit. 15.11.2011].

Dostupná na: < http://www.engr.uconn.edu/~cmli/>

[10] Pavel Hanák, *Segmentation of magnetic resonance images*. Brno 2010

[11] Program ImageJ, [online cit. 1.12.20011].

Dostupný na: < http://rsb.info.nih.gov/ij/docs/index.html >

[12] ImageJ s pužitím Eclipse, [online cit. 20.10.2011]. Dostupný na:

<http://imagejdocu.tudor.lu/doku.php?id=howto:plugins:the_imagej_eclipse_hoo>

[13] Nixon,Mark; Aguado, Alberto. Feature Extraction & Image Processing. Second edition. Geat Britain. : Academic press, 2008.406 s. ISBN 978-0-1237-2538-7

[14] E-Snake, [online cit. 20.10.2011]. Dostupnýna:
<http://bigwww.epfl.ch/algorithms/esnake/>

[15] J. Liang, T. McInerney and D. Terzopoulos, *Interactive Medical Image Segmentation with United Snakes*, Second International Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI99), Cambridge, England, September, 1999, pages 116-127, [online cit. 7.3.2012]. Dostupná na:
<http://www.scs.ryerson.ca/tmcinern/mypapers/miccai99.pdf>

[16] A. Podlasov, E. Ageenko, *Working and Development with ImageJ* [onlinecit.12.11.2011]. Dostupná na:
<http://cs.joensuu.fi/pages/ageenko/public/notes/ea_imagej.pdf >

[17] Li, C.Kao, J.C. Gore and Z.Ding, *Implicit Active Controus Driven by Local Binary Fitting Energy, CVPR 2007*

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

f	Signál v časové oblasti
F	Signál ve frekvenční oblasti, rychlostní funkce
DCT	Discrete Cosine Transform, diskrétní kosinová transformace
JRE	Java Runtime Environment
JAVA	Programovací jazyk
JDK	Java Developmen Kit
MRI	Magnetická rezonance
GM	Šedá hmta
WM	Bílá hmota
CT	Počítačová tomografie
Φ	Level Set funkce
sign(ϕ)	Znaménková funkce
I	Originální obraz
Δ	Diracova funkce
H	Heavisideova funkce
T	časový krok
Ω	Obrazová doména
C	Délka kontury
JPEG	Photographic Experts Group
BMP	Windows Btimap
GIF	Graphics Interchange Format
TIF	Tag image Format
DICOM	Digital Imagin and Communications

SEZNAM PŘÍLOH

A) Diplomová práce - text

B) Obsah CD: **1) Spustitelný program LevelSet.jar s knihovnama**

2) Java kody – Pracovní složky Eclipse

3) Obrázek magentické rezonance mozku pro segmentaci