Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Plzeň, 2014

Pavel Volkovinský

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne: . . . . . .

. . . . . .

vlastnoruční podpis

Anotace

Práce se zabývá zpracováním CT snímků jater za účelem získání jejich cévního systému. První část práce je věnována projektu LISA a seznámení se základními funkcemi jater. Druhá část práce se zabývá problematikou vybraných metod pro segmentaci jaterních cév (filtrování, prahování, binární operace a automatická segmentace), v současné době používanými lékařskými metodami s poukázáním na jejich časovou náročnost a nepřesnost a seznámením s programovacím jazykem Python. Závěrečná část práce srovnává histogramy jater a jejich cévního systému pro různá dodaná CT data.

Klíčová slova:

zpracování obrazu, LISA, játra, segmentace jaterních cév, CT data, cévní systém jater, Python, histogramy jater a jaterních cévních systémů

Summary

This work deals with the processing of liver CT images in order to obtain their vascular system. The first part is devoted to the project LISA and familiarization with the liver basic functions. The second part deals with selected methods for liver veins segmentation (filtering, thresholding, binary operations and automatic segmentation), currently used medical methods and reference to their time consumption and inaccuracy and introduction to the programming language Python. The final part compares histograms of liver and vascular system for various supplied CT data.

Key words:

image processing, LISA, liver, liver veins segmentation, CT data, liver vein system, Python, histograms of liver and of liver vascular systems

Obsah

Seznam obrázků

1 Úvod

Před operačním zákrokem se lékaři snaží získat co nejvíce informací o místech zásahu nebo orgánech pacienta, kterého plánují operovat. K tomu nejčastěji využívají různé softwarové nástroje. Vždy ovšem existuje možnost dané počítačové programy vylepšovat, přidávat nové funkce nebo vytvořit programy jiné.

Základem pro pochopení této práce je fakt, že aktuální lékařské metody jsou časově náročné a nepřesné (manuální odhady) ve výpočtech při odhadnutí řezu lidskými játry. Lékaři potřebují najít specifický řez jater pro ty pacienty, kteří akutně vyžadují operaci, a k tomu je nutné znát rozložení cévního stromu daného orgánu, neboť špatně vedený řez může zapříčinit smrt pacienta.

Problematika je řešena rekonstrukcí 3D modelu jater na základě CT snímků (2D), následnou segmentací za účelem nalezení cévního stromu jater (obsah této práce) a určení specifického řezu játry.

Práce byla řešena pod vedením Ing. Miroslava Jiříka na katedře kybernetiky již v rámci předmětu KKY-PRJ3 při zpracování projektu pro Fakultní nemocnici Plzeň.

Cíle bakalářské práce:

- do modulu pro segmentaci jater naprogramovat a otestovat modul pro manuální a automatické prahování jater za účelem zjištění cévního systému
- kromě testování funkce naprogramovaného modulu zmínit různá úskalí prahování jater a vypracovat zprávu na základě několika ukázkových

dat a tato data vyhodnotit

Než bude možné zhodnotit výsledky naměřených dat, je nutné se seznámit se základy souvisejícími s danou problematikou - a to jsou:

- játra a jejich struktura
- projekt Lisa
- programovací jazyk Python
- metody použité pro segmentaci jater

1.1 Játra

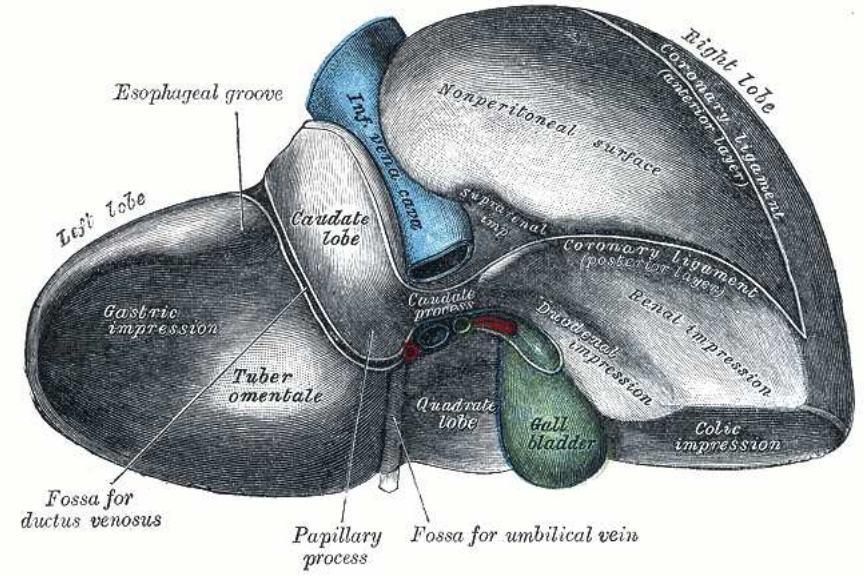
Játra[4] (řecky hepar), jsou největším vnitřním a centrálním orgánem látkové výměny. Mají klíčovou roli při detoxikaci organismu a v metabolismu sacharidů, tuků i bílkovin. Jsou trávicí a endokrinní žlázou (tvorba některých hormonů), ukládají glykogen, železo a některé vitamíny a vytvářejí bílkoviny krevní plasmy.

Játra se skládají ze čtyř jaterních laloků – pravého laloku, levého laloku, lobus quadratus (vpředu mezi pravým a levým), lobus caudatus (vzadu mezi pravým a levým). Jaterní laloky viz obrázek 1.

Játra leží v horní pravé části břicha a váží okolo 1,5 kg. Protékají jimi až 2 litry krve za minutu. K játrům vede jaterní tepna, která přivádí okysličenou krev, a vrátnicová žíla, která přivádí krev nasycenou vstřebanými aminokyselinami a sacharidy z žaludku a střev.

Selhání jater nebo jejich funkcí je život ohrožující stav, objevují se otoky, krvácivé stavy a poruchy funkce mozku. Při CT, MR (magnetická rezonance)

Obrázek 1: Obrázek jater



nebo US (ultrazvukové vyšetření) vyšetřeních je možné nalézt tumory a solitární i vícečetné jaterní cysty, které se zobrazují jako okrouhlé nebo oválné, ostře ohraničené léze [1]. Játra se nedají kompletně uměle nahradit, jediná možnost v případě selhání jater je jejich transplantace. K dispozici jsou i metody jaterní dialýzy, které pomohou prodloužit dobu pro možnost transplantace.

Povrch jater je tvořen vazivovým obalem, který se nazývá Glissonovo pouzdro, capsula Glissoni, též tunica fibrosa. Uvnitř se nachází makroskopicky nečleněný jaterní parenchym, kterým probíhají větvící se cévy a žlučové cesty.

Krevní oběh jater je dvojí: funkční a výživný. Jaterní tepna a její větve představují výživný oběh. Z vrátnicové žíly vychází funkční oběh, který přivádí až 90 % krve.

Větve jaterních žil tvoří samostatný cévní strom, jehož větve jsou propletené s větvemi jaterní tepny, vrátnice a žlučovodů.

Jako základ pro jakékoli další zpracování dat se považují CT snímky pacientových jater.

1.2 Program Lisa

Mezi moduly softwarového nástroje Lisa, který implementuje nalezení cévního systému a specifického řezu jater, patří:

- zobrazení a procházení snímků za účelem výběru oblastí zájmu - to jsou pro nás játra - a následné oříznutí dat, aby ostatní moduly mohly pracovat pouze s takovou množinou dat, ke které chceme dělat rozbor,
- **segmentace jaterních cév** (což je náplň této bakalářské práce)
- a hledání specifického řezu jater na základě známého cévního systému.

Na www stránkách informačního systému výzkumu, experimentálního vývoje a aplikací lze nalézt oficiální zadání projektu pod názvem NT13326.

Název projektu[6]: Zvyšování resekability maligních ložiskových procesů pomocí metod zpřesňujících měření perfúzních parametrů zbytkového jaterního parenchymu - počítačem asistované diagnostiky a softwarového modelování.

Cíle řešení[6]: Projekt si klade za cíl vytvořit model jaterní perfúze, který by umožnil přesné stanovení jaterních segmentů a subsegmentů na základě skutečného proporcionálního cévního zásobování (perfúze) jaterního parenchymu a byl vhodnou zpřesňující pomůckou pro stanovení objemu FLR pomocí racionálních metodik a nikoli estimací. Dále by umožnil navržení virtuální resekce jaterních ložiskových procesů. Na něj by navazoval komplexní systém pro výpočetní tomografii umožňující automatickou detekci a analýzu ložiskových lézí, podezřelých z malignity. Výsledky perfúzního CT

vyšetření budou korelovány s bioptickými nálezy kvantitativní histologickou analýzou cévního zásobení maligních ložisek jater a dynamickým kontrastním ultrazvukovým vyšetřením, což přinese informaci o perfúzi tumoru, která je důležitá pro následnou onkologickou léčbu. Výsledky povedou k zvýšení komplexnosti péče o nemocné s CLM a předpokládáme, že se projeví i zintenzivněním přímé onkochirurgické léčby, ze které mohou tito pacienti profitovat.

Řešení projektu je ve stádiu praktického testování[5] (ukázka operace na obrázku 2). Při testování řešení této problematiky se periodicky nalézají nová úskalí a software je nutno neustále upravovat a rozšiřovat o nové funkce.

Obrázek 2: Ukázka operace jater

Meziuniverzitní tým MUDr. Václava Lišky, Ph.D., testuje na Lékařské fakultě UK v Plzni softwarový model jaterní tkáně, který dokáže vypočítat možnosti regenerace orgánu po rozsáhlé operaci[5]. Už brzy by měl pomáhat hlavně onkologicky nemocným. Pro záchranu života pacienta s jaterními me-

tastázami je operace jedinou možností. Játra se jako jeden z mála orgánů dokáží po zásahu relativně dobře zregenerovat. Pooperační schopnost obnovy je ale u každého člověka jiná a odhadnout ji ještě před operací je téměř nemožné. Výsledky týmu Václava Lišky umožní významným způsobem zpřesnit plánování rozsáhlých resekčních výkonů u pacientů, kde hrozí riziko akutního jaterního selhání a tedy i možnost úmrtí pacienta. Jak Václav Liška uvádí, problematika je velmi složitá a nové otázky se objevují prakticky každý měsíc. Mimo jiné v současnosti tým řeší problém 3D rekonstrukce kapilárního systému jater, který je klíčový pro další práci.

2 Metody

V následující kapitole bakalářské práce se zabývám aktuálně používanými lékařskými metodami pro odhad objemu resekátu a hledání optimálního řezu, jazykem Python a v práci použitými metodami pro segmentaci cévního stromu jater.

2.1 Odhad objemu resekátu

Kapitola přináší informace, jak v současné době lékaři řeší segmentaci jater za účelem nalezení roviny řezu jater a výpočtu objemu resekátu. Prvním krokem je získání objektu jater označováním jednotlivých CT snímků a následný odhad rozložení segmentů jater. Na základě získaných výsledků je možné vypočítat objem resekátu. Resekát je část tkáně nebo orgánu odňatá při resekci, což v chirurgii znamená odstranění celého orgánu nebo jeho části.

Získání objektu jater z CT snímků:

- Lékaři pomocí softwaru prochází každý CT snímek a velice detailně ručně vymezují vnější oblast jater. Nejdříve volí hrubé označení a poté se označení dolaďuje. Pokud se játra rozdělí na více dílů, třeba na dva díly, tak je operátor nucen označit vnější oblast každého dílu zvlášť.
- Uvedený postup je velice časově náročný, protože lékař nebo operátor zpracovává téměř každý snímek. Je možné zpracovávat každý druhý nebo i třetí snímek - v tomto případě dochází k interpolaci dat mezi nejbližšími zpracovanými snímky.

Po získání objektu jater je nutné provést odhad rozložení segmentů jater:

- První způsob: Nalezení hlavního větvení portální žíly. Pojem hlavní v tomto případě znamená první, druhé nebo třetí větvení portální žíly.

Většinou však lékařům postačí najít pouze první. V případě nalezení prvního větvení se játra pomyslně rozdělí na levou a pravou polovinu (dva segmenty - pokud se hledalo jiné větvení, odpovídá tomu i jiný počet segmentů - pro třetí větvení máme osm segmentů jater), což lékařům před operací plně postačí. Určení těchto segmentů se neprovádí softwarově, ale manuálně.

- Druhý způsob: Určení segmentů na základě střední jaterní žíly. Tato žíla se nazývá také drenážní (odvádějící krev). Lékař předpokládá, že pod střední jaterní žílou vede pomyslná středová hranice obou polovin jater (toto rozhodnutí plyne bez jakékoliv segmentace), a ještě přesněji určí směr dělicí roviny podle nepsaného pravidla, že hranice žlučníku je zároveň částí roviny, která dělí játra na poloviny. Podle získaných informací lékaři dále rozhodnou o řezu.

Dle mého zhodnocení jsou lékařské metody buď velice časově náročné nebo velice nepřesné, ačkoli získané informace jsou pro praxi operujícího lékaře postačující. Program LISA popsaný proces zjištění řezu značně ulehčuje (operátor nemusí procházet každý snímek) a upřesňuje (není nutné manuálně určovat důležité oblasti). Přesto mají výsledná data pouze informativní charakter. Rozhodně se nejedná o odborný návod, jakým způsobem a na jakém místě jater provést lékařský zákrok - situaci musí vždy posoudit sám lékař.

2.2 Python

Python je dynamický, objektově orientovaný skriptovací programovací jazyk navržený v roce 1991[3]. Tento programovácí jazyk je vyvíjen jako open source s velkým počtem dodatečných knihoven a díky těmto kontribucím a příspěvkům je jedním z nejrozšířenějších skriptovacích jazyků.

Obrázek 3: Python Hello world

Python patří mezi skriptovací a hybridní jazyky[7]. Umožňuje objektově orientovaná i procedurální paradigmata a v omezené míře i funkcionální. Byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací (včetně GUI — viz například wxPython, který využívá wxWidgets, nebo PySide a PyQT pro Qt a nebo PyGTK pro GTK+).

Nejnovější verzovací řada se uvádí pod označením v3.x, aktuálně v3.4 vydaná 17. března 2014.

2.2.1 Python a jiné jazyky

Python se snadno vkládá do jiných aplikací (embedding), kde pak slouží jako jejich skriptovací jazyk. Tím lze aplikacím psaným v kompilovaných programovacích jazycích dodávat chybějící pružnost. Jiné aplikace nebo aplikační knihovny mohou naopak implementovat rozhraní, které umožní jejich použití v roli pythonovského modulu.

Programátor dokáže velice rychle vytvářet prototypy všeho druhu, které je však poté nutno optimalizovat a zrychlit. V rychlosti vývoje je Python vedoucím jazykem.

Známé implementace v ostatních jazycích jsou Jython (v Javě), CPython (v C/C++) a IronPython (.NET).

Samotný jazyk je například 3 až 5krát rychlejší než PHP. Navíc existuje

snadno použitelná knihovna Psyco[8] (nyní nahrazena knihovnou Pypy), která optimalizuje kód Pythonu na výkon. Některé operace jsou pomocí Psyco urychleny až řádově. Psyco je však podporována pouze pro Intel-386 kompatibilní procesory a využívá hodně operační paměti. Pypy[9] je podporován procesory kompatibilními s Intel x86 (IA-32) a ARM.

Výkonově kritické knihovny jsou většinou implementovány v jazyce C a oproti němu je kód v Pythonu stabilní, robustní a rychle vytvořen, ale je pomalejší v rychlosti provedení instrukcí. Jazyk C++ odpovídá rychlostně úrovni jazyka C, jeho zápisy kódu jsou však modernější a v závěru daleko přehlednější; postrádá ovšem například standardní GUI balíčky pro vývoj jako jazyk C#.

Ve srovnání s programem Matlab je volně dostupný (Matlab licence nepatří mezi nejlevnější). Vědecké knihovny (jako numpy a matplotlib) jsou v posledních letech daleko lépe udržované, a to Python jednoznačně posouvá před Matlab. Největším rozdílem však zůstává, že Matlab není programovacím jazykem, ale programem, který používá pro výpočty a simulace lineární algebru. Tudíž pokud jde o nepočetní problémy, tak Python, námi zvolený jazyk, vyhrává.

2.2.2 Použité knihovny Pythonu

Mezi vybrané a důležité interní a externí použité knihovny Pythonu během vypracování bakalářské práce patří:

- Numpy[10] - nepostradatelná knihovna pro vědecké a matematické výpočty. Obsahuje podporu N-dimenzionálních polí, broadcast funkce, metody implementující lineární algebru, Fourierovy transformace, generování náhodných čísel a nástroje pro integraci do jazyků C, C++

a Fortran. Díky efektivnímu multidimenzionálnímu ukládání dat je knihovnu Numpy možno integrovat se širokou paletou databází.

- Scipy[11] - tento balík knihoven používající Numpy obsahuje například:
  - CPython - rozšíření syntaxe Pythonu o podporu konstukcí jazyka C za účelem zrychlení kritických míst v kódu nebo za účelem integrace s C a C++ knihovnami,
  - Matplotlib - 2D a 3D vykreslování dat
  - a „nose“ - framework pro testování zdrojových kódů v Pythonu.
- cPickle - již v Pythonu implementovaná knihovna pro ukládání a načítání jakýchkoli objektů. Existuje i knihovna Pickle, ale knihovna cPickle je, odhadem již podle názvu, naprogramována v jazyce C, a je proto řádově rychlejší a také efektivnější co se týče zabíraného místa dat.
- PP[12] - Parallel Python Library and Job Server - velice rychlá a spolehlivá knihovna pro paralelní výpočty dat dle zadání programátora.

2.3 Metody zpracování obrazu

K metodám zpracování obrazu se úzce váže i segmentace dat. Pod pojmem segmentace si můžeme představit rozdělení celku na menší části podle určitých pravidel a následné zpracování rozdělených dat. Dalším významem může být získání specifických dat z dat jiných (hledání určité podmnožiny)[2].

Postup segmentace jaterních cév: z CT snímků jater se vytvoří 3D reprezentace dat a na základě jejich rozdělení a filtrování získáme cévní strom. Ideální

by samozřejmě bylo pouze potvrzení dat a jednorázové nalezení cévního stromu, ale kvůli různorodosti lidských jater není možné naprogramovat univerzální algoritmus pro nalezení správného prahu pro všechna data. Existuje mnoho metod, jak najít střední nebo vyvážený práh, ale ty našim účelům nijak neposlouží, protože hledáme určitou část dat a nezajímá nás jejich vyvážení.

Největší problém spočívá v různorodosti dat - metody a určité konstanty uplatněné u jedněch dat nemusí nebo spíše vůbec nebudou pasovat na data jiná. V rámci této bakalářské práce je do určité míry naprogramováno automatické prahování, které slouží jako výborný začátek pro další úpravy a hledání uspokojivého tvaru výstupních dat.

Do programu segmentace jater vypracovaného v rámci bakalářské práce je také zabudován jeden parametr, který může být dle okolností i tím nejdůležitějším parametrem spolu s nastaveným prahem. Tento parametr označuje vrácení počtu největších nalezených objektů a v aplikaci LISA je vždy nastaven na hodnotu 1.

Operátor může pro jedna data najít několik optimálních rozložení parametrů - záleží na požadavku, jakou strukturu mají mít výsledná data. Jako výsledná data je možno mít cévní strom i s nejméně rozlišitelnými cévami, základní větve (dělení) cévního stromu nebo základní cévní strom s cévami střední velikosti.

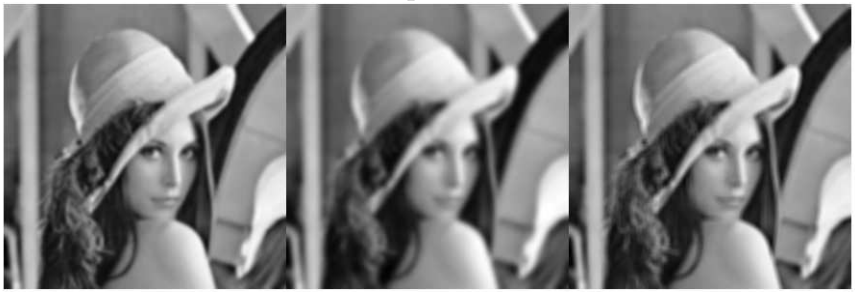
2.3.1 Filtrování dat

Gaussovský filtr umožňuje použití několika druhů parametrů, které ovlivní rozmazání dat. Nejdůležitější parametr představuje proměnná sigma, která je směrodatnou odchylkou, podle níž se řídí rozmazání. Může být nejen kon-

stantou, ale i vektorem, což je v našem případě absolutně ideální, protože rozměry voxelu nemusí být stejné (nemusí se jednat o krychli) a díky vektoru jsme schopni nastavit větší rozmazání v „zahuštěnějším“ směru dat (většinou se jedná právě o směr, který odpovídá samotné šířce řezů).

Obrázek 4 znázorňuje příklad použití Gaussova filtru a pro porovnání i příklad uniformního filtru. Na první a druhou část obrázku je aplikován Gaussův filtr se sigmou 3, poté 5. Třetí část obrázku představuje uniformní filtr s parametrem size roven 11. Z příkladu vyplývá zásadní rozdíl mezi první a druhou částí - parametr sigma je velice citlivý - stačí rozdíl 2 a data ihned vypadají jinak.

Obrázek 4: Ukázka použití Gaussova filtru



2.3.2 Prahování

Segmentace prahováním patří mezi nejjednodušší segmentační postupy. Mnoho objektů nebo oblastí dat je charakterizováno konstantní „odrazivostí“ či „pohltivostí“ svého povrchu. Potom je možné využít určené jasové konstanty prahu k oddělení objektů od pozadí - v našem případě považujeme za pomyslné pozadí jaterní tkáň.

Prahování je nejstarší segmentační metodou a v jednoduchých případech je stále používáno. Vzhledem k výpočetní nenáročnosti je nejrychlejší seg-

mentační metodou.

Operaci prahování definujeme jako transformaci vstupního obrazu f na výstupní (segmentovaný) binární obraz g podle vztahu

$$g = \begin{cases} 1, & f \geq T \\ 0, & f < T \end{cases}$$

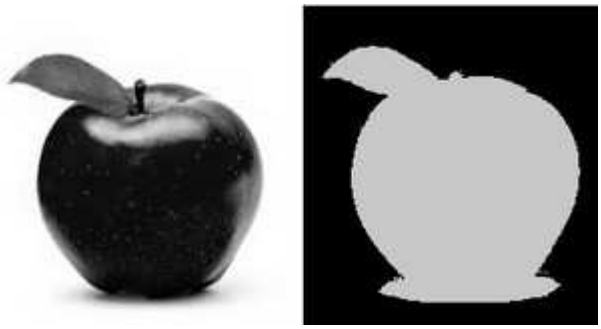
kde T je předem daná konstanta nazývaná práh.

Pokud je třeba mít data po prahování nebinárního charakteru, jsou nutné další úpravy, třeba jako (ukázka z kódu bakalářské práce):

```
if use_min_threshold: data = data * (data >= min_threshold)
if use_max_threshold: data = data * (data < max_threshold)
```

V Pythonu operace $*$ (krát) mezi 2 maticemi vynásobí prvky na stejných pozicích. Potom je možné výsledná data získat se stejnými hodnotami jako vstupní data v místech rovných 1 v binárních datech po prahování. Na obrázku 5 vidíme, jak funguje prahování a oddělení objektu od pozadí.

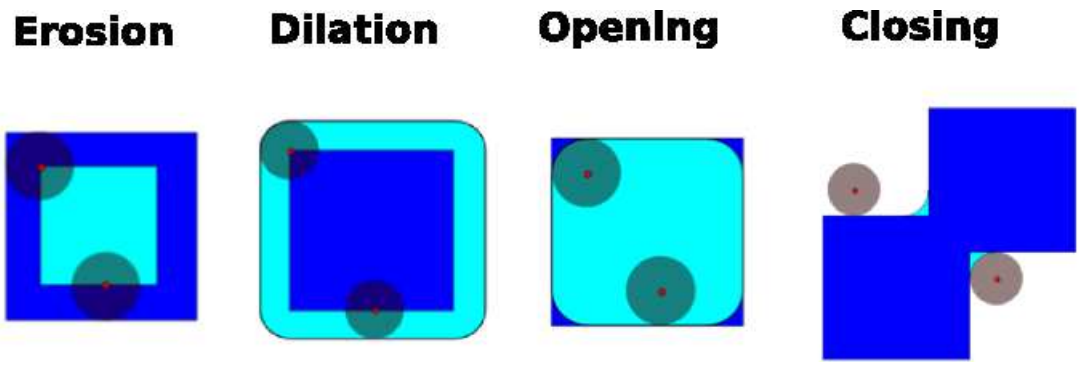
Obrázek 5: Ukázka prahování a vyvážení obrazu



2.3.3 Binární operace

Základem binárních operací je binární eroze a dilatace. Binární otevření a uzavření jsou kombinací operací binární eroze a dilatace. Obrázek 6 zobrazuje princip zmíněných binárních operací.

Obrázek 6: Binární operace eroze, dilatace, otevření a zavření



Binární eroze je matematická morfologická operace, která data podle dané struktury redukuje.

Matice A s dolním indexem podle pořadí iterace binární eroze:

$$A_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Binární dilatace je matematická morfologická operace, která data podle dané struktury rozšiřuje nebo doplňuje.

Matice B s dolním indexem podle pořadí iterace binární dilatace:

$$B_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, B_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, B_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Binární uzavření nejdříve provede erozi a poté dilataci.

Matice C s dolním indexem podle pořadí iterace binárního otevření:

$$C_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, C_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, C_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

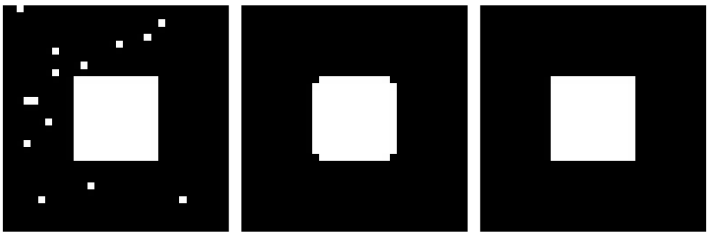
Binární uzavření nejdříve provede dilataci a poté erozi.

Matice D s dolním indexem podle pořadí iterace binárního uzavření:

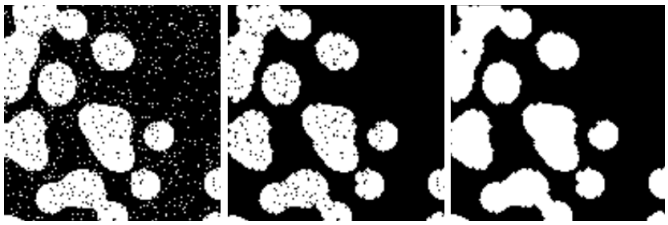
$$D_0 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}, D_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}, D_2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Obrázky 7 a 8 dokumentují použití binárního otevření (zbavení se artefaktů) a poté uzavření (doplnění nebo vyplnění dat).

Obrázek 7: Ukázka binárního otevření a uzavření



Obrázek 8: Ukázka binárního otevření a uzavření 2



3 Výsledky

V této kapitole bakalářské práce se zabývám naprogramovanými skripty, ukázkou jejich funkčnosti a zhodnocením výsledků ukázkových dat.

3.1 Softwarové řešení segmentace jaterních cév v programu Lisa

V rámci řešení projektu a bakalářské práce jsem činný kód rozdělil do několika souborů (skript pro ukázková data, vstupní funkce, segmentace s uživatelským prostředím a knihovna funkcí).

V souboru **PVBPLiver.py** je napsán kód řešící získání výsledků segmentace z ukázkových dat. Skript je součástí pouze bakalářské práce a nijak se nevyužívá při vlastním řešení v běhu programu Lisa. Soubor **segmentation.py** obsahuje vstupní metodu pro segmentaci předaných dat. Během činnosti skriptu si uživatel může vybrat při projekci všech snímků oblasti zájmů (skript poté prioritně vrací vybrané objekty). Soubor **uiThreshold.py** používá knihovnu segmentačních funkcí, do kterých předává parametry zvolené uživatelem z uživatelského prostředí a vizuálně zobrazuje výsledky. Činný kód pro úpravu dat (vlastní segmentace jaterních cév) je možno vidět v příloze 5.2. Soubor **thresholding_functions.py** je knihovnou funkcí pro segmentaci dat (například filtrování, prahování).

Při segmentaci jsem zvolil následující postup operací:

- uplatnění Gaussova filtru - rozmazání dat; data se prahování totiž jeví jako „rozsypaný čaj“, ale výsledná data jsou již vzhledově v pořádku,
- prahování,

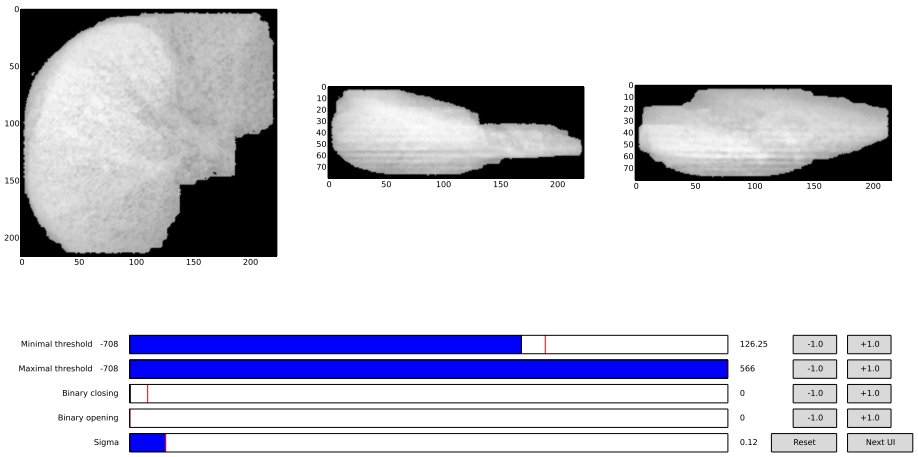
- binární uzavření,
- binární otevření.

3.1.1 Ukázka segmentace jaterních cév

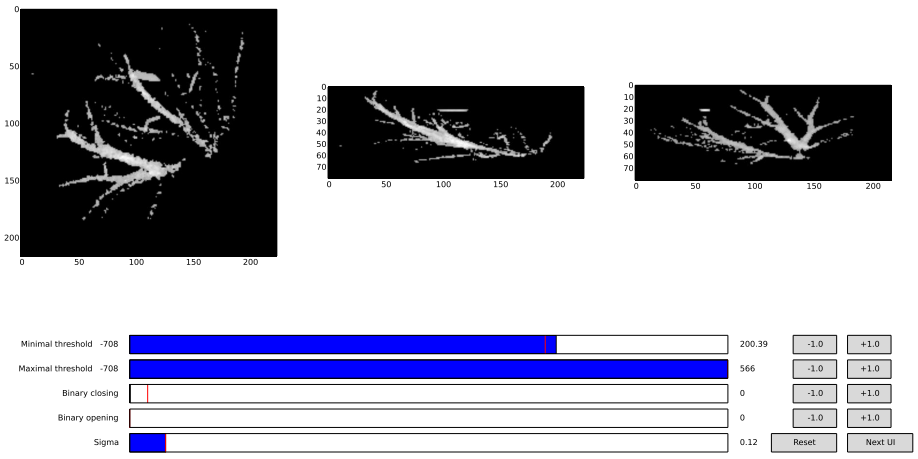
Na obrázku 9 je vidět uživatelské prostředí pro segmentaci. Uživatel si může zvolit minimální a maximální práh, počet binárních uzavření a otevření a velikost parametru sigma. Nad výběrem parametrů jsou tři akumulativní průměty dat v osách x, y a z po uplatnění vybraných hodnot parametrů na vstupní data. V hodnotách prahu je vidět rozmezí nastavitelných hodnot v hounsfieldových jednotkách HU, které lékaři běžně používají. Pro tato data se jedná o interval od -708 do +566.

Pro ukázku průběhu vlastní segmentace použiji data, která již byla na obrázku 9. Pro tato data jsem nezvolil vrácení největších objektů ani vrácení zájmových oblastí (nevybral jsem si část dat k prioritnímu vrácení). Po zvolení prahu 200 (obrázek 10) vypadá výstup dat rozsypaně a je zřetelně vidět, že chybí velké části cév. Po zmenšení minimálního prahu na 176 (obrázek 11) došlo k doplnění chybějících částí cév, ale už vidíme přebytečný šum. Pro odstranění přebytečného šumu jsem změnil parametr binárního otevření (eroze a potom dilatace dat) z hodnoty 0 na 1 (obrázek 12). Na první pohled je zřejmá obrovská citlivost v parametru binárního otevření stejně jako je citlivý parametr prahu. Po uvedené úpravě jsem se zbavil šumu, ale opět došlo k tomu, že chybí části cév.

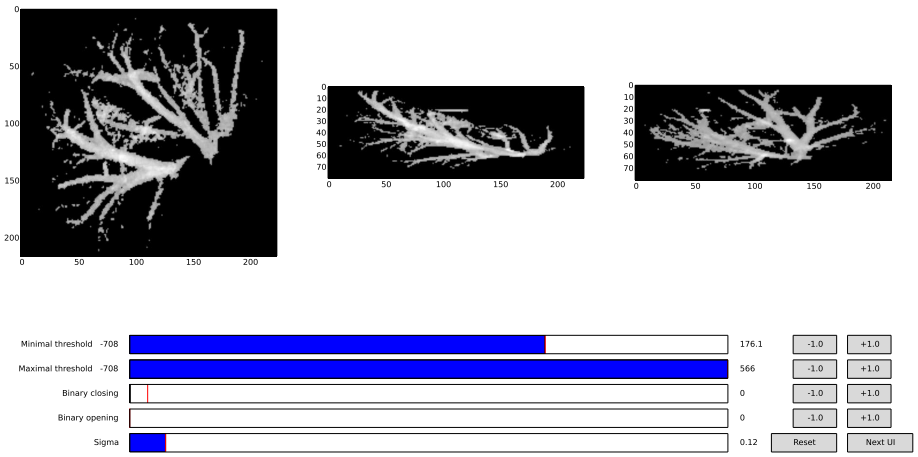
Obrázek 9: Uživatelské rozhraní se vstupními daty



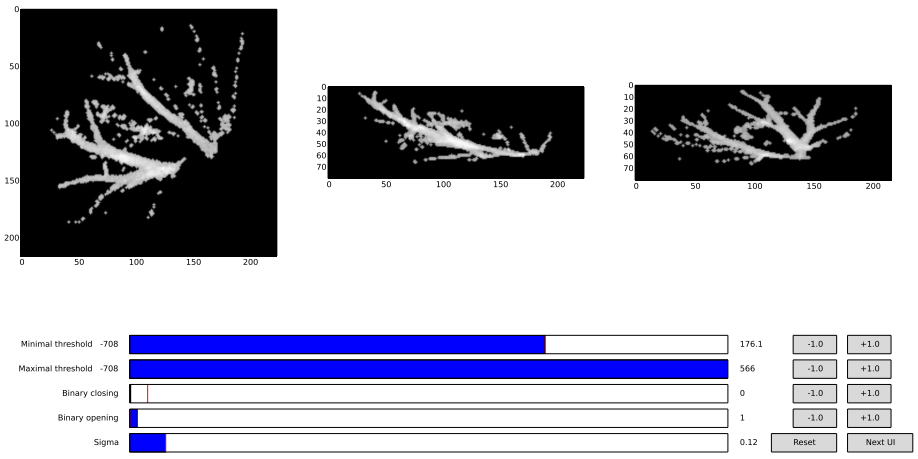
Obrázek 10: Data s neoptimálním prahem



Obrázek 11: Zašuměná data s optimálním prahem

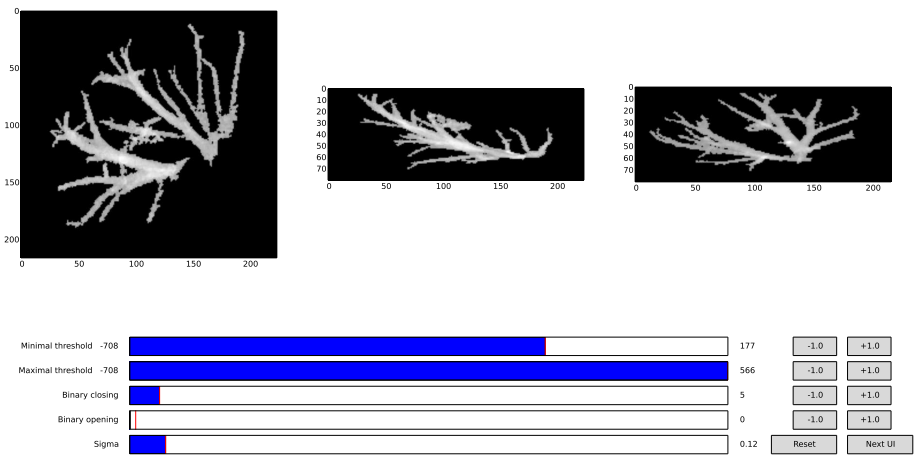


Obrázek 12: Data po binárním otevření



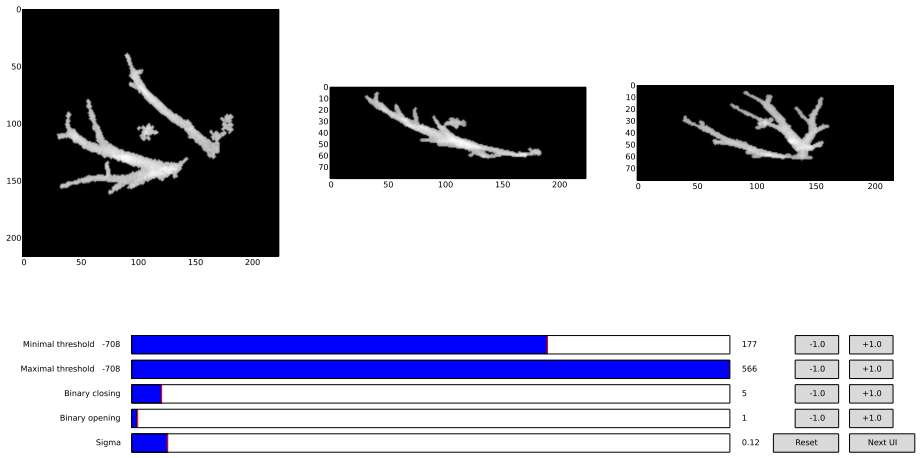
Problém chybějících částí cév lze napravit přenastavením počtu vrácených objektů z 0 (0 je standardní hodnota parametru pro vracení všech objektů) na 3 objekty (program Lisa má standardní hodnotu nastavenou na 1 objekt). Výsledek pro minimální práh 177 je vidět na obrázku 13. Mimo jiné jsem i nastavil parametr binárního uzavření na 5 (tento parametr je málo citlivý a hodnota 5 zajistí doplnění malých prázdných míst uvnitř dat a krátkých mezer mezi vynechanými místy cév). Vrácená data už mají dobrou kvalitu.

Obrázek 13: Data s vrácením 3 největších objektů



Pro další demonstraci velké citlivosti parametru binárního otevření jsem nastavil parametr z hodnoty 0 (z obrázku 13) na 1. Výsledek je vidět na obrázku 14. Tímto krokem se vytratila jakákoliv informativnost výstupních dat, protože se ztratila minimálně polovina cévního systému. Uvedený krok je možné stále napravit. Je možné se vrátit zpět a vypnout binární otevření nebo je možné snížit minimální práh.

Obrázek 14: Data po vrácení největších objektů a binárním otevření



3.1.2 Automatická segmentace

Během programování jsem se pokusil vytvořit určitou funkci pro vrácení ideálního prahu pro prahování, ale tato problematika je pro naše data, játra, složitější než u prahování prostých obrázků, kde je snaha oddělit nebo rozlišit popředí od pozadí nebo jednotlivé objekty od objektů jiných.

Existují určité algoritmy pro výpočet ideálního prahu. Po jejich rychlé a jednoduché implementaci jsem ale zjistil, že nehledám rozdělení dat

vyvážené, ale specifické. V článku[13] o segmentaci jaterních cév jsem se dočetl o možném nalezení hledaného prahu a to aproximací pomocí dvou regresních přímek. V tomto případě se počet voxelů od optimálního prahu po konečný práh lineárně snižuje a od určitého maxima histogramu jater počet voxelů lineárně klesá k optimálnímu prahu, u kterého se nachází zlom.

Po otestování a zprovoznění procedury automatické segmentace jsem došel k názoru, že je tato procedura vhodná pro odhadnutí prvního (počátečního) prahu, od kterého by měl operátor hledat optimální hodnotu. Uvedená procedura se nedá univerzálně uplatnit na všechna data, ale u většiny dat velmi urychlí jejich úpravu - algoritmus najde užitečnou hodnotu prahu, která se ve většině případů blíží ideálnímu prahu, a pro operátora již není problém vyzkoušet několik blízkých prahů.

3.2 Zpracování ukázkových dat

K demonstraci naprogramovaného modulu pro segmentaci jaterních cév bylo vybráno 22 souborů vstupních dat. Výsledky jsem zpracoval a posoudil s využitím histogramů a statistického rozboru dat. Statistický rozbor jsem provedl nad třemi množinami hodnot. První množina hodnot zahrnuje prahy maxim histogramů jater, druhá množina obsahuje hodnoty zvolených prahů pro segmentaci cév a třetí množina zahrnuje rozdíly dvojice hodnot z první a druhé množiny.

Každý histogram představuje křivku, která nás informuje o počtu jednotlivých voxelů pro daný práh. Ke zhodnocení výsledků všech 22 dat jsem vypracoval 3 histogramy:

- 1. histogram tvoří originální data (játra s cévami),

- 2. histogram tvoří pouze cévy,
- 3. histogram tvoří originální data bez cév (játra bez cév).

U každého ze vstupních dat jsem manuální segmentací zjistil potřebné parametry k získání cévního systému jater. Tyto parametry jsem použil pro automatickou paralelní segmentaci. Na základě vstupních a výstupních dat jsem vypracoval všechny potřebné histogramy.

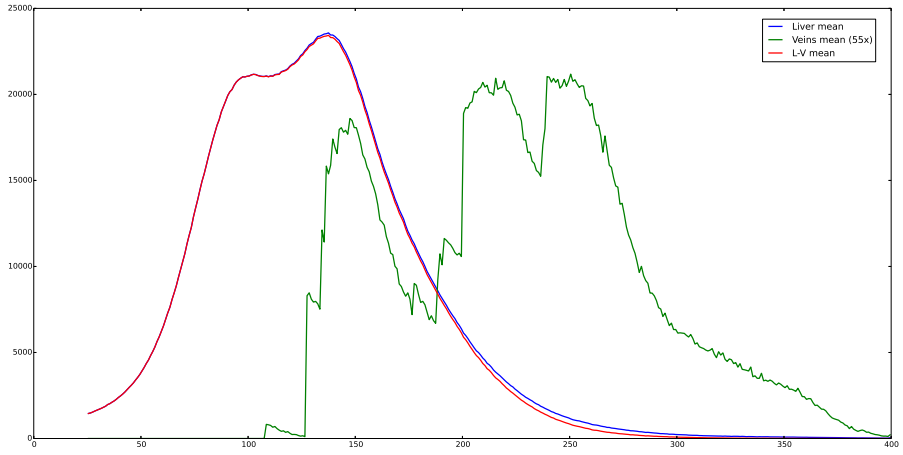
U výsledků nemohu přesně stanovit, zda jsou dobré nebo špatné, ale na základě výsledků mohu hovořit o dobrém nebo špatném formátu nebo uskupení vstupních dat. Dobrá vstupní data jsou taková, u kterých jde jaterní tkáň relativně jednoduše a snadno oddělit od cév.

3.3 Výsledky ukázkových dat

U všech 22 dat jsem použil automatickou segmentaci s malými manuálními úpravami v parametrech. Největší změnu v parametrech jsem zaznamenal u hodnoty prahu, poté u množství hledaných objektů a dále u hodnot binárního otevření a zavření - všechny tyto parametry pro daná data jsou uvedeny v příloze 5.1 u souboru „PVBPLiver.py“.

Následně jsem pro srovnání výsledků vypočítal průměrné hodnoty histogramů a zakreslil je do jednoho grafu - viz obrázek 15. Na ose X se nachází práh v hounsfieldových jednotkách (značené HU). Osa Y je oborem hodnot a reprezentuje počet voxelů pro daný práh. Modrá křivka reprezentuje histogram jater - vstupní data. Zelená křivka znázorňuje 55krát zvětšený histogram cév - pokud bychom měli křivky srovnat bez tohoto zvětšení, nebyly by výsledky vizuálně názorné. Červená křivka představuje rozdíl histogramu jater a nezvětšeného histogramu cév.

Obrázek 15: Průměrné křivky histogramů jater, cév a jater bez cév



V následujících třech bodech jsou vypsány přibližné maximální hodnoty histogramů:

- Histogram jater:

$$max(y_{liver}) \doteq 24000$$

- Histogram cév:

$$y^*_{veins} = 55 \cdot y_{veins} \Longrightarrow max(y_{veins}) \doteq 21000 \div 55 \doteq 382$$

- Histogram jater bez cév:

$$y_{lv} = y_{liver} - y_{veins} \doteq (100\% - 1.6\%) \cdot y_{liver} = 98.4\% \cdot y_{liver}$$

Zajímavý výsledek jsem zaznamenal u histogramu bez cév, který se pro některé hodnoty prahů liší od histogramu jater maximálně o 1,6 %.

Z histogramu cév (dále jen „HC“) (zelená křivka) vyplývá, že během měření a zpracování vybraných dat nebyl použit práh menší než 108 (tuto hodnotu lze najít v příloze 5.1 u souboru „PVBPLiver.py“).

Z průběhu HC je možno vyčíst, že nejsou všechna data přibližně stejná. Na grafu vidíme, že křivka má 3 lokální maxima (prahy 150, 220, 255) a dále několik menších výkyvů, 3 lokální minima (prahy 125, 180, 240), přičemž práh 125 se nachází v okolí použitého minima prahu (108).

Z výše uvedeného průběhu (obrázek 15) lze usoudit, že všechna data poskytují zcela rozdílné výsledky.

Pojmy použité při popisu histogramů:

- rychlý průběh - tj. takový průběh histogramu, který obsahuje málo vypovídajících kladných hodnot histogramu pro různé hodnoty prahu (přibližně 100 a méně jednotek prahu v ose x)
- pomalý průběh - tj. takový průběh histogramu, který obsahuje hodně vypovídajících kladných hodnot histogramu pro různé hodnoty prahu (přibližně více než 100 jednotek prahu v ose x)

Jako dobře nasnímaná CT data se dají označit ta, jejichž histogramy cév mají jak pomalý, tak rychlý průběh. Některé histogramy, které mají rychlé průběhy, mohou naznačovat, že CT snímky nebyly pořízeny ve zcela vhodný čas (kontrastní látka ještě nebyla ve větší části cév) nebo se cévní systém nedal zcela vhodně oddělit od jater, a proto je průběh histogramu tak rychlý (chybí některé hodnoty).

Výsledné HC všech dat jsou velice rozdílné. Tento fakt ihned vypovídá o velkých problémech při programování automatické segmentace (kapitola 3.1.2). O problémech s odhadem prahu se lze přesvědčit při statistickém zpracování 2 souborů hodnot - prahy globálních maximálních hodnot histogramů jater a zvolených optimálních prahů pro segmentaci cév. První soubor hodnot

je v tabulce výsledků označen jako „Data 1“, druhý soubor dat je označen jako „Data 2“ a třetí statistika „Data 3“ se týká vzájemného srovnání hodnot mezi prvními dvěma soubory (srovnání dvojic maxima hodnoty histogramu jater a zvoleného prahu).

	Data 1	**Data 2**	**Data 3**
x_{min}	85,5	108,5	—
x_{max}	181,5	239,5	—
$\bar{x}$	121,11	168,16	47,05
s_x^2	956,52	1648,69	2535,68
s_x	30,93	40,6	50,36

Z charakteristiky dat 1 je patrný rozdíl v polohách maxim histogramů jater - prahy jsou volně rozmístěné přibližně přes 100 jednotek ($x_{1max} - x_{1min} = 181,5 - 85,5 \doteq 100$).

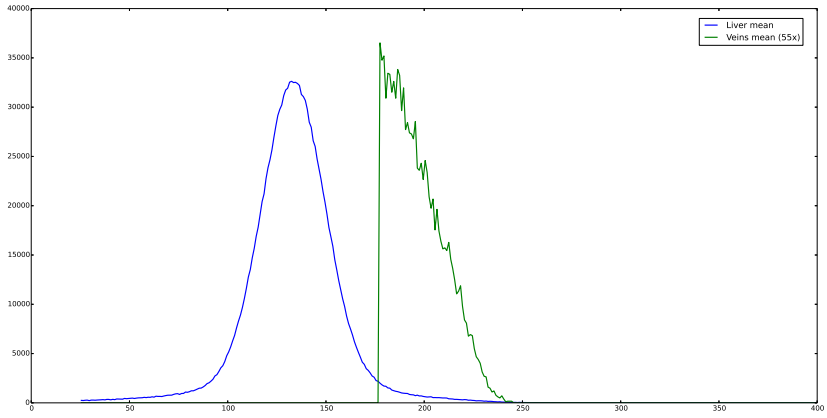
V souboru druhých dat je rozdíl v délce intervalu zvolených prahů přibližně o 30 % větší než rozdíl poloh maxim histogramů jater ($x_{2max} - x_{2min} = 239,5 - 108,5 \doteq 130$) a jsou velké odchylky i v rozptylu a směrodatné odchylce ($s_{2x} \doteq s_{1x} + 10$).

Hodnota průměru třetích dat ukazuje průměrný rozdíl daných dvojic ($\bar{x_2} - \bar{x_1} = 168,16 - 121,11 = 47,05 = \bar{x_3}$). Rozptyl a směrodatná odchylka jsou větší než u prvních dvou souborů dat ($s_{3x} \doteq s_{2x} + 10 \doteq s_{1x} + 20$). Získané hodnoty po srovnání jednotlivých dvojic dat jednoznačně ukazují na náhodné rozmístění dvojic maxima histogramů jater a zvoleného prahu, a tudíž na problémy při automatickém odhadu optimálního prahu, protože se podle pro-

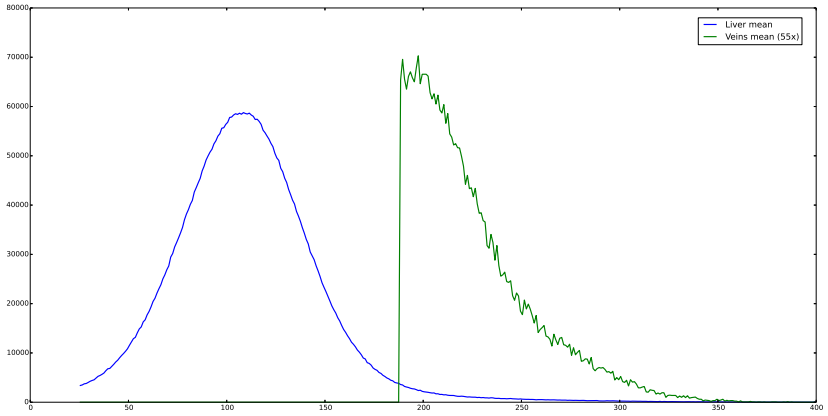
vedené statistiky dat nedá jednoznačně určit optimální práh pro jaterní cévy z hodnoty prahu maxima histogramu jater.

Jednotlivé výsledné histogramy se dají rozdělit na 2 skupiny. První základní skupinou jsou grafy s rychlým nebo pomalým a monotónním průběhem HC. Do této skupiny patří obrázky 16, 17, 18 a 19. Do druhé skupiny se dají zařadit jakékoli histogramy cév, jejichž průběh obsahuje závažnější anomálie. Takové anomálie je možné vidět na obrázcích 20, 21 a 22. Dané grafy s anomáliemi se vyznačují tím, že při klesání HC najednou začnou stoupat hodnoty, a vyskytne se tak druhé lokální maximum pro HC. Tento jev je možné popsat tak, že se od určitého prahu vyskytne shluk světlých cév. Uvedený jev druhého lokálního maxima je nejpatrnější na obrázku 22.

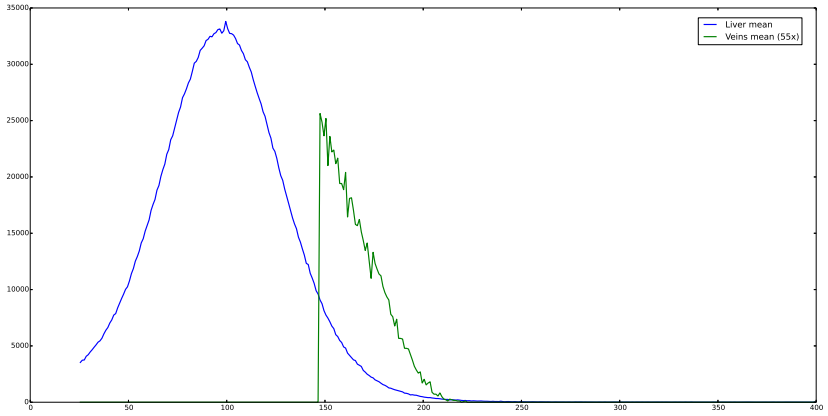
Obrázek 16: Klesající a rychlý průběh histogramu cév



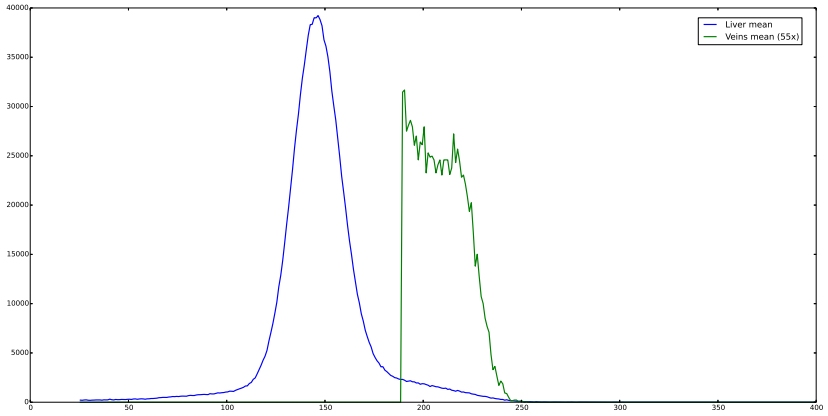
Obrázek 17: Klesající a pomalý průběh histogramu cév



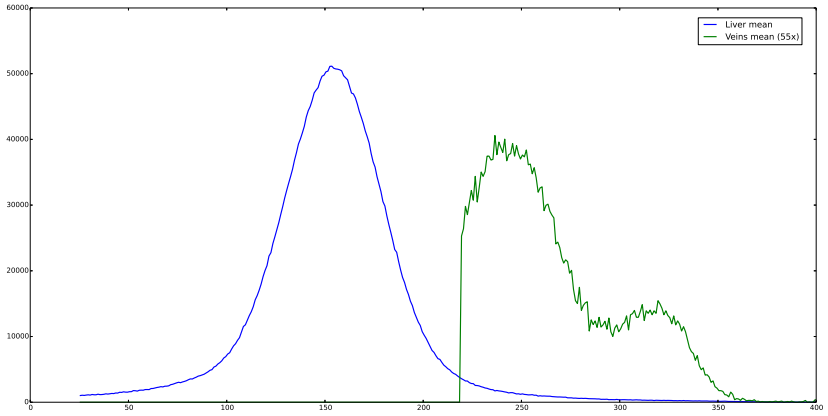
Obrázek 18: Klesající a rychlý průběh histogramu cév 2



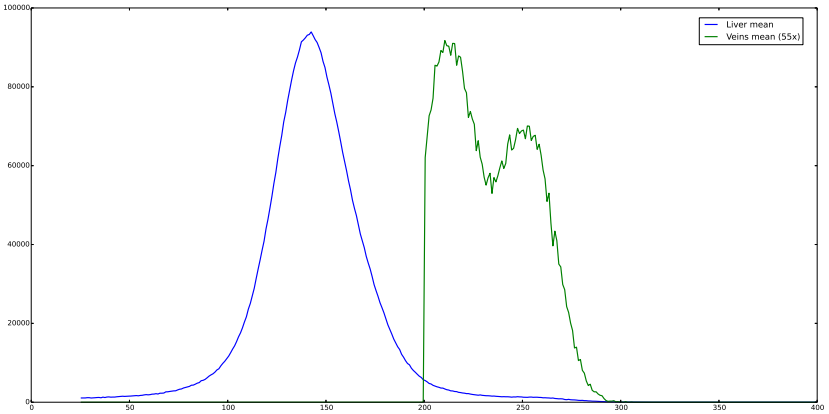
Obrázek 19: Klesající a rychlý průběh histogramu cév 3



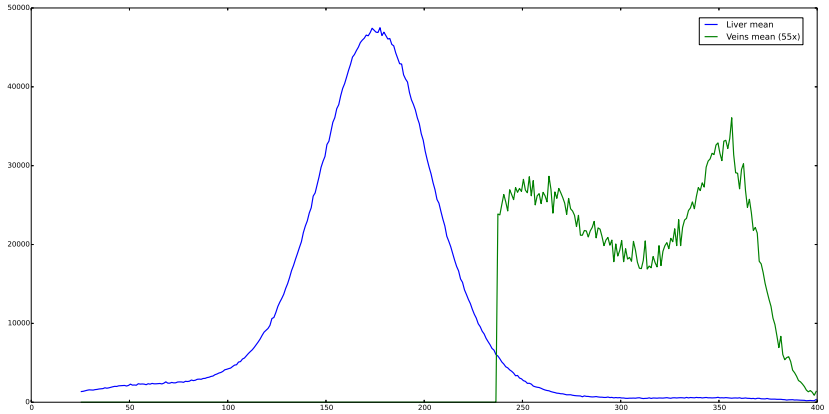
Obrázek 20: Pomalý a nemonotónní průběh s dvěma lokálními maximy



Obrázek 21: Dvě lokální maxima a nemonotónní průběh



Obrázek 22: Ukázka anomálie histogramu cév v druhém lokálním maximu



4 Závěr

Cílem mé bakalářské práce bylo seznámit se s různými metodami pro segmentaci a zpracování obrazu, dále se seznámit s lékařskými postupy pro odhad objemu resekátu, naprogramovat funkce zajišťující segmentaci cév (manuální i automatickou), tento soubor funkcí otestovat na vybraných datech a vyhodnotit výsledky z hlediska splnění funkce naprogramovaných funkcí a z hlediska naměřených dat.

Projekt NT13326[6], který řeší software LISA (kapitola 1.2), do něhož jsem dané funkce pro segmentaci jaterních cév implementoval, se nazývá počítačem asistovanou diagnostikou nalezení řezu játry v potřebné oblasti. Tento postup nemocničním operátorům nebo lékařům, kteří praktikují lékařské metody segmentace a nalezení řezu (označování vnější jaterní tkáně na každém snímku a následném odhadování poloviny jater na základě nepsaných pravidel, více v kapitole 2.1), značně usnadní práci z hlediska času a také značně upřesní polohu optimálního řezu.

Při zpracování vybraných dat jsem se seznámil s různými knihovnami jazyka Python. Pro vykreslení výsledků a průběžnou kontrolu vypočtených dat jsem použil knihovnu Matplotlib, která slouží k vykreslování dat do grafu. Pro rychlé zpracování jsem použil knihovnu PP[12] (Parallel Python Library and Job Server) - použil jsem ji pro paralelní zpracování všech dat najednou, což se se ukázalo jako velice výhodné pro konečné, drobné a zdlouhavé optimalizace algoritmů a vykreslování a ukládání dat a histogramů.

Pro segmentaci jaterního cévního systému jsem použil apriorní informaci od uživatele, který vybere několik voxelů příslušících objektu, o němž ví, že ho chce ve výstupních datech vrátit. Dále jsem použil Gaussův filtr (rozmazání

dat), prahování dat, binární uzavření (zaplnění prázdných míst uvnitř shluků a spojení sousedních voxelů), binární otevření (odebrání artefaktů) a na závěr výběr největších kompaktních shluků (prakticky pouze 1 největší shluk mimo hodnoty 0).

Největší potíže působila snaha naprogramovat automatickou segmentaci (hledání optimálního prahu) a následné občasné korekce všech algoritmů. Automatickou segmentaci se mi z části podařilo naprogramovat - odhadnutý práh je vynikajícím údajem, od jakého prahu by měl operátor hledat optimální práh. Ačkoliv na většině dat odhadnutá hodnota není zcela optimální, přece jen se nachází v jeho okolí.

V rámci zpracování vybraných dat jsem vytvořil histogramy pro srovnání jednotlivých výsledků s průměrným výsledkem a zhodnotil jsem charakter výstupních dat naprogramovaného modulu - za výstup se považuje cévní strom jater. Pro všechna data jsem vytvořil 3 histogramy - histogram jater (nesegmentovaných vstupních dat), histogram cév (výstupní data) a histogram jater bez cév (od vstupních dat jsem odebral průnik s daty výstupními).

U výsledných dat, histogramu cév, jsem z hlediska vizualizace musel změnit měřítko, protože hodnoty histogramu cév dosahovaly v průměru nejvíce 1,6 % hodnot histogramu jater (původních vstupních dat) a v této formě vykreslená v grafu nepodají téměř žádnou informaci. Po změně měřítka (55krát vynásobená výstupní data) již ve výsledcích (grafy histogramů) podávají určitou informaci a dají se přehledně srovnat s histogramem původních dat.

O charakteru vybraných dat nejvíce vypovídá obrázek průměrných histogramů (obrázek 15), na kterém je vidět, že se výstupní data velice liší a že nelze zcela uplatnit parametry použité u jedněch dat na data jiná.

Výstupní data vykazují veliké odchylky v hodnotách histogramů. Několikrát jsou výsledné histogramy i nemonotónního charakteru a nejsou výjimečné ani výskyty anomálií (více lokálních maxim histogramů cév).

Velikou motivací při zpracování této práce a při programování algoritmů do programu Lisa bylo, že se všechny algoritmy využijí v praxi pro získání informací, které lidem mohou zachránit život. Veškeré algoritmy se již testují v praxi a program Lisa lékařům usnadňuje pohled na cévní systém před operací jater[5].

Reference

[1] NEKULA, Josef; HEŘMAN, Miroslav, et al. Radiologie. 3. vyd. Olomouc : Univerzita Palackého v Olomouci, 2005. Dotisk 2008. ISBN 978-80-244-1011-7. S. 205. (cs)

[2] Sonka M., Hlavac V. , Boyle R.: Image Processing, Analysis, and Machine Vision, 3rd edition, Thomson Learning, Toronto, April 2007, 821 p, ISBN 049508252X (2nd edition Brooks/Cole, Pacific Grove, CA, 1999, 1st edition Chapman Hall, London 1993, 4th edition scheduled for 2013)

[3] HARMS, Daryl D. a MCDONALD, Kenneth. Začínáme programovat v jazyce Python. Vyd. 1. Brno: Computer Press, 2003. xviii, 456 s. ISBN 80-7226-799-X

[4] AUTOR NEUVEDEN. Játra - anatomie [online]. [cit. 4.5.2014]. Dostupný na WWW: http://www.kst.cz/web/?page_id=2301

[5] DOSTÁL, Ivo. V Plzni testují počítačový model lidských jater. Pomůže lépe plánovat operace. 2012. Dostupné z: http://iforum.cuni.cz/IFORUM-12340.html

[6] INFORMAČNÍ SYSTÉM VÝZKUMU, EXPERIMENTÁLNÍHO VÝVOJE A INOVACÍ. NT13326 - Zvyšování resekability maligních ložiskových procesů pomocí metod zpřesňujících měření perfúzních parametrů zbytkového jaterního parenchymu - počítačem asistované diagnostiky a softwarového modelování (2012-2015, MZ0/NT) [online]. 2012 [cit. 2014-04-27]. Dostupné z: http://www.isvav.cz/projectDetail.do?rowId=NT13326

[7] PYTHON SOFTWARE FOUNDATION. Python [online]. [cit. 2014-04-27]. Dostupné z: https://www.python.org/doc/

[8] SOURCEFORGE. Psyco [online]. [cit. 2014-04-27]. Dostupné z: http://psyco.sourceforge.net/

[9] PYPY. Pypy [online]. [cit. 2014-04-27]. Dostupné z: http://pypy.org/

[10] NUMPY DEVELOPERS. NumPy [online]. [cit. 2014-04-27]. Dostupné z: http://www.numpy.org/

[11] GOUILLART, Emmanuelle a Gaël VAROQUAUX. Image manipulation and processing using Numpy and Scipy. [online]. [cit. 2014-04-27]. Dostupné z: http://scipy-lectures.github.io/advanced/image_processing/

[12] VANOVSCHI, Vitalii. PARALLEL PYTHON. Parallel Python [online]. 2005 [cit. 2014-04-27]. Dostupné z: http://www.parallelpython.com/

[13] SELLE, Dirk, Bernhard PREIM, Andrea SCHENK a Heinz-Otto PEITGEN. Analysis of Vasculature for Liver Surgical Planning. IEEE Transactions on medical imaging. č. 21, s. 15.

5 Příloha

V kapitole příloh jsou pouze názorné a nejdůležitější bloky kódu (parametry pro segmentaci ukázkových dat a činný kód pro vlastní segmentaci). Veškeré zdrojové kódy, ukázková vstupní data a výsledná data jsou uložena na přiloženém DVD.

5.1 PVBPLiver.py

Soubor PVBPLiver.py - úryvek zdrojového kódu z metody pro nastavení segmentace dat po jejich manuálním přezkoumání.

```
# -*- coding: utf-8 -*-
"""
-----------------------------------------------------------------
Purpose:     (CZE-ZCU-FAV-KKY) Liver medical project

Author:      Pavel Volkovinsky
Email:       volkovinsky.pavel@gmail.com

Created:     2014/02/21
-----------------------------------------------------------------
"""

def _setup_data():

    dataPath = []
    threshold = []
```

```
sigma = []
dilationIterations = []
nObj = []
binaryClosingIterations = []
binaryOpeningIterations = []
biggestObjects = []

dataPath.append('org-liver-orig001.mhd-3mm_alpha45.pklz')
threshold.append(134)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(3)
binaryClosingIterations.append(4)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig001.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(134)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(3)
binaryClosingIterations.append(4)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig002.mhd-3mm_alpha45.pklz')
threshold.append(127)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(1)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig002.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(127)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(1)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig003.mhd-3mm_alpha45.pklz')
threshold.append(136)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig003.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(136)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig004.mhd-3mm_alpha45.pklz')
threshold.append(239)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig004.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(238)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig005.mhd-3mm_alpha45.pklz')
threshold.append(200)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig005.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(200)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig006.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(189)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(1)
binaryClosingIterations.append(1)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig007.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(147)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(3)
binaryClosingIterations.append(3)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig008.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(188)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig009.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(215)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(1)
binaryClosingIterations.append(1)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig010.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(237)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(1)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig011.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(200)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(3)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig012.mhd-alpha_45_vs_25_smoothing.pklz')
threshold.append(191)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(2)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig013.mhd-alpha_45_vs_25.pklz')
threshold.append(139)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(3)
binaryClosingIterations.append(3)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig014.mhd-alpha_45_vs_25.pklz')
threshold.append(108)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(2)
binaryClosingIterations.append(3)
binaryOpeningIterations.append(0)
biggestObjects.append(True)

dataPath.append('org-liver-orig015.mhd-alpha_45_vs_25.pklz')
threshold.append(219)
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(3)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig016.mhd-alpha_45_vs_25.pklz')
threshold.append(142)
```

```
sigma.append(0.15)
dilationIterations.append(2)
nObj.append(4)
binaryClosingIterations.append(5)
binaryOpeningIterations.append(1)
biggestObjects.append(True)

dataPath.append('org-liver-orig017.mhd-alpha_45_vs_25.pklz')
threshold.append(177)
sigma.append(0.12)
dilationIterations.append(2)
nObj.append(2)
binaryClosingIterations.append(3)
binaryOpeningIterations.append(0)
biggestObjects.append(True)
```

5.2 uiThreshold.py

Soubor uiThreshold.py - metoda pro vlastní segmentaci dat.

```
def updateImage(self, val):
"""
Hlavni update metoda.
Cinny kod pro Gaussovo filtrovani, prahovani, binarni uzavreni
a otevreni a vraceni nejvetsich nebo oznacenych objektu.
"""
if (sys.version_info[0] < 3):
import copy
self.imgFiltering = copy.copy(self.data)
else:
self.imgFiltering = self.data.copy()

## Filtrovani

## Zjisteni jakou sigmu pouzit
if(self.firstRun == True and self.inputSigma >= 0):
sigma = numpy.round(self.inputSigma, 2)
else:
sigma = numpy.round(self.ssigma.val, 2)
sigmaNew = thresholding_functions.calculateSigma(
self.voxel, sigma)

self.imgFiltering = thresholding_functions.gaussFilter(
self.imgFiltering, sigmaNew)
```

```
del(sigmaNew)

## Prahovani (smin, smax)

max_threshold = -1
min_threshold = self.threshold

if self.interactivity:
self.smin.val = (numpy.round(self.smin.val, 2))
self.smin.valtext.set_text('{}'.format(self.smin.val))
self.smax.val = (numpy.round(self.smax.val, 2))
self.smax.valtext.set_text('{}'.format(self.smax.val))

min_threshold = self.smin.val
max_threshold = self.smax.val

self.threshold = min_threshold

if (self.threshold == -1) and self.firstRun:
min_threshold = thresholding_functions.calculateAutomaticThreshold(
self.imgFiltering, self.arrSeed)

self.imgFiltering = thresholding_functions.thresholding(
self.imgFiltering, min_threshold, max_threshold, True, self.interactivity)

## Operace binarni otevreni a uzavreni.
```

```
## Nastaveni hodnot slideru.
if (self.interactivity == True) :
closeNum = int(numpy.round(self.sclose.val, 0))
openNum = int(numpy.round(self.sopen.val, 0))
self.sclose.valtext.set_text('{}'.format(closeNum))
self.sopen.valtext.set_text('{}'.format(openNum))
else:
closeNum = self.ICBinaryClosingIterations
openNum = self.ICBinaryOpeningIterations

self.imgFiltering = thresholding_functions.binaryClosingOpening(
self.imgFiltering, closeNum, openNum, True)

## Zjisteni nejvetsich objektu.
self.getBiggestObjects()
## Vykresleni dat
if (self.interactivity == True):
self.drawVisualization()
## Nastaveni kontrolnich hodnot
self.firstRun = False
garbage.collect()
```