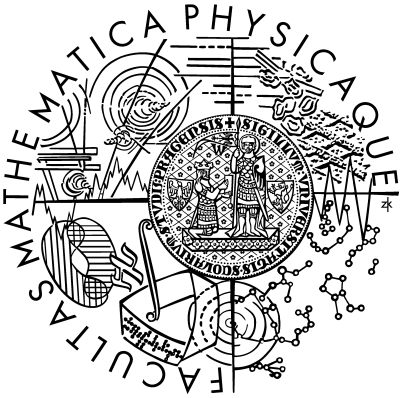
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Miloš Doubek

Sledování arteriálního řečiště na CT datech

Kabinet software a výuky informatiky

červen 2008

Vedoucí diplomové práce: RNDr. Josef Pelikán
Studijní program: Informatika
Studijní obor: Počítačová grafika

Děkuji vedoucímu práce RNDr. Josefu Pelikánovi za mnoho cenných rad a odbornou pomoc při vedení této práce a za mnoho životních zkušeností získaných při společných konzultacích.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 7. srpna 2008 Miloš Doubek

Obsah

Název práce: Sledování arteriálního řečiště na CT datech
Autor: Miloš Doubek
Vedoucí diplomové práce: RNDr. Josef Pelikán
e-mail vedoucího: Josef.Pelikan@mff.cuni.cz

Abstrakt:

Práce se zaměřuje na zpracování dat z počítačového tomografu (CT) pořízených při vyšetření cévního řečiště — CT angiografii. Vyšetřovanou oblastí jsou karotické tepny od odstupu z oblouku aorty (ev. truncus brachiocephalicus) po bazi lební, tedy celý jejich extrakraniální průběh. Vzhledem k zásadnímu podílu karotických tepen na cévním zásobení mozku je toto vyšetření často prováděným výkonem, majícím rozhodující vliv na diagnostickou rozvahu a následný terapeutický postup při patologických stavech cévního řečiště.

Současný klinický software dodávaný s tomografy neposkytuje dostatečný komfort pro snadnou prostorovou orientaci v datech, potřebný pro spolehlivou diagnostiku. Možným řešením a současně velkou výzvou v této oblasti je automatická segmentace a diagnostika cév.

Pro segmentaci výše zmiňovaného úseku karotické tepny byl použit paralelní a robustní algoritmus z kategorie algoritmů založených na registraci modelu a reálných dat. Z parametrů modelu lze pak tepnu zhodnotit a upozornit například na zúžení v jejím průběhu. Algoritmus byl vyzkoušen na syntetických i reálných datech.

Jádro systému je implementováno jako soustava oddělených ITK filtrů. Využívá tedy všech výhod softwarového návrhu knihovny ITK, jako je objektový přístup, paralelní a streamované zpracování dat, multiplatformnost nebo snadné znovuvyužití vytvořených filtrů. Uživatelskou přívětivost aplikace zajišťují Windows Forms z .NET frameworku.

Klíčová slova: segmentace cév, registrace modelu, ITK, ManagedITK

Title: Artery tracking in CT data
Author: Miloš Doubek
Department: Department of Software and Computer Science Education
Supervisor: RNDr. Josef Pelikán
Supervisor's e-mail address: Josef.Pelikan@mff.cuni.cz

Abstract:

This thesis focuses on the processing of CT data received during the CT examination of the vessels — CT angiography. The area of interest is the extracranial course of the carotic arteries. Regarding the major role of these vessels in the blood supply of the brain cells this examination is frequently performed and influences the diagnostic decision and following therapy in case of pathological changes of the vessels.

The contemporary clinical software included in the tomographs does not provide sufficient comfort for easy 3-dimensional orientation which is necessary for the diagnostics. The automatic segmentation and classification of the vessels might be a possible solution to this problem and also a great challenge.

For the segmentation of the extracranial part of the carotic arteries the parallel and robust model fitting algorithm was used. The parameters of the model make it possible to classify the vessel and to draw attention to its stenosis, for example. The algorithm was tested on both synthetic and real data.

The core of the system is implemented as a set of separate ITK filters. It exploits all the advantages of the software architecture of the ITK framework (object-oriented design, parallel and stream data processing, multi-platform, reusable components). The user friendliness is ensured by the Windows Forms of .NET framework.

Keywords: vessel segmentation, model registration, ITK, ManagedITK

Kapitola 1

Úvod

Tato kapitola podává stručný úvod do problematiky segmentace cév z dat pořízených pomocí počítačové tomografie. Počítačová tomografie je v dnešní době zcela běžné vyšetření. Trojrozměrný obraz, který při vyšetření vzniká, je třeba rychle a přesně zpracovat pro stanovení diagnózy. K tomu lékařům pomáhá software, který jim umožňuje zlepšit prostorovou orientaci v získaných datech. Klasický software umí vytvářet řezy v libovolných rovinách či časově náročné prostorové rekonstrukce. Při prohlížení dvojrozměrných řezů (slice) často ztrácíme návaznost na předchozí řez a tím prostorovou orientaci. To je typický problém při trasování tepen, kdy hodnotíme, jak jsou tepny průchodné a zda nedochází k hypoperfuzi jimi zásobeného orgánu.

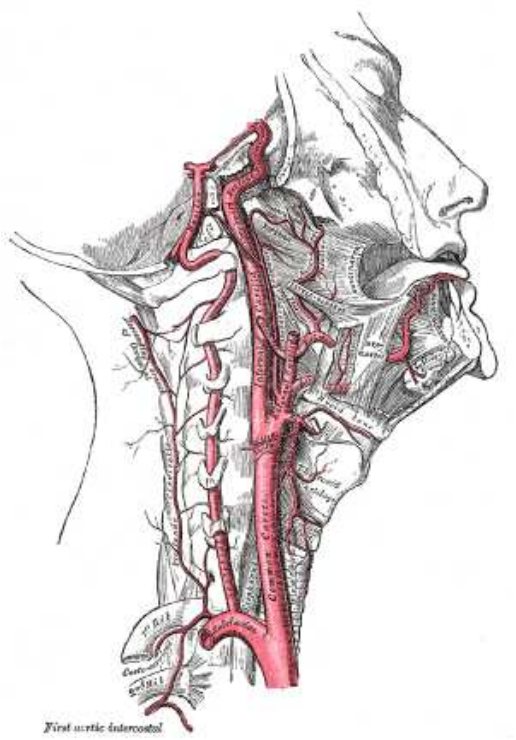
V této oblasti zpracování medicínských dat je velký prostor pro vývoj a implementaci sofistikovaných algoritmů, které pomohou lékaři s rychlou a správnou diagnostikou. Takové algoritmy musí řešit mnoho problémů spojených jak s velkým objemem a špatnou kvalitou dat, tak s omezeními, která jsou dána použitou zobrazovací technikou. Vrcholem této snahy je automatická či poloautomatická segmentace tkání a orgánů následovaná analýzou takto vybraných dat (například pomocí jejich parametrizace). Podobné postupy jsou základem počítačem podporované diagnózy (CAD — Computer Aided Diagnosis), což jsou systémy, které lékaře upozorní na podezřelé oblasti a pomohou tak výrazně snížit riziko přehlédnutí patologických změn.

1.1 Vytyčení problému a cíle práce

Cílem této práce je implementovat robustní metodu pro segmentaci krčních tepen (obrázek 1.1), poté tepny parametrizovat tak, aby došlo k usnadnění jejich automatického zpracování a následné diagnostické rozvahy. Zvolený postup bude podrobně popsán, bude zhodnoceno jeho další použití pro data z jiné oblasti, například břicha. Při zobrazování tepny vybereme takovou metodu, která umožní snadný náhled na průběh celé tepny.

Tento algoritmus bude pracovat výhradně na datech získaných počítačovou tomografií.

Postup otestujeme na syntetických i reálných datech a výsledky zpracujeme. Algoritmus by měl rozumně volit kompromis mezi robustností a rychlostí výpočtu. Zvyšování rychlosti postupuje cestou paralelizace úloh, proto i zvolený algoritmus by měl být implementován jako paralelní.



Obrázek 1.1: Zkoumaná část krční tepny — společná krkavice a vnitřní krkavice (převzato z anatomického atlasu [44]).

1.2 Struktura práce

Předkládáme strukturovaný soupis kapitol s jejich popisem pro usnadnění orientace v textu.

Kapitola 2 poskytuje medicínský pohled na zobrazování krčních tepen. Jsou zde uvedeny možnosti a postupy jejich zobrazování počítačovou tomografií a jinými metodami.

Kapitola 3 obsahuje přehled metod, které se používají k segmentaci cév. Na začátku kapitoly je soupis problémů, s nimiž se popsané algoritmy potýkají. Několik metod je zde podrobně vysvětleno.

Kapitola 4 obsahuje popis několika zobrazovacích metod, které se specializují přímo na vizualizaci cév — jde především o typy *Curved Planar Reformation* (CPR). Tato kapitola kromě podrobného vysvětlení každého typu CPR obsahuje i mnoho názorných obrázků.

Kapitola 5 detailně popisuje metodu segmentace cév, která byla vybrána a implementována v rámci této práce. Jde o metodu, která lícuje válcovitý model k reálným datům. Je zde také vysvětlena strategie její paralelizace.

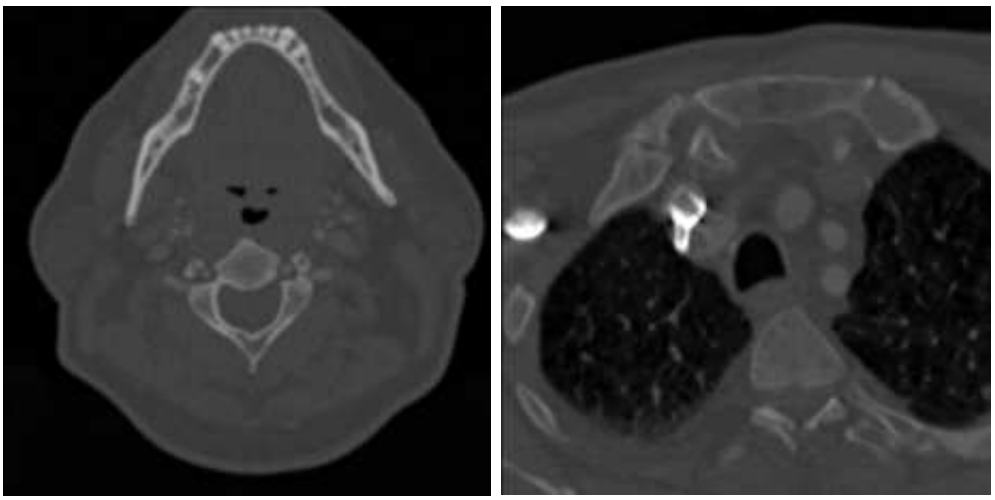
Kapitola 6 přináší zhodnocení použitého algoritmu, popisuje několik jeho slabých míst a možnosti jejich řešení. Dále ve formě grafů prezentuje výsledky testů jak na syntetických datech, tak na reálných datech poskytnutých MUDr. Martinem Horákem z fakultní nemocnice Na Bulovce.

Příloha A je stručným úvodem do problematiky knihovny ITK s přehledem jejích možností. Dále jsou zde přílohy B a C, což je uživatelská a programátorská dokumentace.

Kapitola 2

CT-angiografie

Počítačová tomografie (CT) je zobrazovací metoda, založená na detekci rentgenových paprsků, procházejících pacientem v axiální rovině, ze kterých se poté rekonstruuje výsledný obraz. Angiografie je obecně metoda zobrazující cévy, princip tohoto zobrazení se liší podle použité modality. Příklady dat pořízených pomocí CT při vyšetření tepen (CTA) jsou na obrázku 2.1.



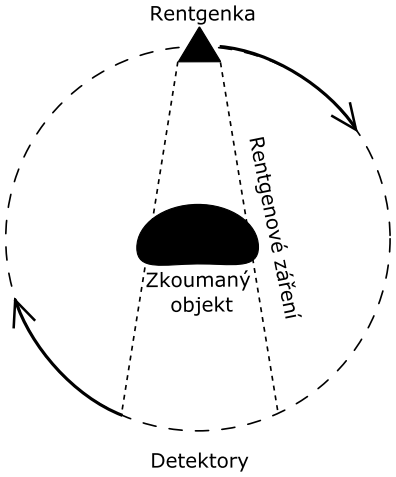
Obrázek 2.1: Vlevo je řez oblastí nad větvením karotid, vpravo je řez v úrovni těsně po odstupu cévních kmenů pro horní polovinu těla z oblouku aorty.

2.1 Počítačová tomografie

V roce 1979 byla za uvedení této zobrazovací metody pánům Hounsfieldovi a Cormackovi udělena Nobelova cena v oblasti lékařství. Tato metoda se od té doby stala jednou s nejpopulárnějších zobrazovacích metod v lékařství a úspěšně se dále rozvíjí.

Z pohledu zpracování dat pomocí CT je největší předností této metody zobrazování normovaných hodnot obrazové funkce v rozmezí -1000 až 3095 HU (Hounsfield unit). Různým tkáním je podle jejich absorpce rentgenového záření přiřazena pevná hodnota (nebo interval) z Hounsfieldovy stupnice. Kalibračními hodnotami jsou -1000 HU pro vzduch a 0 HU pro vodu. Pak například víme, že krev v tepnách má denzitu 40–60 HU na všech počítačových tomografech.

Principem CT je rekonstrukce výsledného obrazu z mnoha projekcí pořízených na kruhové dráze rentgenky (obrázek 2.2). Pro rekonstrukci snímaného prostoru se používá mnoha algoritmů, některé můžeme najít například v [35, 36]. Všechny rekonstrukce ovšem trpí několika nedostatky, které jsou dány především technickými možnostmi přístroje. Jedním z nich je *Partial volume efekt* — prostorová rozlišovací schopnost přístroje nedokáže zachytit detailní struktury nebo přechody dvou oblastí s dostatečnou přesností. V takových bodech je pak výsledná hodnota průměrem hodnot, z nichž některé spadají do jedné a některé do druhé oblasti. To má za následek, že se hrany ve výsledném obrázku jeví jako rozmazané a že nelze spolehlivě změřit denzitu velmi malých ložisek a rozhodnout na jejím základě o charakteru daného ložiska. Dalším je šum — obecný problém při pořizování obrazových dat daný mnoha vlivy jako například nedokonalostí snímačů a obecně technickým řešením snímání prostorových dat. Některé další neduhy CT jsou popsány na začátku kapitoly 3.



Obrázek 2.2: Schématický princip počítačového tomografu (převzato z Wikipedie).

2.2 Metody vyšetření tepen

Klasickou angiografií rozumíme postup, při němž je pomocí katetru přímo do zobrazované cévy aplikována jodová kontrastní látka, detekovatelná RTG zářením. Je-li kontrastní látka aplikována do tepny, jedná se o arteriografii, je-li aplikována do žíly, hovoříme o venografii. Široce užívanou metodou zpracování obrazu při klasické angiografii je digitální subtrakce, kdy je vytvořena obrazová maska (několik snímků dané oblasti bez přítomnosti kontrastní látky), která je následně odečtena od snímků s kontrastní látkou, což zlepšuje přehlednost vyšetřovaného cévního řečiště. Klasická angiografie je pro dlouhou dobu užívání a bohaté zkušenosti považována za zlatý standard. Její významnou výhodou je možnost sledování průtoku kontrastní látky cévou v reálném čase (skiaskopie) a možnost bezprostřední intervence v případě nálezu patologických stavů. Nevýhodou je invazivita metody (nutnost zavedení katetru nejčastěji cestou stehenní či podpažní tepny s následným nasondováním kýžené cévy, například krkavice, jež s sebou nese diskomfort pacienta a možnost komplikací výkonu), organizačně je překážkou opakovaná expozice zdravotnického personálu RTG záření.

CT-angiografie je metoda, při níž zobrazujeme cévy počítačovou tomografií. Kontrastní látka je aplikována do periferní žíly (obvykle v loketní jamce) a snímání pomocí CT přístroje je zahájeno s adekvátním zpožděním v okamžiku, kdy se kontrastní látka nachází v cévách vyšetřované oblasti. Odpadá tím invazivita klasické angiografie, zvyšuje se množství získaných informací (např. zobrazení cévní stěny,

rovněž tak struktur v okolí cévy), zkracuje se doba vyšetření, neboť není potřeba sondovat jednotlivé cévy.

Zcela odlišnou modalitou je vyšetřování cév magnetickou rezonancí, které ovšem není v současnosti široce používáno, jednak s ohledem na dostupnost metody, především však proto, že data takto získaná nemají větší výtěžnost nežli data získaná při CT vyšetření, které je rychlejší a levnější.

Další metodou je vyšetření cév pomocí ultrazvuku, které je vyšetřením zcela běžným — poskytuje informace o cévní stěně a s použitím Dopplerova principu poměrně přesně měří rychlost toku v cévách. Na základě informace o rychlosti průtoku lze určit relativní zúžení dané cévy. Limitací metody je možnost průniku ultrazvukových vln — není možno vyšetřovat oblasti, kdy se mezi sondou a vyšetřovanou strukturou nachází kost, vzduch nebo kdy vyšetřovaná struktura je uložena hluboko pod povrchem těla.

2.3 Medicínský pohled

Ve vyspělých zemích s vysokým podílem kardiovaskulárních onemocnění na nemocnosti a úmrtnosti populace je především aterosklerotické postižení karotických tepen velmi časté, zvláště ve vyšších věkových kategoriích. Protože tyto tepny zajišťují přívod krve do mozku, má jejich zúžení či uzávěr vážné, leckdy fatální následky, proto je kvalitní diagnostika v této oblasti nezbytná.

Vzhledem k dobré dostupnosti, nízkým nákladům a minimální zátěži pro pacienta je v neakutních stavech metodou první volby vyšetření ultrazvukové, které umožní selektovat pacienty pro další diagnostický postup. V případech, kdy je plánována endovaskulární intervence[1] (trombolýza[2], zavedení stentu, plastika cévy), je namístě klasická angiografie s bezprostředně následujícím terapeutickým výkonem.

V situacích akutních (např. podezření na uzávěr karotické tepny) je indikováno vyšetření výpočetní tomografií, které je možné realizovat v poměrně krátkém časovém intervalu (řádově desítky minut včetně přípravy pacienta a vyhodnocení vyšetření). Nezanedbatelnou výhodou CT vyšetření v takovéto situaci je možnost zhodnotit v jedné době i stav mozkové tkáně včetně její perfuze. Je nasnadě, že

[1] zákrok uvnitř cévy

[2] rozpouštění krevní sraženiny

v případě akutního uzávěru tepny je každé vylepšení zpracování dat, které vede ke zkrácení vyšetřovací doby, významným posunem. Zlepšení přehlednosti cévního řečiště může být velkým přínosem i v jiných lokalizacích nežli jsou karotické tepny. Nabízejí se například tepny ledvinné či řečiště trávicího traktu, které je velmi špatně přehledné a diagnostika jeho patologických stavů (především trombotických uzávěrů) je nezřídka pozdní.

Potenciál vyšetření pomocí CT je velký, zvláště s představou nově vyvíjených přístrojů, které jsou schopny zobrazit vyšetřovanou oblast v několikanásobně větším rozlišení, než je možné dnes. S nárůstem velikosti generovaných dat v průběhu vyšetření přichází nezbytnost automatické či poloautomatické diagnostiky těchto dat.

Kapitola 3

Metody segmentace cév

Segmentace cév je jedna z nejčastěji řešených úloh v oblasti zpracování medicínských dat. Zároveň je to klíčový krok v automatické či poloautomatické diagnostice cév, o jejímž významu hovoří kapitola 2.3. Za tímto účelem byla vyvinuta celá řada metod a mnoho jejich kombinací. Je tedy s podivem, že tento problém dosud nebyl uspokojivě vyřešen. Důvodů pro to je hned několik:

Velikost dat — může se pohybovat až v řádech gigabytů, navíc s použitím moderních přístrojů, které jsou schopny získat data ve velkém rozlišení, objem dat ještě výrazně vzrůstá. Standardní rozlišení v medicínské praxi je 512x512 bodů na jeden řez (slice). Avšak již dnes jsou testovány CT přístroje, které jsou schopny snímat řezy v rozlišení 2048x2048 a vyšším.

Šum, artefakty a rozmazání pohybem — data získaná pomocí počítačové tomografie (CT) nebo magnetické rezonance (MR) jsou velmi zatížena šumem, bohužel i míra šumu se snímek od snímku liší. Taktéž obsahují množství artefaktů, způsobených přítomností vysoce denzních objektů, například zubů či kovových implantátů (endoprotéz). Čas, po který snímá počítačový tomograf pacienty, může dosahovat i několika sekund, během nichž nelze zastavit některé pohyby jako například srdeční akci nebo mimovolní svalový třes. V určitých oblastech pak dochází následkem výše popsaných skutečností k chybné rekonstrukci dat.

Variabilita snímaných dat — variabilita může být dána jak přirozenou fyzickou odlišností pacientů (tělesné rozměry, pohlaví), tak vrozenou atypickou stavbou

či lokalizací segmentovaných orgánů, což je problém hlavně u metod, které jsou založeny na vyhledávání vzorů.

Nehomogenita denzity kontrastní látky — kontrastní látka aplikovaná do cévního řečiště se mísí s krví a je společně s ní distribuována. Denzita tkáně či orgánu v daném okamžiku je tedy ovlivňována například množstvím a rychlostí aplikace kontrastní látky, stavem kardiovaskulárního systému pacienta nebo mírou prokrvení dané oblasti. Obecně v periferních cévách s menším průměrem je denzita jejich kontrastního obsahu mnohem menší než v cévách větších průměrů.

Partial volume efekt — typickou vlastností dat pořízených počítačovou tomografii je prolínání denzity na rozhraní snímaných orgánů. Například kalcifikované tepny mají přibližně stejnou denzitu jako kost, která se může nacházet v bezprostřední blízkosti takto postižené tepny. V tomto místě pak není možné rozeznat hranici tepny a kosti.

S tímto vším by si měl ideální segmentační algoritmus umět poradit. Bohužel žádný takový algoritmus nebyl dosud publikován. Většina algoritmů dobře fungujících na syntetických datech selhává na datech reálných, nebo je nelze se stejným počátečním nastavením použít pro jiná reálná data.

Stěžejní práce [1], která poskytuje přehled algoritmů, vhodných pro segmentaci cév, řadí tyto do šesti kategorií: (1) techniky vyhledávající vzory, (2) aproximace modelu, (3) trasování cév, (4) umělá inteligence, (5) neuronové sítě, (6) detekce válcovitých tvarů. Bohužel až příliš mnoho metod nelze do těchto kategorií jednoduše rozdělit. Téměř všechny moderní metody se snaží kombinovat několik různých přístupů. V této práci se budeme zajímat pouze o techniky, které lze použít pro data získaná pomocí počítačové tomografie. Ostatní techniky, které lze použít pro jiné modality, jako magnetická rezonance nebo ultrazvuk, jsou velmi odlišné a nejsou náplní této práce. Z těchto důvodů zavádíme vlastní rozdělení algoritmů vhodných pro segmentaci cévního řečiště z CT dat. Pro lepší orientaci v této kapitole uvádíme strukturované rozdělení algoritmů, včetně jejich podkategorií.

1. **Multi-scale technika** — podkapitola 3.1
2. **Matematická morfologie** — podkapitola 3.2
3. **Diferenciální geometrie** — podkapitola 3.3

   (a) **Zvýrazňující filtry** — podkapitola 3.3.1

4. **Trasování cév** — podkapitola 3.4

   (a) **Narůstání oblastí** — podkapitola 3.4.1

   (b) **Přímý postup** — podkapitola 3.4.2

   (c) **Hledání minimální cesty v grafu** 3.4.3

5. **Detekce středu cévy** — podkapitola 3.5

6. **Deformovatelné modely** — podkapitola 3.6

   (a) **Aktivní kontury** — podkapitola 3.6.1

   (b) **Level-sety** — podkapitola 3.6.2

7. **Parametrické modely** — podkapitola 3.7

   (a) **Šablony(Template Matching)** — podkapitola 3.7.1

   (b) **Eliptické modely** — podkapitola 3.7.2

   (c) **Válcovité modely** — podkapitola 3.7.3

3.1 Multi-scale technika [1]

Jak je uvedeno výše, jedním z největších problémů při segmentaci medicínských dat je jejich velikost a tedy i doba potřebná k jejich zpracování. Tato metoda řádově eliminuje dobu segmentace. Myšlenka je jednoduchá, vstupní data se zmenší v určitém poměru r, následně se takto zmenšená data segmentují. Je nezbytné v segmentačním algoritmu pracovat s r jako s parametrem. Následně se přejde k segmentaci dat v původní velikosti, tedy s parametrem $r = 1$. Klíčové je nalézt způsob využití informací získaných v předchozí segmentaci tak, aby se co nejvíce vymezila data, která je potřeba segmentovat ve větším rozlišení — například při přechodu do většího rozlišení rozpoznávat vzor pouze v oblastech, kde byl tento vzor nalezen v datech s menším rozlišením. Důležitá je také volba měřítek, aby doba potřebná na vygenerování dat v menším rozlišení nebyla větší než doba segmentace ve větším rozlišení. Tato technika je pro svou úspěšnost využívána v mnoha algoritmech.

[1] V česky psaných dokumentech se používá též výraz pyramidový přístup.

3.2 Matematická morfologie

Operátory této skupiny pracují většinou na binárních obrázcích, zde jde o rozšíření na obrázky s větším počtem stupňů šedi. Základními dvěma druhy operátorů jsou dilatace a eroze. Zde si uvedeme algoritmus, který pomocí těchto operátorů zvýrazní cévy v datech z počítačového tomografu. Jedná se o *top-hat* otevření obrázku. Vstupní obrázek označíme I, eroze obrázku za použití okolí bodu B je pak $\varepsilon_B(I)$. Pro každý bod x obrázku $\varepsilon_B(I)$ platí

$$[\varepsilon_B(I)](x) = \min_{b \in B} I(x + b).$$

Podobně dilatace $\delta(I)$ v okolí B bodu x

$$[\delta_B(I)](x) = \max_{b \in B} I(x + b).$$

Otevření obrázku $\gamma(I)$ s okolím B pak

$$[\gamma_B(I)](x) = \delta_B[\varepsilon_B(I)].$$

Konečná morfologická operace *white top-hat* odečte od původního obrázku jeho otevřený obraz, čímž dojde k odstranění některých částí obrázku, například šumu na pozadí tepen s kontrastní látkou.

$$WTH_B(I) = I \ - \delta_B[\varepsilon_B(I)].$$

Metoda uspokojivě plní svůj účel, leč její popularita je poněkud zastíněna *zvýrazňujícími filtry*, které jsou popsány v následující kapitole a které dosahují výrazně lepších výsledků.

3.3 Diferenciální geometrie

Algoritmy této kategorie vnímají data jako funkci ve vyšší dimenzi. Tuto funkci pak zpracovávají pomocí diferenciálního a integrálního počtu tak, že hledají některé její vlastnosti, například hřebenové linie. Nalezené linie pak prohlásí za střed cévy. Jednodušší algoritmy nacházejí využití zejména při zpracování obrázků pořízených metodou digitální subtrakce (odečtení nativního obrázku od obrázku s kontrastní látkou). Do této kategorie patří také oblíbené *zvýrazňující filtry*, jimiž se budeme podrobněji zabývat v následující kapitole.

3.3.1 Zvýrazňující filtry

Tyto filtry mají za cíl zvýraznit válcovité struktury v datech, čehož mnoho různých algoritmů využívá při přípravě vstupních dat. Jedna z nejpoužívanějších metod je popsána v [5] další podobná v [6]. V obou případech se jedná o filtraci dat, která je založena na porovnávání vlastních čísel Hessovy matice (matice druhých derivací) v každém bodě. Nejdříve jsou vstupní data zmenšena v několika měřítcích. Na všechna takto získaná data je aplikován zvýrazňující filtr. V každém měřítku totiž filtr zvýrazní válcovité tvary jiných průměrů, po finální kompozici jsou zvýrazněny cévy různých poloměrů. Se znalostí vlastních čísel Hessovy matice λ_1, λ_2 *a* λ_3, která jsou pojmenována tak, aby splňovala podmínku $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$, můžeme vypočítat skalární hodnotu, kterou se zvýrazní bod válcovité struktury, různými způsoby. Například v práci [6] je použit následující vzorec:

$$\nu(\lambda_1, \lambda_2, \lambda_3) = (1 - e^{-\frac{\mathcal{R}_\mathcal{A}{}^2}{2\alpha^2}}) e^{-\frac{\mathcal{R}_\mathcal{B}{}^2}{2\beta^2}} (1 - e^{-\frac{\mathcal{S}^2}{2\gamma^2}}), \tag{3.1}$$

kde

$$\mathcal{R}_\mathcal{A} = \frac{|\lambda_2|}{|\lambda_3|}, \quad \mathcal{R}_\mathcal{B} = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda 3|}}, \quad \mathcal{S} = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}$$

α, β a γ jsou parametry této metody, o jejichž bližším významu se lze dočíst v [6]. Tento vzorec se při výpočtu použije pouze v případě, kdy vlastní čísla Hessovy matice λ_1, λ_2 a λ_3 splňují podmínku $\lambda_2 \leq 0$ a $\lambda_3 \leq 0$ (podrobněji v algoritmu 1). Ideální případ nastává, pokud splňují podmínku $|\lambda_1| \approx 0$, $|\lambda_1| \ll |\lambda_2|$ a $\lambda_2 \approx \lambda_3$. Pak je bod, ve kterém byla tato vlastní čísla vypočtena, součástí válcovité struktury zcela jistě. Výsledná hodnota pro každý bod i matice $\mathcal{V}$ se vypočte jako

$$\mathcal{V}(i) = \max(\mathcal{V}^*(i, \sigma)), \text{ kde } \sigma \in [\sigma_{min}, \sigma_{max}]. \tag{3.2}$$

Filtry jsou velmi úspěšné a populární, mají v podstatě jedinou nevýhodu, kterou je velká časová náročnost, zvláště při aplikaci na celá vstupní data.

3.4 Trasování cév

Tyto metody směřují pozornost do známého bodu uvnitř cévy (*seed point*) a na základě apriorních informací a vlastností obrázku v tomto bodě nebo jeho okolí usuzují na směr, ve kterém céva pokračuje. V takto nově nalezeném bodě algoritmus

Algoritmus 1 Zvýraznění válcovitých tvarů pomocí vlastních čísel Hessovy matice

1. Vstupní data tvoří trojrozměrná matice bodů, kde každý z bodů má určitou hodnotu denzity, dále interval $[\sigma_{min}, \sigma_{max}]$ a krok σ_{step}.

2. Pro každé měřítko $\sigma = \sigma_{min} + k\sigma_{step}$, pro které platí $\sigma \leq \sigma_{max}$, a pro každý bod i vstupní matice.

   (a) spočítej Hessovu matici s měřítkem σ v bodě i (konvoluce obrázku s druhou derivací Gaussova jádra s rozptylem σ)

   (b) spočítej vlastní čísla Hessovy matice

   (c) pokud $\lambda_2 > 0$ nebo $\lambda_3 > 0$, pak výsledná hodnota $\mathcal{V}^*(i, \sigma) = 0$, jinak vypočti $\mathcal{V}^*(i, \sigma) = \nu(\lambda_1, \lambda_2, \lambda_3)$ podle vzorce 3.1

3. Výstupem je trojrozměrná matice $\mathcal{V}$ stejných rozměrů jako vstupní, vypočtená podle vzorce 3.2.

pokračuje další iterací. Lze tedy říci, že tyto metody začínají ve startovním bodě, pokoušejí se najít cestu cévou do konečného bodu či do konečné množiny bodů.

3.4.1 Narůstání oblastí

Narůstání oblastí (Region Growing) je snadná metoda založená na záplavovém algoritmu. Dvě nejdůležitější kritéria při postupu vlny do sousedního bodu jsou založena na podobnosti hodnot nebo jejich prostorové blízkosti. Postup tepnou je možný, dokud denzita sousedního bodu spadá do intervalu, který definuje hodnoty intenzity uvnitř cévy. Algoritmus není příliš úspěšný, nekvalitní data zatížená šumem jsou pro něj nepřekonatelnou překážkou. Existuje několik vylepšení, která si do jisté míry dovedou s některými z těchto překážek poradit. Například adaptivní narůstání oblastí použité v práci [19]. Metoda se používá spíše pro zpracování skiagramů (prostých RTG snímků) nebo jako menší součást segmentačních procedur pracujících na CT či MR datech. Výjimkou je práce [23], která představuje algoritmus založený na záplavové vlně (flood fill), při jejímž postupu se detekuje rozvětvení cévy a tak se postupně generuje graf cévního řečiště. Rozvětvení je detekováno, pokud body,

které jsou přidávány v aktuální vlně, nejsou spojité (nenavazují na sebe). Zda bylo rozvětvení nalezeno správně, je kontrolováno v dalším kroku, detailněji v [23].

3.4.2 Přímý postup

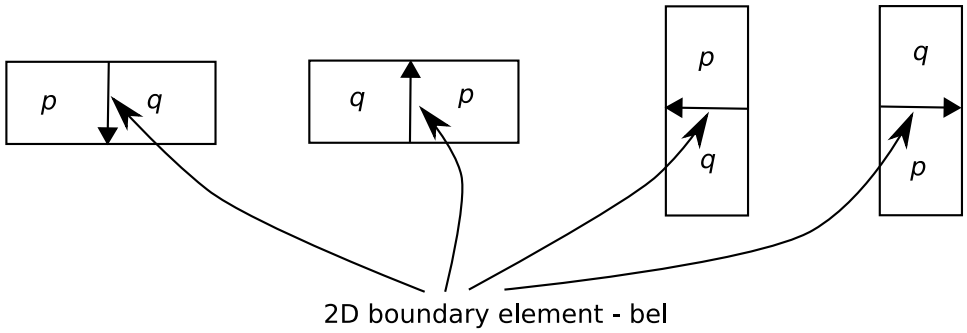
Algoritmy na svém vstupu potřebují kromě počátečního bodu A (seed point) také směr $\mathbf{u}$, kterým se céva ubírá. Poté se tímto směrem vypraví a vygenerují bod C, který vypočtou pomocí vzorce $C = A + (\alpha d_{min})\mathbf{u}$, kde $\alpha \in [0, 1]$ je parametr algoritmu a d_{min} je dosavadní minimální průměr. V bodě C je rekonstruována rovina kolmá na vektor $\mathbf{u}$, v této rovině se pak různými způsoby upřesňuje pozice bodu C tak, aby byl lokalizován ve středu cévy. Lze použít metodu *Vrhání paprsků* z kapitoly 3.5. Lepší řešení je v práci [24], kde pozice bodu C nemusí být uvnitř cévy. Paprsky totiž nejsou vysílány z bodu C, ale ze všech bodů čtverce, jehož střed je právě bod C. Velikost hrany čtverce je βd_{min}, kde β je také parametr algoritmu. Metoda má klasické problémy při určení konce paprsku (hranice cévy), blíže popsané v následující kapitole.

3.4.3 Hledání minimální cesty v grafu

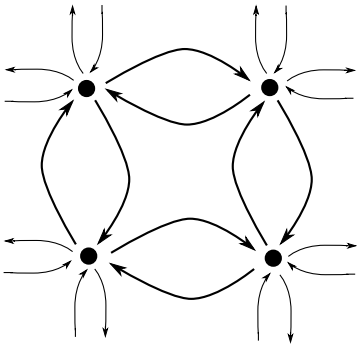
Každý bod v obrázku, kde počítačový tomograf vypočetl hodnotu denzity, odpovídá uzlu v grafu a každý přechod z jednoho bodu do sousedního odpovídá hraně grafu. Pro 3D data má tedy každý uzel (krom hraničních) 26 hran do sousedních uzlů. Většina algoritmů hledajících minimální cestu v grafu se od klasických algoritmů jako Dijkstrův algoritmus a Floyd-Warshallův algoritmus liší pouze způsobem, kterým vypočítávají ohodnocení hrany grafu. Uvedeme zde příklad algoritmu *live wire*, který byl původně vyvinut pro interaktivní segmentaci objektů nebo uzavřených oblastí, ale hodí se také pro rychlé hledání cesty mezi dvěma body. Algoritmus prošel od svého prvního uvedení podstatnými změnami, my ho zde uvádíme tak, jak je popsán v [26] a některá jeho vylepšení z [25] nebo [27].

Hranice je orientovaná, uzavřená a spojitá křivka (Jordanova), tvořená množinou ohodnocených orientovaných hran (bel — boundary element) 3.1. Součet ohodnocení hran (belů) na hranici je minimální. Algoritmus pracuje ve 2D prostoru s 4-sousedností, ve 3D prostoru s 6-sousedností bodů. Každý bel má přiřazenou sadu vlastností, ze kterých lze vypočítat jednu skalární hodnotu, a to jeho ohodnocení. Výpočet ohodnocení se ve všech dostupných popisech tohoto algoritmu

liší, většina používá k výpočtu lineární, Gaussovu nebo hyperbolickou transformaci vlastností belu, jako například gradientu nebo denzity obou bodů přiléhajících k belu. V takto připraveném grafu 3.2 se pomocí Dijkstrova algoritmu hledá minimální cesta z počátečního bodu do koncového bodu (při interaktivní segmentaci je to bod pod kurzorem myši).



Obrázek 3.1: Příklad orientované hrany (belu) mezi dvěma body p a q (převzato z [25]).



Obrázek 3.2: Graf, ve kterém se hledá hranice metodou *live wire* (převzato z [25]).

Algoritmus lze řádově urychlit, pokud diskretizujeme výpočet ohodnocení hran a v Dijkstrově algoritmu použijeme místo prioritní fronty speciální datovou strukturu. Detailně o tomto vylepšení v kapitole 5.1. Další vylepšení *Live Wire on the Fly* je popsáno v [25]. Urychlení algoritmu přináší také metoda *Intelligent Scissors* [29], která omezuje prostor, ve kterém se hledá cesta. Výhody a nevýhody tohoto postupu jsou popsány v kapitole 5.1.

3.5 Detekce středu cévy

Zde popsané algoritmy hledají cestu, která prochází středem cévy. Spojením těchto cest vzniká stromová struktura cévního řečiště. Tyto algoritmy mohou být velmi rozmanité, avšak všechny zde uvedené předpokládají apriorní znalost cesty cévou. Tuto cestu je možné odhadnout pomocí metod založených na hledání minimální cesty v grafu. Jedna z nejlépe propracovaných metod hledání minimální cesty cévou je v práci A. Kanitsara [7]. I další část mají zde uvedené algoritmy společnou. V každém bodě odhadnuté cesty určí rovinu kolmou na tuto cestu. V nalezené rovině pak algoritmy různými způsoby hledají střed cévy. Aby bylo možné zapisovat body této roviny a počítat s nimi, je třeba zvolit bázi a střed její souřadné soustavy. Ta je pro každý bod cesty jiná, je tedy potřeba výsledný střed převést zpět do prostoru získaných dat. Více o hledání této roviny a volbě báze v kapitole 5.2.

Vrhání paprsků (Ray Casting)

V současné praxi se jedná zřejmě o nejpoužívanější metodu. Její snadná implementace a dostačující výsledky na zdravých cévách dlouhý čas bránily rozšíření jiných, sofistikovanějších algoritmů. Předpokladem tohoto algoritmu je známý bod uvnitř cévy. Dalším vstupem je omezující podmínka, která paprsku směřujícímu z vnitřku cévy říká, zda je ještě uvnitř nebo již mimo cévu. Tato podmínka může být dvojího druhu.

První, použitá například v práci [7], podle vstupního intervalu hodnot $[vessel_{min}, vessel_{max}]$ jednoduše rozhodne, zda je paprsek ještě v tepně či nikoli. Hraniční hodnoty se získávají empiricky, jejich ideální hodnotu je velmi obtížné získat.

Druhá podmínka je pouze hodnota změny intenzity. Pokud bod na paprsku překročí maximální povolenou změnu intenzity svého průběhu, pak je právě zpracovávaný bod považován za koncový bod paprsku. Zvolení správné maximální hodnoty je opět velmi obtížné, ideální hodnota musí být větší než změny intenzity způsobené šumem.

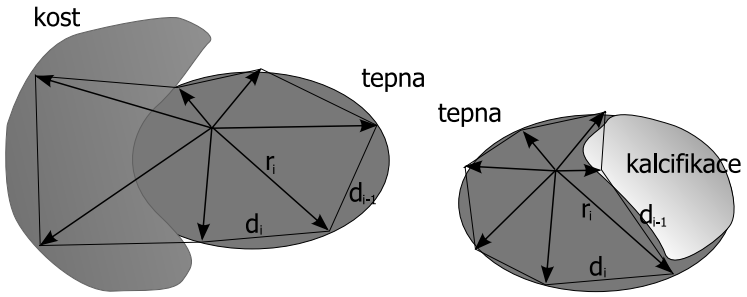
Po nalezení hranice cévy první či druhou omezovací podmínkou se výsledný střed cévy spočte pomocí následujícího vzorce.

$$[x_c, y_c] = [\frac{\sum_{i=1}^{n} x_i(d_{i-1} + d_{i \bmod n})}{2\sum_{i=0}^{n-1} d_i}, \frac{\sum_{i=1}^{n} y_i(d_{i-1} + d_{i \bmod n})}{2\sum_{i=0}^{n-1} d_i}] \tag{3.3}$$

Souřadnice vypočteného centra tepny jsou označeny $[x_c, y_c]$, číslo n značí počet paprsků vyslaných k okraji cévy. Souřadnice i-tého bodu na hranici cévy jsou $[x_i, y_i]$, vzdálenost mezi i-tým a i+1 bodem je d_i. Celý vzorec 3.3 je vlastně vážený průměr, kde váhou každého bodu je vzdálenost k jeho sousedům.

Že tento algoritmus není ideální, není třeba zdůrazňovat, několik příkladů jeho selhání je vidět na obrázku 3.3. Bohužel žádný algoritmus založený na prahování nemůže být ze své podstaty úspěšný na nekvalitních datech. Na druhou stranu tento algoritmus je velmi rychlý a na kvalitních datech dosahuje uspokojivých výsledků.

Ve všech ohledech lepších výsledků dosahuje první omezující podmínka. Druhá, sledující změnu intenzity na okraji cév, výrazně zaostává hlavně na cévách malých průměrů.



Obrázek 3.3: Příklady chybného určení středu tepny v metodě vrhání paprsků. Vlevo je v blízkosti cévy kost a její intenzita splývá s intenzitou tepny. Vpravo je tepna s kalcifikovaným plátem, který má o mnoho větší intenzitu než kontrastní náplň tepny.

Posun středu (Block Matching)

Algoritmus je založen na jednoduché myšlence, kdy hledáme posun středu cévy, k němuž došlo v jednom kroku algoritmu. Ze znalosti středu cévy v předchozím kroku se vypočte střed cévy v aktuálním kroku.

$$[x_c, y_c] = [x'_c + x_d, y'_c + y_d] \tag{3.4}$$

Bod $[x'_c, y'_c]$ je střed cévy, který byl vypočten v předchozím kroku. Zbývá tedy vypočíst posun středu $[x_d, y_d]$. Ten získáme při minimalizaci funkce

$$\sum_{i,j} [plane'(i,j) - plane(i + x_d, j + y_d)]^2, \tag{3.5}$$

kde $plane'$ a $plane$ jsou parametrizace roviny kolmé na cestu v předchozím a současném kroku. Hodnoty x_d a y_d jsou parametry optimalizačního algoritmu, pomocí nichž se funkce minimalizuje. Několik metod, které lze použít pro minimalizaci funkce 3.5, je popsáno v kapitole 5.2.2.

Tento postup má bohužel mnoho nevýhod — největší je velká časová náročnost při minimalizaci funkce 3.5. Výsledky publikované v práci [11] poukazují na to, že časová složitost tohoto algoritmu v porovnání s metodou vrhání paprsků je o několik řádů horší a dosažené výsledky víceméně stejné. Další nevýhodou je implementační náročnost požadovaného optimalizačního algoritmu.

Těžiště (Center Of Gravity)

V práci [12] je popsána metoda hledání objektu v černobílém obrázku pomocí určení těžiště. Po aplikaci tohoto algoritmu společně se znalostí cesty procházející cévou se střed cévy vypočítá následovně:

$$[x_c, y_c] = \left(\frac{\sum_{[x,y]\in\Omega} xw(x,y)}{\sum_{[x,y]\in\Omega} w(x,y)}, \frac{\sum_{[x,y]\in\Omega} yw(x,y)}{\sum_{[x,y]\in\Omega} w(x,y)} \right), \tag{3.6}$$

celý výpočet středu cévy (těžiště) se odehrává v rovině $plane$. Tato rovina je kolmá na cévu v právě zpracovávaném bodě cesty. Ω je množina obsahující všechny body tepny. Funkce $w(x, y)$ je váha bodu cévy o souřadnicích $[x, y]$. Ta je definována jako

$$w(x,y) = plane(x,y) - m, \quad m = \min_{[x,y]\in\Omega} (plane(x,y)).$$

Zbývá určit body, které obsahuje množina Ω, ty jsou podobně jako v metodě *Vrhání paprsků* určeny intervalem minimální a maximální hodnoty intenzity cévy. Tím tato metoda dědí některé potíže spojené s určením příslušného intervalu, ovšem i přes tento neduh patří mezi nejúspěšnější při zpracování syntetických dat, a to jak časovou náročností, tak přesností určení středu. Více o této metodě lze nalézt v práci [11] a [12].

3.6 Deformovatelné modely [2]

3.6.1 Active Contour Models (snakes)

Aktivní kontury jsou založeny na minimalizaci energetického funkcionálu E_{snake} v následující rovnici

$$E^*_{snake}(\mathbf{v}) = \int_0^1 E_{snake}(\mathbf{v}(t))dt, \quad E_{snake} = E_{int} + E_{image} + E_{con} \tag{3.7}$$

je $\mathbf{v}$(t) parametrická křivka, přes jejíž body (*snaxel*) se provádí integrace. Na každý bod působí interní a externí síly. Zjednodušeně řečeno, interní síly E_{int} udržují křivku vyhlazenou a "elastickou", proti nim působí externí síly E_{image}, které body křivky "vytahují" směrem k hledaným konturám. Poslední člen E_{con} v rovnici 3.7 obsahuje uživatelem zadané omezující podmínky, které určitým způsobem ovlivňují výpočet. Pěkně zpracované *deformovatelné objekty* včetně jejich odvození a několika rozšíření jsou v diplomové práci Václava Krajíčka [16].

Vylepšení aktivních kontur při segmentaci složitých tvarů řeší [22] zavedením topologické informace do postupu jejich výpočtu. Autor této práce dokládá úspěšnost topologických aktivních kontur právě na segmentaci cév na angiografických datech i na datech vyšší dimenze pořízených pomocí MR nebo CT. Přesto se této metody při segmentaci cév příliš nevyužívá. Preferují se spíše metody založené na *level-setech*, jejichž popis následuje.

3.6.2 Level-sets

Hlavní myšlenkou těchto algoritmů je reprezentovat hledanou křivku jako vrstevnici nulté hladiny funkce ve vyšší dimenzi než jsou vstupní data. Tedy místo hledání křivky jako v *aktivních konturách* metoda hledá *level set* funkci, jejíž součástí je tato křivka. Na *level set* funkci opět působí síly, které definují její výsledný tvar. Tyto síly jsou obvykle kombinací sil odvozených ze zpracovávaného obrázku a vlastností generované *level set* funkce.

Jedna z prací, která využívá těchto metod pro segmentaci cév na CT datech a úspěšně ji kombinuje s několika již popsanými metodami jako *multi-scale* nebo *zvýrazňující filtry*, je [18]. Metoda využívá pro zpřesnění segmentace cévy *level-sety*. Data jsou zpracována v několika krocích, z nichž první je nalezení středové cesty

[2] V anglicky psané literatuře známé jako Deformable Models.

tepnou. Cesta se hledá mezi dvěma uživatelem definovanými body jako minimální cesta v grafu. Ohodnocení hrany přecházející z jednoho bodu na druhý je vypočteno jako rozdíl zvýrazněné hodnoty tepny (vesselness value) obou bodů. Více o výpočtu této hodnoty pomocí vlastních čísel Hessovy matice v kapitole 3.3.1. Druhý krok segmentuje tepny pomocí *level set* funkce, pro počáteční nastavení funkce využívá informaci o poloze středové cesty tepnou. Funkci přitom deformují síly založené na kombinaci tří hodnot vypočtených v každém bodě. První je hodnota denzity (grey-level) bodu, druhá je hodnota gradientu (gradient-level) a třetí hodnota je zvýraznění tepny (vesselness value). Metoda vznikla původně pro data z MR, ale byla úspěšně testována i na CT datech.

Další práce [17] stejně jako předchozí využívá *zvýrazňujícího filtru* podle [6]. Hodnoty vypočtené filtrem normuje do intervalu $[0, 1]$ tak, aby hodnota 1 znamenala střed tepny, hodnota 0 hranici tepny a záporná hodnota body mimo tepnu. Metoda poté minimalizuje energii $E = \alpha E_{vessel} + (1 - \alpha)E_{boundary}$, kde konstanta $\alpha \in [0, 1]$ a E_{vessel} je integrál normovaných hodnot přes vnitřek tepny. Pro $E_{boundary}$ je využita teorie *geodetických křivek*. *Geodetické křivky* jsou vylepšením aktivních kontur. Kombinují vnější a vnitřní energie v jeden člen, který bere v úvahu jak vlastnosti obrazu (hrany), tak vlastnosti křivky (vyhlazenost, délka). Práce porovnává několik postupů výpočtu geodetických křivek, k jejich nalezení ve všech případech používá *gradientní metodu* (gradient descent). Více o geogetických křivkách v práci [21] nebo [16].

3.7 Parametrické modely

3.7.1 Šablony (Template Matching)

Tyto metody se pokoušejí vyhledat šablonu nebo šablony v obrázku. Šablona vznikne z reálných dat odbornou manuální segmentací. Můžeme mít například různorodou množinu šablon cévního řečiště, kde jsou zastoupena data od pacientů různého věku a pohlaví. Důležité je, že všechny šablony jsou určitě správně vytvořeny. Existuje mnoho metod, které se následně snaží nějakou šablonu z množiny vhodně umístit na vstupní data. Oblíbené je spojení s deformovatelnými objekty, které mohou zahájit výpočet ze znalosti tvaru reálné šablony. Další možností je použít některou z optimalizačních metod z kapitoly 5.2.2 spolu s *multi-scale* urychlením.

3.7.2 Eliptické modely

Metoda logicky vychází z eliptického tvaru cévy v jejím příčném řezu. Tyto algoritmy podobně jako algoritmy z kapitoly 3.5 znají počáteční cestu cévou. Každý bod této cesty se pak snaží co nejpřesněji posunout do středu cévy. Střed cévy naleznou jako střed elipsy, kterou umístí přesně na hranici cévy. V práci [30] je popsáno a porovnáváno několik sofistikovaných modelů testovaných na datech MR angiografie, ale použitelných i pro data CT angiografie. Jako zástupce těchto algoritmů si ovšem uvedeme jednodušší metodu *Lícování elipsy*.

Lícování elipsy (Ellipse Fitting)

V každém příčném řezu počáteční cesty cévou algoritmus aplikuje v okolí bodu cesty Cannyho hranový detektor [13]. Tím získá množinu bodů P, které jsou na hranici cévy. Z této množiny bodů P pak pomocí optimalizačního algoritmu získá parametry výsledné elipsy a z nich již jednoduše určí střed cévy nebo také její průměr.

Algoritmus řeší problém, jak proložit kuželosečku $C(\mathbf{a}) = \{x|F(\mathbf{a};x) = 0\}$ množinou bodů P. Hledáme tedy takový vektor $\mathbf{a}$, aby všechny body z množiny P ležely v množině $C(\mathbf{a})$. Obecná rovnice kuželosečky je dána vztahem

$$F(\mathbf{a};x) = a_0x_i^2 + a_1x_iy_i + a_2y_i^2 + a_3x_i + a_4y_i + a_5 = \mathbf{x_i} \cdot \mathbf{a^T},$$

kde $\mathbf{a} = [a_0, a_1, a_2, a_3, a_4, a_5]$ a $\mathbf{x_i} = [x_i^2, x_iy_i, y_i^2, x_i, y_i, 1]$.

Přesně tento problém řeší i práce [14]. Na deseti stranách se zabývá otázkou, jaký minimalizační algoritmus zvolit a jak je který náchylný na nekompletní data či data zatížená šumem. Například klasický postup, který minimalizuje sumu kvadrátů geometrické vzdálenosti každého bodu množiny P od množiny $C(\mathbf{a})$ pomocí simplexové metody, je oproti ostatním algoritmům tak pomalý, že jej v této práci odmítli šířeji otestovat. Na druhou stanu dokument uznává, že je to postup nejkomplexnější a lépe se vypořádá s reálnými daty. Za zmínku také stojí práce Radima Halíře a Jana Flussera [15], která prezentuje neiterativní metodu nalezení optimálně proložené elipsy. Autoři přebírají myšlenku navrženou A. W. Fitzgibbonem, autorem práce [14], tu pak vylepšují tím, že pomocí vhodné dekompozice matice a jejích vlastních čísel vypočtou parametry elipsy. Tato metoda má být velmi rychlá a odolná vůči nekvalitním datům.

První a zároveň největší potíže přináší nastavení Cannyho hranového detektoru tak, aby označil body pouze na hranici tepny. Poté, co jsou známy body na

hranici cévy, lze pomocí některého z algoritmů z prací [14] nebo [15] získat výsledné parametry elipsy.

Metoda lícování elipsy nejlépe ze všech uvedených algoritmů v kapitole o hledání středu tepny detekuje střed tepny na syntetických datech. Jejímu nasazení na reálná data nejvíce brání malá robustnost, která pramení z detekce hran cévy pomocí Cannyho filtru.

3.7.3 Válcovité modely

Válcovité modely s eliptickým či kruhovým průřezem jsou nasnadě, pokud rozmýšlíme, jakému modelu připodobnit cévy. Metody veskrze fungují na obdobném principu, který obsahuje tři kroky. V prvním kroku odhadnou bod, který je uvnitř nebo v blízkosti cévy. V druhém kroku do tohoto bodu umístí válcovitý model a pomocí různých optimalizačních metod se snaží odhadnout parametry modelu tak, aby model s výslednými parametry co nejpřesněji lícoval s hledanou cévou. Takto postupuje po celé délce segmentované cévy. V třetím kroku je schopen z parametru modelu podél celé délky cévy zrekonstruovat nebo klasifikovat cévu. Obvyklými parametry takového modelu jsou pozice, poloměr, rotace a jiné podle definovaného modelu. Práce [31] představuje válcovitý model, který se iterativně umísťuje na cévu v reálných datech pomocí Kalmanova filtru. Kalmanův filtr rekurzivně odhaduje a zpřesňuje parametry modelu a tím minimalizuje měrnou funkci. Měrná funkce počítá čtverec rozdílu mezi modelem a daty, na které se model umísťuje. Zmiňovaná práce hodnotí úspěšnost tohoto postupu na datech magnetické rezonance. Alexandra La Cruz ve své disertační práci [11] popisuje použití téměř shodného modelu na datech počítačového tomografu. Jelikož součástí této diplomové práce je implementace podobného modelu jako v dílech [11] a [31], ponecháváme jeho další popis do kapitoly 5.

Kapitola 4

Metody vizualizace tepen

Diplomová práce se zaměřuje spíše na segmentaci cév nežli na jejich zobrazování, proto zde podrobně vysvětlíme pouze zobrazovací metodu CPR (Curved Planar Reformation) a její tři podtypy. Tato metoda byla vyvinuta přímo za účelem zobrazení cév, proto byla také v rámci této práce implementována. Ostatní obecné metody jako je *Maximum Intenzity Projection* (MIP) nebo *Direct Volume Rendering* (DVR) zmíníme v závěru kapitoly.

4.1 CPR — Curved Planar Reformation

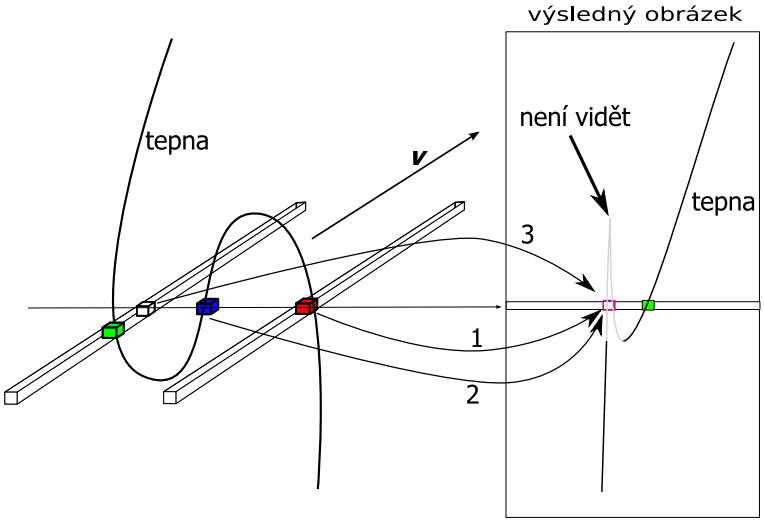
CPR je zobrazovací technika, která umožňuje zobrazit celou cévu v jednom dvojrozměrném obrázku. Metoda byla rozpracována v rámci projektu ADAPT na Technické univerzitě ve Vídni. Její popis byl poté publikován v práci Armina Kanitsara [8] a v o rok mladší vylepšené verzi [9].

Ač tato metoda neposkytuje ideální prostorovou představu o průběhu cévy, je mezi radiology velmi populární. Ti totiž mohou cévu v tomto zobrazení snadno prohlížet a detekovat její patologické změny. Jedná se hlavně o rozpoznání stenóz, aneuryzmat a přítomnosti kalcifikovaných plátů v cévní stěně.

Metoda je založena na znalosti cesty, která prochází středem cévy. Cesta je reprezentována jako sekvence voxelů. S takovou vstupní informací pracují všechny tři druhy této metody popsané v následujících podkapitolách.

4.1.1 Projected CPR

Jedná se o projekci úzkého řezu vstupních dat do výsledného dvojrozměrného obrázku. Onen úzký řez je definován vstupní cestou a vektorem pohledu **v**. Každým bodem cesty je proložena přímka ve směru vektoru **v**, tato přímka je poté promítnuta do výsledného obrázku. Důležité je, že vektor **v** je během zpracování všech bodů cesty neměnný. Metoda má i několik nevýhod, které lze vidět na obrázku 4.1, a které jsou popsány níže.

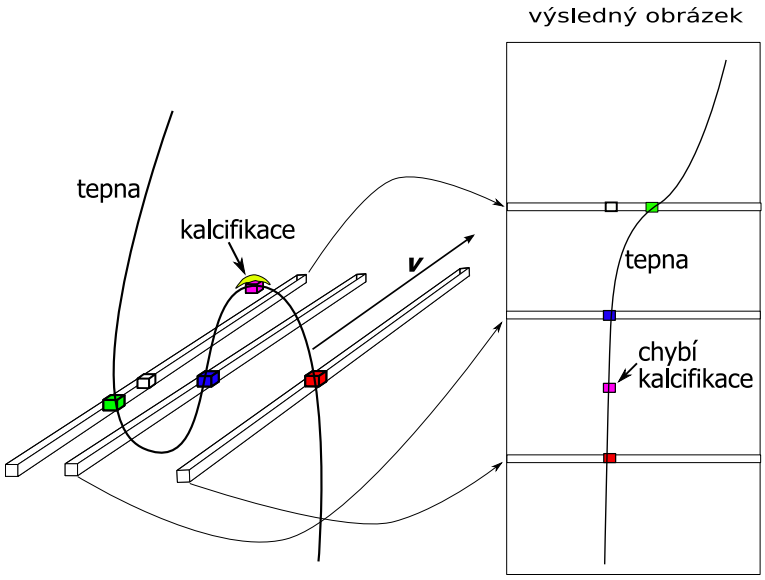


Obrázek 4.1: Projected CPR — princip vykreslení cévy.

1. Některé úseky cévy mohou být překryty částmi, které algoritmus zpracovává později. Ve výsledném obrázku pak mohou chybět i velké části zobrazované cévy, což je názorně vidět na obrázku 4.1. Částečně lze tento problém řešit pomocí *Maximum Intenzity Projection* (MIP) překrývajících se řádků případně počítáním průměrné hodnoty (AVG) těchto řádků.
2. Zobrazení Projected CPR nezachovává izometrii, tedy délka cévy na výsledném obrázku není stejná jako její skutečná délka. V takovém obrázku pak není možno provádět měření.
3. Pokud se ve stěně tepny vyskytuje menší kalcifikovaný plát a pokud přímka ve směru pohledu plát neprotíná, pak se tento ve výsledku nezobrazí.
4. Některé části zobrazované cévy se zobrazují nekompletní a imitují tak přítomnost stenózy, dokonce i v oblastech, kde je naopak přítomno rozšíření cévy v podobě aneuryzmatu.

4.1.2 Stretched CPR

Vylepšení *Projected CPR*, které již nepřekrývá vykreslené body a zachovává izometrii. Možnost měření skutečných vzdáleností ve výsledném obrázku a obstojná prostorová orientace v obrázku jsou jeho hlavními výhodami. Na druhou stranu metoda trpí všemi ostatními neduhy uvedenými v předchozí podkapitole. Princip jejího fungování je zřejmý z obrázku 4.2. Vstupní cesta musí mít "sub-voxelové" rozlišení (k dopočítání hodnot bodů nacházejících se na cestě mezi body cestu definujícími se používá lineární interpolace).



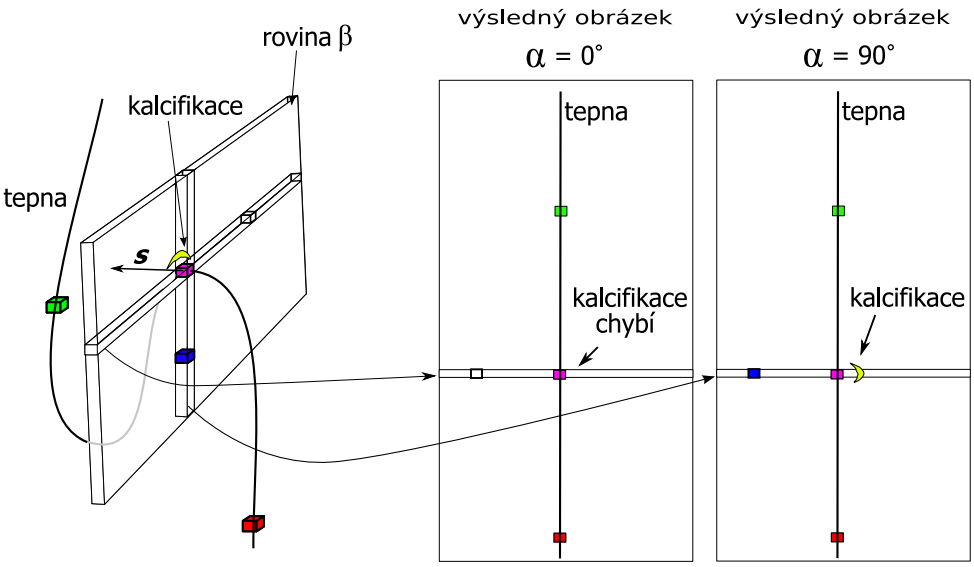
Obrázek 4.2: Stretched CPR — princip vykresnení cévy.

4.1.3 Straightened CPR

Algoritmus se od ostaních typů CPR zobrazení odlišuje tím, že zkoumanou cévu natáhne do přímky. Na takto natažené cévě jsou lehce viditelné hledané stenózy nebo aneuryzmata, za což platíme horší prostorovou orientací v obrázku. Vstupem tohoto algoritmu je jako u předchozích metod cesta středem cévy (center-line), od této cesty je navíc požadována schopnost generovat v každém bodě směrový vektor. Dalším vstupem je úhel, pomocí kterého se počítá vektor pohledu na cévu (vector of interest). Výstupem je obrázek natažené cévy, který má výšku shodnou s délkou cesty.

Algoritmus 2 Algoritmus výpočtu Straightened CPR.

1. Na vstupu algoritmu je cesta cévou (center-line), úhel α a vstupní obrázek.
2. Pro každý bod cesty i vygenerujeme i-tý řádek bodů do výsledného obrázku. Tedy výška výsledného obrázku bude odpovídat počtu bodů na cestě cévou.
   (a) V bodě i vypočteme směrový vektor cesty $\mathbf{s}$.
   (b) Rovina β je definována bodem i a vektorem $\mathbf{s}$ (Rovina β je příčným řezem cévy v bodě i).
   (c) V rovině β zvolíme bázové vektory. Pomocí bázových vektorů a úhlu α vypočteme vektor pohledu $\mathbf{v}$.
   (d) Na přímce dané bodem i a vektorem pohledu $\mathbf{v}$ zvolíme body, které budou tvořit i-tý řádek výsledného obrázku. Prostřední bod musí být současně bodem i. Hodnoty v těchto bodech získáme interpolací vstupního obrázku.
3. Výstupem je 2D obrázek s libovolnou šířkou a výškou odpovídající počtu bodů na vstupní cestě.



Obrázek 4.3: Straightened CPR — princip vykreslení cévy.

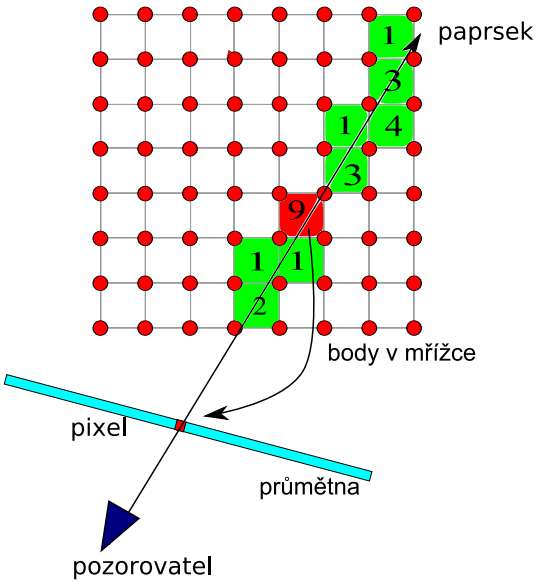
Tato technika je schopna si poradit se všemi nevýhodami, jmenovanými v podkapitole *Projected CPR*, kromě třetího bodu. Tento bod řeší dvě vylepšení algoritmu.

1. Do jednoho bodu výsledného obrázku se nepřenáší pouze jeden bod vstupního obrázku, ale více bodů, které jsou v rovině β a zároveň na kolmici k vektoru pohledu. Z nich pak vybereme například bod s největší intenzitou (MIP).

2. Další vylepšení se nazývá *Rotating CPR*, místo jedné *Straightened CPR* s úhlem pohledu α se vygeneruje více snímků s různými úhly, například $0°, 30°, 60°, 90°, 120°, 150°, 180°$ — jako na obrázku 4.3.

4.2 MIP — Maximum Intensity Projection

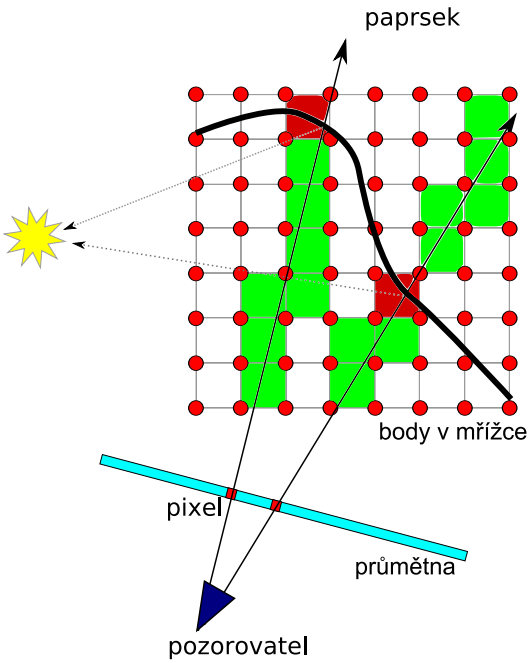
Nejčastěji používaná zobrazovací metoda v klinické praxi. Je založena na zobrazení voxelu s největší hodnotou intenzity, kterým prochází zobrazovací paprsek 4.4. Tato metoda je velmi snadná na implementaci, navíc lékaři jsou již ni zvyklí a dovedou se v datech tohoto typu dobře a rychle orientovat. Při znalosti segmentované cévy lze metodu vylepšit a body mimo ni nezobrazovat, to dává komfortnější podmínky pro její hodnocení. Na druhou stranu se tím zhorší prostorová orientace v datech a je nutný předpoklad správné segmentace cévy. Časová náročnost této metody je lepší než u metody DVR (kapitola 4.4), ale horší než u generování izoploch (kapitola 4.3).



Obrázek 4.4: Princip MIP (převzato od Lukáše Maršálka).

4.3 Izoplochy

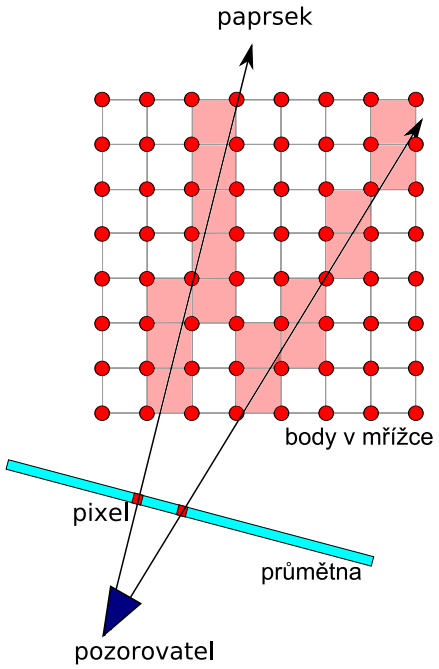
Metoda generuje plochy z objemové reprezentace dat. Zobrazuje pouze část objemu, a sice buňky protínající izoplochu definovanou zadaným prahem. Nejpoužívanějším algoritmem generujícím izoplochy z objemových dat je známý *marching cube*, který generuje síť trojúhelníků v každé buňce. Toto je rychlá metoda s výsledky dostačujícími pro mnohé účely.



Obrázek 4.5: Princip generování izoploch metodou *marching cube* (převzato od Lukáše Maršálka).

4.4 DVR — Direct Volume Rendering

Zobrazovací metoda, která zobrazuje celý objem. Výsledek zobrazení závisí na definici přechodové funkce. Ta umožňuje různým hodnotám buněk přiřadit různé vlastnosti, podle kterých se počítá jejich příspěvek do výsledného zobrazení. Metodu si lze představit jako průchod světla objemem dat. Existuje několik modelů, podle kterých se světlo procházející objemem simuluje. V lékařství je nejpoužívanější *Vyzařující pohlcující model*, který zanedbává rozptyl světla.



Obrázek 4.6: Simulace světla (převzato od Lukáše Maršálka).

Kapitola 5

Zvolený postup segmentace

Tato kapitola se věnuje výběru a popisu vhodného postupu segmentace tepen, který bude v rámci této práce implementován. V kapitole 3 bylo uvedeno množství algoritmů, které lze pro tento účel použít. Příslušné algoritmy můžeme rozdělit do dvou skupin podle toho, kolik a jak kvalitních informací poskytují při postupu tepnou.

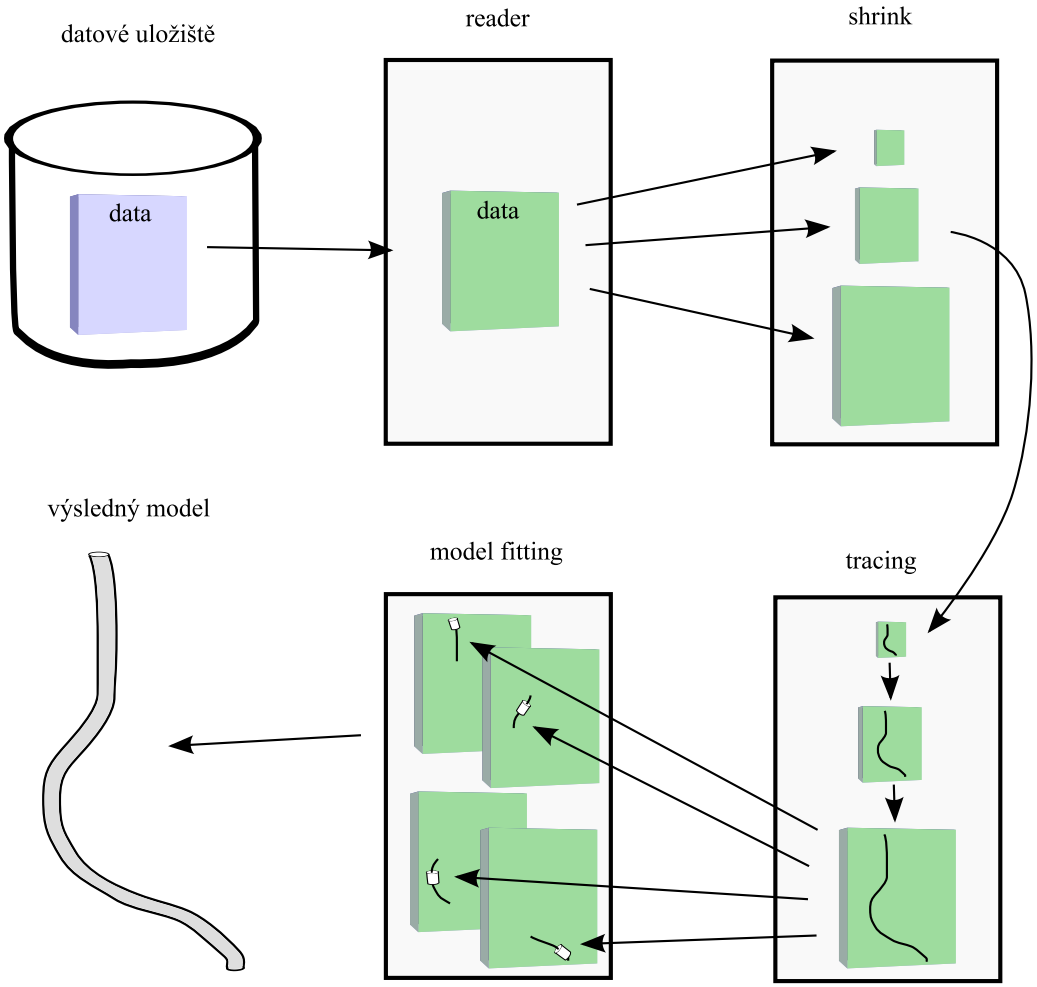
Prvním typem jsou algoritmy, které procházejí cévní řečiště z počátečního bodu do koncového, případně více koncových bodů. Při použití některých z mnoha možných vylepšení jsou tyto algoritmy poměrně rychlé, mají ovšem jednu podstatnou nevýhodu — nejsou schopny poskytovat plnohodnotnou informaci o segmentované tepně. Jejich výstupem je většinou pouze cesta procházející touto tepnou.

Druhým typem jsou algoritmy, které nějakým způsobem optimalizují vztah modelu a reálných dat. Tímto jsou míněny algoritmy spadající do kategorie deformovatelných modelů a parametrických modelů z kapitoly 3.6 a 3.7. Tyto algoritmy jsou schopny poskytovat více informací při postupu tepnou, například její průměr, směr nebo pozici středových bodů tepny. Jejich jedinou, avšak podstatnou nevýhodou je časová náročnost, která závisí na počtu parametrů a na zvoleném optimalizačním algoritmu.

Při návrhu algoritmu se pokusíme vhodně zkombinovat oba druhy výše zmíněných algoritmů. Využijeme rychlého vyhledání cesty tepnou a pomocí této znalosti budeme optimalizovat cestu tak, aby procházela středem tepny. Toho docílíme parametrickou optimalizací modelu a navíc získáme další informace o tepně, které by pak měly postačit ke správné diagnostice.

Důvodem využití prvotně vyhledané cesty je pozdější snadná paralelizovatelnost metody při časově náročné optimalizaci parametrického modelu. Druhým přístupem

je inicializovat parametry pomocí výsledků získaných v minulém kroku optimalizace. To ovšem vede k sekvenčnímu algoritmu, při kterém bychom nemohli využít největší přednosti knihovny ITK (příloha A) — tou je paralelizace pomocí proudového zpracování dat. Na obrázku 5.1 je zachycen návrh algoritmu s vyznačeným paralelním zpracováním časově náročných úloh.



Obrázek 5.1: Schéma algoritmu, paralelní zpracování (vlákna) jsou naznačena šipkami.

5.1 Hledání cesty

Na voxely vstupního obrázku budeme nahlížet jako na vrcholy grafu. Pokud dva voxely mají společný alespoň jeden vrchol, pak definujeme mezi těmito vrcholy grafu neorientovanou hranu. U trojrozměrného obrázku se tedy jedná o 26-ti sousednost. Pro

každou hranu je spočítáno její ohodnocení podle toho, zda je či není součástí tepny. Funkce f_{cost}, která počítá ohodnocení hran, je jediná, která může ovlivnit výpočet výsledné cesty. Z tohoto důvodu se jí budeme věnovat podrobněji v podkapitole 5.1.4, bezprostředně po popisu Dijkstrova algoritmu a několika jeho vylepšení.

5.1.1 Dijkstrův algoritmus

Algoritmus pro hledání nejkratší cesty v grafu prvně popsaný v Dijkstrově práci [32], pracuje pouze na grafech s kladným ohodnocením hran. Nechť tedy graf $G(V, E)$ je takový graf. Množina Z obsahuje algoritmem navštívené vrcholy grafu, k těmto vrcholům je již známa nejkratší cesta z počátečního vrcholu s. Množina N naopak obsahuje dosud nenavštívené vrcholy. Dále označme $d[v]$ délku cesty z vrcholu v do počátečního vrcholu s. Pokud je $d[v] = \infty$, pak cesta z vrcholu s do vrcholu v dosud nebyla nalezena. Algoritmus v každém kroku i najde všechny vrcholy z množiny N, které mají některý sousední vrchol v množině Z. Množinu, kam patří právě tyto nalezené vrcholy, pojmenujeme R_i, pro všechny její vrcholy q spočteme $d[q]$. Poté vybereme takový vrchol q_i z množiny R_i, který má nejmenší $d[q]$ — vzdálenost od počátečního vrcholu s. Tento vrchol zařadíme do množiny Z a odstraníme z množiny N. Takto pokračujeme, dokud není množina N prázdná. Algoritmus 3 popisuje hledání cesty z bodu s do bodu e a zavádí operace *delete_min* a *decrease_key*.

Časová složitost algoritmu je počet vrcholů n vynásobený časovou složitostí operace *delete_min* plus počet hran m vynásobený časovou složitostí operace *decrease_key*. Například při implementaci prioritní fronty pomocí pole dostaneme celkový čas $\mathcal{O}(n^2 + m)$. Je tedy vidět, že "úzkým hrdlem" tohoto algoritmu je implementace prioritní fronty.

Prvním vylepšením je nevkládat do prioritní fronty vrcholy, jejichž $u[v] = \infty$. To algoritmus sice trochu urychlí, ale praktické zkušenosti s takovouto implementací ho navzdory tomu zařadily do kategorie téměř nepoužitelných algoritmů.

5.1.2 Diskretizace ohodnocení hran

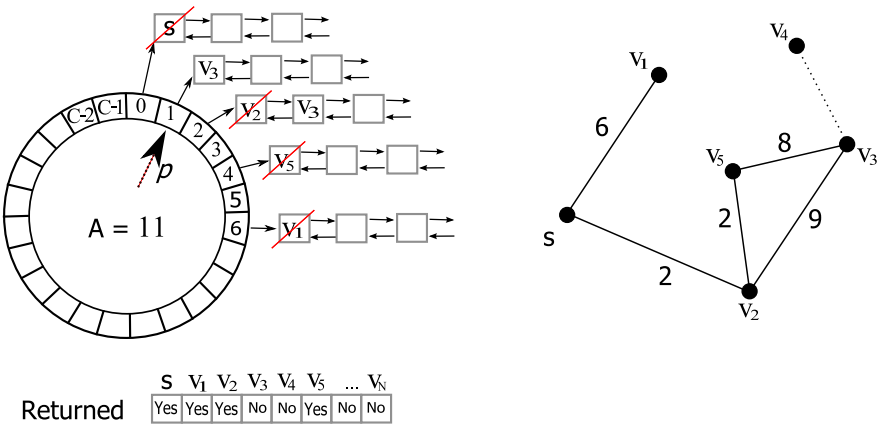
Řádové urychlení přináší diskrétní prioritní fronta popsaná v [25] a [28]. Jejím předpokladem je diskrétní a omezená funkce počítající ohodnocení hran f_{cost}. Předpokládáme tedy, že hrany grafu mají hodnoty v oboru přirozených čísel včetně nuly z intervalu $[0, C]$. Pokud je ohodnocení hrany rovno číslu C, pak je tato hrana

Algoritmus 3 Dijkstrův algoritmus.

1. Vstupem je graf $G(V, E)$, který má kladně ohodnocené hrany, bod počáteční bod cesty s a koncový bod cesty e.
2. Nastavíme $d[s] = 0$, $\forall v \in V \backslash s : d[v] = \infty$ a $\forall v \in V : odkud[v] = nil$.
3. Vytvoříme prioritní frontu F z vrcholů V podle priority $d[v]$.
4. Dokud vrchol e je zařazen v prioritní frontě F,
   (a) $v = delete\_min(F)$ {Najdeme vrchol v s nejkratší cestou do vrcholu s a odstraníme ho z fronty.}
   (b) $Z := Z \cup v$ {Přidáme vrchol do množiny vrcholů s již nalezenou cestou.}
   (c) $\forall vw \in E$, kde $v \in Z$ a $w \in F$ a $u[v] + f_{cost}(vw) < u[w]$ spočítáme nové $u[w] = u[v] + f_{cost}(vw)$ a $odkud[w] = v$. Těmto vrcholům také aktualizujeme priority v prioritní frontě F operací $decrease\_key(w, F)$.
5. Cestu zpětně rekonstruujeme pomocí informací uložených v poli *odkud*.

pro algoritmus nedostupná, jako by hodnota hrany byla nekonečno. Diskrétní prioritní fronta je ve své podstatě cyklická fronta, ve které jsou uloženy ukazatelé na obousměrné spojovací seznamy. Fronta dále udržuje číslo A (uprostřed na obrázku 5.2), které odpovídá maximální délce cesty v dosud navštívené části grafu. Také je to délka cesty z počátečního bodu s do právě zpracovávaných vrcholů (ty jsou v obousměrném seznamu, na který ukazuje ukazatel p uprostřed obrázku).

Následuje obrázek, kde je patrno, jak fronta pracuje na grafu z pravé části obrázku. Algoritmus je právě při zpracování vrcholu v_3. Tečkované hrany grafu jsou dosud nenavštívené části grafu.



Obrázek 5.2: Diskrétní prioritní fronta při zpracování grafu vpravo pro $C = 10$.

Od této datové struktury požadujeme rychlé operace *delete_min* a *decrease_key*. U operace *decrease_key* si pomáháme mapou *Returned*, do které si poznačíme všechny vrácené vrcholy v operaci *delete_min*. Díky tomu můžeme do prioritní fronty vkládat jeden vrchol vícekrát, prioritní fronta ho vrátí jen jednou, a to pokud je do tohoto vrcholu nalezena nejkratší cesta. Dále pro lepší přehlednost uvedeme algoritmický popis obou metod:

Algoritmus 4 Operace *delete_min* na diskrétní prioritní frontě.

1. Nastavíme počitadlo $D = 0$.
2. Je-li $D > C$, algoritmus končí — prošel celým grafem. Jinak pokračujeme dalším krokem.
3. Pokud je obousměrný spojový seznam, na který ukazuje ukazatel p, prázdný, pokračujeme bodem 6, jinak dalším bodem.
4. Odstraníme první vrchol v ze seznamu, na který ukazuje p.
5. Pokud $Returned[v]$ = 'Yes', pokračujeme dalším bodem. Jinak nastavíme $Returned[v]$:= 'Yes' a vrátíme vrchol v.
6. Přesuneme ukazatel p na další pole v cyklické frontě. Zvýšíme $D := D + 1$ i $A := A + 1$ a jdeme na krok číslo 2.

Algoritmus 5 Operace *decrease_key* na diskrétní prioritní frontě.

1. Vstupem je vrchol v a délka nalezené cesty (do vrcholu s) $lenght$.
2. Spočteme index do cyklické fronty $idx := lenght \mod C$.
3. Do spojového seznamu, do něhož ukazuje ukazatel s indexem idx, přidáme na konec vrchol v.

Časová složitost operace *delete_min* je $\mathcal{O}(C)$, u operace *decrease_key* je $\mathcal{O}(1)$. Tedy celková časová složitost takto upraveného Dijkstrova algoritmu je $\mathcal{O}(nC + m)$. V praxi tato optimalizace přinesla urychlení algoritmu několiksetkrát.

5.1.3 Multi-scaling[1]

Dijkstrův algoritmus při procházení tepnou navštíví velké množství vrcholů, které pak nevyužije při rekonstrukci výsledné cesty. Omezení počtu těchto vrcholů dosáhneme využitím *Multi-scale* techniky popsané v kapitole 3.1. Myšlenka je jednoduchá. Nejdříve se najde cesta tepnou ve zmenšeném obrázku. Poté, při hledání cesty ve větším obrázku, se prochází pouze ty body, které jsou součástí již nalezené cesty nebo s ní sousedí. Tímto se výrazně omezí počet algoritmem navštívených bodů. Postup má ovšem také svá negativa — pokud je cesta ve zmenšeném obrázku nalezena chybně, pak ve zvětšeném se cesta buď vůbec nenajde nebo je chybná. Tento problém vzniká při hledání cesty v příliš zmenšených datech, ve kterých má navíc segmentovaná tepna výrazně vinutý průběh.

Celkový popis použití pyramidového přístupu je zapsaný v pseudokódu na konci této kapitoly. Tato technika přináší významné urychlení při velkém objemu vstupních dat a jejich členitosti.

5.1.4 Ohodnocení hran

Jak již bylo zmíněno, ohodnocení hran je jedním z mála způsobů, jak ovlivnit průběh výpočtu cesty v datech a kvalitu jeho výstupu. Existuje mnoho faktorů, které mohou průběh cesty ovlivnit. Například práce [7] definuje ohodnocující funkci $f_{cost}(x, y)$ mezi vrcholy x a y jako:

[1] Česky též nazývaný pyramidový přístup.

$$f_{cost}(x, y) = c_{step} + f_I(y) + f_G(x, y) + f_L(y) \tag{5.1}$$

První faktor c_{step} je počet kroků do vrcholu x. Zapříčiní, že výsledná cesta bude mít co možná nejmenší počet kroků, jde tedy o jakési vyhlazení cesty.

Další faktor $f_I(y)$ je založen na prahování hodnot a definován jako:

$$f_I(y) = \begin{cases} \infty & f(y) < border_{min} \\ \omega_{lower}(vessel_{min} - f(y)) & border_{min} \leq f(y) < vessel_{min} \\ 0 & vessel_{min} \leq f(y) \leq vessel_{max} \\ \omega_{upeer}(f(y) - vessel_{max}) & vessel_{max} < f(y) \leq border_{max} \\ \infty & border_{max} < f(y) \end{cases}$$

Omezuje počet vrcholů, které budou při hledání cesty procházeny. Parametry $border_{min}, border_{max}, vessel_{min}, vessel_{max}$ [2] jsou prahové hodnoty denzity tepny. Další parametry ω_{lower} a ω_{lower} jsou penalizace, pokud cesta jde přes vrcholy, jejichž denzita není v intervalu $[vessel_{min}, vessel_{max}]$. Tím lze dosáhnout toho, že cesta přes kalcifikovaný plát, který má vysokou denzitu, nebude tak penalizována jako cesta, která vybočí mimo tepnu, kde je hodnota denzity naopak nízká.

Faktor f_G počítá velikost změny denzity mezi vrcholy x a y. Je počítán jako $f_G(x, y) = |f(x) - f(y)|$. Preferuje směr v ose tepny, kde se denzita nemění, narozdíl od směru k okraji tepny.

Čtvrtý člen f_L rovnice 5.1 je hodnota konvoluce obrazu v bodě y získaná pomocí Laplaceova operátoru a omezená prahovou hodnotou $c_{Laplace}$.

$$f_L(y) = \begin{cases} \infty & (L * f)(y) > c_{Laplace} \\ 0 & jinak \end{cases}$$

Tento faktor by měl penalizovat cestu, která směřuje ven z tepny do kostních struktur. Ovšem v praxi se tento faktor příliš neosvědčil.

Během implementace bylo dále vyzkoušeno několik dalších faktorů, které by mohly mít naději na úspěch. Některé nyní popíšeme, zhodnotíme z praktického hlediska a složíme z nich výslednou ohodnocující funkci f_{cost}.

Faktor preferující aktuální směr cesty f_A je vyjádřením následující myšlenky: Pokud šla cesta nějakou dobu daným směrem, pak je pravděpodobné, že tímto

[2] Podrobnější popis těchto parametrů je na konci kapitoly v závěrečném shrnutí.

směrem půjde i nadále. Z pole *odkud* můžeme získat informaci, kudy šla cesta do právě zpracovávaného bodu x. Nechť vrchol w je vrcholem, vzdáleným N kroků zpět po cestě od vrcholu x, a $\tau(x)$ je označením pozice vrcholu x. Vypočteme tedy směrový vektor cesty $\mathbf{t} = \tau(x) - \tau(w)$[3]. Dále vypočteme vektor hrany, jejíž ohodnocení se právě počítá $\mathbf{b} = \tau(y) - \tau(x)$ (směr, kam cesta může pokračovat). Nakonec pomocí skalárního součinu můžeme vypočítat úhel, který oba vektory svítají. Tuto hodnotu lze přímo použít jako výsledek. Tedy $f_A(x, y) = arccos(\frac{\mathbf{t} \cdot \mathbf{b}}{|\mathbf{t}||\mathbf{b}|})$.

Podobný je faktor preferující směr k cíli, pouze vektor $\mathbf{t}$ se počítá jako $\mathbf{t} = \tau(e) - \tau(x)$, kde e je koncový bod cesty. Tento faktor preferuje hrany, které jsou natočeny směrem k cílovému vrcholu.

Faktor preferující střed tepny f_M dá přednost vrcholu, jehož sousedé mají hodnoty denzity stejné jako je průměrná hodnota tepny $vessel_{avg}$. Je vyjádřen vzorcem $f_M(x) = \sum_{u \in \Omega(x,N)} |vessel_{avg} - f(u)|$, kde $\Omega(x, N)$ je množina všech okolních vrcholů vrcholu x do vzdálenosti N. Faktor udržuje cestu poblíž středu tepny, pokud ta ovšem obsahuje větší kalcifikace, vyhýbá se jim a cestu raději hledá na okraji tepny.

Faktor preferující válcovitou strukturu je založen na stejném principu jako *Zvýrazňující filtry* v kapitole 3.3.1. Tedy při ohodnocení hrany spočítá pro oba vrcholy vlastní čísla Hessovy matice a podle rovnice 3.1 vypočte hodnoty ν_x a ν_y. Za výslednou hodnotu lze pak brát například $|\nu_x - \nu_y|$.

Faktor preferující blízkost cíle f_T počítá vzdálenost vrcholu x od cíle e. Lze počítat jak klasickou Euklidovskou vzdálenost $f_T(x) = \sqrt{\sum_{i=0}^{2} (x_i - e_i)^2}$, tak Manhattan vzdálenost $f_T(x) = \sum_{i=0}^{2} (x_i - e_i)$.

Vytvořit výslednou ohodnocující funkci f_{cost} jako lineární kombinaci těchto faktorů je nelehký úkol. Pro každý faktor lze uvést příklad dat, na kterých nebude správně fungovat. Praktické testy navíc ukázaly, že ideální kombinace faktorů se pro různá data velmi liší a nelze tudíž odvodit žádné pravidlo nebo vybrat jen ty úspěšné. Zajímavá by jistě byla práce věnující se dynamickému určování vah jednotlivých faktorů podle dosavadního průběhu cesty a zpracovaných dat. V této práci zvolíme kompromisní řešení, které přináší uspokojivé výsledky na všech testovaných datech. Výsledný vzorec pro výpočet funkce f_{cost} zapisujeme jako:

$$f_{cost}(x, y) = \alpha f_I(y) + \beta f_M(y) + \gamma f_T(y), \tag{5.2}$$

[3]Vektor $\mathbf{t}$ lze spočítat i lépe, podobně jako se počítá směr cesty v kapitole 5.2.4.

kde α, β a γ jsou váhy jednotlivých faktorů. Ty mohou být stejné pro různé typy dat. Nakonec je třeba ohodnocení hrany zaokrouhlit na celé číslo a "normalizovat" tak, aby spadalo do intervalu $[0, C]$.

5.1.5 Shrnutí

Celkový algoritmus lze zapsat takto:

Vstupy algoritmu:

- Vstupní 3D data — obrázek, ve kterém se nachází kontrastní látkou zvýrazněná tepna.
- Souřadnice počátečního a koncového bodu cesty s a e.
- Interval $[vessel_{min}, vessel_{max}]$, kde $vessel_{min}$ je minimální hodnota denzity ideální tepny a $vessel_{max}$ je maximální hodnota denzity tepny.
- Interval $[border_{min}, border_{max}]$ obsahuje hodnoty denzity, které se mohou vyskytovat v tepně, například hodnoty kalcifikací. Měla by platit podmínka $border_{min} \ll vessel_{min}$ a zároveň $vessel_{max} \ll border_{max}$.
- Hodnota $vessel_{avg}$ — jde o průměrnou hodnotu denzity tepny.
- Parametr τ říká, v jaké míře se mají při pyramidovém zmenšování vstupního obrázku preferovat průměrné hodnoty tepny. Parametr je z intervalu $[0, 1]$.

Výstup algoritmu:

- Seznam za sebou zřetězených bodů, které představují průchod tepnou od počátečního bodu ke koncovému.

Algoritmus 6 Algoritmus výpočtu cesty tepnou.

1. Vypočítáme několik obrázků zmenšených v různém měřítku. Při výpočtu hodnot zmenšených obrázků použijeme parametr τ, jak říká následující bod.

2. Pro každý voxel zmenšeného obrázku zjistíme, které voxely mu odpovídají ve větším obrázku. Spočteme jejich průměr p_{avg} a najdeme v nich hodnotu intenzity p_{near}, která je nejbližší hodnotě $vessel_{avg}$. Výsledná hodnota voxelu je pak $(1 - \tau)p_{avg} + \tau p_{near}$.

3. Pomocí vylepšeného Dijkstrova algoritmu vyhledáme cestu tepnou v nejmenším obrázku. Pokud je cesta úspěšně vyhledána, pokračujeme dalším bodem. Pokud vyhledána není, nejmenší obrázek se odstraní a pokračujeme znovu tímto bodem.

4. S pomocí informace o cestě vyhledané v předchozím kroku upřesňujeme cestu v obrázcích většího rozlišení. Pokud se cesta v právě zpracovávaném rozlišení nenajde, všechny obrázky menšího rozlišení se odstraní a pokračuje se předchozím bodem.

5. Pokud jsme nalezli cestu v obrázku s největším rozlišením, pak je dána na výstup a algorimus končí.

Při správně zadaných vstupních parametrech algoritmus vždy najde cestu uvnitř tepny. Oproti klasickému Dijkstrovu algoritmu je řádově rychlejší. Díky velké rychlosti není potřeba používat paralelní verzi tohoto algoritmu [33, 34].

5.2 Hledání středu cévy

Myšlenka těchto metod již byla popsána v kapitole 3.7.3. Zde uvedeme konkrétní příklad takovéto metody, která vychází z práce [11] a [31]. Tuto metodu jsme zvolili z důvodu proklamované velké robustnosti a přesnosti výsledných parametrů spolu s malou náchylností na nekvalitní vstupní data. Tato kapitola kromě detailního popisu vybrané metody prezentuje i několik jejích vylepšení, jako například redukci počtu optimalizovaných parametrů.

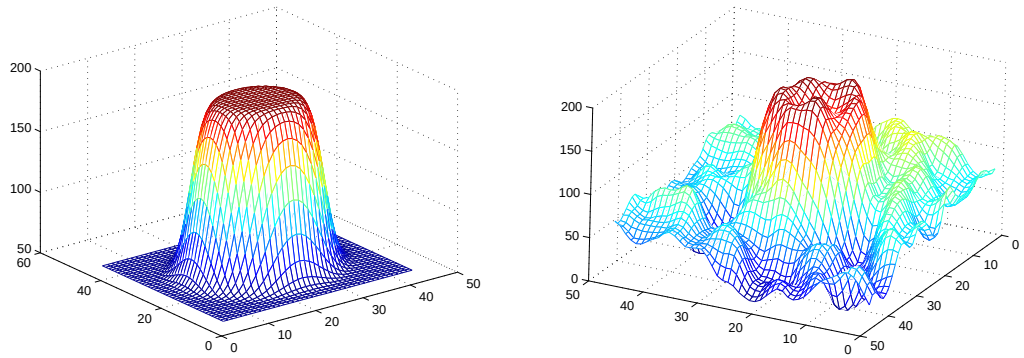
5.2.1 Model

Při získávání dat počítačovou tomografií dochází z důvodu konečného počtu detektorů k rozmazání obrázku (blurring). Toto rozmazání je charakterizováno pomocí *Point-Spread Function* (PSF), u CT dat je pak často tato funkce aproximována konvolucí s trojrozměrným Gaussovým jádrem a rozptylem σ. O problematice *Point-Spread* funkcí a možnostech jejich aproximace na CT datech lze číst v článku [37].

Náš model vychází z představy tepny jako homogenní ostře ohraničené vinuté trubice. Následně je třeba na tuto představu aplikovat konvoluci s trojrozměrným Gaussovým jádrem G_σ. Tato operace odstraní všechny ostré přechody a přiblíží výše uvedenou představu reálným datům. Model lze zapsat jako

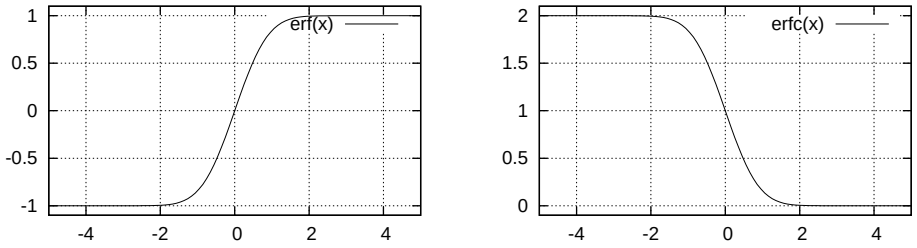
$$model = (b + V \cdot \eta) * G_\sigma, \tag{5.3}$$

kde b je konstantní funkce s hodnotou pozadí, V je také konstantní funkce s hodnotou denzity tepny a funkce η je definována jako 1 pro všechny body, které jsou součástí tepny, a jako 0 pro body ostatní.



Obrázek 5.3: Na obrázku vlevo je námi definovaný model, který se lícuje na reálná data vpravo. Pro lepší zobrazení je model na obrázku převeden do nižší dimenze.

Práce s tímto modelem by byla analyticky příliš složitá, proto se pokusíme v zápisu 5.3 zbavit konvoluce. Protože se jedná o konvoluci s Gaussovým jádrem, je aproximace jednoduchá. Použijeme *erf — error function*, ta je definována jako integrál z Gaussovy funkce, tedy přesně jako $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. My budeme pro zpřehlednění zápisu používat její doplňkovou funkci *erfc* definovanou jako $erfc(x) = 1 - erf(x)$. Průběh obou funkcí je na obrázku 5.4.



Obrázek 5.4: Průběh funkce erf (error function) a její doplňkové funkce erfc.

Po této úpravě vypadá rovnice modelu v bodě x následovně:

$$model(x) = b + V \cdot \text{erfc}\left(\frac{distance(x)}{\sigma}\right) \tag{5.4}$$

Parametr σ ovlivňuje rozmazání ostrých hran (blurring) modelu. Je zde také nová funkce *distance*, která je definována jako vzdálenost bodu x od stěny tepny. V této funkci je skryta složitost odstraněné konvoluce. Funkci *distance* můžeme odhadnout pomocí vzorce

$$distance(x) = \frac{f(x)}{\|\nabla f(x)\|}, \tag{5.5}$$

kde f je implicitní funkce geometrického objektu. V naší práci, jak ukazuje kapitola 5.2.3, se obejdeme bez znalosti takovéhoto odhadu. V práci [11] naopak pokračují v této myšlence a definují funkci f jako válec v ose z, který je možno natáčet kolem osy x (úhel α) a y (úhel β) s pozicí středu (x_0, y_0, z_0) a s r_x jako poloměrem válce v ose x, r_y poloměrem v ose y — jedná se tedy přesněji o eliptický válec. Jeho analytické vyjádření je

$$\begin{aligned} f(x,y,z) = \; & \tfrac{[(x-x_0)\cos(\beta)+(y-y_0)\sin(\alpha)\sin(\beta)+(z\ -z_0)\cos(\alpha)\sin(\beta)]^2}{r_x^2} + \\ & + \tfrac{[(y-y_0)\cos(\alpha)-(z\ -z_0)\sin(\alpha)]^2}{r_y^2} - 1 \end{aligned} \tag{5.6}$$

Tyto tři rovnice mám definují kompletní model části tepny s následujícími parametry:

- střed válce (x_0, y_0, z_0)
- poloměry eliptického válce r_x a r_y
- otočení α (osa x) a β (osa y)
- rozmazání vstupních dat modelující PSF σ
- denzita tepny V
- denzita pozadí b

Takový model má deset parametrů. Důvod zanedbání rotace kolem osy z není v uvedené práci popsán. Důležité je, které parametry jsou ukryty ve funkci *distance*, jsou to pouze ty parametry, jež definují část modelu porovnávanou s reálnými daty. Toho využijeme při optimalizaci modelu.

Nejdříve definujeme měrnou funkci. Budeme klasicky minimalizovat součet čtverců rozdílu mezi modelem (*model*) a naměřenými daty (*image*).

$$\chi^2(\mathbf{a}) = \sum_{x \in \Omega(\mathbf{a})} \left(model(x, \mathbf{a}) - image(x)\right)^2, \tag{5.7}$$

kde $\mathbf{a}$ je vektor všech parametrů modelu. Množina Ω obsahuje všechny body, v nichž se měří rozdíl mezi modelem a naměřenými daty. Je tedy vhodné, aby tato

množina obsahovala pouze ty body, které jsou součástí definovaného válce nebo jsou v jeho těsné blízkosti. Z tohoto důvodu má také množina Ω parametr **a**.

Funkci χ^2 z rovnice 5.7 lze minimalizovat pomocí některého z nelineárních optimalizačních algoritmů popsaných v následující kapitole.

5.2.2 Optimalizační metody

Mezi nejpoužívanější metody, které minimalizují nelineární model, patří gradientní metoda (gradient descent), Gauss-Newtonova metoda a Levenberg–Marquardtova metoda. Gragientní metoda je velmi populární pro svou jednoduchost a snadnou implementaci. Bohužel ze své podstaty je velmi pomalá. Pokud žádáme rychlou metodu, nezbývá než použít Levenberg–Marquardtovu metodu. Ta optimálně kombinuje gradientní metodu a Gauss-Newtonovu metodu. V této podkapitole popíšeme odvození všech tří metod.

Gradientní metoda

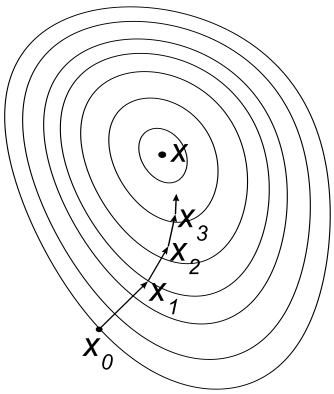
Metoda hledá lokální minima funkce $F(x)$, jak je znázorněno na obrázku 5.5. Začínáme v bodě x_0 a vydáváme se proti směru gradientu (tam, kde funkce nejrychleji klesá) podle vzorce:

$$x_{n+1} = x_n - \gamma \nabla F(x_n),\ n \geq 0 \tag{5.8}$$

Pokud je reálná funkce F v okolí bodu x_n lipschitzovsky spojitá, pak pro dostatečně malá γ větší než nula platí $F(n+1) \leq F(n)$. V praxi je klíčovým krokem volba konstanty (délky kroku) γ. Ta musí být dostatečně malá, aby metoda konvergovala, na druhou stranu čím menší je hodnota γ, tím více iterací musí algoritmus udělat, než splní podmínku ukončení. Taková podmínka může vypadat například jako $||\nabla F|| < \varepsilon$, kde ε je parametr algoritmu. Od této základní metody je odvozeno mnoho algoritmů, které se liší pouze výpočtem délky kroku v každé iteraci.

Gauss-Newtonova metoda

Gauss-Newtonova metoda minimalizuje součet čtverců měrné funkce bez požadování jejích druhých derivací, což je jedna z jejích největších výhod. Na tuto metodu může být nahlíženo jako na modifikaci klasické Newtonovy metody hledající minima funkce. Z toho také vychází následné odvození.



Obrázek 5.5: Postup gradientní metody na funkci F (převzato z Wikipedie).

Nechť dvojice $(\mathbf{t}_i, y_i)$, $i = \{1, .., m\}$ jsou naměřená data, kde $\mathbf{t}_i \in \mathbb{R}^k$ jsou body, ve kterých jsou naměřeny hodnoty y_i. Dále máme matematický model $M : \mathbb{R}^{k+n} \to \mathbb{R}$, který je závislý na n volných parametrech $x_1, x_2, .., x_n$ a pro který požadujeme, aby $M(\mathbf{t}_i, \mathbf{x}) \approx y_i$, pro $i = \{1, .., m\}$ a $\mathbf{x} = (x_1, x_2, .., x_n)$. Chceme, aby hodnoty $r_i(\mathbf{x}) = M(\mathbf{t}_i, \mathbf{x}) - y_i$ byly v absolutní hodnotě co nejmenší. Měrnou funkci definujeme jako $f(x) = \sum_{i=1}^{m} r_i(\mathbf{x})^2 = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$.

Nyní vyjádříme gradient g měrné funkce f a aproximujeme její Hessian G. Gradient g lze rozepsat jako $g(\mathbf{x}) = 2\sum_{i=1}^{m} \nabla r_i(\mathbf{x}) \cdot r_i(\mathbf{x}) = 2J(\mathbf{x})^T \mathbf{r}(\mathbf{x})$, kde $J(\mathbf{x}) \in \mathbb{R}^{n \times m}$ je *Jakobiho matice* (i-tý řádek matice J($\mathbf{x}$) je gradient r_i v bodě $\mathbf{x}$). Hessian je roven $G(\mathbf{x}) = 2\sum_{i=1}^{m} \nabla r_i(\mathbf{x}) \cdot \nabla r_i(\mathbf{x})^T + 2\sum_{i=1}^{m} \nabla^2 r_i(\mathbf{x}) \cdot r_i(\mathbf{x}) \approx 2\sum_{i=1}^{m} \nabla r_i(\mathbf{x}) \cdot \nabla r_i(\mathbf{x})^T = 2J(\mathbf{x})^T J(\mathbf{x})$.

Pomocí vzorce z klasické Newtonovy metody pro výpočet směru posunu z bodu $x^{(k)}$ do $x^{(k+1)}$ a v předchozím odstavci vypočtených vzorců dostáváme

$$G^{(k)} \cdot \delta\mathbf{x}^{(k)} = -g^{(k)} \qquad \Rightarrow \qquad J^{(k)^T} \cdot J^{(k)} \cdot \delta\mathbf{x}^{(k)} = -J^{(k)^T} \cdot r^{(k)}, \tag{5.9}$$

což je tzv. *normální rovnice* (obyčejná soustava lineárních rovnic). Výsledný vektor $\delta\mathbf{x}^{(k)}$ přičteme k $\mathbf{x}^{(k)}$ a dostáváme $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}$ — nový vektor optimalizovaných parametrů.

Levenberg–Marquardtova metoda

Tato metoda řešení úlohy nejmenších čtverců byla navržena Levenbergem již roku 1944 v práci [39]. Jedná se o metodu s omezeným krokem (restricted step method).

To znamená, že se hledá krok metody, tedy vektor $\delta\mathbf{x}^{(k)}$, na omezené oblasti, ve které je kvadratická aproximace minimalizované funkce dostatečně přesná, což je v okolí bodu $x^{(k)}$ (trust region). Řešení minimalizačního problému pro tuto kvadratickou aproximaci se určí z *rozšířené normální rovnice*

$$\left(J^{(k)^T} \cdot J^{(k)} + \nu I\right) \cdot \delta\mathbf{x}^{(k)} = -J^{(k)^T} \cdot r^{(k)} \tag{5.10}$$

pro vhodné ν (damping parametr), kde I je jednotková matice.

Pro výpočet damping parametru během iterace existuje mnoho postupů, které lze najít například v knize [42]. Obecný postup říká, že pokud změna vektoru parametrů $\mathbf{x}$ o $\delta\mathbf{x}$, kde $\delta\mathbf{x}$ bylo vypočteno z *rozšířené normální rovnice* 5.10, vede k redukci chyby $\mathbf{r}(\mathbf{x})$, je tato změna akceptována a proces pokračuje s klesajícím damping parametrem ν. V opačném případě parametr ν roste, dokud není pomocí *rozšířené normální rovnice* nalezena taková změna vektoru parametrů $\delta\mathbf{x}$, pro kterou chyba $\mathbf{r}(\mathbf{x})$ klesá. Parametr ν se adaptivně přizpůsobuje dosavadnímu průběhu algoritmu. Je-li $\nu \to 0$, výpočet přechází v *Gauss-Newtonovu* metodu, zatímco pro $\nu \to \infty$ definuje $\delta\mathbf{x}^{(k)}$ směr největšího spádu jako v *gradientní metodě*. Algoritmus ukončí hledání optimálních parametrů, pokud je splněna některá z následujících podmínek:

- velikost gradientu $\mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})$ poklesne pod práh ϵ_1 (dalšími iteracemi se nedosahuje výrazného zlepšení)
- relativní změna velikosti $\delta\mathbf{x}$ poklesne pod práh ϵ_2 (je dosaženo správných hodnot)
- chyba $\mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})$ poklesne pod práh ϵ_3 (je dosaženo správných výsledků)
- je překročen počet maximálního počtu iterací

Více o tomto algoritmu a jeho vylepšeních v článcích [40, 41, 43] nebo v knize [42].

5.2.3 Optimalizace

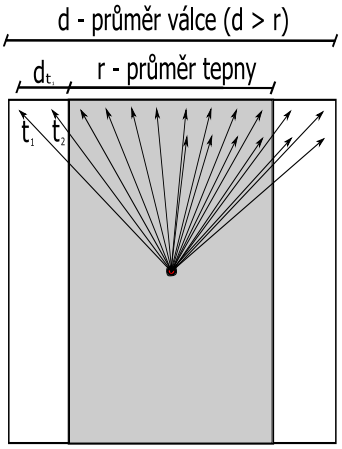
Po odbočce k optimalizačním algoritmům se v této podkapitole vrátíme zpět k minimalizaci měrné funkce

$$\chi^2(\mathbf{a}) = \sum_{x \in \Omega(\mathbf{a})} \left(model(x, \mathbf{a}) - image(x)\right)^2 . \tag{5.11}$$

Budeme se zabývat optimalizací této funkce z praktického hlediska, včetně několika implementačních urychlení optimalizačního algoritmu.

Jelikož počet bodů, ve kterých se model lícuje k obrázku, je konečný, nepotřebujeme mít při optimalizaci přesné analytické vyjádření našeho modelu, tak jak ho popisují rovnice 5.3, 5.4 a 5.6, a jak předpokládá práce [11]. To znamená, že v našem případě není třeba odhadovat funkci *distance* pomocí vzorce 5.4 nebo vyčíslovat geometrickou funkci (rovnice 5.6) v každé iteraci, a to dokonce ani při změně tvaru válce.

Na celý proces optimalizace budeme nahlížet jako na registraci dat a geometrického objektu, v našem případě válce. Následuje popis postupu při takovéto registraci. V prvním kroku nastavíme počáteční velikost válce, která definuje pouze v kolika bodech se bude počítat rozdíl naměřených dat a modelu. V tomto válci najdeme seznam vektorů T směřujících z centrálního bodu válce do všech ostatních bodů ležících ve válci. Protože je válec v základní poloze, lze jednoduše spočítat vzdálenost koncového bodu každého vektoru ze seznamu T od stěny válce, který má průměr pouze r, jak je to znázorněno na obrázku 5.6. Tím vznikne seznam



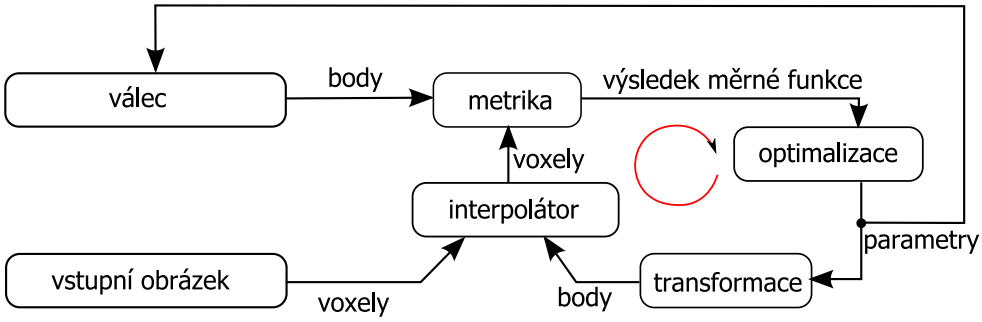
Obrázek 5.6: Schéma výpočtu vzdálenosti bodů od stěny tepny(šedá plocha je tepna).

D vzdáleností bodů od stěny tepny. Oba seznamy T i D jsme vypočetli předem, abychom nemuseli v každé iteraci počítat rovnice 5.4 a 5.6.

Nyní do každého bodu vstupní cesty umístíme vektory z množiny T (obrázek 5.10). V takto vzniklých bodech vypočteme hodnoty modelu. K tomu potřebujeme znát pouze vzdálenost bodu od stěny tepny, kterou máme uloženou v seznamu D, a

optimalizované parametry modelu. Výslednou hodnotu odečteme od interpolované hodnoty obrázku v tomto bodě. Získaný rozdíl uložíme do vektoru, který vrátíme optimalizačnímu Levenberg-Marquardtovu algoritmu.

Patří-li mezi optimalizované parametry pozice středu nebo úhel natočení válce, nesmíme zapomenout vektory ze seznamu T před jejich použitím adekvátně transformovat. Pokud je mezi parametry například poloměr válce, je třeba transformovat také hodnoty v seznamu D. Na obrázku 5.7 je znázorněn proces iterací v jednom bodě cesty, jehož výsledkem jsou optimální parametry modelu v tomto bodě. Tento postup je třeba zopakovat pro všechny body cesty.



Obrázek 5.7: Registrace modelu a vstupního obrázku (obrázek převzat z práce [4]).

5.2.4 Omezení parametrů modelu

Rychlost *Levenberg-Marquardtova* algoritmu, který byl vybrán pro účel registrace modelu a dat, významně ovlivňuje počet optimalizovaných parametrů. Proto se pokusíme jejich počet redukovat.

Zvážíme, jaké informace máme k dispozici před samotnou optimalizací — jsou to medicínské znalosti o segmentovaných tepnách, matematický model, který chceme optimalizovat, a cesta tepnou, kterou využíváme pro snadnou paralelizovatelnost procesu optimalizace.

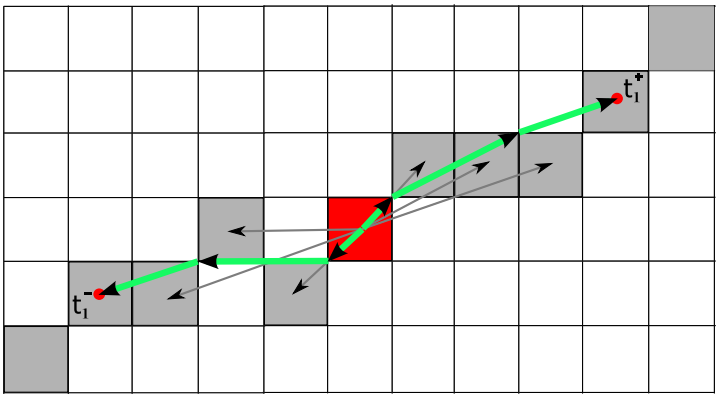
Zkoumáním medicínských dat můžeme dojít k závěru, že krční tepny mají vždy téměř kruhový průřez, a to i v případě, že obsahují kalcifikovaný plát či jsou zúžené. Z toho můžeme usoudit, že aproximace tepny pomocí válce s kruhovým průřezem je dostatečně přesná. V programu napsaném v rámci této práce lze měnit model a jeho parametry velmi snadno. Praktické testy těchto dvou modelů (kruhový a eliptický válec) předpoklad potvrdily — pro oba modely přinesly prakticky totožné výsledky. Touto úvahou podpořenou několika testy jsme se zbavili dvou parametrů modelu

— otočení eliptického válce kolem osy z a dvou poloměrů eliptického válce, které nahradil jediný poloměr shodný v ose x i y.

Cestu vypočtenou v první fázi metody využijeme k redukci parametrů α a β. Protože jsme při počítání ohodnocení hran použili vyhlazovací faktory, má výsledná cesta tepnou "rozumný" průběh. Toho můžeme nyní využít a vypočítat z jejího průběhu tečný vektor $\mathbf{t}$. Cestu τ budeme parametrizovat parametrem l, $l \in [0, M-1]$, kde M je počet bodů cesty, mezi kterými se lineárně interpoluje. Výpočet provedeme podle vzorce

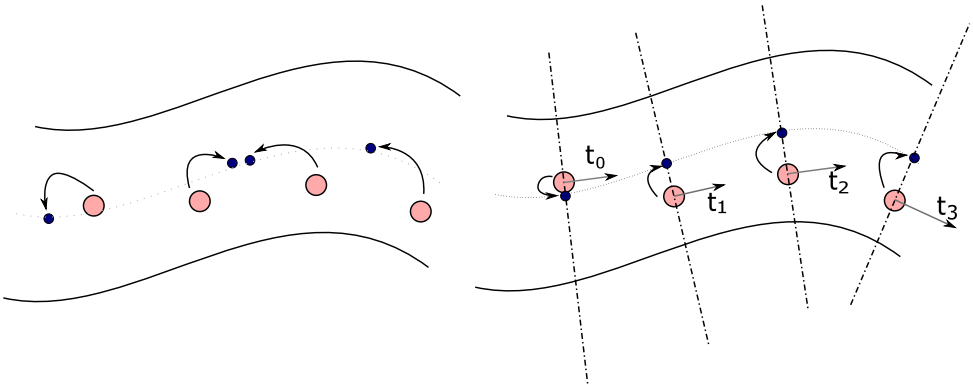
$$\begin{aligned} &\mathbf{t}_l = t_l^+ - t_l^-, \quad \text{kde} \\ &t_l^+ = \tau(l) + \textstyle\sum_{i=1}^{2K} \left[\left(-\left|\frac{i}{K} - 1\right| + 1\right)(\tau(l+i) - \tau(l))\right] \text{ a} \\ &t_l^- = \tau(l) + \textstyle\sum_{i=1}^{2K} \left[\left(-\left|\frac{i}{K} - 1\right| + 1\right)(\tau(l-i) - \tau(l))\right], \end{aligned} \tag{5.12}$$

konstanta K značí, jak daleko od pozice l leží bod s největší váhou. Váhy bodů se zvyšují lineárně, po dosažení bodu s největší váhou se opět lineárně snižují. Více objasní obrázek 5.8. Drobné potíže má tento postup pouze na koncích cesty, kdy takto vypočtený tečný vektor může být nepřesný. Stěžejní je správná volba konstanty K.



Obrázek 5.8: Výpočet tečného vektoru $(t_l^+ - t_l^-)$ cesty v červeném bodě pro konstantu $K = 2$.

Znalost tečného vektoru cesty nám také poslouží k řešení potíží při optimalizaci pozice středu válce. Při testování výsledných středů tepny se ukázalo, že jsou sice správně umístěny uprostřed tepny, ale jejich pozice není na nejbližším takovém místě, jak ukazuje obrázek 5.9 vlevo. Řešení, znázorněné na obrázku 5.9 vpravo, je hledat pozici středu tepny v rovině kolmé na cestu v právě zpracovávaném bodě. Protože budeme hledat středový bod v rovině, stačí k určení jeho pozice pouze dva parametry. Tím jsme opět snížili počet parametrů o jeden.

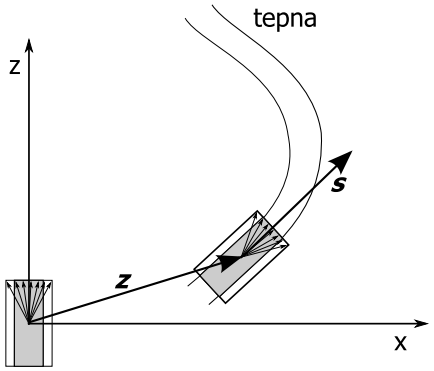


Obrázek 5.9: Příklad chybné optimalizace pozice středu tepny na obrázku vlevo. Na obrázku vpravo je příklad správného určení středu tepny.

Nyní se pokusíme zaměřit na zvolený model a jeho analytické řešení. Přepíšeme proto měrnou funkci χ^2 do diskrétního tvaru:

$$\chi^2 = \sum_{\mathbf{v} \in D} \left[image(\mathbf{v}, \mathbf{z}, \mathbf{s}) - b - V \left(\mathrm{erfc}(\mathbf{v}, r) \right) \right]^2 \tag{5.13}$$

Pro lepší vysvětlení některých parametrů rovnice 5.13 následuje obrázek 5.10.



Obrázek 5.10: Průřez optimalizací.

V rovnici 5.13 si můžeme všimnout, že parametry b a V nejsou členy žádné složitě derivovatelné funkce, což vede k myšlence, že je možné tyto parametry vypočítat rovnou. Hledáme tedy klasicky lokální minimum funkce, t.j. funkci derivujeme podle b a podle V a výsledky položíme rovné nule.

$$\frac{\partial \chi^2}{\partial b} = \sum_{\mathbf{v} \in D} -2 \left[image(\mathbf{v}, \mathbf{z}, \mathbf{s}) - b - V \ \mathrm{erfc}(\mathbf{v}, r) \right] = 0 \tag{5.14}$$

$$\sum_{\mathbf{v}\in D} image(\mathbf{v},\mathbf{z},\mathbf{s}) = bN + V \sum_{\mathbf{v}\in D} \mathrm{erfc}(\mathbf{v},r) \tag{5.15}$$

$$\frac{\partial \chi^2}{\partial V} = \sum_{\mathbf{v}\in D} \mathrm{erfc}(\mathbf{v},r)\left[image(\mathbf{v},\mathbf{z},\mathbf{s}) - b - V\ \mathrm{erfc}(\mathbf{v},r)\right] = 0 \tag{5.16}$$

$$\sum_{\mathbf{v}\in D} \mathrm{erfc}(\mathbf{v},r) image(\mathbf{v},\mathbf{z},\mathbf{s}) = b\sum_{\mathbf{v}\in D} \mathrm{erfc}(\mathbf{v},r) + V \sum_{\mathbf{v}\in D} \mathrm{erfc}(\mathbf{v},r)^2 \tag{5.17}$$

N je počet vektorů v množině D. Rovnice 5.15 a 5.17 jsou dvě rovnice o dvou neznámých. Z nich vyjádříme parametry b a V. Nejdříve ale zavedeme substituci

$$e = \mathrm{erfc}(\mathbf{v},r), \quad i = image(\mathbf{v},\mathbf{z},\mathbf{s}) \quad \text{a} \quad \textstyle\sum = \sum_{\mathbf{v}\in D}\text{, pak}$$

$$b = \frac{\sum e^2 \sum i - \sum e \sum e \cdot i}{N\sum e^2 - \sum e \sum e} \tag{5.18}$$

$$V\ = \frac{\sum e \sum i - N \sum e \cdot i}{\sum e \cdot \sum e - N \sum e^2} \tag{5.19}$$

Tímto jsme z optimalizace odstranili také parametry b a V.

5.3 Shrnutí

V této kapitole byla popsána metoda segmentace cév, kterou je možné rozdělit na dvě části. První část metody spočívá v nalezení posloupnosti na sebe navazujících bodů (cesty), lokalizovaných uvnitř nebo v těsné blízkosti tepny. Posloupnost těchto bodů začíná i končí uživatelem (lékařem) zadanými body. Pro nalezení cesty mezi těmito body použijeme upravený Dijkstrův algoritmus. Druhá část metody vychází ze znalosti cesty tepnou a pomocí optimalizace válcovitého modelu se snaží získat další informace o segmentované tepně. Výsledné parametry lze použít k následnému automatickému zhodnocení segmentované tepny.

Celý postup lze shrnout do několika bodů, které velmi dobře popisuje úvodní obrázek 5.1.

Tato práce využívá myšlenek klasických metod a inovativně je aplikuje na daný problém. Jde především o popsaný pyramidový přístup k metodě hledání cesty v objemových datech. Takto byla vylepšena již známá metoda *live-wire*, která je popsána v dílech v [25, 28]. Byly zde nově vyzkoušeny některé faktory, které různými způsoby ovlivňují ohodnocení hran v grafu a tedy i výslednou cestu.

Druhá část metody přejímá myšlenku parametrického válcovitého modelu z děl [11, 31] a pohlíží na optimalizační úlohu jako na registraci prostorového modelu a

Algoritmus 7 Postup segmentační metody popsané v této kapitole.

1. Získání počátečního a koncového bodu segmentované tepny a ostatních parametrů této metody. U mnoha parametrů lze použít stejné hodnoty pro různá vstupní data.
2. Načtení kompletních vstupních dat do paměti.
3. Zmenšení vstupních dat v několika měřítcích. Postupné vyhledání cesty od menších měřítek k větším (kapitola 5.1).
4. Rozdělení cesty na části podle počtu procesorů. Spuštění optimalizace parametrů válcovitého modelu (kapitola 5.2) — každou část zpracovává jiný procesor.
5. Závěrečné zkompletování parametrů modelu pro každý bod cesty.
6. Vygenerování výsledného obrázku metodou *Straightened CPR* (kapitola 2).

vstupních dat. Tím je možné mnoho výpočtů provést ještě před vlastní optimalizací. Nově jsou zde prezentovány postupy, jak omezit počet parametrů modelu a tím urychlit jeho výpočet. Zatímco v práci [11] prezentují výsledný čas konvergence parametrů modelu v jednom řezu 0,9 sekundy (není uveden HW), podařilo se nám dosáhnout řádového zlepšení - během testů bylo dosaženo výsledného času 0,008 sekundy (viz kapitola 6.4).

Hodnocení popsané metody včetně upozornění na několik slabých míst a možnosti jejich řešení poskytuje následující kapitola. Obsahuje také množství testů časové náročnosti metody a úspěšnosti její paralelizace.

Kapitola 6

Měření a výsledky

V klinické praxi se vyskytuje množství různě kvalitních dat, která se mohou stát vstupem naší metody. V této kapitole proto otestujeme, jaké výsledky má optimalizační algoritmus na reálných i syntetických datech. Syntetická data používáme pro ověření správné konvergence parametrů. Výsledné parametry modelu by měly být stejné jako parametry, které byly použity při generování syntetických dat. Jelikož jsme v metodě nepoužili žádný preprocessing, prezentujeme rovněž přesnost naší metody na vstupních datech s různým stupněm šumu. Velmi důležitá je také rychlost aplikace a možnosti její paralelizace, čemuž jsme přizpůsobili i strukturu celé metody. Několik měření na toto téma je v závěru kapitoly.

6.1 Popis použitých dat

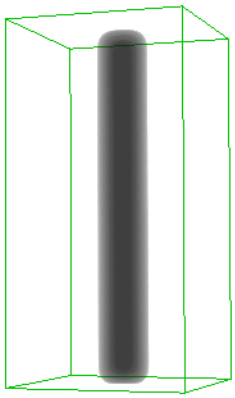
Jak je již zmíněno výše, použili jsme na testování data dvou typů. První typ jsou námi vygenerovaná syntetická data. Druhý typ jsou reálná data anonymních pacientů, která poskytl MUDr. Martin Horák (Radiologické oddělení, nemocnice Na Bulovce).

6.1.1 Syntetická data

K otestování naší optimalizační metody jsme použili tři obrázky, jež pojmenujeme jako *simple*, *noise* a *spiral*[1].

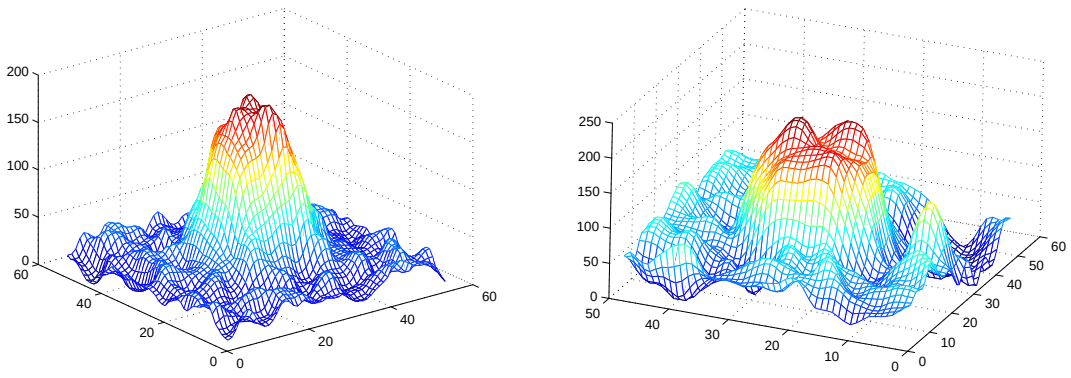
[1]Tento obrázek byl vytvořen Danem Muellerem na Queensland University of Technology

První obrázek *simple* představuje válec v základní poloze, na který byl aplikován Gaussův filtr — rozmazání obrázku představuje aproximaci vlastností (blurring) CT přístroje při pořizování reálných dat. Takto vygenerovaná data jsou na obrázku 6.1.



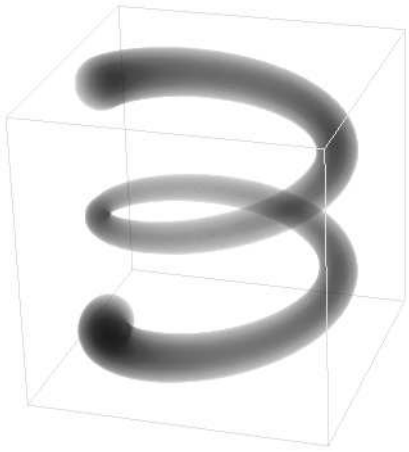
Obrázek 6.1: Vygenerovaná syntetická data *simple* zobrazená metodou MIP.

Pro vytvoření obrázku *noise* byl použit obrázek *simple*, ke kterému byl přidán trojrozměrný Gaussův šum s vhodnými parametry tak, aby výsledek co nejvíce odpovídal reálným datům. Na obrázku 6.2 je porovnání řezu námi vytvořenou umělou tepnou a řezu skutečnou krční tepnou.



Obrázek 6.2: Vlevo jsou vytvořená syntetická data, vpravo skutečný průřez krční tepnou (krkavicí).

Poslední syntetická data *spiral* jsou na obrázku 6.3. Je to vygenerovaná spirálovitá tepna s postupně se snižujícím průsvitem a poloměrem, od poloviny délky tepny se tyto veličiny začínají zpět zvětšovat až do svých počátečních hodnot.



Obrázek 6.3: Spirálovitá umělá data *spiral* určená k testování správně nalezené cesty.

6.1.2 Reálná data

Reálná data jsou pomocí CTA (kapitola 2) nasnímané oblasti krku. K dispozici máme snímky tří pacientů. Pojmenujeme tato data jako *pacient* 1, *pacient* 2 a *pacient* 3. Patologické nálezy u pacientů můžeme popsat následovně:

pacient 1 — výrazně vinutý průběh pravé vnitřní krkavice typu kinking a kalcifikace ve stěně obou vnitřních krkavic

pacient 2 — těsná stenóza odstupu pravé vnitřní krkavice

pacient 3 — drobná kalcifikace ve stěně společné krkavice vpravo a stenóza odstupu vnitřní krkavice vlevo

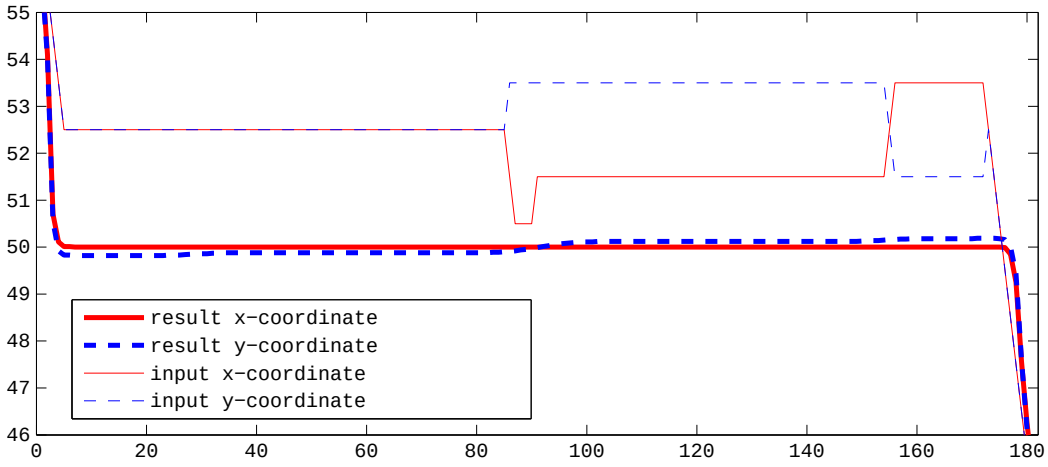
U CTA je zvykem pořizovat řezy o menší tloušťce než u klasického CT, kde tloušťka řezu je kolem 5 milimetrů. Důvodem je zobrazení malých cév, které by na klasickém CT nemusely být vidět, a lepší možnost trasovat cévu. Nevýhodou je větší šum v takto pořízených datech. Všechny obrázky z naší testovací množiny mají tloušťku řezu 1 milimetr a méně. Přiložená anonymizovaná data nejsou nijak upravena, pouze převedena z DICOM formátu do formátu MHD (Meta Header Data).

6.2 Výsledné parametry na syntetických datech

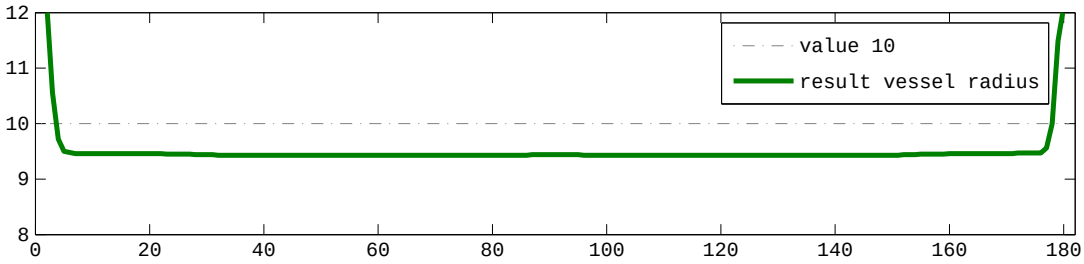
Syntetická data by měla do značné míry odpovídat našemu modelu. Z toho pro tato data plyne rychlá a přesná konvergence parametrů, jak je vidět na následujících grafech.

U všech parametrů můžeme pozorovat chybnou konvergenci na začátku i na konci cesty. To je dáno počátečním postavením válcovitého modelu, kdy jeho část nespadá do oblasti tepny. Díky tomu dochází v těchto částech cesty k velkým výchylkám hodnot parametrů a tudíž není možné relevantně počítat střední odchylku a jiné statistické veličiny vypovídající o přesnosti metody. Proto místo výpočtu statistických veličin prezentujeme grafy hodnot těchto parametrů. Na ose x je vždy odpovídající krok tepnou. Na ose y jsou hodnoty daného parametru v jednotlivých bodech cesty.

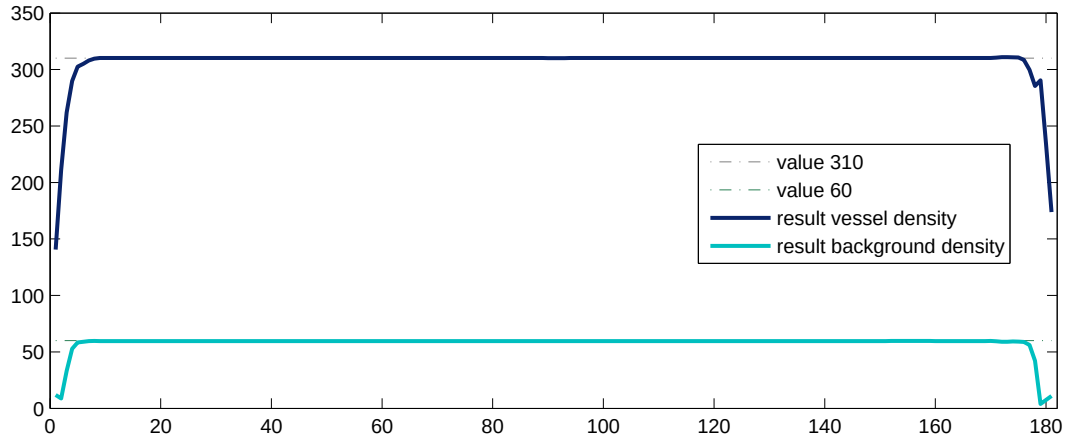
Výsledné parametry u dat *simple* jsou dle očekávání přesné. Parametry určující pozici středu tepny jsou vidět na grafu 6.4. Správný střed tepny je v každém bodě na pozici 50 pro obě souřadnice (x,y). Výsledné parametry jsou na grafu vyznačeny silnou červenou a modrou křivkou. Na dalším grafu 6.5 je poloměr tepny — ve všech bodech cesty je nepatrně nižší než správná hodnota 10, což je způsobeno simulací *PSF* pomocí konvoluce s diskrétním Gaussovým jádrem. Hodnoty denzity pozadí b a denzity tepny V nejsou parametrem optimalizačního algoritmu (Levenberg-Marquardt), ale jsou počítány přímo (jak je vysvětleno v kapitole 5.2.4). Z grafu 6.6 je patrno, že správná konvergence ostatních optimalizovaných parametrů zapříčiní správnou konvergenci těchto vypočtených parametrů, které se od správných hodnot (60 pro b a 310 pro V) neliší. Výsledný obrázek 6.8 generovaný metodou *Straightened CPR* je díky správné konvergenci všech parametrů a absenci šumu velmi pěkný.



Obrázek 6.4: Srovnání počáteční pozice (tenká křivka) bodu cesty a výsledné pozice (silnější křivka) bodu cesty, který zkonvergoval do středu tepny v průběhu celého zpracovávaného úseku.



Obrázek 6.5: Výsledný poloměr tepny v celém jejím průběhu. Hodnota 10 značí správný poloměr tepny.

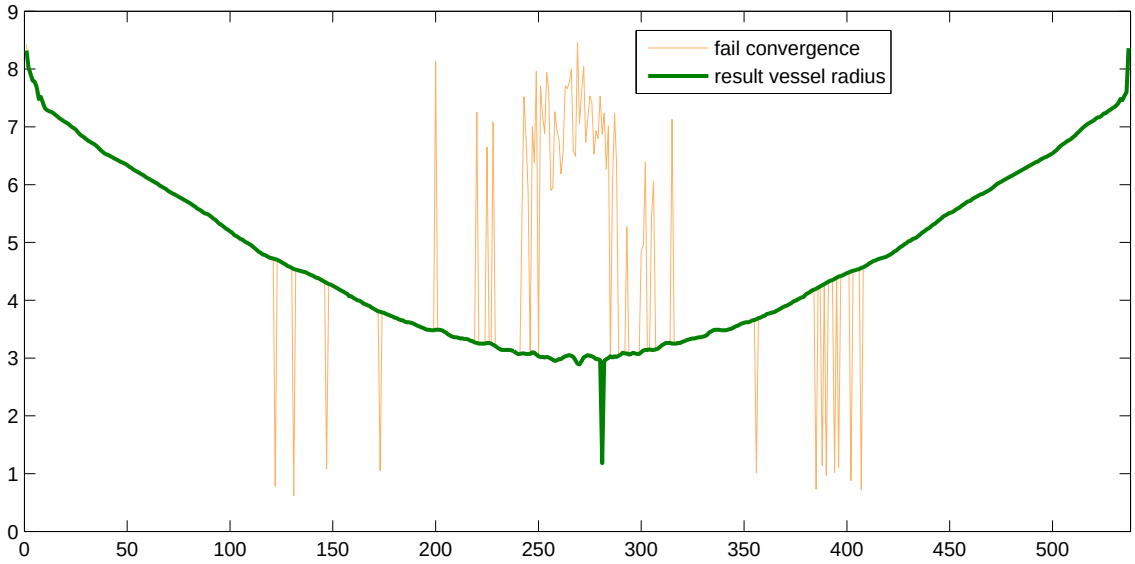


Obrázek 6.6: Parametry modelu, denzita pozadí b a denzita tepny V.

Obdobné měření jsme provedli pro data *spiral*, což má za cíl otestovat odolnost algoritmu na klesající poloměr a průsvit tepny. Některé výsledné grafy jsou v příloze D. Zde je uveden pouze graf hodnot poloměrů tepny podél nalezené cesty 6.7. Graf prezentuje správnou konvergenci poloměru tepny (silnější zelená křivka) a chybnou konvergenci (slabší křivka). K chybnému výpočtu poloměru, ale i jiných parametrů, dochází, pokud je počet bodů, ve kterých se počítá rozdíl modelu a vstupních dat, příliš malý — to je případ slabší křivky v grafu 6.7.

Na silnější výsledné křivce je vidět chybná konvergence modelu při kroku číslo 280. V řezu, ve kterém dochází k této chybě, je totiž hodnota bodu, kterým prochází počáteční cesta, rovna hodnotě $b + \frac{V}{2}$. V takovém bodě se může stát, že parametr odpovídající poloměru tepny "zkolabuje" do nuly. Téměř všechny body, ve kterých se pak měří rozdíl modelu a vstupních dat jsou právě v tomto bodě vstupního obrázku, přičemž jejich vzdálenost od stěny tepny je nula. Hodnota funkce $erfc$ v bodě 0 je $1/2$ a tedy hodnota modelu v těchto bodech je $b + \frac{V}{2}$, z čehož vyplývá, že rozdíl modelu a vstupních dat je téměř ve všech bodech, kde se měří, roven nule, přestože hodnoty parametru neodpovídají správných hodnotám. K této chybné konvergenci dochází především v případě, že prvotní cesta tepnou jde v těsné blízkosti její stěny, kde denzita bodů může být rovna hodnotě $b + \frac{V}{2}$. Proto jsme při hledání cesty tepnou, konkrétně při ohodnocení hran, zavedli faktor f_M, který preferuje cestu středem tepny. Vhodné je také zvolit minimální denzitu tepny $vessel_{min}$ (vstupní parametr naší metody) tak, aby platilo $vessel_{min} > b + \frac{V}{2}$.

Na obrázku 6.8 jsou výsledné obrázky vygenerované metodou *Straightened CPR* pro data *simple* a *spiral*.

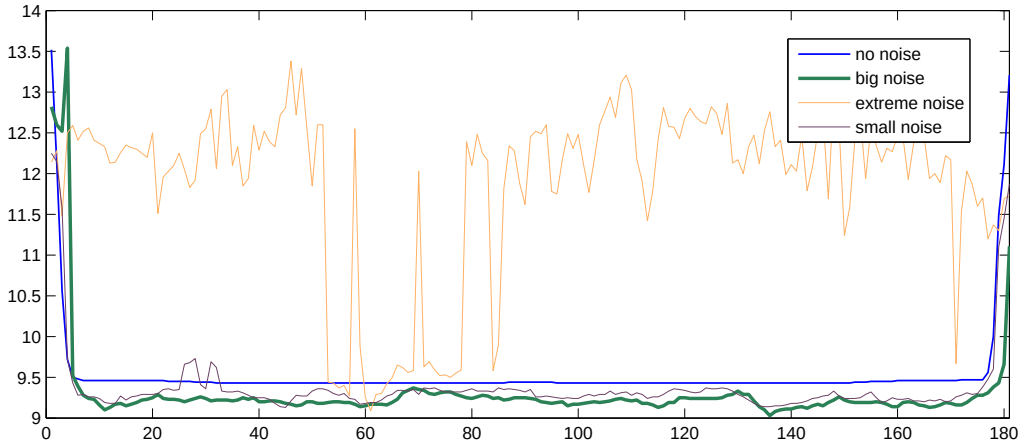


Obrázek 6.7: Výsledný poloměr tepny na datech *spiral* v celém jejím průběhu.

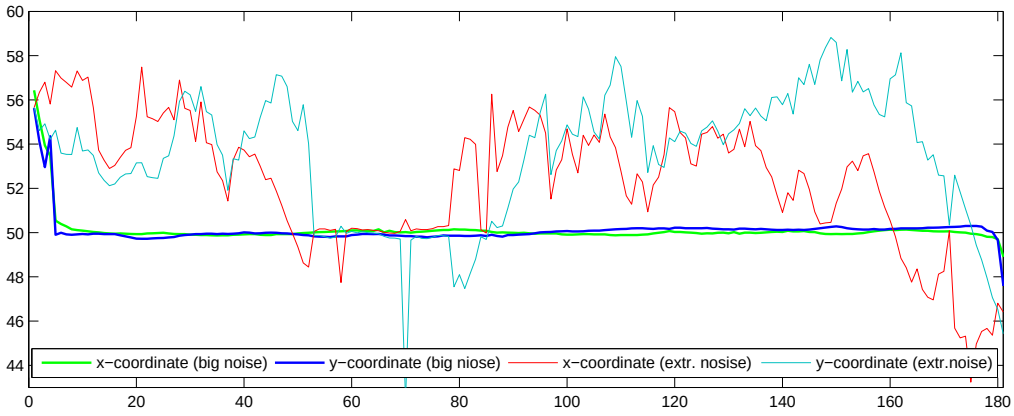
Obrázek 6.8: Výsledné obrázky vygenerované metodou *Straightened CPR* — **a)** vstupní data *simple*, **b)** data *spiral*.

Poslední test na syntetických datech hodnotí náchylnost správné konvergence parametrů modelu na úroveň šumu vstupního obrázku. Ke vstupnímu obrázku byl přidám Gaussův šum s různou amplitudou γ . Konkrétně pro šum *small noise* je amplituda $\gamma = 70$, pro *big noise* je $\gamma = 100$ a pro *extreme noise* je $\gamma = 130$. Na grafu 6.9 je konvergence poloměru tepny při různém zatížení vstupního obrázku

šumem. Následuje graf 6.10, kde je konvergence pozice středu tepny. Z grafu 6.9 je vidět, že pokud je γ příliš velká, metoda si již s takto nekvalitními daty nedokáže poradit. Zajímavé je, že metoda dlouhou dobu odolává rostoucímu šumu, pokud ten pak překročí jistou mez, parametry začnou konvergovat chybně.



Obrázek 6.9: Graf konvergence poloměru tepny při různém stupni šumu.



Obrázek 6.10: Graf konvergence pozice středu tepny při různém stupni šumu.

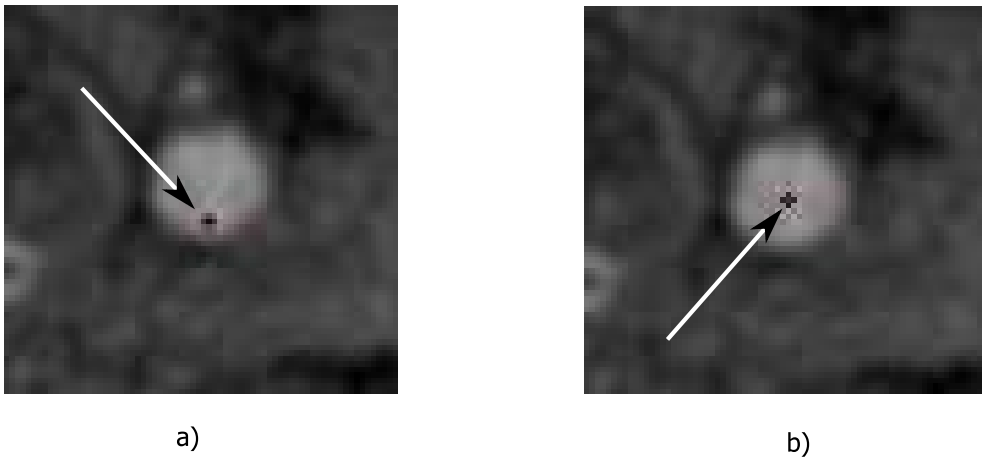
6.3 Výsledné parametry na reálných datech

Při testování metody na reálných datech se negativně projevuje jejich velký šum. Pokud ten překročí určitou mez, způsobí chybnou konvergenci většiny parametrů modelu, například poloměru nebo denzity tepny. Naopak pozice parametrického

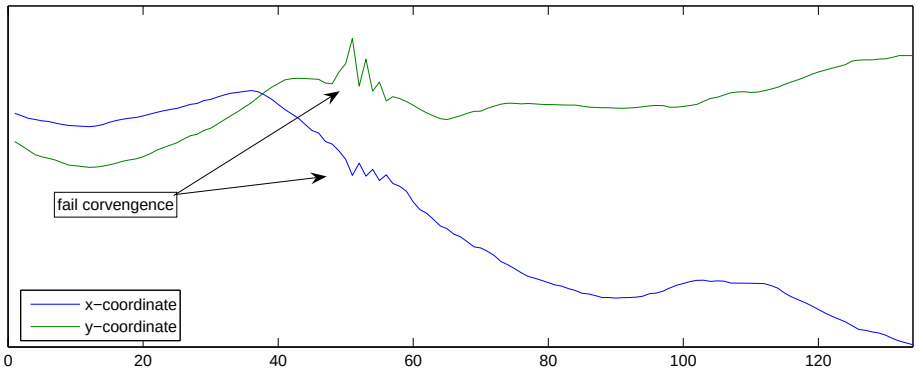
válce konverguje i při nekvalitních datech správně do středu tepny. Díky tomu filtr generující segmentovanou tepnu metodou *Straightened CPR*, který požaduje přesnou cestu středem tepny, dává uspokojivé výsledky. Na přiloženém DVD a obrázku 6.11 jsou ukázky reálných dat, na nichž je možné se přesvědčit o pozici optimalizované cesty. Téměř na všech těchto datech můžeme v bodech s hodnotou denzity rovnající se přibližně $b + \frac{V}{2}$ pozorovat chybnou konvergenci modelu. Důvod byl popsán již v kapitole 6.2, na reálných datech se jev ovšem projevuje ve větší míře. Na výsledných obrázcích je tento problém vidět jako řádky, které přesně nenavazují na předchozí.

Tento problém není možné jednoduše vyřešit penalizací optimalizačního algoritmu během iterací, pokud se parametry dostanou do chybných hodnot, například pokud hodnota poloměru se blíží nule. Optimalizační algoritmus by pak v takovém případě rovněž skončil chybně, pouze na jinou ukončující podmínku. Mezi možná řešení patří optimalizace středu tepny pomocí metody *Center Of Gravity* z kapitoly 3.5 těsně před optimalizací parametrů modelu. Vycházíme z toho, že denzita bodů je v blízkosti stěny tepny rovna hodnotě $b + \frac{V}{2}$. Dalším řešením je použít *simulované žíhání*, do tohoto optimalizačního algoritmu je přidán prvek náhodnosti, což má za následek, že algoritmus "neuvízne" v lokálním minimu měrné funkce, ale snaží se dohledat globální minimum funkce. Tato metoda je ovšem již ze svého principu velmi pomalá.

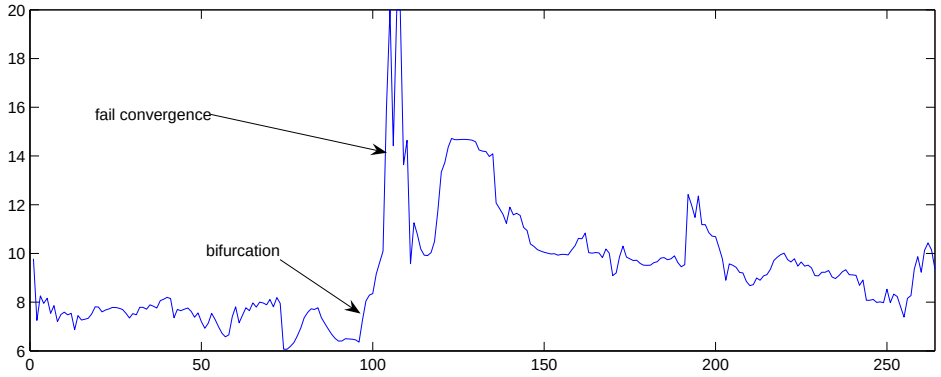
Následují grafy výsledných parametrů modelu v průběhu segmentované části. Na konci jsou prezentovány výsledné obrázky vytvořené metodou *Straightened CPR*.



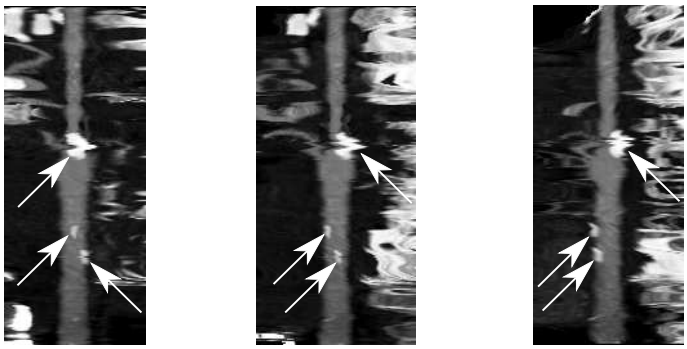
Obrázek 6.11: Optimalizace středu tepny na reálných datech — **a)** neoptimalizovaný bod cesty, **b)** optimalizovaný bod cesty (střed tepny).



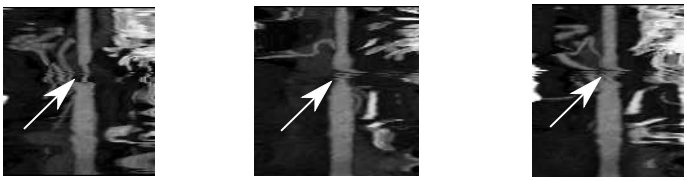
Obrázek 6.12: Graf souřadnic středů krkavice — na označených místech je vidět chybná konvergence válce do středu tepny.



Obrázek 6.13: Graf poloměru tepny (krkavice) na datech *pacient* 1 — na označených místech je vidět chybná konvergence a větvení tepny (cesta začíná ve vnitřní krkavici a jde proti směru toku krve do společné krkavice).



Obrázek 6.14: Výsledný obrázek generovaný pomocí Straightened CPR z dat *pacient* 1 v různém úhlu pohledu, konkrétně $0°, 30°, 90°$. Bílou šipkou jsou označeny kalcifikace.



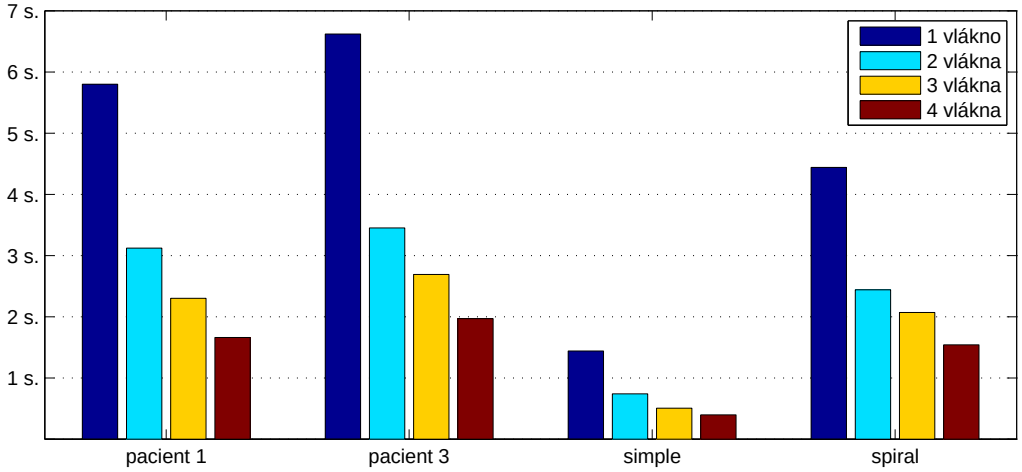
Obrázek 6.15: Výsledný obrázek generovaný pomocí Straightened CPR z dat *pacient* 2 v různém úhlu pohledu, konkrétně $0°, 30°, 90°$. Bílou šipkou je označena stenóza, zároveň je v tomto místě chybná konvergence parametrů (řádky na sebe nenavazují).

6.4 Měření rychlosti metody

Měření rychlosti byla prováděna na počítači s čtyřjádrovým procesorem Intel Core 2 Quad o frekvenci 2,4GHz a 8GB paměti RAM. Jak je vidět z grafu a tabulky, je metoda velmi rychlá. Výborné výsledky má také test paralelního běhu optimalizace cesty — graf 6.16.

Data	Velikost (MB)	Cesta (s)	Délka cesty	Optimalizace (s)	Celkový čas (s)
pacient 1	132	2,18	264	1,64	4,44
pacient 2	124	2,09	281	1,79	4,53
pacient 3	143	2,51	282	1,93	5,02
simple	8	0,23	181	0,39	0,65
spiral	9	0,61	537	1,54	2,17

Tabulka 6.1: Tabulka časů jednotlivých částí metody. Časy byly naměřeny na paralelní verzi metody.



Obrázek 6.16: Časové výsledky paralelizace algoritmu.

Kapitola 7

Závěr

Tato práce prezentuje ucelený přehled metod segmentace cév v objemových lékařských datech, konkrétně v datech pořízených počítačovým tomografem (CT). Z velkého množství různých metod popsaných v kapitole 3 vybírá metodu z kategorie parametrických válcovitých modelů, která je v rámci práce implementována a důkladně testována.

Od začátku práce byl kladen velký důraz na rychlost a paralelizovatelnost metody, což je dnes jeden z klíčových prvků při zvyšování rychlosti algoritmů. Z tohoto důvodu byla vybraná metoda zkombinována s metodou vyhledání cesty cévou.

Vyhledávání cesty cévou je realizováno upraveným Dijkstrovým algoritmem, kdy na objemová data nahlížíme jako na graf. Z důvodu optimalizace rychlosti byla z algoritmu odstraněna pomalá prioritní fronta, kterou nahradila speciální datová struktura. Ta využívá normovaného a diskrétního ohodnocení hran a řádově urychluje vyhledání cesty. Protože vhodné ohodnocení hran grafu je jedna z mála možností, jak omezit počet navštívených vrcholů grafu a tím urychlit hledání cesty, bylo otestováno několik faktorů, které počítají ohodnocení hran. Jmenujme například faktor preferující střed cévy nebo faktor odhadující délku cesty. Výsledná hodnota hrany je pak lineární kombinací těchto faktorů. Za zajímavé považujeme další omezení počtu zpracovávaných vrcholů využitím pyramidového přístupu k hledání cesty, což přináší významné urychlení zejména při zpracování velkého objemu dat.

Optimalizace parametrů válcovitého modelu ve vztahu k reálným datům je druhou částí metody. Je zde získána množina parametrů cévy popisujících její průběh. Tyto parametry lze použít například pro automatické zhodnocení cévy či

její rekonstrukci. Model, který byl použit, lze zapsat vzorcem takto:

$$model(x) = b + V \cdot \operatorname{erfc}\left(\frac{distance(x)}{\sigma}\right), \tag{7.1}$$

kde $distance(x)$ je vzdálenost bodu x od stěny tepny, V je denzita cévy a b denzita pozadí, parametr σ určuje stupeň rozmazání vstupních dat. Parametry modelu 7.1 jsou optimalizovány pomocí rychlého Levenberg-Marquardtova algoritmu, který minimalizuje měrnou funkci χ^2,

$$\chi^2(\mathbf{a}) = \sum_{x \in \Omega(\mathbf{a})} \left(model(x, \mathbf{a}) - image(x)\right)^2, \tag{7.2}$$

zde $\mathbf{a}$ je vektor parametrů, *image* jsou vstupní data a množina Ω obsahuje body, ve kterých se měří rozdíl modelu a dat. Rychlost optimalizace závisí jak na vlastnostech měrné funkce χ^2, tak na počtu parametrů. Kapitola 5.2.4 popisuje omezení jejich počtu — například parametry b a V není třeba optimalizovat, ale pomocí derivace χ^2 a vzniklé soustavy dvou rovnic o dvou neznámých výsledné hodnoty rovnou vypočíst.

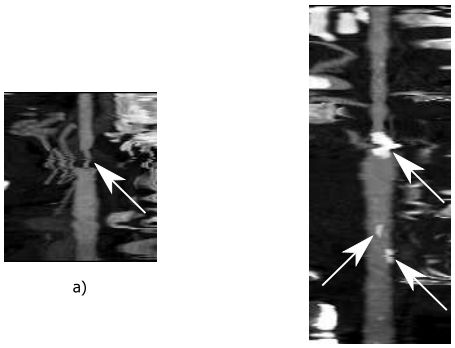
Testy této metody přinesly zajímavé výsledky. Zatímco první fáze metody, tj. hledání cesty, se ukázala jako velmi robustní — na všech datech z testovací množiny našla správně cestu, druhá fáze metody, tj. optimalizace parametrů modelu, se ukázala býti náchylnou na nekvalitní vstupní data zatížená šumem. Podrobili jsme tedy optimalizaci parametrů modelu v kapitole 5 širšímu testování. Jeho výsledky je možno shrnout asi takto:

Metoda velmi dobře pracuje na kvalitních datech, která jsou jen málo nebo středně zatížena šumem. Všechny parametry modelu pak velmi přesně konvergují k správným hodnotám. Pokud ovšem šum překročí kritickou míru, konvergence některých parametrů se skokově výrazně zhorší, což platí například pro parametr symbolizující průměr tepny. Jiné parametry, například pozice středu tepny, nejsou mírou šumu výrazněji ovlivněny.

V praxi je bohužel velké procento nasnímaných dat zatíženo šumem, který sice lze vhodným nastavením CT přístroje redukovat, v běžné lékařské praxi se to ale neděje. Proto výsledné obrázky vytvořené touto metodou nejsou zatím zcela přesvědčivé.

Pro zobrazování výsledků byla vybrána metoda *Straightened CPR*, která zobrazí celou segmentovanou tepnu v dvojrozměrném obrázku. Výsledky vytvořené pomocí této metody můžeme vidět na následujícím obrázku 7.1.

Práce by se měla ubírat směrem k vylepšení použité metody vizualizace tepen — *Straightened CPR* nebo k implementaci *Stretched CPR*, která ve výsledném obrázku poskytuje lepší prostorovou představu o pozici tepny. Některá další vylepšení byla popsána během testování v kapitole 6.3.



Obrázek 7.1: Výsledné obrázky získané pomocí postupu popsaného v této práci. Na obrázku vlevo je ukázka zúžení tepny (krkavice), vpravo na obrázku je bílou šipkou vyznačeno několik kalcifikovaných plátů ve stěně tepny. Obě vstupní data jsou velmi zatížena šumem.

Dodatek A

The Insight Toolkit — ITK

ITK je knihovna zaměřená na segmentaci a registraci medicínských dat. Vznikla v roce 1999 jako projekt s otevřeným kódem sponzorovaný Národní lékařskou knihovnou USA. Dnes je projekt sponzorovaný mnoha organizacemi včetně několika komerčních, z nichž hlavní je firma Kitware, Inc. ITK je pod otevřenou licencí a je možné ji bezplatně používat dokonce i pro komerční účely.

Velkou výhodou knihovny je infrastruktura, která je kolem ní postavena a bez níž by nebyla správa tak velkého open-source projektu možná. Jde například o automatickou kompilaci, automatické testování nového kódu, reportování chyb nebo volně přístupný publikační systém. Jelikož je knihovna open-source, vývojáři z celého světa ji mohou rozšiřovat. Kvalita kódu je ovšem velmi pečlivě střežena. Před tím, než kdokoli přispěje svým kódem, musí tento kód projít mnoha automatickými testy. Pro tento účel byl firmou Kitware vyvinut testovací systém *ITK Testing Dashboard*. Dále kód projde oponenturou ostatních vývojářů, kteří se vyjadřují jak ke zvolenému algoritmu a jeho výstižnému popisu v přiloženém dokumentu, tak například k úpravě a správnosti kódu. K pohodlnému publikování takového kódu spolu s teorií a možností komentovat a hodnotit práci slouží *The Insight Journal*. To, co z ITK dělá skutečně použitelnou knihovnu, je volně dostupná dokumentace, jde především o knihu *The ITK Software Guide* [4] a o dokumentaci automaticky vygenerovanou z kódu. Na velmi dobré úrovní je také podpora uživatelů knihovny, pro něž existují dvě konference, kde mohou klást své otázky a vždy dostanou několik odpovědí.

Navíc ITK je hlavní součástí mnohem rozsáhlejšího projektu jménem NAMIC(The National Alliance for Medical Image Computing). Součástí tohoto projektu jsou velmi zajímavé aplikace, které lze použít spolu s ITK, proto zde některé

uvádím:

VTK — Visualization Toolkit, knihovna sloužící k zobrazování a interakci se zobrazovanými daty. Často se používá jako zobrazovací jednotka dat zpracovaných pomocí ITK.

3D Slicer — program sloužící pro zobrazení medicínských dat a následnou práci s nimi. Program používá obě knihovny — ITK i VTK.

KWKidgets — knihovna poskytující uživatelské rozhraní založené na Tcl/Tk s C++ API. Vhodná k tvorbě GUI pro aplikace využívající spojení ITK a VTK.

CMake — aplikace nezávislá na operačním systému sloužící k překladu knihoven ze zdrojových souborů. Všechny uvedené projekty mají překlad řízený pomocí CMake. Příkazy řídící průběh kompilace jsou uvedeny v souboru `CMakeLists.txt`, jenž má podobnou strukturu jako soubor `Makefile`.

DART — testovací server, který používá další nástroje jako je CDash, CTest, CPack a CMake. Na tomto serveru je postaven "Dashboard" — systém pro testování všech zde uvedených projektů. Jedním z nich je již uvedený *ITK Testing Dashboard*.

Teem Libraries and Command Line Tools — nástroje pro reprezentaci, zpracování a zobrazení vědeckých dat. Jen pomocí příkazové řádky lze číst, filtrovat, segmentovat či jinak zpracovávat různá data. V kombinaci například s Bournovým shellem je to opravdu mocný nástroj.

XNAT — The Extensible Neuroimaging Archive Toolkit, podpora pro bezpečné úložiště a správu dat velkých objemů včetně například webového rozhraní pro přístup k datům.

Batchmake — nástroj pro jednoduché dávkové zpracování velmi obsáhlých dat. Umí zpracovat data i na distribuovaných systémech a analyzovat jejich průběh jak v uživatelsky přívětivé aplikaci, tak pomocí webové aplikace. Celý průběh je řízen pomocí skriptu, který má velmi podobnou syntax jako CMake.

A.1 Koncepce knihovny

Pro implementaci ITK byl použit programovací jazyk C++. ITK využívá jeho vlastností, jako je generické programování (šablony), jen málo tříd ponechává nešablonovaných. Podpora více vláken je dnes již samozřejmostí pro každý úspěšný systém. Další žádanou vlastností je nezávislost na platformě, autoři uvádějí až 42 podporovaných platforem. Při návrhu knihovny byla použita celá řada návrhových vzorů, jako například *factory* pro nové instance objektů nebo *command/observer* pro generování událostí. Paměťový model je založen na chytrých ukazatelích (smart-pointer), ty obsahují čítač, který počítá počet referencí na objekt, na nějž ukazuje. Pokud se takový chytrý ukazatel alokuje na zásobníku a program doběhne mimo jeho platnost, klesne-li přitom čítač na nulu, je objekt automaticky odalokován. Tímto se zamezuje velkému počtu těžko dohledatelných chyb.

ITK naopak neposkytuje žádné možnosti vizualizace dat nebo grafického uživatelského prostředí, proto je často kombinována například s použitím knihovny VTK.

Práce [4] rozděluje knihovnu ITK do několika oblastí, každou oblast popíšeme. Získáme lepší představu o možnostech knihovny ITK.

Numerická knihovna VNL — numerická knihovna, která obsahuje mnoho algoritmů a datových struktur jako například: matice, vektory, polynomy, konstanty, dekompoziční algoritmy, optimalizační algoritmy. Okolo všech těchto tříd je napsána obalová třída (wrapper) v ITK.

Reprezentace dat — pro reprezentaci dat jsou v ITK použity dvě třídy `itk::Image` a `itk::Mesh`. Obě jsou dostatečně obecné pro reprezentaci dat různých dimenzí. K datům se v obou případech přistupuje pomocí různých druhů iterátorů. Obě třídy dědí od jedné z nejdůležitějších tříd v ITK knihovně a to `itk::DataObject`.

Zpracování dat (roura) — procesy neboli filtry jsou klasické objekty, které zpracovávají datové objekty. Každý proces je odvozen od další velmi důležité třídy `itk::ProcessObject`. Celá knihovna je postavena na architektuře proudového zpracování dat (data-flow), což znamená, že data a procesy jsou střídavě propojeny v rouře proudového zpracování. Roura pak řídí průběh zpracování dat, tedy rozhoduje o tom, která data se budou zpracovávat. Pokud je povoleno více vláken, rozhodne, jak budou data rozdělena vláknům, která je pak zpracují.

Vstup a výstup — jsou to procesy(zdroje), které stojí na začátku a na konci roury. Typickými zástupci těchto procesů jsou `itk::ImageFileReader` a `itk::ImageFileWriter`. Obě třídy podporují mnoho vstupních a výstupních formátů dat(souborů).

Spatial objects — tyto třídy reprezentují v ITK geometrické tvary jako například elipsu, válec, krychli,... Tyto objekty se mohou spojovat a vytvářet hierarchie. Zjednodušeně má každý prostorový objekt svoji transformaci vůči nadřazenému objektu a podřízené prostorové objekty.

Registrace — podpora několika druhů registračních postupů. Například registrace obrázků, registrace geometrických tvarů, pyramidová (multiresolution) registrace, non-rigid registrace pomocí B-Spline, registrace založená na rovnicích parciálních derivací (PDE), registrace založená na metodě konečných prvků (FEM).

Level-sety — podpora pro tvorbu filtrů, které různými metodami řeší rovnice parciálních derivací na obrázcích. Je zde implementováno několik pokročilých numerických metod řešících tyto rovnice. 3.6.2.

Knihovna ITK byla použita při implementaci algoritmů popsaných v kapitole 5. Některé informace založené na programátorské zkušenosti s touto knihovnou jsou v programátorské dokumentaci v příloze C.

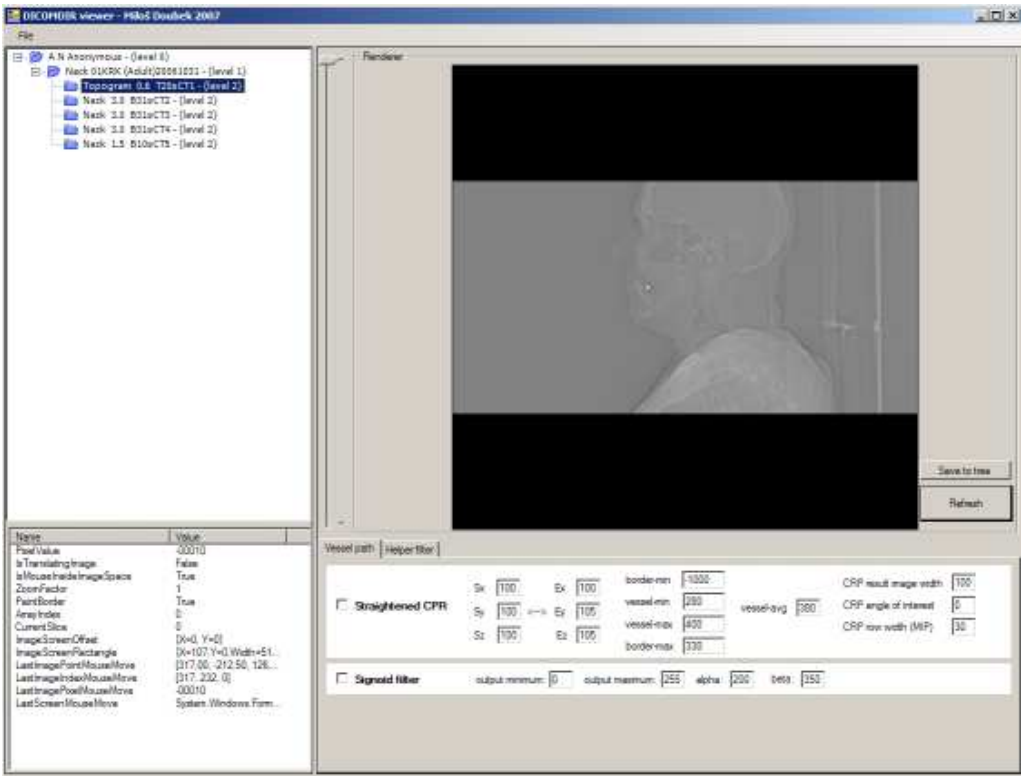
Dodatek B

Uživatelský manuál

Přestože na uživatelské rozhraní v této práci nebyly kladeny velké nároky, podařilo se díky ideálnímu spojení jádra systému a *Windows Forms* dosáhnout rychlé a zároveň uživatelsky příjemné aplikace. Tvorba grafického rozhraní pomocí *Windows Forms* je pak například ve vývojovém prostředí *Microsoft Visual Studio* velmi rychlá a variabilní. Následuje krátký popis hlavního okna aplikace.

B.1 Popis aplikace

Hlavní okno aplikace je na obrázku B.1, v levé části nahoře je stromový seznam pacientů: v první úrovni (level 0) jméno pacienta, v druhé úrovni (level 1) je načtená studie a ve třetí úrovni (level 2) je již série snímků. Největší část aplikace vpravo nahoře zabírá zobrazovací komponenta, která zobrazuje vybranou sérii snímků. V pravé části této komponenty je posuvník (track bar), který mění zobrazovaný řez vybrané série. Oblast vlevo dole je informační tabule, kde jsou uvedeny různé hodnoty zobrazovaného obrázku. Poslední oblast vpravo dole je určena k nastavení parametrů různých filtrů.



Obrázek B.1: Hlavní okna aplikace.

B.2 Funkce aplikace a její použití

Funkci aplikace popíšeme pomocí odpovědí na několik otázek:

Je třeba aplikaci instalovat? — aplikaci lze jednoduše bez instalace spustit. Je ovšem třeba mít nainstalovaný *.NET Framework 2.0* a *Microsoft Visual C++ 2008 Redistributable Package (x86)*, vše je na přiloženém DVD v adresáři `install`.

Jak načíst obrázek? — obrázek lze do naší aplikace načíst několika způsoby. Prvním je použít hlavní menu — `Open file - Ctrl + O` ve výběrovém okně je možné vybrat soubory typu MHD, DICOM nebo DICOMDIR. Pokud chce uživatel načíst celý dataset, který je uložený jako množství DICOM souborů v jednom adresáři, zvolí v menu `Open series + Ctrl+R` a vybere daný adresář. Další možností je použít metodu drag&drop a přenést soubor nebo více souborů do naší aplikace, například z Windows Exploreru, a konečně

poslední možností je asociovat danou příponu souboru (mhd nebo dcm) s naším programem.

Po otevření souboru jsou načtena pouze meta data obrázku, jako je jméno pacienta, popis studie,... Samotná data jsou načtena až v momentě, kdy uživatel označí serii (level 2), kterou chce zobrazit. Při načítání změní tlačítko 'Refresh' svoji barvu na červenou. Po načtení a zobrazení dat v zobrazovací části změní tlačítko svou barvu zpět na šedou.

Jak pracovat se zobrazovací částí? — zobrazovací číst uživatel ovládá pomocí myši. Se stisknutým pravým tlačítkem se posouvá celý obrázek, kolečko myši zmenšuje a zvětšuje obrázek. Levým tlačítkem myši je možno vybírat body, které se předvyplní v parametrech filtrů. Přechod mezi řezy trojrozměrného obrázku je realizován pomocí posuvníku na levé straně zobrazovací části. Pokud je obrázek pouze dvojrozměrný, je tento posuvník neaktivní.

Jak filtrovat obrázek? — vždy je vybraný pouze jeden obrázek, který lze filtrovat, a to ten, který je právě zobrazován. Uživatel vybere filtr, který chce použít, a vyplní jeho parametry. Více o každém parametru je si lze přečíst v tooltipu každého textového políčka. Je možné použít i více filtrů za sebou. Filtrace se spustí velkým tlačítkem 'Refresh' v zobrazovací části.

B.3 Testování jádra

Jádro systému lze otestovat samostatně — na přiloženém DVD jsou v adresáři *data* uloženy dávkové soubory, jejichž spuštěním se spustí metoda popsaná v této práci, a pomocí metody CPR je vygenerován výsledný obrázek. Toto testování se hodí zejména k testování časové náročnosti metody.

Dodatek C

Programátorská dokumentace

Program, který byl vytvořen v rámci této práce, je rozdělen do dvou částí. První část je uživatelské rozhraní, což je projekt s názvem `ArTra`. Druhá část je jádro aplikace — projekt s názvem `MinimalPathFilter`. Obě části jsou naprosto rozdílné, na každou z nich byly kladeny jiné požadavky. Zatímco od uživatelského rozhraní jsme požadovali rychlý vývoj GUI (graphic user interface), od jádra aplikace požadujeme rychlost, multi-platformitu a paralelní běh. Každé části se budeme věnovat v samostatné podkapitole.

C.1 Jádro systému

Jádro aplikace je napsáno v jazyku C++ jako soustava ITK filtrů, které jsou propojeny do roury proudového zpracování dat. Podrobněji o knihovně ITK se lze dočíst v příloze A. Podmínkou pro psaní ITK filtrů je znalost obsahu knihy [4]. ITK filtry mají danou strukturu a povinné parametry, jejichž absence může způsobit nesprávnou funkčnost filtru. Obecně se filtry dědí od vhodného předka, jehož výběr je závislý na typu vstupních a výstupních dat. Dobrý zdroj informací o tom, jak se který filtr používá, jsou testy, které ověřují správnou funkčnost filtrů (soubory končící na `test.cxx`). Následuje seznam všech filtrů a pomocných tříd, které byly během práce implementovány.

`DiscretePriorityQueue` — pomocná třída, která implementuje diskrétní prioritní frontu, tak jak ji popisuje kapitola 5.1.2 a obrázek 5.2.

`FastSmoothShrinkImageFilter` — ITK filtr typu "image to image". Na svém vstupu má shrink faktor r konstantu τ a průměrnou hodnotu tepny $vessel_{avg}$, na

výstupu pak zmenšený obrázek podle zadaných hodnot. Popis jeho výstupu a vysvětlení významu vstupních hodnot je v kapitole 5.1.4.

`MinimalPathImageFilter` — filtr typu "image to path", ve vstupním obrázku najde cestu tepnou, jak popisuje kapitola 7. Ke zmenšení obrázků třída používá `FastSmoothShrinkImageFilter`. Dále obsahuje `PathOptimalizer` a `ThresholdOptimalizer` třídy počítající ohodnocení hran grafu podle faktorů, které jsou popsané v kapitole 5.1.4.

`ObliqueSectionImageFilter` — filtr typu "image to image", ze vstupního 3D obrázku vygeneruje 2D obrázek (řez) podle zadaného vstupního bodu (střed výsledného obrázku), vektoru pohledu $view - vector$ a vektoru, který ve výsledném obrázku směřuje vzhůru $up - vector$. Inicializace použité transformace je v samostatné třídě `LookAtTransformInitializer`.

`PolyLineWithDeriveParametricPath` — datová struktura, sloužící k uložení seznamu bodů tvořících cestu. Třída používá lineární interpolaci k dopočítání bodů ležících na cestě. Směrový vektor (derivace v bodě) se počítá podle vzorce 5.12. Tato datová struktura je použita jako výstup filtru `MinimalPathImageFilter`.

`StraightenedCPRImageFilter` — filtr typu "image and path to image" podle algoritmu 2 vypočte 2D obrázek ze vstupního 3D obrázku a seznamu bodů definujících cestu.

`VesselOptimalizeLMPathFilter` — filtr typu "image and path to spatial object". Filtr, který lícuje model ke vstupnímu obrázku tak, jak to popisuje celá kapitola 5.2. Výstup filtru je objekt, který reprezentuje cévu (kromě pozice jeho bodu je zde uložen například i průměr cévy). Výpočet metriky pro LMA (Levenberg-Marquardt algorithm) obstarává třída `MultipleValuedVesselToModelAlignmentCostFunction`.

C.2 Grafické rozhraní

Grafické rozhraní je zcela odděleno od jádra, je napsáno v jazyce *C#* s pomocí *Windows Forms*. Přechod mezi *managed* a *unmanaged* kódem je realizován v projektu `ArTraCore`, který je napsaný v jazyce *managed C++*. Jako obalové třídy, které přenáší *unmanaged* instance do *managed* kódu, se používají třídy z knihovny

ManagedITK [1] [3]. Celá aplikace je rozdělena do několika projektů, jejichž výčet následuje:

`ArTra.Viewer` — základní spouštěcí bod grafického rozhraní, obsluhuje základní okno aplikace.

`ArTra.Core` — jádro aplikace, zpracovává vstupní data. Používá knihovnu *ManagedITK* pro zpřístupnění funkcí z knihovny ITK.

`ArTraCore` — pokud funkce z knihovny ITK není dosud zpřístupněna přes *ManagedITK*, přistupuje tento projekt k takovým funkcím přímo. V budoucnu, až budou v *ManagedITK* zpřístupněný všechny funkce a datové struktury, by tento projekt měl zaniknout.

`ArTra.GdiRenderer` — komponenta, která zobrazuje objemové obrázky po řezech. Jednotlivé řezy umí zvětšovat a odečítat hodnoty v konkrétním bodě. Pro zobrazování dat používá pouze *Microsoft Windows GDI+*.

`ArTra.VtkRenderer` — komponenta zobrazující obrázky s hardwarovou podporou pomocí knihovny VTK. V současné aplikaci není použita.

`ArTra.Dicom` — projekt se stará o načítání DICOM souborů a získavání informací z DICOMDIR souboru včetně jeho kompletního načtení. K tomu používá knihovnu `Gobosh.Dicom`.

`Anonymizer` — program anonymizuje lékařská data, dovede procházet adresářovou strukturu a pokud jméno souboru vyhovuje zadané šabloně, pak data anonymizuje.

`Aga.Controls` — komponenta zobrazující data ve stromové struktuře.

`Aga.Diagnostics` — obsahuje pomocné třídy pro logování událostí a přesné počítání časového intervalu.

C.3 Další vývoj

Další vývoj by měl jít směrem eliminace projektu `ArTraCore`, což vyžaduje dopsání .NET wrapper tříd okolo ITK tříd, které jsou potřeba pro běh aplikace.

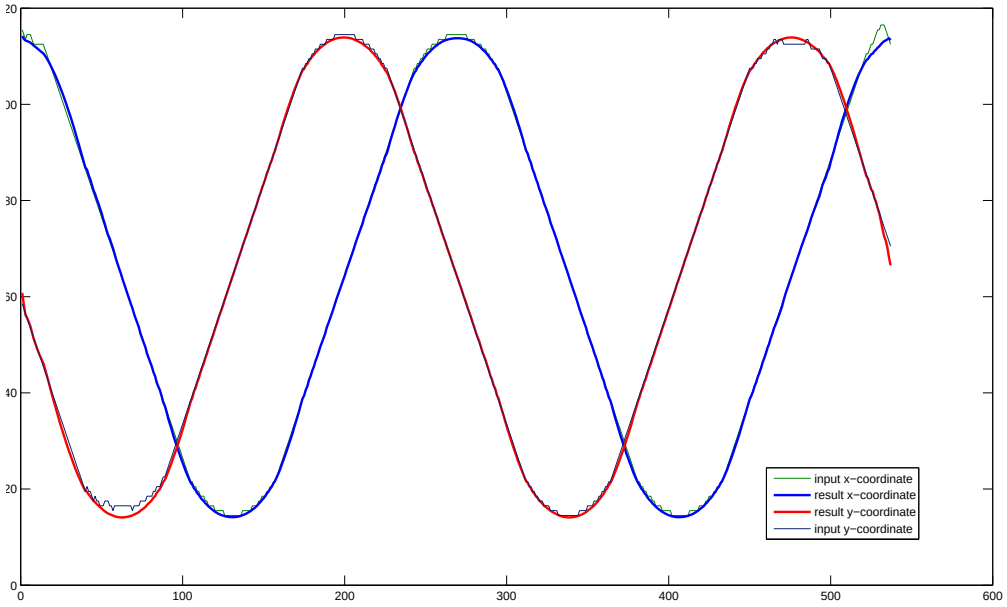
Psaní samotných filtrů v *managed* kódu s podporou knihovny ManagedITK tak, jak to popisuje [3], nemůžeme z hlediska optimální rychlosti doporučit. Lepší

[1] .NET ITK wrapper

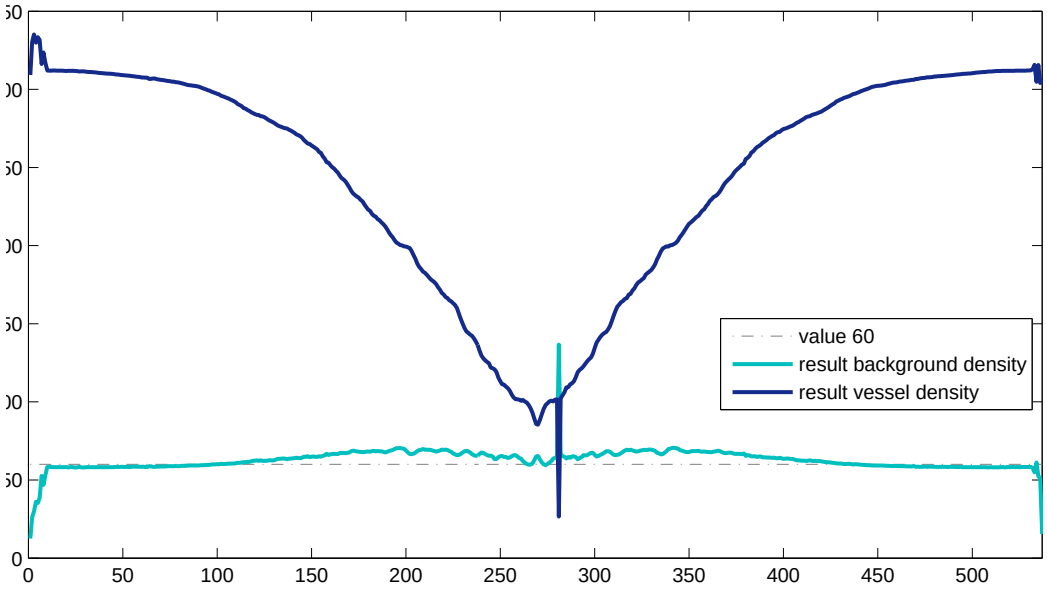
řešení je zpracovat obrázek pomocí klasického ITK filtru a do *managed* kódu přenést pouze výsledek, jak to dělá naše aplikace. Je třeba dbát na to, aby nedocházelo k častým přechodům mezi *managed* a *unmanaged* částí, neboť to může způsobit výrazné zpomalení aplikace.

Dodatek D

Grafy výsledných parametrů válcovitého modelu



Obrázek D.1: Srovnání počáteční pozice bodu cesty (slabá křivka) a výsledné pozice středu tepny (silná křivka) v průběhu celého zpracovávaného úseku tepny na datech *spiral*.



Obrázek D.2: Výsledné parametry při optimalizaci dat *spiral* — b (denzita pozadí) a V (denzita tepny). Správná hodnota pro denzitu pozadí je 60. Denzita tepny se v průběhu cesty mění, jak správně ukazuje tmavě modrá křivka.

Literatura

[1] Cemil Kirbas, Francis Quek : *A review of vessel extraction techniques and algorithms*, VISLab, January 2003.

[2] Dan Mueller : *SharpImage: An Image Processing Prototyping Environment*, Queensland University of Technology, Brisbane, Australia, July 2007.

[3] Dan Mueller : *ManagedITK: .NET Wrappers for ITK*, Queensland University of Technology, Brisbane, Australia, February 2008.

[4] Luis Ibanez, Will Schroeder, Lydia Ng, Josh Cates : *The ITK Software Guide, Second Edition*, Insight Software Consortium, November 2005.

[5] Yoshinobu Sato, Shin Nakajima, Nobuyuki Shiraga, Hideki Atsumi, Shigeyuki Yoshida, Thomas Koller, Guido Gerig, Ron Kikinis : *3d multi-scale line filter for segmentation and visualization of curvilinear structures in medical images*,IEEE Medical Image Analysis, vol. 2, pp. 143168, June 1998.

[6] A. F. Frangi, W. J. Niessen, K. L. Vincken, M. A. Viergever : *Multiscale vessel enhancement filtering*,Medical Image Computing and Computer-Assisted Intervention - MICCAI'98, volume 1496 of LNCS, pages 130–137. SpringerVerlag, Germany, 1998.

[7] A. Kanitsar, R. Wegenkittl, P. Felkel, D. Fleischmann, D. Sandner, E. Groller : *Computed Tomography Angiography: A Case Study of Peripheral Vessel Investigation*,IEEE Visualization, pages 477 480, October 2001.

[8] Kanitsar, A.; Fleischmann, D.; Wegenkittl, R.; Felkel, P.; Groller, E. :*CPR - curved planar reformation*, Visualization, 2002. VIS 2002. IEEE Volume , Issue , 1-1 Nov. 2002 Page(s): 37 - 44

[9] Kanitsar, A.; Wegenkittl, R.; Fleischmann, D.; Groller, M.E. :*Advanced curved planar reformation: flattening of vascular structures*, Visualization IEEE Volume, Issue 24-24, October 2003.

[10] Lukáš Maršálek :*3D vizualizace v lékařství*, CGG MFF UK – CGUdS, PRAGUE ACM SIGGRAPH, leden 2008. `http://cgg.ms.mff.cuni.cz/SIGGRAPH/2008-01-31-marsalek.pdf`

[11] Alexandra La Cruz : *3D Modeling and Reconstruction of Peripheral Arteries*, Institute of Computer Graphics and Algorithms Vienna University of Technology, Austria, January 2006.

[12] H.C van Assen, M. Egmont-Petersen and J.H.C. Reiber : *Accurate Object Localization in Gray Level Images Using the Center of Gravity Measure*, IEEE Transaction Imaging Processing, 11(12):13791384, December 2002.

[13] J. Canny : *A Computational Approach to Edge Detection*, IEEE Transactions Pattern Analysis Machine Intelligence, 8(6):679698, November 1986.

[14] A. Fitzgibbon and R. Fisher : *A Buyer's Guide to Conic Fitting*, Department of Artificial Intelligence, Edinburgh University, 1995. `http://research.microsoft.com/ awf/papers/buyers.ps.gz`

[15] R. Halír and J. Flusser : *Numerically Stable Direct Least Squares Fitting of Ellipses*, Department of Software Engineering, Charles University, 1998. `http://autotrace.sourceforge.net/WSCG98.pdf`

[16] Václav Krajícek : *Merení objemu v 3D datech*, Kabinet software a výuky informatiky, Matematicko-fyzikální fakulta, Karlova Univerzita, 2007. `http://195.113.19.34/ vajicek/dipl/`

[17] Gang Yu, Yalin Miao, Peng Li and Zhengzhong Bian : *Multiscale Vessel Segmentation: A Level Set Approach*, School of Life Science and Technology, Xi'an Jiaotong University, October 2005.

[18] C. M. van Bemmel, L. J. Spreeuwers, M. A. Viergever and W. J. Niessen : *Level-Set Based Carotid Artery Segmentation for Stenosis Grading*, Image Sciences Institute, University Medical Center Utrecht, January 2002.

[19] Jaeyoun Yi, Jong Beom Ra : *A locally adaptive region growing algorithm for vascular segmentation*, International Journal of Imaging Systems and Technology, vol. 13, pages 208-214, June 2003. http://dx.doi.org/10.1002/ima.10059

[20] Pierre Soille : *Morphological Image Analysis: Principles and Applications*, Springer-Verlag New York, Inc., 2003.

[21] V. Caselles and R. Kimmel and G. Sapiro : *Geodesic active contours*, ICCV '95: Proceedings of the Fifth International Conference on Computer Vision, 1995.

[22] Tim McInerney and Demetri Terzopoulos : *T-snakes: Topology adaptive snakes*, Department of Computer Science, University of Toronto, August 1999.

[23] C. Zahlten : *Reconstruction of branching blood vessels from CT-data*, Visualization in Scientific Computing, Springer-Verlag, pages 41-52, 1995.

[24] O. Wink,W. J. Niessen, M. A. Viergever : *Fast delineation and visualization of vessels in 3-D angiographicimages*, Medical Imaging, IEEE Transactions on Volume 19, Issue 4, Pages 337-346, April 2000.

[25] A. X. Falcao, J. K. Udupa, and F. K. Miyazawa : *An ultra-fast user-steered image segmentation paradigm: Live wire on the fly*, IEEE Transaction on Medical Imaging, 19(1): 5562, January 2000.

[26] A. X. Falcao : *Paradigmas de Segmentacao de Imagens Guiada pelo Usuario: Live Wire, Live-Lane e 3D-Live-Wire*, PhD thesis, Institute of Computing - IC State University of Campinas - UNICAMP, 1997.

[27] Lauren O'Donnell : *Semi-Automatic Medical Image Segmentation*, Master of Science in Computer Science and Engineering, Massachusetts Institute of Technology, October 2001.

[28] Petr Felkel : *Segmentation of Vessels in Peripheral CTA Datasets*, SCCG 2001, pages 25-28, VRVis Center, Austria 2001.

[29] A. Vilanova i Bartrolí : *Interactive segmentation of medical images based on intelligent scissors*, Projektarbeit in Fach Informatik, TU Wien, April 1997

[30] Karl Krissian, Grégoire Malandain, Nicholas Ayache, Regis Vaillant and Yves Trousse : *Model Based Detection of Tubular Structures in 3D Images*, Technical Report 3736, INRIA, Sophia Antipolis, France, 1999.

[31] Stefan Worz and Karl Rohr : *A New 3D Parametric Intensity Model for Accurate Segmentation and Quantization of Human Vessels*, In Medical Image Computing and Computer-Assisted Intervention - MICCAI. Springer, September 2004.

[32] E. W. Dijkstra : *A note on two problems in connexion with graphs*, Numerische Mathematik. 1, pages 269–271, 1959.

[33] Chang-le Lu, Yong Chen : *Using Multi-Thread Technology Realize Most Short-Path Parallel Algorithm*,World Academy of Science, Engineering and Technology (PWASET), volume 15, october 2006.

[34] A. Crauser and K. Mehlhorn and U. Meyer and P. Sanders : *A Parallelization of Dijkstra's Shortest Path Algorithm*, Lecture Notes in Computer Science, volume 1450, pages 722-739, 1998.

[35] Kai Zeng, Erwei Bai, Ge Wang : *A fast CT reconstruction scheme for a general multi-core PC*, Journal of Biomedical Imaging, Volume 2007, Issue 1, January 2007.

[36] B. M. Carvalho, G. T. Herman, S. Matej : *Abstract ART for helical cone-beam CT reconstruction*, Department of Computer and Information Science, University of Pennsylvania.

[37] Gil Schwarzband and Nahum Kiryati : *The point spread function of spiral CT*, School of Electrical Engineering, Tel Aviv University, November 2005.

[38] Mgr. Radka Svobodová Vareková : *Optimalizace - PV027*, National Center for Biomolecular Research, Masarykova univerzita, 2005. `http://ncbr.chemi.muni.cz/ n19n/vyuka/optimalizace/`

[39] Levenberg, K. : *A Method for the Solution of Certain Problems in Least Squares*, Quart. Appl. Math. 2, 164-168, 1944.

[40] Ananth Ranganathan : *The Levenberg-Marquardt Algorithm*, June 2004.

[41] Manolis I.A. Lourakis and Antonis A. Argyros : *Is Levenberg-Marquardt the Most Eficient Optimization Algorithm for Implementing Bundle Adjustment?*, Institute of Computer Science, Foundation for Research and Technology, 2005.

[42] Jorge J. Moré : *The Levenberg-Marquardt algorithm: Implementation and theory*, Lecture Notes in Mathematics, Numerical Analysis, November 2006.

[43] Martin Střelec : *Využití metody nelineárních nejmenších čtverců pro rekonstrukci přechodové charakteristiky*, Západočeská univerzita v Plzni, Katedra kybernetiky, 2006.

[44] Henry Gray : Anatomy of the Huma Body, 1918. - Bartleby.com's version of Henry Gray's Anatomy of the Human Body