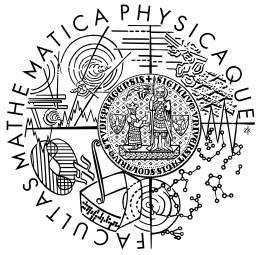

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Jan Horáček

Volumetric data processing for CT enterography

Department of Software and Computer Science Education

Supervisor of the master thesis:	RNDr. Josef Pelikán
Study programme:	Computer Science
Study branch:	Software Systems

Prague 2016

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Volumetric data processing for CT enterography

Author: Jan Horáček

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Josef Pelikán, Department of Software and Computer Science Education

Abstract: The overall goal of our work is to develop algorithms for efficient processing, segmentation and tracking of the small intestine in CT enterography scans. The small intestine is a complex organ, the shape of which can vary considerably between patients: and in addition to this, its location and shape can change significantly between subsequent scans of the same person. The CT enterography process uses contrast agents to improve the visibility of the intestine, so that various potentially problematic features, such as inflammations, obstructions and so on, can be properly seen. However, due to the convoluted shape of the organ, manual diagnosis of raw CT enterography data is still a difficult and time-consuming task, and is prone to diagnostic errors. We have prepared a set of methods for automatic preprocessing, segmentation and tracking of such data that aims at providing a much clearer data visualization: such tools can greatly improve the diagnostic process.

Our first contribution is to make a high quality denoising method for volumetric data practically usable: so far, it had been impractical due to its too high computational cost. This is solved by devising a GPU-friendly implementation scheme of the algorithm in question. Lowering its computation time from tens of minutes, or even hours, to a few minutes at most (depending on HW), finally makes it possible to use this algorithm for everyday practical work: and such a high quality denoising step is crucial for a later successful segmentation of the data. The next contribution is a system for computing the probability of intestinal lumen and intestinal walls on watershed-segmented regions. We propose a system for computing this probability based on several statistical features that are computed over the watershed regions. We also provide a discussion of the performance and suitability of the most promising subset of these features.

Using these computed lumen and wall probabilities, we then propose a robust algorithm for tracking the small intestine path through the bowel area: within this technique, we also address problems caused by the data imperfections that are typical of real CT scans. Furthermore, we propose an algorithm for precise segmentation of the lumen that is usable for wall analysis on tracked data. Finally a set of visualization possibilities is presented, as suggestions for practical usage of the results that are provided by the proposed pipeline.

Overall, we manage to create an automatic pipeline for processing noisy thin-slice CT enterography scans into segmented and tracked data: the result is much more suitable for diagnostic purposes than the original raw CT data. The only manual processing step in our pipeline is a simple removal of certain unwanted features, such as a distended colon, that are very similar in appearance to the small intestine. Our techniques allow for visualization of the entire bulk volume of the intestine to show topological locations and possible regions of interest, as well as a detailed visualization along individual intestinal segments for closer inspection.

Keywords: segmentation, GPU, denoising, CT enterography

Název: Zpracování objemových dat pro CT enterografii

Autor: Jan Horáček

Katedra: Katedra software a výuky informatiky

Vedoucí: RNDr. Josef Pelikán, Katedra software a výuky informatiky

Abstrakt: Primárním cílem našeho výzkumu bylo vyvinout algoritmy pro zpracování, segmentaci a sledování průběhu tenkého střeva na snímcích CT enterografie. Tenké střevo je spletitý orgán, u každého člověka výrazně odlišný a ani dva skeny jednoho pacienta nemusí být nutně podobné. Metoda CT enterografie využívá kontrastních materiálů ke zvýraznění a vyčištění střeva na snímcích výpočetní tomografie. Tím pádem mohou být diagnostikovány různé zánětlivé choroby, obstrukce a podobně. Ovšem manuální zpracování těchto snímků je velmi náročné. Vyžaduje mnoho času a úsilí kvůli tvaru střevního traktu, velkému množství šumu na snímcích a celkové velikosti dat. To může vést k chybám či přehlédnutím. Kvůli zjednodušení diagnostiky jsme připravili sadu metod pro automatické předzpracování, segmentaci a sledování průběhu, jejichž výsledky jsou vhodné pro přehlednou vizualizaci.

Prvním krokem bylo umožnění využití algoritmu pro vysoce kvalitní odstraňování šumu na objemová data. Navrhli jsme implementační schéma vhodné pro GPU a tím snížili časovou náročnost z desítek minut až několika hodin na jednotky minut (dle použitého hardware). Takto je možné nasazení i v každodenní praxi. Kvalitní algoritmus pro odstraňování šumu je důležitým krokem pro úspěšnou segmentaci těchto dat. Dalším krokem je systém pro výpočet pravděpodobnosti střevního lumenu a stěny v rámci regionů vzniklých watershed segmentací. Navrhli jsme systém na odhad této pravděpodobnosti založený na několika statistických ukazatelích počítaných přes tyto regiony. Uvedli jsme i rozbor vhodnosti a výkonu nejslibnějších statistických ukazatelů.

Dále jsme navrhli robustní algoritmus pro sledování průběhu střeva s využitím těchto pravděpodobností. Věnovali jsme se vyřešení problémů spojených s aplikací na reálná CT data a chybami s nimi spojenými. Navrhli jsme algoritmus pro přesnou segmentaci střevního lumenu nad takto sledovaným průběhem kliček - využitelný pro podrobnější analýzu střevní stěny. Nakonec jsme ukázali několik nápadů, jak data získaná našimi metodami efektivně zobrazovat pro praxi.

Podařilo se nám vytvořit automatickou sadu kroků pro zpracování tenkých CT-enterografických řezů s velkým množstvím šumu. Výsledkem jsou vysegmentovaná a trackovaná data s výrazně usnadněnými možnostmi diagnostiky. Jediným manuálním krokem zůstává jednoduché odstranění nechtěných částí těla, která jsou na CT enterografických snímcích velmi podobná tenkému střevu - například tlusté střevo. Naše metoda umožňuje nejen zobrazení vysegmentovaného kompletu pro získání informace o umístění v těle, zvýraznění zajímavých částí, ale také zobrazení a virtuální narovnání jednotlivých střevních kliček pro podrobnou analýzu.

Klíčová slova: segmentace, GPU, odstraňování šumu, CT enterografie

Dedication:

I would like to thank my supervisor Josef Pelikán for leading me through the perils of research work and giving guidance in my efforts. Another thanks go to my colleagues - mainly Jan Kolomazník, Petr Kmoch and Václav Krajíček - they were creating a pleasant research environment, giving fruitful discussions and providing interesting (and sometimes also crazy) ideas.

I would also like to thank my closest family members - my parents and my girlfriend - for supporting me everyday over any obstacle during my study and research years, tolerating long night working sessions and also reviewing some of the non-research related aspects of my work summarized in this thesis.

Finally an important thanks goes to Dr. Martin Horák, who made this whole work possible by providing invaluable experience about the medical aspect. The range of information needed for a successful research was enormous, ranging from the acquisition process to the final analysis of the data.

Contents

Introduction

In this thesis we would like to show and summarize our research on efficient volumetric data processing, especially the processing of computed tomography (CT) and CT enterography (CTE). We present several new algorithms to preprocess, segment and track the intestine.

Thesis goal

The primary goal is to provide significant support for the diagnosis of the small bowel and the inflammatory diseases of the intestine, such as Crohn's disease.

We have tried to automate as many of the computational steps as possible, so that the only intervention of a human user would be to provide data and then after some computation directly traverse through the segmented intestinal segments.

The result is a series of algorithmic steps whose parameters have to be set-up once per scan type and then may be executed automatically. The algorithms are dependent on the setting of the CT scanner and on the preparation of the patient. However, they should be consistent within a group of similarly set-up scans coming from one machine type, which we tried to demonstrate on routinely scanned datasets.

The analysis and final diagnosis at the end must be done by a radiology specialist, but we are trying to give as much information about the patient as possible while removing the inherent complexity of the intestinal path through the body to speed up the exploration of the dataset and point at potentially important places in the organ.

The reduction of a radiologist's time and effort spent on one dataset is important, because a significant help for the medical expert should lead to lowered probability of an error, overlooked problem and wrong diagnosis. Good visualization in 3D also helps in planning a potential surgical intervention.

Thesis overview

The structure of the thesis follows both our research progress and the actual sequence of operations that are performed during the processing. It is divided into the following chapters:

- **Chapter 1** gives a brief introduction to the nature of the data that we are working on, including the acquisition process, reconstruction and specific details of CTE.

- **Chapter 2** contains a crucial step in our processing pipeline - denoising. We describe the sources of noise in the input data and our ways how to improve the data quality for further segmentation operations. We propose a fast implementation scheme of a high quality denoising algorithm adapted for our data.

- **Chapter 3** shows the subdivision of the dataset into watershed regions and discusses various parameters describing these regions. We propose a classification schema for the computation of probability - whether the region belongs to intestinal lumen, intestinal wall or other parts of the body and evaluate the results on real CT enterography datasets.

- **Chapter 4** describes our algorithm for tracking the small intestine where we estimate the continuity of individual intestinal segments. The result is a set of paths through the dataset, which follow well distended parts of the small bowel.

- **Chapter 5** uses the result from chapter 4 and describes our approach to a precise lumen border location estimation along each individual intestinal segment.

- **Chapter 6** uses tracked path and lumen segmentation results to visualize the structure of small intestine. We have tried several methods to visualize the segmented intestinal segments - in original CT space, in space orthogonal to the path, with 3D or polar representation of the lumen/wall intersection.

- **Chapter 7** gives an insight into our implementation process including the whole pipeline workflow and practical information about the programs used.

- **Chapter 8** briefly summarizes results obtained with our methods and concludes this thesis.

- **Appendix A.1** shows more in depth comparisons of feature spaces for probability computation in chapter 3. This chapter contains a comparison between noisy and denoised datasets and a side by side comparison of two different datasets.

- **Appendix A.2** discusses the stability of our classification process on an erroneously matched training set.

Chapter 1

Source data

1.1 Acquisition - Computed tomography

The advancements of computing technology in the last 10 years enabled a very significant step in the possibilities of data acquisition, large dataset processing and analysis. What was impossible a few years ago is now in common use.

Lets take for example computed (or in some sources computerized) tomography (CT). The base for this method is the detection of X-ray photon attenuation by a source and a detector (details behind the physics of X-rays can be found for example in Als-Nielsen and McMorrow [2011]). By sending high-energy photons through an object or patient, we detect only a subset of them on the detector cells or film.

The amount of photons detected gives us an estimation of the probability of transmission through an object. Materials such as air or extruded polystyrene have a very low probability of high energy photon absorption. Other materials like bone tissue and metals have a very low transmissivity, blocking also high energy photons (with energy in the range of MeV).

Comparing these probabilities gives us an idea about the internal structure of an object without the need to disassemble it. Such methods in industry are called nondestructive analysis. In medicine they are called noninvasive methods. In either case the main idea is to obtain internal structural information without the need to destroy or harm the object/patient.

The simplest (and also very frequently used) such method based on high energy photon absorption is called simply an X-ray image. It can be used both with analog and digital technology. Analog being a traditional X-ray sensitive film and digital being usually a flat-panel with a scintillator (basically a very large digital camera detector covered with a luminescent layer that emits visible light under ionizing radiation). It is a very common and relatively cheap technique to analyze broken bones, large tumors or (in industry) cracks, holes, bubbles and other discontinuities in manufactured parts. A nice recent survey of techniques used for X-ray image analysis is in Mery [2015]. However it is not very applicable for the analysis of a more complex structure of analyzed objects. More projections might be used, but even that is not enough in some cases of abdominal diseases or brain analysis.

Next step in the effort to *look inside objects* is laminography. This technique was developed for analog acquisition on traditional x-ray sensitive film. Currently

it has a digital reconstruction-based version as well. The principle of analog laminography is similar to X-ray imaging, but with the difference that the source and detector/film are simultaneously moving. The movement is synchronized so that only one plane of the inspected object always projects on the same part of the detector/film. The result is an image with one focused plane and everything above and below this plane is blurred. The shape of the blur is dependent on the trajectory of the source and detector/film movement. Usually a linear or circular movement is used. This way we can focus on an exact plane within the object. It is not enough for a complete reconstruction, but especially with flat objects the structure at a given height can be seen well enough. An analogy can be found in traditional microscopy, where there is also a very narrow depth of field and everything out of focus is blurred. The digital version closely resembles a CT reconstruction, but with some unavoidable artifacts.

When we need a full reconstruction of the internal structure, our best tool would be CT. The CT was invented in 1972 by Godfrey Hounsfield (in Hounsfield [1972]) and the principle is relatively simple - rotate the object 360° (with static source-detector pair, see figure 1.2) or rotate the source-detector (with static object, see figure 1.1), project the X-ray shadow from many angles to the detector and then reconstruct the volume. There are several possible geometric configurations of the source-detector pair (see for example Herman [2009]), but the principle stays the same: for each reconstructed point have as many projections as possible from different angles.

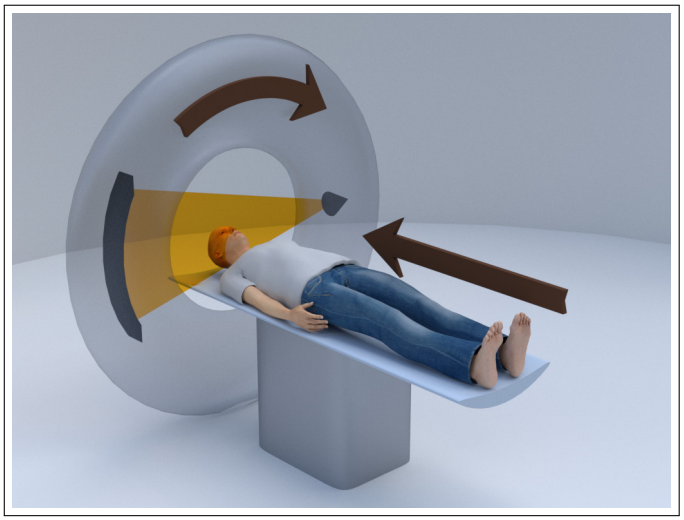

Figure 1.1: A schematic model of a CT used in health care.

The main problem with CT is the computational complexity of the reconstruction process and the final size of the data. A typical scan in medicine usually has hundreds of MB, industrial scans with higher resolutions may have dozens or even hundreds of GB. This is the perfect place to use the current computational power of modern computers, especially the massive parallel architectures such as GPUs. The reconstruction is only one step, further processing is necessary to enable the inspection and the analysis of such large datasets and this processing is usually very power consuming as well. We present several such algorithms that can leverage the available power for a thorough processing of CT datasets.

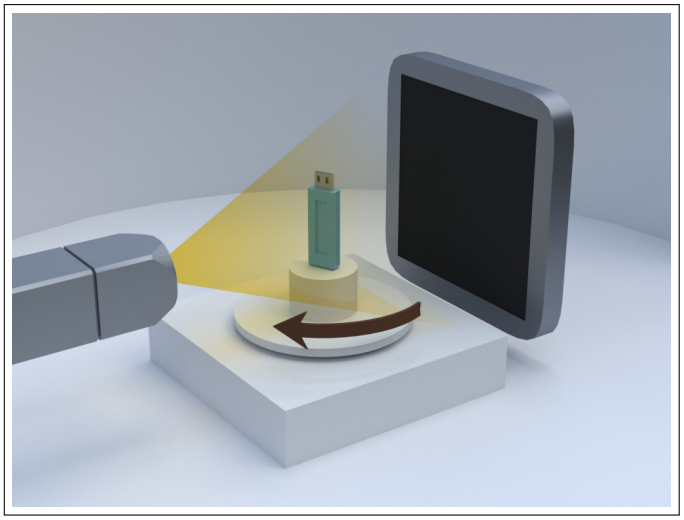

Figure 1.2: A schematic model of an industrial CT scan.

1.2 CT reconstruction

CT reconstruction is the process of obtaining information about the internal structure of an object from a set of X-ray projections. This section shows a very brief introduction to the reconstruction process to better understand the data we are working with. Definitions, equations and citations in this chapter are taken from Herman [2009], chapter 2.4 and 2.6.

Lets define the *linear attenuation coefficient* $\mu_{\overline{e}}^{t}$ of a tissue t at energy $\overline{e}$ as follows: Let ρ be the probability that a photon of energy $\overline{e}$ which enters a uniform block of the material t of unit thickness on a trajectory perpendicular to the face of the block, will not be absorbed or scattered in the material. We define:

$$\mu_{\overline{e}}^{t} = -\ln\rho \tag{1.1}$$

We usually do not work directly with *linear attenuation coefficient* as defined here, but rather a *relative linear attenuation coefficient* at energy $\overline{e}$ defined as $\mu_{\overline{e}}^{t} - \mu_{\overline{e}}^{a}$, where t is the material present during the actual measurement and a is the material present during calibration measurement. A CT number (or a number in Hounsfield unit - HU) is a linearly transformed relative attenuation with the background material water and scaled so that air has a CT number of approximately -1000.

This is an important fact, that the CT numbers can be *calibrated* and repeatable and a given CT number at an energy $\overline{e}$ represents an absolute property of the tissue. This is in contrast with other modalities (such as MRI, where the situation is not that simple). In the rest of this book we will be dealing with CT numbers in Hounsfield units as our source data.

The main goal of reconstruction is the computation of the linear attenuation coefficient (or CT number) from ρ, which is estimated from X-ray projections. The process is mostly one of the two types: reconstruction based on Inverse Radon transform (methods based on this are usually called filtered backprojection (FBP)) or iterative approaches, such as SART (Andersen and Kak [1984]).

An inverse Radon transform was created in 1917 by Johann Radon (Radon

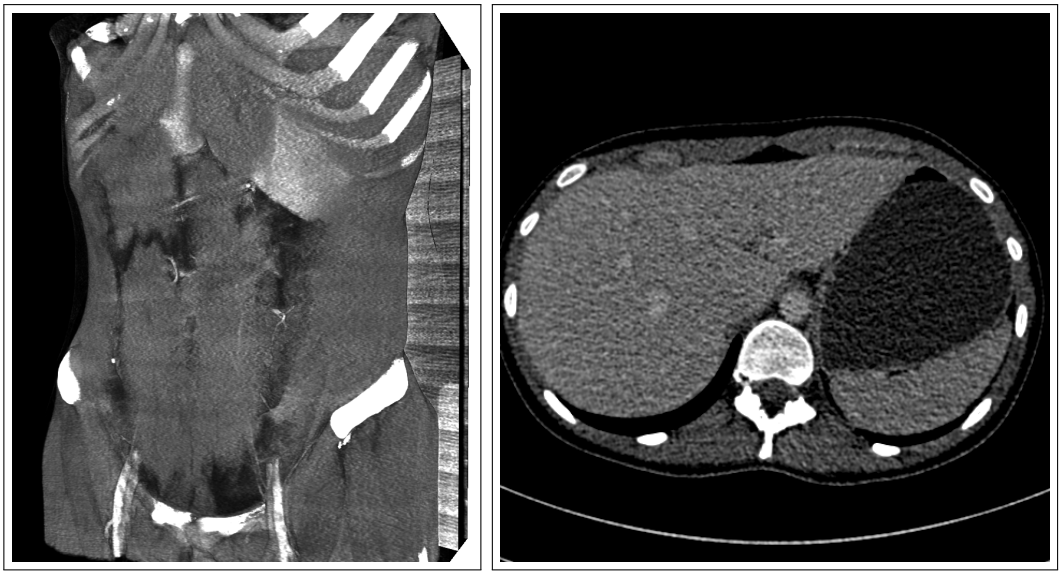

Figure 1.3: A 3D reconstruction from a medical CT. A DVR rendering of the whole dataset and one axial slice through liver and stomach.

[1917], English translation of the paper is in Radon [1986], a survey on several variants and usage is in Herman [2009] and Deans [1983]). It can be described (from Herman [2009]) as "the distribution of the relative linear attenuation in an infinitely thin slice is uniquely determined by the set of all its line integrals".

The equation is as follows: let L be the line of the photon trajectory, l the distance of L from the origin, *theta* the angle made with the x axis by the perpendicular drawn from the origin to L and let $m(l, \theta)$ be the integral of $\mu_{\bar{e}}(x, y)$ along the line L. Then:

$$\mu_{\bar{e}}(x, y) = -\frac{1}{2\pi^2} \lim_{\epsilon \to 0} \int_{\epsilon}^{\infty} \frac{1}{q} \int_{0}^{2\pi} m_1(x \cdot cos\theta + y \cdot sin\theta + q, \theta) d\theta dq \tag{1.2}$$

where $m_1(l, \theta)$ is the partial derivative of $m(l, \theta)$ with respect to l.

We will not go into details of solving this equation, but it is a well explored region of numerical mathematics (mostly dealing with a robust estimation and interpolation of $m_1(l, \theta)$ from noisy discrete points) and there are fast algorithms for computing this reconstruction. This is however easily applicable only to circular or helical geometry (which is the case of most full-body medical CT scanners, see figures 1.1 and 1.3).

On the other hand iterative approaches such as ART (Gordon et al. [1970] Hounsfield [1972]), SIRT (Gilbert [1972] Herman [1980]) and SART (Andersen and Kak [1984]) are trying to solve or approximate a set of equations over several iterations. This easily enables other geometries than circular/helical (such as Maisl et al. [2014]) or usage in ultrasound or microwave tomography. See for example figure 1.4.

Current medical CT scanners have begun to replace old FBP algorithms with the more computationally expensive iterative approaches. The reason for that is the simplicity of implementation of some improvements directly into the reconstruction process instead of post-processing the already reconstructed data. The

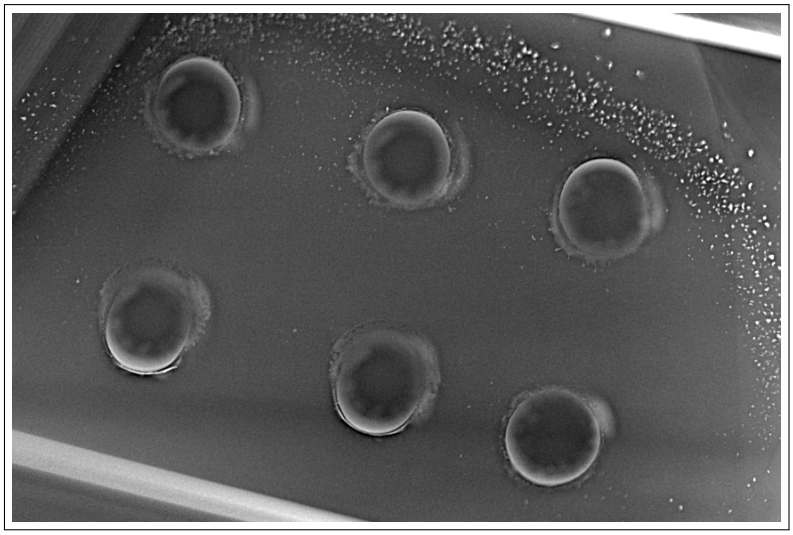

Figure 1.4: A nondestructive inspection of welds using X-ray tomography. Projections scanned with 220keV, 0.3mA.

scanning process is currently more limited during the physical acquisition by the sensitivity of detectors than the computational power of current computers.

1.3 CT enterography

CT examination is currently an invaluable technique for medical diagnosis, in some cases with no other alternative. It is relatively cheap (compared to for example MRI), fast (one revolution of source-detector pair takes around 0.5-1 seconds, full chest+abdomen scan is usually done within 2-5 seconds on a MDCT - Multi-detector CT) and has very good resolution in respect to bones and some organs with the possibility of a 3D visualization (especially important in case of complicated bone fractures such as in femoral neck, see an example in Horáček et al. [2009]). The speed of CT is especially important in medical diagnosis of patients, it is able to produce full tomographic projection within a second, minimizing the influence of peristalsis, respiration and heartbeat.

CT is based on radio-density measurement (stored in HU), which uses X-rays of various wavelengths to estimate the attenuation within a human body. Thus materials that have similar composition (like human internal organs with a large amount of water) behave similarly.

There have been several improvements to the standard CT process in case of internal organ diagnosis, such as post-contrast scanning. There are many variants, mostly using intravenous positive contrast agent with enhanced attenuation of X-rays to highlight parts of the body with high blood circulation. This way we can mark blood veins, circulation in the brain, enhance kidneys, tumors in the liver and other parts of the body.

CT enterography (CTE) is a special example of post-contrast computed tomography to simplify the diagnosis of bowel. It differs from traditional CT by using intravenous contrast material, large volumes of a neutral oral contrast agent and is scanned on a high resolution MDCT scanner (Booya et al. [2006], Paulsen et al. [2006], Federle [2007]). Neutral contrast agent is used to improve intestinal

lumen distention, intravenous agent is for enhancing intestinal wall, possible inflammation areas and additional complications. An example of difference between standard CT and CT enterography is shown in Figure 1.5.

Figure 1.5: A comparison of normal CT to CT enterography scan. Left image is contrast enhanced CT enterography (120 keV, 310mA). Right image is standard (non-contrast) CT scan (120 keV, 150mA).

Neutral enteric contrast agent might be a sufficient amount of water (cheap, well tolerated, available, but might pose problems due to fast absorption rate). There are also alternatives, such as methyl-cellulose solution, polyethylene glycol solution, and low-attenuation barium suspensions (Booya et al. [2006], Paulsen et al. [2006]). Oral administration of neutral contrast agent is usually sufficient, so no intubation is necessary, making the procedure much more pleasant for the patient (compared to for example CT enteroclysis). Positive intravenous contrast agent is injected right before the CT scan and precise timing is necessary to obtain good results (scan is done usually within 30-60 seconds after administering the intravenous contrast agent, exact timing varies between studies).

There is a range of other techniques for small bowel diagnosis (Cohen [2011]), such as endoscopy, capsule endoscopy, serological markers, fecal calprotectin and lactoferrin analysis and CT enteroclysis (Maglinte et al. [2007]) to name a few.

The main advantages of CTE over other techniques are its non-invasiveness, relative simplicity of patient preparation, well tolerated by patients, ability to clearly show wall thickening, prominence of vasa rectae (Madureira [2004]), abscesses, fistulae and mural stratification (Choi et al. [2003], Zalis and Singh [2004]) and compared to capsule endoscopy - no risk of capsule retention. It is thus very suitable for active Crohn's disease (CD) diagnosis, but also for other inflammatory diseases, bowel obstruction, etc.

The disadvantage is ionizing radiation with similar doses like a traditional CT scan, the inability to display intestinal wall color (necessary for example in the case of a mild or early disease, in which case a direct visualization of the mucosal surface using endoscopy or capsule endoscopy is necessary - Afifi and Kassem [2012]Sostegni et al. [2003]) and occasional intolerance to iodinated contrast material.

The accuracy of CTE when diagnosing CD has been reported to be 93.9%, with specificity 100% and sensitivity 85.7% (Mazzeo et al. [2001]).

1.4 Data

The only source data we use for our computation are CTE scans results. The nature of the data is volumetric with a regular 3D grid structure, where each voxel contains the information about the relative linear attenuation coefficient converted to HU. Preferred data are the ones with cubic or close to cubic shape of voxels (i.e. similar size in each dimension), usually the size of a voxel is around 0.5^3 to 0.75^3 mm.

The complete dataset consists of a set of axial slices with spatial resolution of 512×512 pixels/voxels. The amount of slices depends on the settings during the scanning, there are usually 300-800 slices per scan.

The HU values of the input data are provided as 12-bit values stored in 16-bit words. The storage/transfer format used is widely known as a DICOM standard. We use DCMTK toolkit for loading the data. Input values range from -1000 or -1024 HU for air, 3072 or 3096 for metals/maximal density and 0 for water (different values depend on dataset/machine settings, this information was included in source DICOM format in fields *RescaleIntercept* and *RescaleSlope*).

We have included only datasets from patients that have signed an agreement to use their data in a research project. All datasets were anonymised before using them for tests, we only kept necessary information with technical details of acquisition and reconstruction (scanning parameters, resolution, dimensions, etc.). Our datasets were scanned with 120keV power, tube current varied depending on each case. We had 34 datasets prepared this way for this research.

Internal format used for all computations was a custom volumetric format with 32-bit float as the basis for voxel value. This was used for all steps starting from the output of the denoising step (chapter 2 with the exception of markers (used for marking regions of segmentation results), where a 32-bit integer was used (the 24-bit mantissa of 32-bit float was insufficient).

The total size of the smallest input dataset was around 220MB (436 slices), the largest input dataset was around 390MB (784 slices). The intermediate data stored in 4-bytes per voxel took around twice that size per dataset. This size is currently supported without problems on modern GPUs, several steps (especially the denoising step) were performed on a GPU accelerator to significantly speed up the computation and enable the practical usage of our processing.

Chapter 2

Denoising volumetric data on GPU

2.1 Introduction

This chapter focuses on a much more general approach to volumetric data processing than the rest of this thesis. We are discussing denoising which can be used on many regular 3D volumetric datasets, independently on the data format. An efficient and fast GPU variant of the NLM (Nonlocal Means) algorithm (Buades et al. [2005]) is presented.

The reason behind this is that a very high quality denoising is crucial for the rest of this work. Original data without denoising were strongly inapropriate for standard segmentation algorithms. Examples of such data are shown in section 2.6.

However, a suitable denoising algorithm (such as NLM), which gives a very good result and performs optimal denoising is computationally very expensive. The results were sufficient for the rest of the segmentation process, but the computational burden being hours on a modern PC was a serious setback and prevented the rest of the segmentation process from being practically useful.

Because of this we had to create a new approach to the computation of NLM on an OpenCL-capable GPU to enable its use in practice. Here we would like to present our results from Horáček et al. [2011].

2.1.1 Denoising volumetric data

Many current medical imaging methods produce volumetric data, such as computed tomography (CT), magnetic resonance (MRI) and others. This data provides a good insight into the workings of the human body, but is also quite large and it is difficult to process with more advanced processing techniques. We are interested mainly in CT for our case.

For the examination of complicated structures and small details we need thin slices and also low radiation doses to avoid any unnecessary harm to the patient (a computation of a CT ionizing radiation dose to the patient can be found for example in DeMarco et al. [2005] and in Bazalova and Verhaegen [2007]). But thin slices bring a lot of noise with it and are thus very hard to follow and segment with automatic or semiautomatic methods. Many current segmentation

algorithms need a robust edge detection. In the case of many thin-slice CT enterography scans, the standard deviation of the *homogeneous* inner parts of the intestine (lumen filled with neutral contrast agent) might almost reach the level of the difference between the mean value of the lumen and the contrast enhanced intestinal wall (depending on the scanner settings, reconstruction process and the efficiency of the intravenous contrast). Therefore we need a robust denoising approach to apply some proven and efficient segmentation algorithm.

We will focus only on random noise, which is very apparent in the intestinal part of the body. Acquisition artifacts, such as *star* artifacts resulting from dense objects being present in the body are not so apparent, because unless the patient has some sort of metal implant this part of the body usually contains only the spine and upper part of pelvis and no other dense bones or objects. For example Gu et al. [2006] discusses a method of star artifact removal.

2.1.2 Noise in low-dose CT

CT reconstruction process described in section 1.2 shows us a mathematically simple solution to the problem of the X-ray tomographic inspection. However, we need precise estimations of the line integral 2.1 of relative linear attenuation along the line between the X-ray source and the detector and we also need an infinite number of them.

$$\int_0^D \mu_{\bar{e}}(x, y)dz \tag{2.1}$$

That is not practically achievable and from the discrete and noisy estimations rise errors that damage the result of the reconstruction. The exact analysis of the errors and noise is not simple, we will try to analyze the most prominent ones and the ones we are able to significantly reduce using techniques mentioned in this chapter.

The first and for us also most important source of errors is the physical nature of X-ray imaging - photons. Photon emission, interaction with material and detection are all stochastic in nature. Combined with the need to reduce the dose of high energy photons in medical imaging to prevent ionizing radiation damage to the patient, this fact elevates it to the most important problem of X-ray imaging.

As a side note: a similar problem, but arising from a different reason is found in industrial CT imaging. The doses are also kept to a possible minimum, although usually much higher than in health care. The reason is mainly the maximal sustainable power of micro-focus X-ray sources, service life and degradation of the detector (which might be one of the most expensive parts of an industrial CT) and time constraints due to generally larger number of projections needed for a good resolution in the reconstruction.

An example from an industry-grade detector is shown in 2.1. The acquisition parameters are different than in health care. Medical CT scanners usually operate with less voltage and higher current. But the complete exposition time is around 0.5s-2s (all projections), compared to 0.5s *per projection* in industrial CT (the total exposition time of a complete scan must be multiplied by the number of projections, such as 400-1600, depending on the resolution). We do not have a

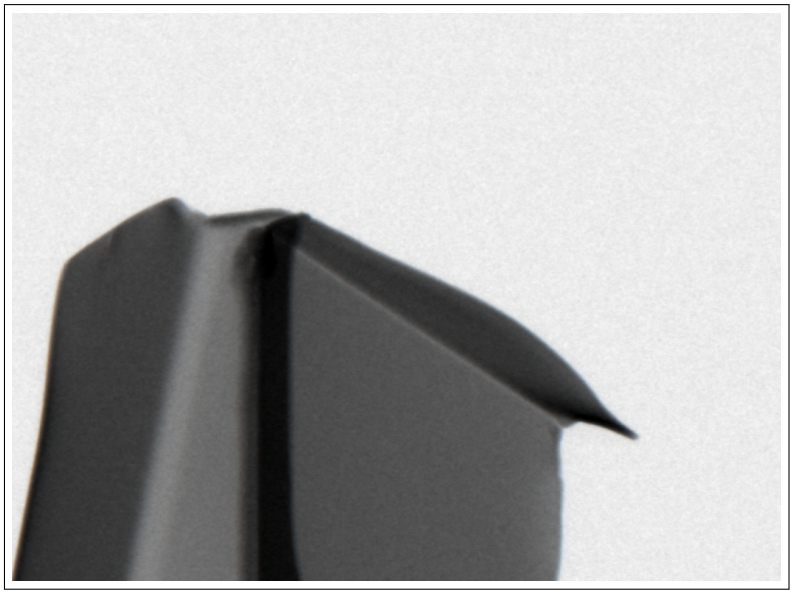

Figure 2.1: An example of an X-ray image (in the image is a folded metal sheet) from an industry-grade detector. Acquisition parameters are 160keV, 0.5mA, 0.5s. The slight noise pattern might not be visible in print, please check also the digital version of this work to examine the noise behavior.

similar example for medical CT, because the projection data are not captured on a flat-panel detector, but rather by several rows of 1D detectors (Goldman [2008]) and thus the source data are not suitable for direct visualization (they are in a form of so-called *sinograms*). But it is clear that with a significantly smaller dose the noise level would definitely increase.

The photon emission probability can be modeled (Macovski [1983], Herman [2009]) with the Poisson probability law:

$$p_{Y_\lambda}(k) = \frac{\lambda^k \cdot e^{-\lambda}}{k!} \tag{2.2}$$

where Y_λ is a discrete random variable describing the number of photons emitted in a unit period of time in the direction of the detector by a stable X-ray source that emits on average λ photons. By the central limit theorem, for large λ the behavior of Y_λ resembles normal distribution with mean λ and variance λ.

Let's denote ρ the transmittance through the material and σ the probability that a photon arriving at the detector is counted, then the number of photons detected is a sample of a Poisson random variable with parameter $\lambda\rho\sigma$. An experimental study on the detector noise properties is in Wang et al. [2008].

We have not found a good study about the noise behavior in the reconstructed data. The majority of studies only discuss the amount and variation of the noise, but not its other properties. We have made an estimation, that the behavior would be similar to Gaussian white noise, because a large number of factors with a similar random probability would affect the result (each point is reconstructed from many line integrals, all of which carry the error discussed above).

An experiment has been performed on parts of real CT scans of human bodies. We have selected parts that should be homogeneous (ie. parts filled with water) and performed a histogram analysis. An example of a fitted Gaussian curve on

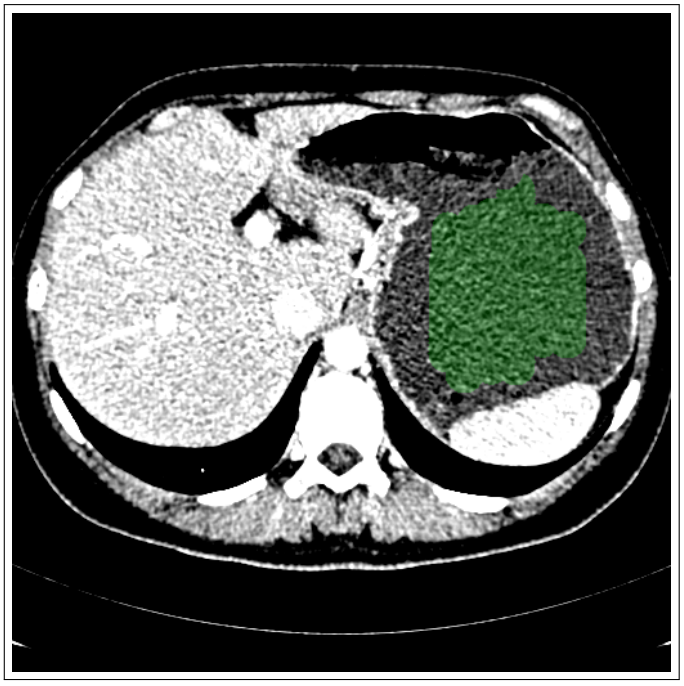

Figure 2.2: Example of a source dataset from which we estimated the behavior of CT reconstruction noise. Extracted data are painted in green.

the extracted data histogram can be seen in figure 2.3.

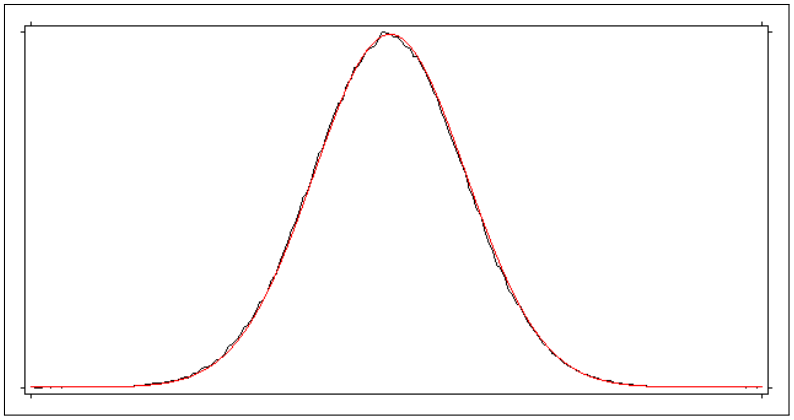

Figure 2.3: Histogram of water from figure 2.2 (black) and fitted Gaussian curve (red).

The next topic was the analysis of (in)dependence of neighboring voxels. We have performed correlation analysis on the same (supposedly homogeneous) data. The result can be seen in figure 2.4.

By analyzing only neighbors in axial slice or only along the Z axis we get results shown in figure 2.5. Due to the fact that the reconstruction is performed per axial slice, the dependence of noise between slices is very low (correlation coefficient around 0.20-0.23). But correlation within one slice is much stronger (around 0.70-0.75). Total correlation coefficient between all neighbors is around 0.5-0.6. This slightly breaks our assumption about white Gaussian noise, so we definitely cannot use a 2D image based denoising method per input slice. We have to rely on the low correlation between slices and use a 3D-based denoising

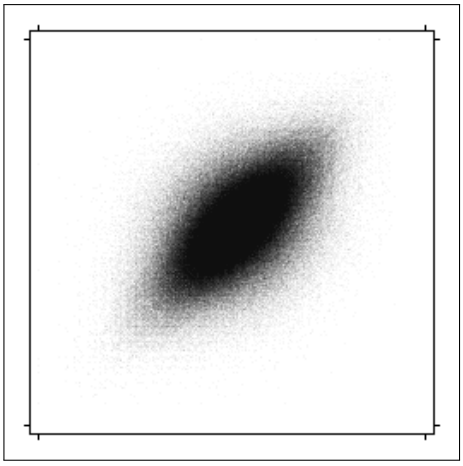

Figure 2.4: Dependence of voxel HU value (horizontal axis) on the HU value of its neighbors (vertical axis). Neighbors taken in all 3 dimensions (XYZ), correlation coefficent is 0.567.

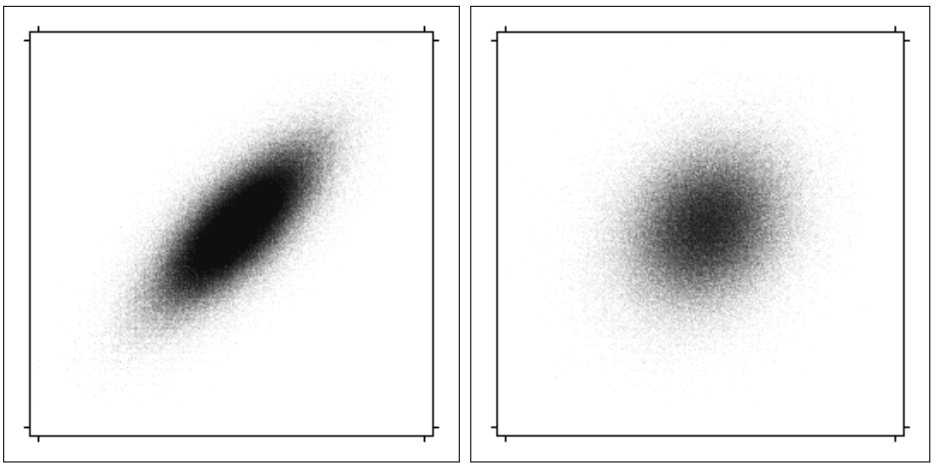

Figure 2.5: Dependence of voxel HU value (horizontal axis) on the value of its neighbors (vertical axis). (a) Neighbors taken in axial slice (XY), correlation coefficient is 0.743. (b) Neighbors taken only along Z axis, correlation coefficient is 0.216.

process.

The noise behavior was validated on 3 datasets with similar results. We have deemed such similarity as sufficient for our decision about the denoising algorithm.

2.1.3 Other error sources in low-dose CT

Other important sources of error worth mentioning causing visible degradation of medical CT images are:

- Beam hardening
- Partial volume effect
- Wrong calibration of source/detector or their instability

Beam hardening is the result of a wide X-ray spectra instead of single wavelength photons and different absorption properties of tissues in regard to X-ray photon energy. See Bazalova and Verhaegen [2007] for an example of a traditional tungsten X-ray source spectra and its simulation. A very apparent result of beam hardening on metal object in human body is shown in figure 2.6. Any X-ray tomography with non-monochromatic X-ray and non-homogeneous scanned object suffer from beam hardening, but it is really apparent only on object scans with very significant differences in material absorption. This means mostly metallic implants, teeth, dental amalgam, etc. in medical CT area.

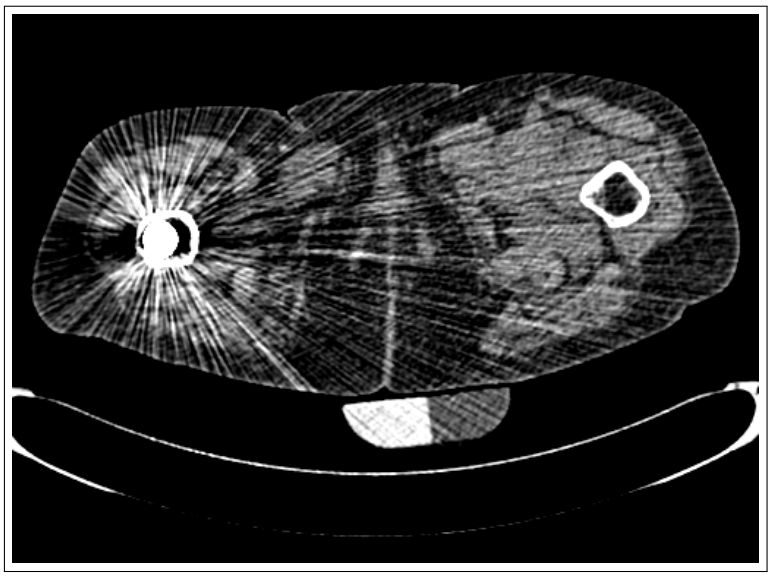

Figure 2.6: An example of a strong beam hardening artifact caused by a metal endoprosthesis in femur. Scanned part is an axial slice just below the femoral neck. Both femoral bones are visible, bone on the left side of the image contains metal endoprosthesis, bone on the right side is without any artificial material.
set of patients

Fortunately, the only significant source of beam hardening artifacts in CT enterography is the spine and the results are not so apparent, because the strength of these artifacts created by vertebrae is not so apparent and in the area of small bowel the "star beams" are already a low-frequency background with a low amplitude. Many figures in this thesis show axial slices including the spine and it can be seen that the beam hardening artifacts in the bowel area almost cannot be noticed.

Partial volume effect is caused by the finite resolution of the detectors and non-linearity during reconstruction. This can be improved basically only by increasing the detector resolution.

Wrong calibration or defects on detector/source may cause errors in transmissivity estimation. Especially defective cells on the detector may cause so called *ring artifacts*. (All our datasets were properly calibrated with a stable X-ray source, so this was not a problem for our processing.)

All three mentioned sources of errors can be improved either during acquisition or reconstruction phase (beam hardening) or by directly changing the mechanical properties of the scanner (adding filters for pre- or post-hardening the X-rays,

increasing resolution of detector, detector calibration, etc.).

Fixing these errors is beyond the scope of a single general-purpose denoising algorithm. A sophisticated approach is necessary to reduce any of these errors, ideally directly during the acquisition or reconstruction process. Especially the beam hardening creates artifacts (figure 2.6), that are not easily removable by a denoising method (sadly denoising algorithms tend to further enhance them).

We will not discuss removing these other errors more in depth as they did not cause significant problems in our case of CT enterography processing.

2.2 Denoising approaches

We have tried implementing our own iterative reconstruction algorithm based on SART iterative method (see chapter 1.2), but without significant improvements over the reconstruction algorithms supplied with CT scanner equipment. It turned out that the knowledge about the exact internal structure of the machine (including filters, detector/scintillator composition, collimator, etc.) is crucial in designing an algorithm with sufficient quality. In this regard an algorithm supplied by the manufacturer surpassed our efforts both in quality and speed. We were left only with the option of post-processing data supplied by the manufacturer's reconstruction process.

The selection of denoising algorithms applicable on volumetric data is much narrower than a selection of algorithms for 1D signals or 2D images. We refer to currently used algorithms, such as Gaussian filter, median filter (in Gallagher and Wise [1981]), total variation minimization (in Rudin and Osher [1981], Rudin et al. [1992]) and nonlocal means filter and a survey given in Buades et al. [2005] with improvements in Coupe et al. [2008].

We have not considered the BM3D denoising method (Dabov et al. [2006]) or its 3D variant BM4D (Maggioni et al. [2013]), because even though it should have slightly better denoising performance compared to NLM, its computational complexity is also very high. For our purposes the GPU variant of NLM was sufficient both in the quality and speed performance area. We have yet to find a way to speed up the BM4D to the performance level of our NLM implementation.

In this chapter we present the result of our previous work in Horáček et al. [2011]. On a side-note there has been recently published also a newer variant of NLM implementation utilizing multiple GPUs in Cuomo et al. [2014] and another algorithm in Eklund et al. [2011], which is slightly slower than our solution, although with a different algorithm implementation.

2.3 Nonlocal-means algorithm

First introduced by Buades et al. [2005] this algorithm is still nowadays considered one of the best approaches quality-wise. It has very good properties in respect to detail preservation and very successfully removes white noise. However the computational complexity is very high, especially for 3D data.

The definition is as follows:

$$NL(u)(x_i) = \sum_{x_j \in \Omega^3} w(x_i, x_j) u(x_j) \tag{2.3}$$

$$w(x_i, x_j) = \frac{1}{Z_i} e^{-\frac{||u(N_i) - u(N_j)||^2_{2,a}}{h^2}} \tag{2.4}$$

where u is the original noisy image, Ω^3 is the definition range of the image, w is the weight computed from the similarity of the local neighborhoods of two voxels and N_i, N_j are local neighborhoods around given voxels, h is a filtering parameter and Z_i is a normalization constant.

This can be explained as follows: Each voxel is reconstructed by the weighted average of the most similar voxels in its vicinity. The similarity of two voxels is computed from the L2 norm of their neighborhoods - voxel-wise.

It has been proven in Buades et al. [2005] that NL-Means is a very efficient algorithm (quality-wise) and performs an optimal denoising. But the computational complexity ($O(n * m * k)$, where n = number of voxels, m = size of the search space and k = size of local neighborhoods for computing similarity, all of which are 3-dimensional parts of the dataset) is nowadays too large for a daily medical practice.

2.3.1 NL Means Optimizations

NLM has been tried on GPU, mostly for photos and other 2D or 1D datasets (e.g. Kharlamov and Podlozhnyuk [2007]). The optimization was using block approach without overlaps, which brings strong block artifacts and the whole implementation is done only for 2D images.

Good optimizations for 3D data are given in Coupe et al. [2008]. The original aim was denoising of the MRI datasets of head and brain, but they are good for CT images as well. The basis is a selection of relevant voxels based on local mean and variance values before the L2 norm is computed.

Combining this voxel selection with block-wise approach from Buades et al. [2005] brings a very significant speedup, the authors claim as much as 40x to 66x. No implementation details are given though. We did not manage to achieve such performance without drastically reducing the quality of the result.

As a result of these optimizations, the algorithm is able to run in merely dozens of minutes instead of hours. Our implementation has achieved approximately 20-30 minutes on a quad core i7 processor for a volume of size 512x512x548.

2.4 OpenCL Capable GPU

The current GPUs supporting OpenCL interface (Khronos OpenCL Working Group [2009]) are designed to efficiently process large tasks that are parallelizable. The sheer computational power is at least one order higher than a commodity-grade CPU. However, efficient use needs some practice and optimizations that are unnecessary (and some also inefficient) on CPU (NVIDIA Corporation [2008b], NVIDIA Corporation [2009]).

The main theme of GPU programming is a so called SIMT (Single Instruction - Multiple Threads) execution model. This acronym was inspired by a similar

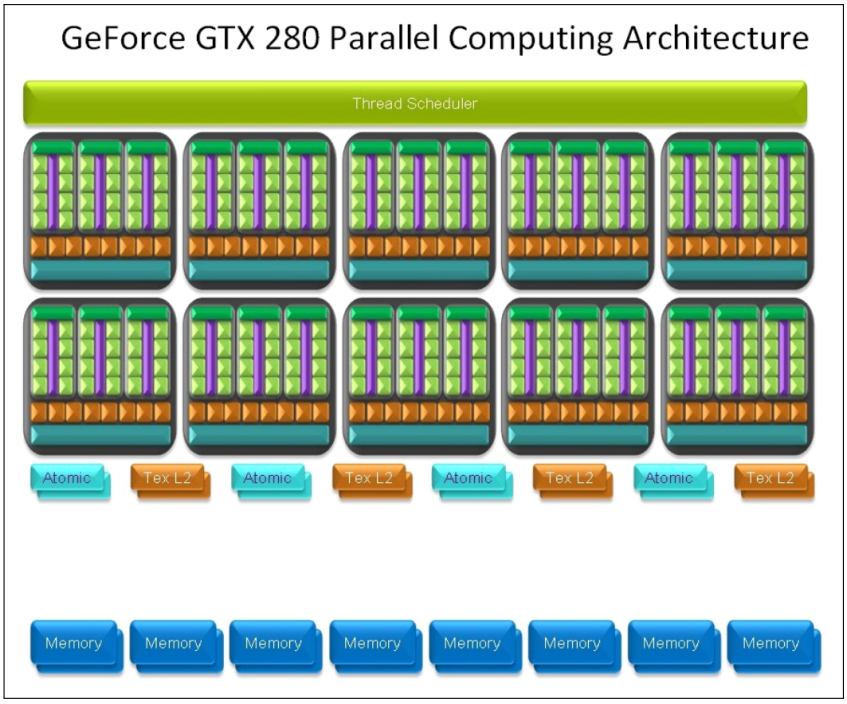

Figure 2.7: Architecture of an nVidia GeForce GTX 280 processor. Image from NVIDIA Corporation [2008a].

acronym in parallel computing - SIMD (Single Instruction - Multiple Data). It was used by NVIDIA in the presentation of the Tesla scalable unified graphics and parallel computing architecture (Lindholm et al. [2008]).

Although many different manufactures produce OpenCL (or a proprietary alternative Cuda) capable hardware (the most known being NVIDIA Corporation, Intel Corporation and Advanced Micro Devices, Inc.) and each manufacturer has several generations of such hardware with different capabilities, the principle stays always the same. One chip (usually called GPU, for example nVidia GeForce GTX 280 in fig. 2.7) contains several so called streaming multiprocessors (SM, shown in fig. 2.8). Each multiprocessor contains individual processing cores - usually 8 or more.

Multiprocessors can perform different tasks, the shared part of the chip between SMs is not very large (and differs between GPU generations). The magic happens inside these SMs, on the individual cores.

All cores in a single multiprocessor process the same instruction on different data, because they share the instruction cache, scheduler and other parts that are usually devoted to individual cores in CPUs. Also the size of caches is significantly reduced compared to a CPU, leading to much higher latency when accessing the GPU main memory. On the other hand, a much larger part of the chip is devoted to ALUs and a special HW support is present to provide almost no-cost switching of threads on the cores. So the SM can switch a thread that is waiting for data from main memory with another thread that can perform computation.

The most effective way of utilizing the GPU is by running literally thousands of threads that perform the same instructions (defined in a so-called *kernel*) on different data. Also specific techniques such as memory coalescing, manual

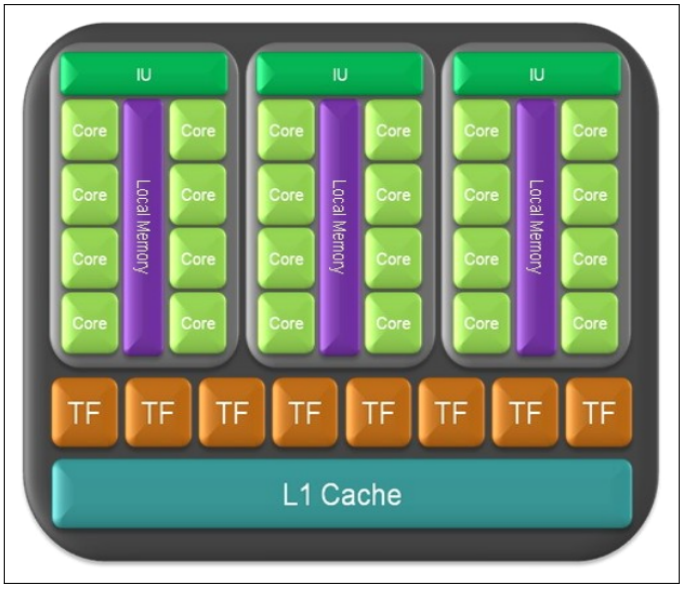

Figure 2.8: A detail of a single TPC (Thread Processing Cluster) consisting of 3 SMs (Streaming Multiprocessors). Image from NVIDIA Corporation [2008a].

caching in a fast but small local shared memory and other latency hiding methods are required when running memory-intensive algorithms.

Practically, the threads are subdivided into blocks of threads, which are grouped into a grid. A kernel is run on a grid, blocks are then assigned automatically to SM for processing. They may be executed in random order, so special synchronization techniques are necessary in case some operations need to be serialized. In our case we are using only serialization per block, by issuing special serialization instructions that block the computation of threads untill the whole block is finished. Blocks themselves are independent and may be run in any order.

Specific techniques necessary for a proper GPU utilization vary slightly between individual hardware generations. That may mean the dimensionality of blocks and grids, their sizes, available synchronization techniques, etc. Our algorithm was originally designed on the first generation of Cuda-capable GPUs, so forward compatibility is not a problem.

For a thorough discussion of individual hardware specifics, please refer to the manufacturer's documentation. For example: searching for OpenCL or Cuda Tuning guide by NVIDIA, AMD, Intel or Apple might show up to date information (note thought that Cuda is a proprietary NVIDIA technology). An example is in: NVIDIA Corporation [2016].

2.4.1 Basic HW Implementation of Optimized Version

After implementing optimized NL means for the OpenCL interface, we have found out that on a GeForce GTX 275 it is only about 6 times faster than on a single 2.6GHz CPU. However, current processors usually have more than one core and thus are already able to perform parallel computations. Running this version on a Core i7 with 4 cores and hyper-threading, the speed difference was even after

careful optimization of the GPU version negligible, even though a GeForce was running many more threads concurrently.

This is caused by a very slow memory access and no local cache on this version of GeForce. About 90% of the computational time was spent on two memory reads. Also the process of choosing which voxels will be computed breaks the continuity of the thread warps and slows down the computation instead of speeding it up. This algorithm is thus unsuitable for direct implementation on GPU and needs to be changed.

2.4.2 HW Implementation of the Original NL Means

We tried re-implementing the original NL-means, this time to maximally utilize the capabilities of the GPU with minimized branching and global memory access. Meaning pre-loading the data to the fast on-chip memory and no voxel selection.

The parameters of NL-means filter for best quality are taken from Coupe et al. [2008]: local neighborhood radius 2 voxels, search radius 4 or 5 voxels (only small difference in quality) and automatic smoothing parameter computed from pseudo-residuals.

Thus, the memory consumption of one voxel reconstruction with a search radius 4 would be: number of voxels $= (2 \cdot (4 + 2) + 1)^3 = 2197$, so source data size $= 2197 \cdot 4$ bytes per float $= 8788$ bytes. We also add temporary memory for weights $= (2 \cdot 4 + 1)^3 = 729$ floats $= 2916$ bytes and additional memory for summing weights $= 360$ bytes.

This fits completely into the local memory even on the first generation HW. Search radius of 5 voxels would not fit into the on-chip memory on some cards, but the quality difference between 4 and 5 voxels in means of signal to noise ratio is shown in Coupe et al. [2008] to be small enough.

The algorithm itself:

1. Each thread reads the first $2 \cdot (4 + 2) + 1$ voxels in the Z direction from global memory to local memory (Figure 2.9(a, b))
2. For each relevant voxel, compute the weight function with central voxel (Figure 2.10(a))
3. Compute the sum of weights
4. Compute the sum of weighted values for each relevant voxel
5. Normalize with weight sum
6. Store result into global memory (each threads writes to a unique position)
7. In each thread move all voxels in local memory by 1 and read one new voxel (Figure 2.10(b))
8. Continue with step 2

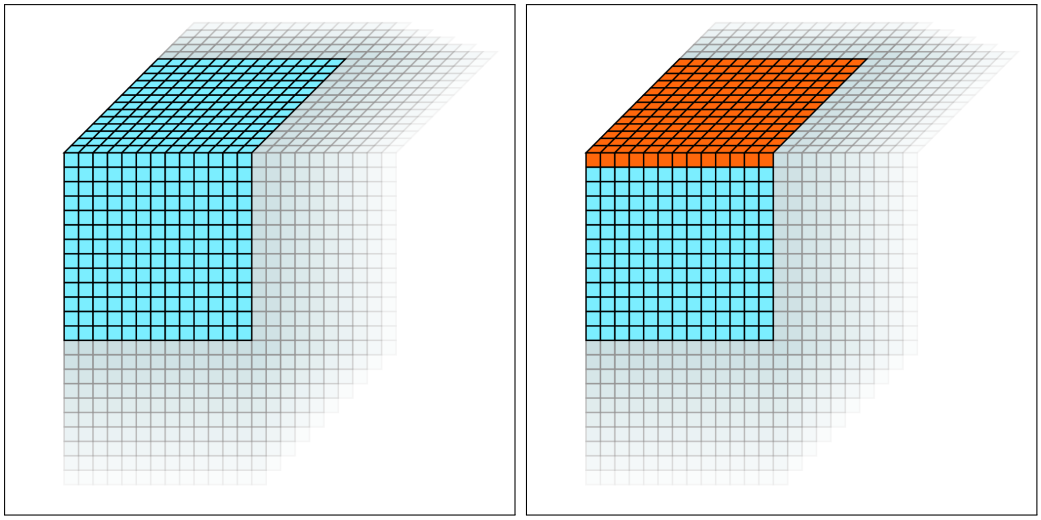

Figure 2.9: Blue voxels are the search space, orange voxels are highlighted for explanation, currently processed voxel is in the middle of the blue voxels: (left) Volume needed for one voxel. (right) One thread is executed for each column.

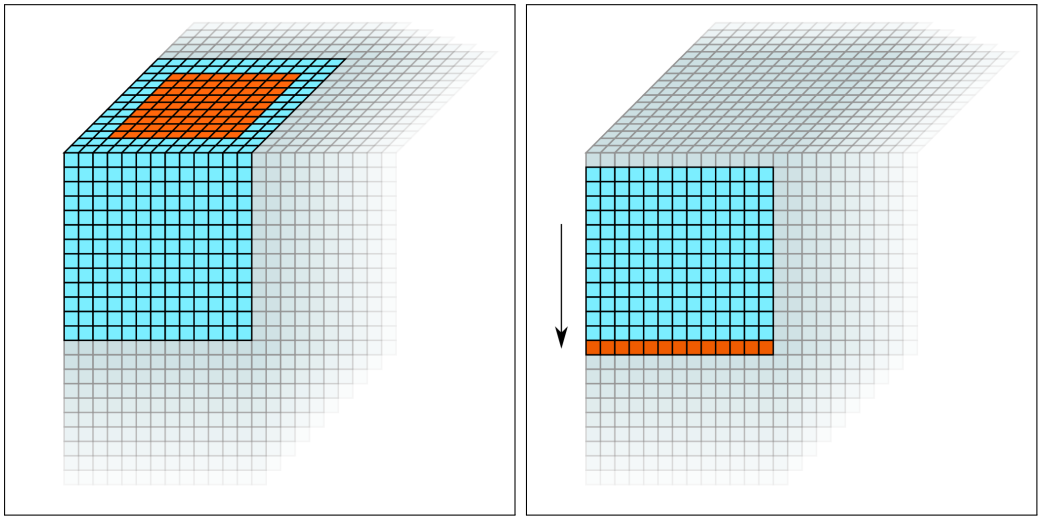

Figure 2.10: (left) Threads actually computing weights and sum. (right) Loading next slice.

2.4.3 New HW Implementation of the Optimized Version

We also tried optimized selection of computed voxels by using this presented algorithm with good utilization of the local memory. However, the results were much worse than on the CPU. The algorithm has been running about 2x slower than unoptimized version in 2.4.2. It was running about the same speed as the unoptimized version only with drastic reduction in quality. The speed was strongly dependent on the actual data (how much of the data was discarded).

Algorithm	Processor	Threads	Time
NL-Means	i7 3.07GHz	8	5:38:15
Block NLM	i7 3.07GHz	8	0:26:35
NL-Means	C2Q 2.4GHz	4	> 10 hours
Block NLM	C2Q 2.4GHz	4	0:55:57
NL-Means	GF 8800GT	13^2	0:13:41
NL-Means	GF 275GTX	13^2	0:06:44
NL-Means	GF 460GTX	13^2	0:04:22
NL-Means	GF 980GTX	13^2	0:00:59

Figure 2.11: Measured on dataset of size 512x512x548. Thread count on GPU is the number of threads per block - multiple blocks may be running on the whole GPU.

2.5 Results

2.5.1 Performance

The times measured for given algorithms are shown in figure 2.11. As you can see, the simple and computationally very expensive algorithm runs much faster on current GPU than the most optimized CPU version on 8-threads on a good current CPU. It should be stressed, that the results on CPU are already parallelized and the comparison is done with the *full CPU power* and not a single-threaded version.

The application performance has been validated with a profiler. The *instruction throughput* on a GT200 was 0.921813. The *retired instruction per cycle* on a GF104 was 1.86737.

2.6 Quality

We have performed denoising on 34 datasets with the same settings for the NLM filter and all results were visibly improved. We have not found any wrongly removed feature, everything was visible and much clearer than in the original data. An example comparison of different results is shown: thin slices without processing, thick slices without processing and thin slices denoised.

Thin slices in figure 2.12 in our example have a thickness of 0.5mm, so the voxels are cubic and ideal for segmentation, because all directions in the dataset have the same scale. However thin slices have also the noisiest appearance.

Thick slices in figure 2.13 have a much better SNR, but significantly lack the resolution in the orthogonal direction to the slice, so many small details might be lost. In this particular example, there is a resolution reduction of 1:5. See figure 2.15 to see the results of thick slice reconstruction.

Denoised thin slices in figure 2.14 are the best candidate for a successful segmentation. They excel both in spatial resolution and feature visibility.

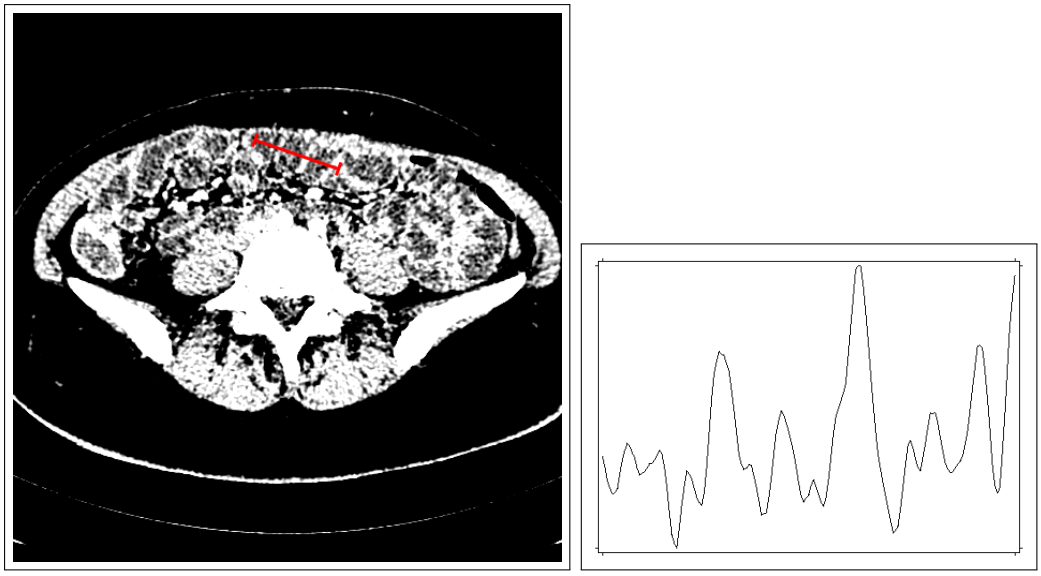

Figure 2.12: Axial plane of thin slice reconstruction. Red line in the data intersecting 3 intestinal walls is shown as an intensity profile.

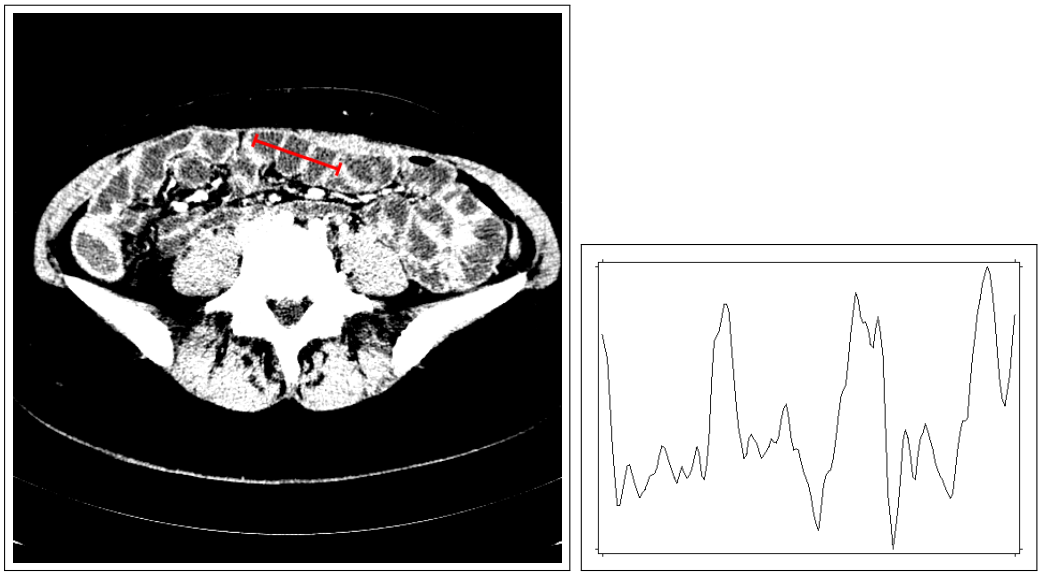

Figure 2.13: Axial plane of thick slice reconstruction. Red line in the data intersecting 3 intestinal walls is shown as an intensity profile.

2.7 Conclusion

GPU version on an OpenCL-enabled graphics card allows denoising of the whole patient dataset in merely minutes instead of hours and does not bring any artifacts created with aggressive optimizations needed for CPU. Thus we consider this a good approach for processing such intensive problems.

We have also validated that the original brute-force version without voxel selection is faster than the CPU optimized one. The explanation of this behavior is the GPU architecture briefly described in 2.4. Each multiprocessor performs a single instruction for all its threads. In case of the brute-force processing all

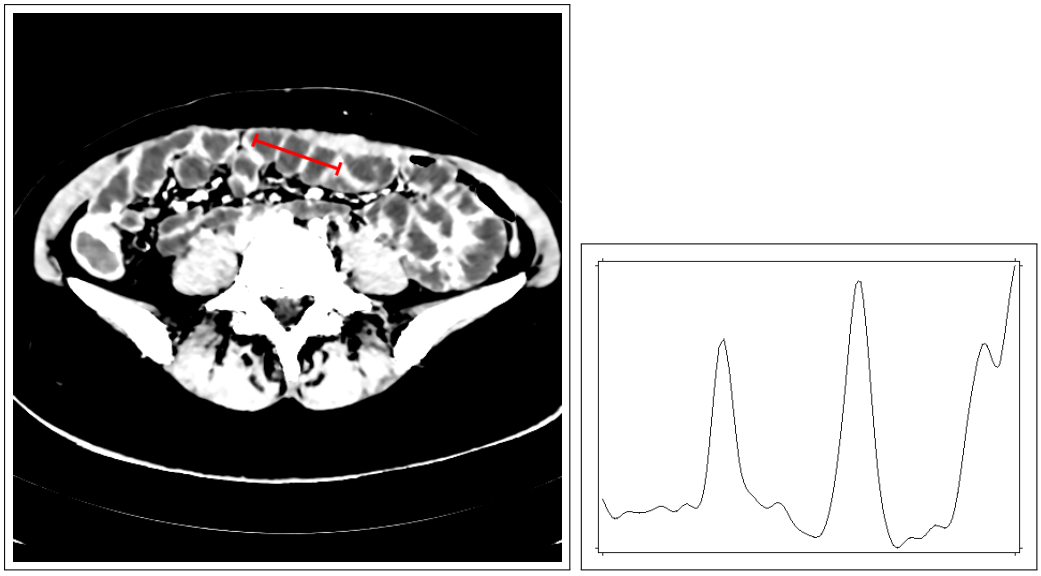

Figure 2.14: Axial plane of thin slice reconstruction after denoising with NLM algorithm. Red line in the data intersecting 3 intestinal walls is shown as an intensity profile.

instructions always align and the only problem is the memory latency, which can be hidden by thread switching (by lowering the occupancy of registers), memory coalescing (by reading subsequent bytes from main memory) and simple manual caching in local memory shared over all threads.

The algorithm optimized for CPUs with relevant voxel selection has a problem on GPU with input-data-dependent branching and thus *increasing* the number of instructions (instead of *decreasing*). This is caused by broken thread coherency. The problem is so apparent that a GPU with hundreds of ALUs gives us no real advantage over a CPU with 4 cores.

A full evaluation of denoising quality is not given in this thesis, it can be found in Coupe et al. [2008]. We have presented a brief effect on available CT enterography data.

The profiling has shown that we have successfully eliminated the memory bandwidth problem, on a GT200 architecture is room only for circa 8% improvement onto the peak theoretical performance.

However, on the tested GF104 the performance is about half of the theoretical peak. This is caused by heavy usage of the on-chip memory - two warps do not fit on a single SM and this prevents the HW from executing 4 instructions per cycle. We have not found a way to reach this limit without drastically increasing the bandwidth dependency or quality of denoising.

Future improvements may be in implementing the block-wise approach for GPU. Reconstruction by blocks was proposed by the original author of NLM in Buades et al. [2005]. It performs similarly to per-voxel NLM, but does not reconstruct a single voxel per step, but rather overlapping blocks over which the L2 norm is computed (in equation 2.4). It has superior speed, as the number of computed L2 norms per voxel is significantly lower, but it comes at the cost of quality.

Another improvement may be overcoming the one-warp-per-multiprocessor

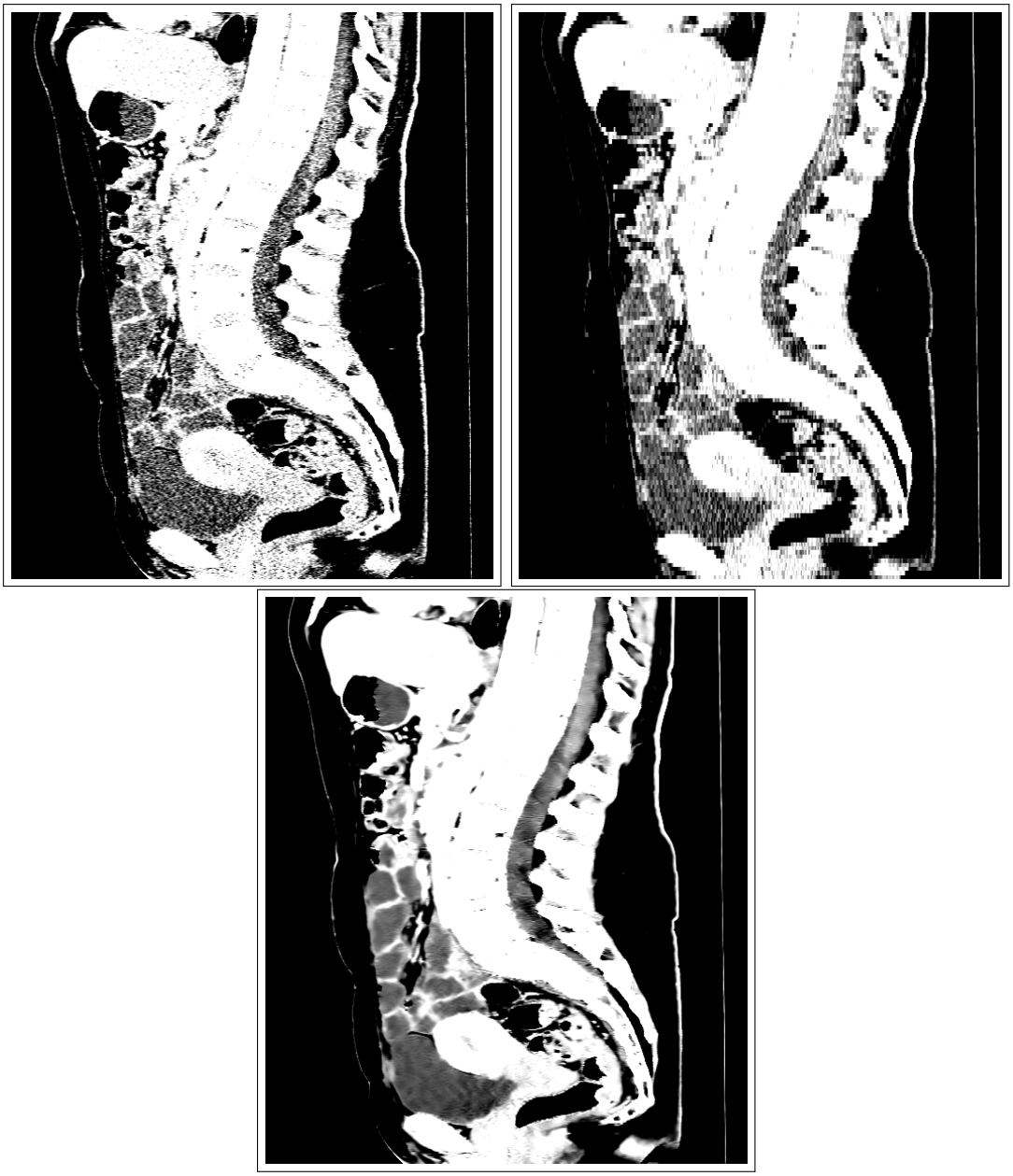

Figure 2.15: Sagittal plane of all three compared datasets - thin, thick and denoised slices. It is apparent that thick axial slices have a significant deficiency in orthogonal resolution.

limit when running on a GF104 and newer architectures or we could try to implement a GPU version of Darbon et al. [2008], which has even lower computational complexity than the original NLM.

We have made a Cuda version of the algorithm implementation available in an open-source library CUGIP (Cuda Generic Image Processing - Kolomazník [2013-2016]).

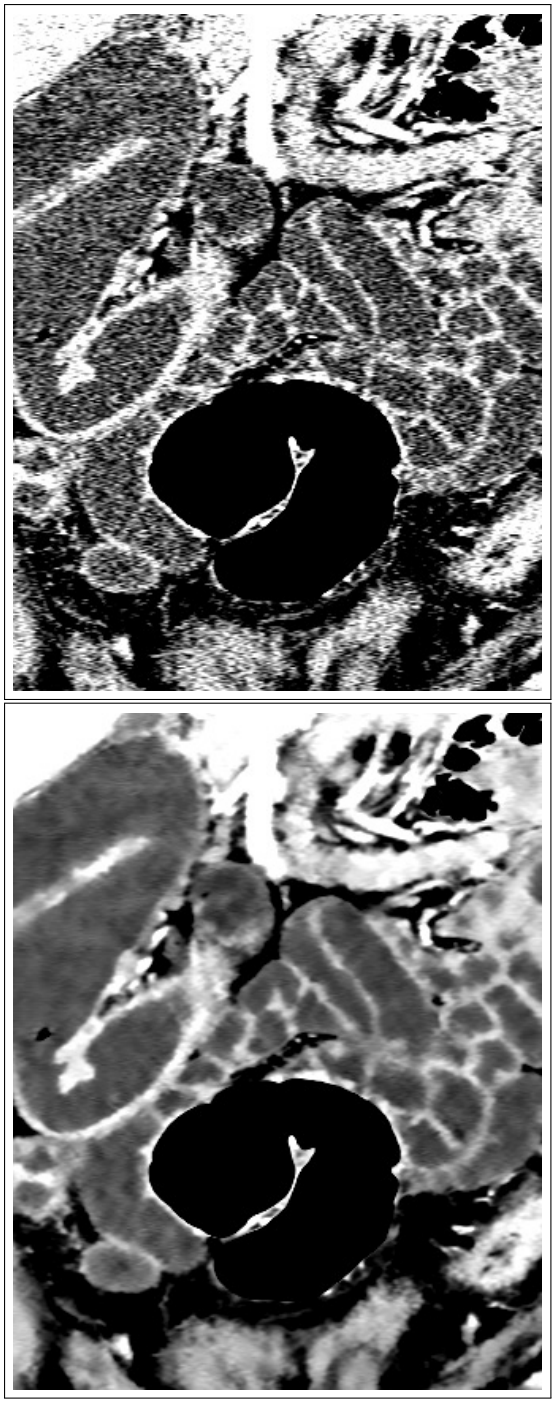

Figure 2.16: Close up example of denoised result: (a) Original data. (b) Denoised data with OpenCL GPU implementation of the original algorithm.

Chapter 3

Processing CT enterography using watershed segmentation

3.1 Introduction

This chapter is the first to be fully focused on working directly with CTE data. We present here the first part of the segmentation process from Horacek et al. [2015].

The processing system is based on a subdivision of the volume by a watershed transformation into smaller regions and then classification of these regions to obtain the main volume of the small intestine. A discussion of suitable region descriptors is given and the result are probability maps of intestinal lumen and wall - an example is shown in figure 3.1.

3.1.1 Other work

A significant amount of work has been done on the segmentation of colon for virtual colonoscopy and some work also on the segmentation of small intestine. However, a robust technique suitable for its tracking has not been identified yet.

We have surveyed the existing techniques in Fidler et al. [2009], Holmes et al. [2010], Näppi et al. [2010] and Näppi et al. [2012] and tried various threshold-based and voxel-based region growing algorithms without much success in tracking the intestine reliably. These methods had very frequent leaks into the neighboring folds and the colon that could not be practically corrected, even manually.

The reason might be a slightly different scanning procedure, an excessive amount of noise or a different contrast method. None of the denoising methods mentioned in chapter 2 were sufficient to make a voxel-based growing usable. The 3D region growing procedure on voxels is very sensitive even to a single discontinuity in the scanned volume (due to noise, partial volume artifact or imperfect distension with neutral contrast agent, see table 3.1). The healthy intestinal wall is around $1 - 2mm$ thick (Herlinger et al. [2001]), which may lead to *holes* in the data in case of imperfect contrast agent saturation and/or diagonal direction and close proximity of several folds.

A segmentation method directly for CT enterography based on region growing is given in Näppi et al. [2010] and its improved version in Näppi et al. [2012]. In our case it did not work for us in the general case due to very frequent leaking.

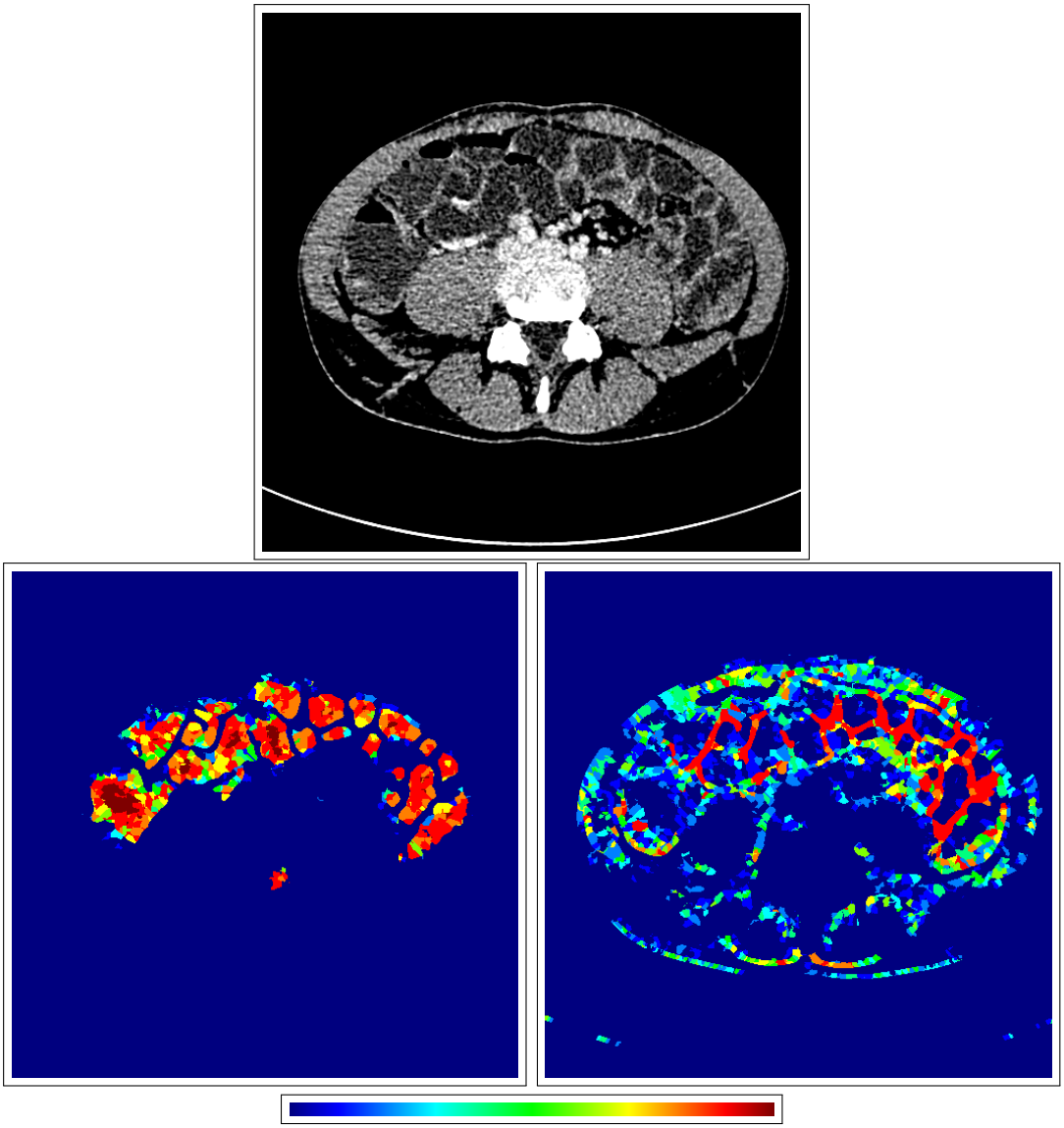

Figure 3.1: Example of result from our watershed processing pipeline. Individual watershed regions have a constant probability, which is color coded with a so-called *jet* ramp LUT. (a) Source slice. (b) Lumen probability. (c) Intestinal wall probability. (d) *Jet* ramp of values from 0 to 1.

However, a wall inflammation analysis method described there seems promising.

Another interesting method based on tracking the mesenteric vasculature is mentioned in Zhang et al. [2013]. This method performs very well in segmenting the overall small bowel volume, but does not perform any path tracking. Its output is only the rough bulk of the organ.

There are also other methods for diagnosing small intestine and Crohn's disease, namely using MRI, like MR enterography (Fidler et al. [2009], Gourtsoyiannis et al. [2006] and a method similar to ours in Holmes et al. [2010]). However, our focus has been specifically the result of a CTE scan.

3.1.2 Symbols used

We will be using the following symbols throughout the rest of this chapter:

- s ... one whole watershed region
- A ... a class of regions (i.e. lumen, wall, ...)
- $P_A(s)$ ... a probability that the region s belongs to A
- $N(s)$ ... a set of regions directly touching the region s
- $K_k(s)$ ... k nearest neighbors of s in a feature space
- v ... a voxel in a 3D dataset
- $\tau_{decision}$ ... a threshold for an algorithmic decision, all thresholds in this chapter must be optimized to input data acquisition parameters

3.2 Input data filtering

3.2.1 Denoising

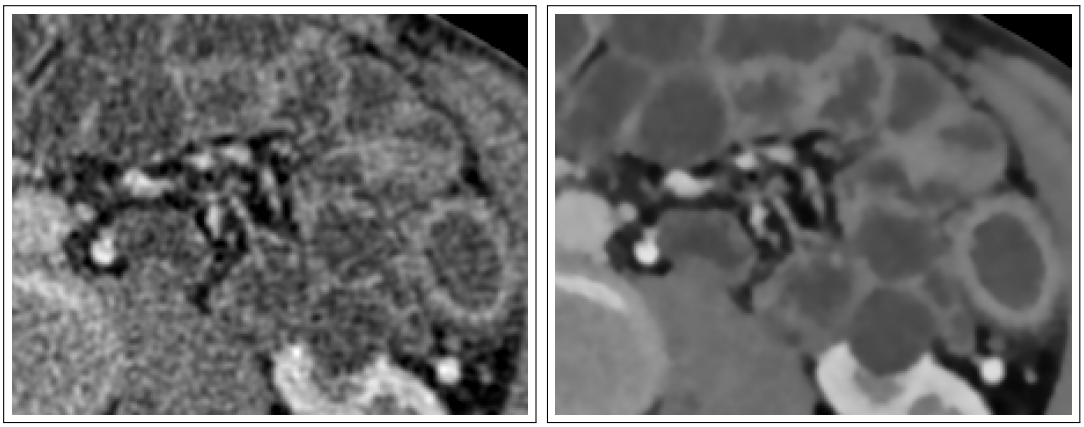

Figure 3.2: Original and denoised CT enterography axial slice example. An inflammation is visible in the right part of the image.

We need roughly cubic voxels with the size of around $0.5^3 mm$ or smaller to prevent partial volume artifacts on the relatively thin intestinal wall (healthy wall is around $1 - 2mm$ thick). CT enterography data with thin slices (0.5-0.75mm) unfortunately have a large amount of noise. Reconstruction details and noise origin are discussed in chapter 2.1.2. Thick slices are not an option, slices thicker than 2mm significantly reduce any possibility to diagnose the intestinal wall parallel to the slice.

There are many methods for denoising, starting with a Gaussian or a box filter and ending with anisotropic filters and NLM (non-local means). We have tried several methods and the best results were with the NLM (Buades et al. [2005]Coupe et al. [2008]Cuomo et al. [2014]).

NLM removes noise inside the lumen and at the same time keeps planar structures such as thin walls intact. The vague metric called here as the *best* denoising

	Wall		Lumen	
	Mean	Std. deviation	Mean	Std. deviation
Original	131.09	59.83	49.26	44.69
Denoised	117.64	34.33	54.16	15.92

Table 3.1: Standard deviation on intestinal lumen and intestinal wall before and after denoising. This was measured on the same voxels and datasets as the histograms on figure 3.3.

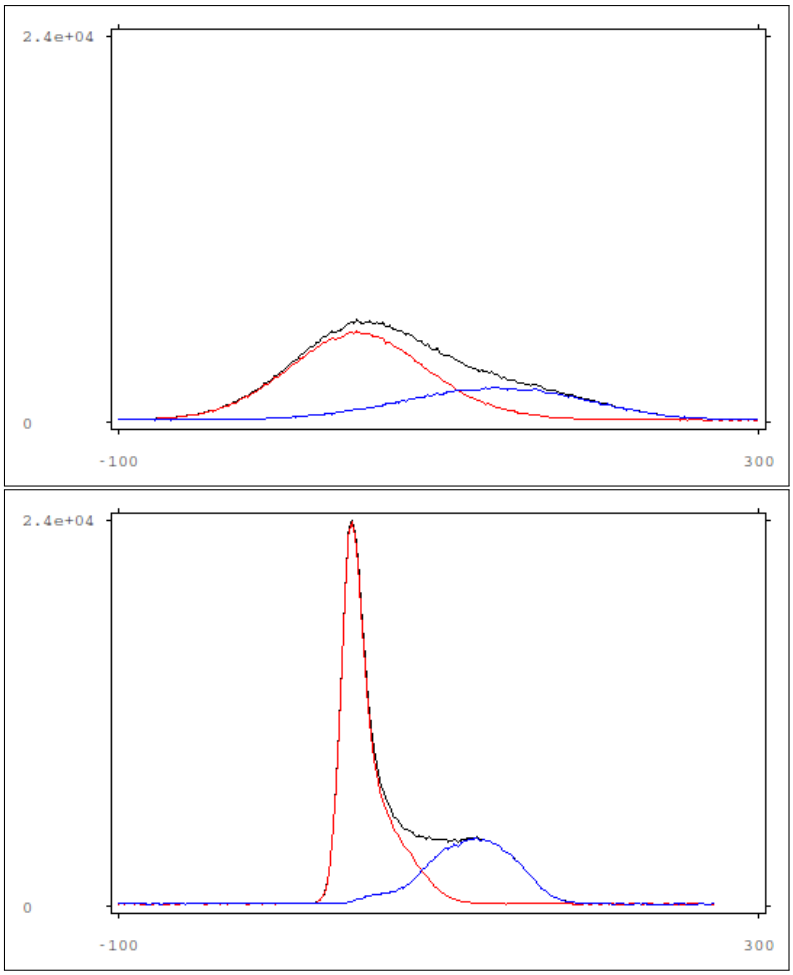

Figure 3.3: Histograms of samples from (a) original and (b) denoised data from a well distended and well contrast-saturated dataset. The same part (voxel-wise) of dataset is used - a part of manually segmented intestine (lumen+wall), no other body parts are included. Red line is lumen, blue line is wall, black line is the sum of all marked voxels.

method is based on the optimality of the NLM Buades et al. [2005] and visual evaluation of the presence and the visibility of anatomical structures such as intestinal walls and villi. Another metric was the amount of leaks in subsequent processing (for example region growing), with NLM minimizing such events compared to other algorithms (Gauss and adaptive Gauss filtering, diffusion-based methods, median filtering, morphometry, etc.).

Here we have used our earlier implementation from Horáček et al. [2011] (described in chapter 2). An example of the denoised data is shown in fig. 3.2.

As can be seen in table 3.1, the denoising process significantly reduces the standard deviation of lumen.

A histogram comparison from manually segmented part of the intestine is shown in figure 3.3. Denoised data in fig. 3.3(b) have two apparent parts in the full histogram (black), lumen with a narrow high peak of lower-intensity HU values and wall with much wider higher-intensity HU values. Those two peaks cannot be easily identified in full histogram (black) on figure 3.3(a).

The difference in mean values of both lumen and wall before and after the denoising step in table 3.1 can be explained as a result of a very strong denoising filtering needed due to large amounts of noise present. Thus values in lumen are slightly increased because they are completely enveloped with higher-valued voxels from the contrast-enhanced walls and even though the weight of voxels from walls in NLM denoising is very low, it is slightly larger than zero. The same applies to intestinal walls. We have performed a test on the whole dataset to be sure the overall mean value of the dataset stays the same after denoising and the difference was negligible (less than 1 HU, namely 0.15 HU on the whole dataset from figure 3.3).

We have also sometimes used Gaussian low-pass filter. Although not for the primary denoising of the source data. It has been used to smoothen intermediate data between algorithms, such as the search for average gradient in some area. Each usage in the following chapters is commented to show the reason for such filter.

3.2.2 Watershed transformation

The watershed transformation is directly usable only on a very narrow set of problems and in the case of real-world images, it usually leads to an over-segmentation (a practical application of watersheds can be seen for example in Peter et al. [2008]). However, it can be very useful in the preprocessing as a tool for reducing complexity of other algorithms or to provide some further information for the next segmentation step.

The goals for our watershed transformation preprocessing is to provide a subdivision into smaller regions that would ideally meet these demands:

1. *lumen*: cover large part of the lumen and regions neighboring with intestinal wall should have precise borders usable for final segmentation

2. *wall*: one region would span across the whole width of the wall to cover all statistically significant voxels

3. regions themselves would have statistically significant values and those would be usable for classification

To satisfy these conditions we used the following schema:

1. Use a **denoised** dataset with (as much as possible) uniformly cubic voxels as the input (see figure 3.4 for a comparison between original and denoised dataset used as source for the Sobel magnitude operator)

2. Compute a 3D **Sobel** magnitude operator as a an edge detector

$$SOB_{3D} = \sqrt{SOB_X^2 + SOB_Y^2 + SOB_Z^2} \tag{3.1}$$

3. Compute **watersheds** on the Sobel magnitude operator output.

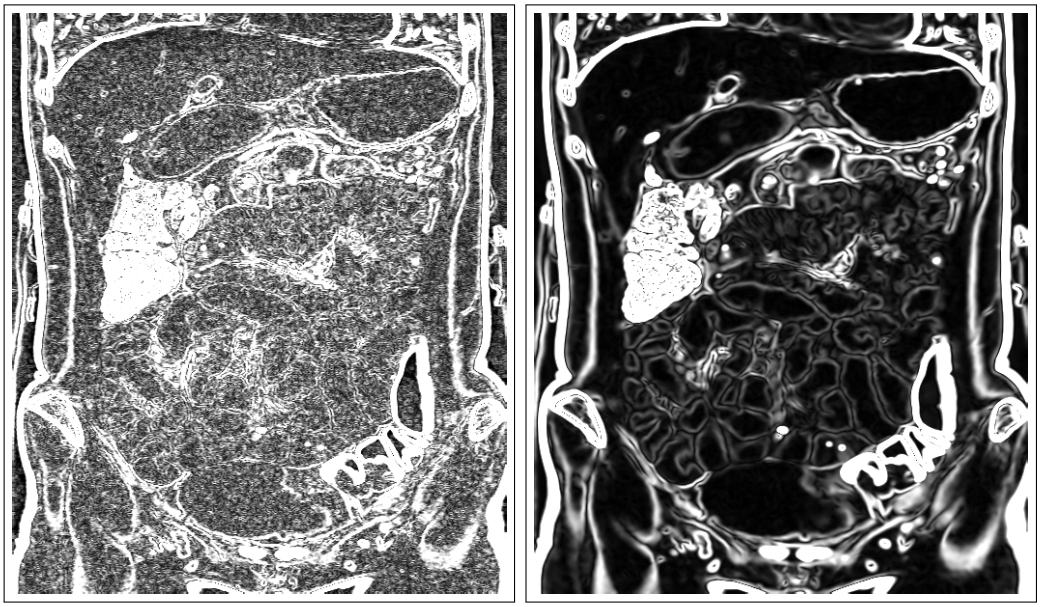

Figure 3.4: 3D Sobel operator magnitude. (a) Computed over original data. (b) Computed over denoised data.

To emphasize the importance of the denoise step we show two results of a manual segmentation on watershed-preprocessed data in fig. 3.5. Both datasets were segmented by manually selecting individual regions representing the lumen as close as possible. You can see that the watershed region borders on the denoised dataset match the true lumen/wall borders much more precisely. The shown denoised dataset was then used as one of the training sets for the automatic segmentation process.

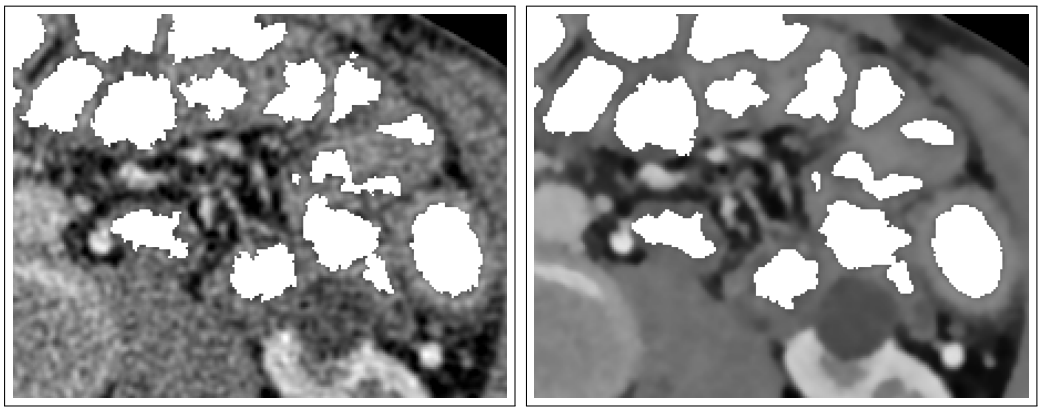

Figure 3.5: Manual segmentation of the lumen by selecting watershed regions computed on the (a) original and (b) denoised data.

The Sobel gradient estimator as a base for the watershed transformation was used for its properties of good ability to locate image/volume value edges and its slight implicit low-pass filtering in the perpendicular direction smoothens noise residuals. With it we can create watershed borders at both sides of the intestinal wall due to its increased HU values by a positive contrast agent. In an optimal case (without any noise, with good distention and contrast saturation) that would be enough to segment the lumen completely in one region and a healthy wall should have a thicknes of approximately one region as well.

For the watershed transformation we have used watershed regions with implicit borders, because in contrary to 2D images explicit borders in volume data would take up too much voxels and significantly reduce the size of watershed regions for further analysis.

3.3 Probabilities

For the final processing of the watershed-processed data we used probabilities of belonging to lumen or wall classes based on kNN sampling of a manually segmented training set.

3.3.1 Training set

The training data source was selected as a representative part of one dataset that included all relevant parts of the body - well-distended small intestine, bones, other internal organs, air, fat, muscles, etc.

Manual marking of watershed regions was performed, indicating one of three classes for each region - *lumen*, *wall* and *other*. Manual segmentation of one patient takes over 10 hours of work. The advantage is a precise segmentation even in unclear areas, disadvantage is obviously time. The representative part is around $\frac{1}{16}$ of the dataset and its manual segmentation took around 3 hours.

A set of features was computed and used as a training set for the probability estimation.

You can see a part of the training dataset it in fig. 3.5, second image. It is important to note that the training dataset must be scanned with the same preparation procedure and scanner settings as the datasets we want to segment automatically.

The whole patient dataset used for training was then included in the evaluation with other datasets, but is explicitly marked as such to differentiate it from the rest of the results.

3.3.2 Features selection

A number of various features comes into mind when we want to differentiate regions into several classes. Some of them are computed as a property of the whole region (such as standard deviation), some are averaged values computed per voxel. Each of them should express some individual property of the watershed region. We have tried the following set of them:

Mean value:

$$\overline{s} = \frac{1}{|s|} \cdot \sum_{v \in s} v \tag{3.2}$$

This is obviously the most important feature, because visually we recognize the lumen and wall as having different intensity (due to neutral and positive contrast agents used, see Paulsen et al. [2006]). Using the mean value was without question, so we used it also for the visualization for the following features (figures 3.6 and further).

Standard deviation:

$$\sigma_s = \sqrt{\frac{1}{|s| - 1} \cdot \sum_{v \in s} (v - \overline{s})^2} \tag{3.3}$$

The basic premise for this feature is that denoised lumen should have significantly lower standard deviation than non-homogeneous wall. See figure 3.6.

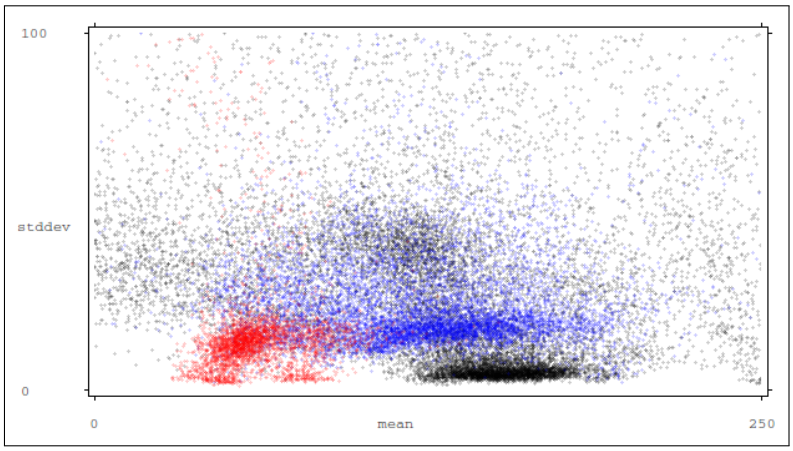

Figure 3.6: Mean value vs standard deviation per region on denoised data. Red is lumen, blue wall, black other regions.

Average multi-scale objectness filter based on Hessian matrix eigenvalues (Antiga [2007]) - this filter can enhance planar or tubular objects in the dataset and thus very easily highlight the intestinal walls. We will further refer to this filter as *objectness*. The definition for the generalized filter is as follows: let's denote as $\lambda_1, \ldots, \lambda_N$ the eigenvalues of a $N \times N$ Hessian matrix such that $\lambda_1 \leq \cdots \leq \lambda_N$. When we want to enhance M dimensional objects in N dimensional dataset, we define the following values:

$$R_A = \frac{|\lambda_{M+1}|}{\prod_{i=M+2}^{N} |\lambda_i|^{\frac{1}{N-M-1}}} \tag{3.4}$$

$$R_B = \frac{|\lambda_M|}{\prod_{i=M+1}^{N} |\lambda_i|^{\frac{1}{N-M}}} \tag{3.5}$$

$$S = \sqrt{\sum_{j=1}^{N} \lambda_j^2} \tag{3.6}$$

For the special case $M = N - 1$ we let $R_A \to \infty$ and in case $M = 0$ we let $R_B = 0$.

The objectness value is then computed from Hessian eigenvalues computed over a σ-Gaussian smoothed source data, thus detecting areas with changing intensity of scale σ:

$$O(\lambda)_\sigma = \begin{cases} (1 - e^{-\frac{R_A^2}{2\alpha^2}}) \cdot e^{-\frac{R_B^2}{2\beta^2}} \cdot (1 - e^{-\frac{S^2}{2\gamma^2}}) & if \lambda_j < 0 for M < j \leq N \\ 0 & otherwise \end{cases} \tag{3.7}$$

with α, β, γ being user-defined weights.

We compute the objectness for planar surfaces inside volumetric data, i.e. $M = 2$, $N = 3$, final objectness computed as $max(O(\lambda)_{1.5}, O(\lambda)_2$ and user-defined parameters $\alpha = 0.5$, $\beta = 1.0$, $\gamma = 5.0$. I.e. we are searching inside a 3D volume for 2D planes of thickness 3-5 voxels.

Objectness is a property of one voxel, the final region feature value is computed as an average over the region. See figure 3.7.

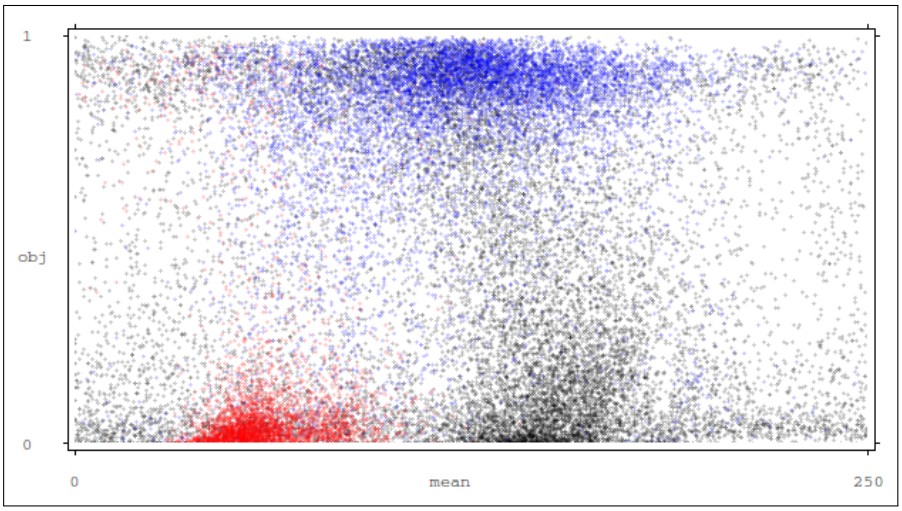

Figure 3.7: Mean value vs average objectness per region. Red is lumen, blue wall, black other regions.

Skewness:

$$\gamma_1 = \frac{1}{|s|} \cdot \sum_{v \in s} \left[\left(\frac{v - \overline{s}}{\sigma_s} \right)^3 \right] \tag{3.8}$$

A statistical analysis of the intestinal wall has shown different (higher) skewness values in the wall when compared to intestinal lumen. The reason for the skewness of the wall region histograms is the intensity shape of the wall - with a larger number of lower values on the borders, almost no voxels below the mean value of the lumen (except for noise and neighboring fat/air) and a small amount of high-valued voxels in the middle. See figure 3.8.

Kurtosis:

$$\beta_2 = \frac{\frac{1}{|s|} \cdot \sum_{v \in s} (v - \overline{s})^4}{\sigma_s^4} \tag{3.9}$$

We have tried kurtosis as another standardized moment and tried to reveal, if it had any advantage for our case. Its benefit turned out to be of minor importance. See figure 3.9.

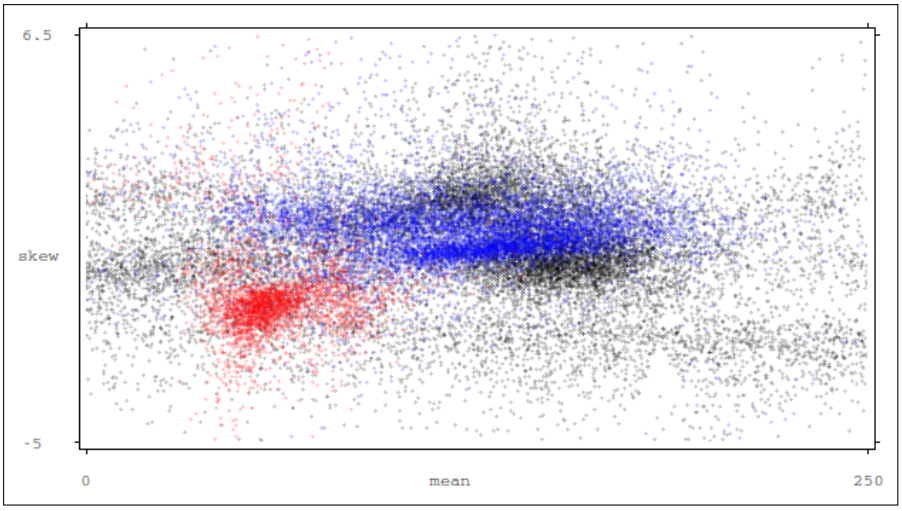

Figure 3.8: Mean value vs skewness per region. Red is lumen, blue wall, black other regions.

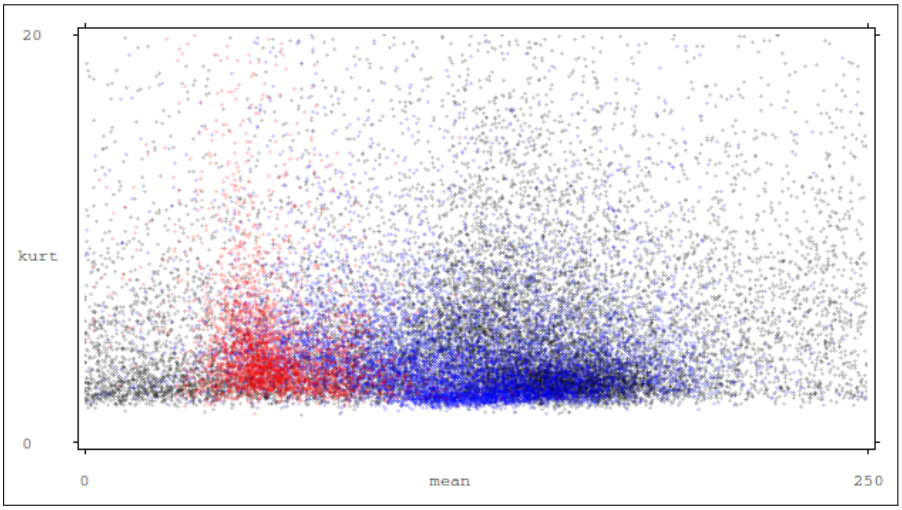

Figure 3.9: Mean value vs kurtosis per region. Red is lumen, blue wall, black other regions.

Size of average gradient and **average size of gradient** - chosen for the same reasons as standard deviation, but turned out to be of less information value.

We have selected a set of 4 features as the basis for our classifier: *mean, standard deviation, objectness* and *skewness*. The bare minimum for a relevant classification (albeit with many errors) is the *mean value* and *objectness*. Both the *standard deviation* and *skewness* make the final classification more precise with a stronger certainty on lumen and wall

Tables 3.2, 3.3 and 3.4 show the selection of features. We have used a manually segmented training set as a source of features and then using these values (shown in graphs in this chapter) tried to compute the probability of each region belonging to its class (using a 2D to 4D k-NN classifier with $k = 10$ mentioned later in section 3.3.3) on the same dataset. This way we have evaluated all regions from the training set, thresholded the probability by 0.5 and computed total error in classification. Tables 3.2, 3.3 and 3.4 have 4 columns: *Lumen* and *Wall*

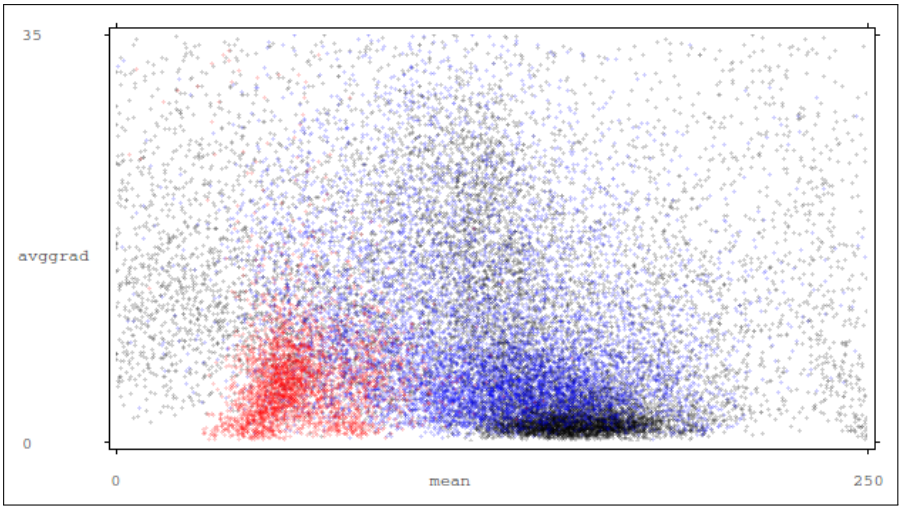

Figure 3.10: Mean value vs size of average gradient per region. Red is lumen, blue wall, black other regions.

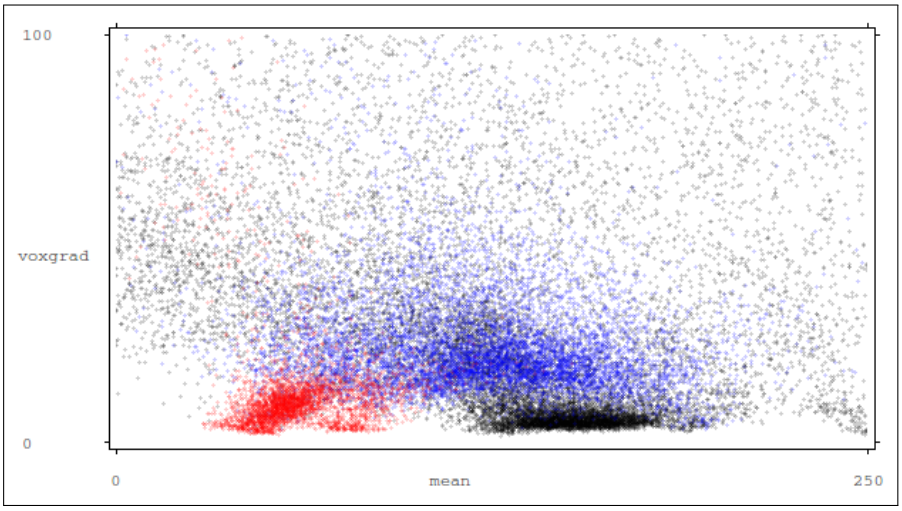

Figure 3.11: Mean value vs per-region average size of per-voxel gradients. Red is lumen, blue wall, black other regions.

indicate the percentage of regions within manual segmentation of lumen and wall belonging to the correct class (higher number is better). Columns $\neg Lumen$ and $\neg Wall$ indicate number of erroneously marked voxels as *Lumen* or *Wall* while not belonging to the given class (false positives).

The visualization of the feature space can be seen in figure 3.12. The representation of individual features using grey values in volume space is shown in figures 3.13.

3.3.3 Probability estimation

We have used a form of kNN (k nearest neighbors) probability estimator. This estimator was chosen for these reasons:

- Good fitting to the training data with class clusters of various shapes
- Robust in case of noisy data

Mean + ...	Lumen	Wall	¬ Lumen	¬ Wall
Avg. grad. size	75.76	60.61	1.52	7.93
Objectness	**83.02**	75.84	2.17	7.68
Kurtosis	63.73	49.06	2.72	6.43
Skewness	77.25	67.32	1.79	7.66
Standard deviation	82.02	69.08	1.32	6.57
Avg. size of grad.	82.94	72.58	0.9	8.02

Table 3.2: Classification precision of training dataset. Using 2 per-region features: Mean value + value on row.

Mean + Objectness + ...	Lumen	Wall	¬ Lumen	¬ Wall
Avg. grad. size	81.42	68.81	1.41	6.59
Kurtosis	81.05	69.03	2.24	5.25
Skewness	81.59	73.99	1.9	5.81
Standard deviation	**84.51**	70.81	1.23	6.09
Avg. size of grad.	83.88	74.24	0.89	7.57

Table 3.3: Classification precision of training dataset. Using 3 per-region features: Mean value + Objectness + value on row.

Mean + Obj. + Stddev + ...	Lumen	Wall	¬ Lumen	¬ Wall
Avg. grad. size	83.97	70.63	1.3	6.12
Kurtosis	86.25	72.87	1.16	5.28
Skewness	**88.23**	75.92	0.88	5.5
Avg. size of grad.	87.85	76.72	0.97	5.07

Table 3.4: Classification precision of training dataset. Using 4 per-region features: Mean value + Objectness + Standard deviation + value on row.

- Relative simplicity - easy implementation for testing, easy debugging

The speed of kNN is not very good in case of a brute-force version, but acceleration structures can greatly improve the time complexity (such as a k-D tree).

The training set creates a 4D feature space with individual points representing regions from the training set. Depending on the size and density of the training set, we need to choose a number k for neighbor selection.

When processing a sample region s, we select k nearest neighbors (set $K_k(s)$) from the training set in the feature space using Euclidean distance and compute the probability that s is in class A:

$$P_A(s) = \frac{|\{b|b \in K_k(s) \wedge b \in A\}|}{k} \tag{3.10}$$

The training set consisted of all three classes: lumen, wall and other regions. We needed the probabilities only for P_{wall} and P_{lumen}.

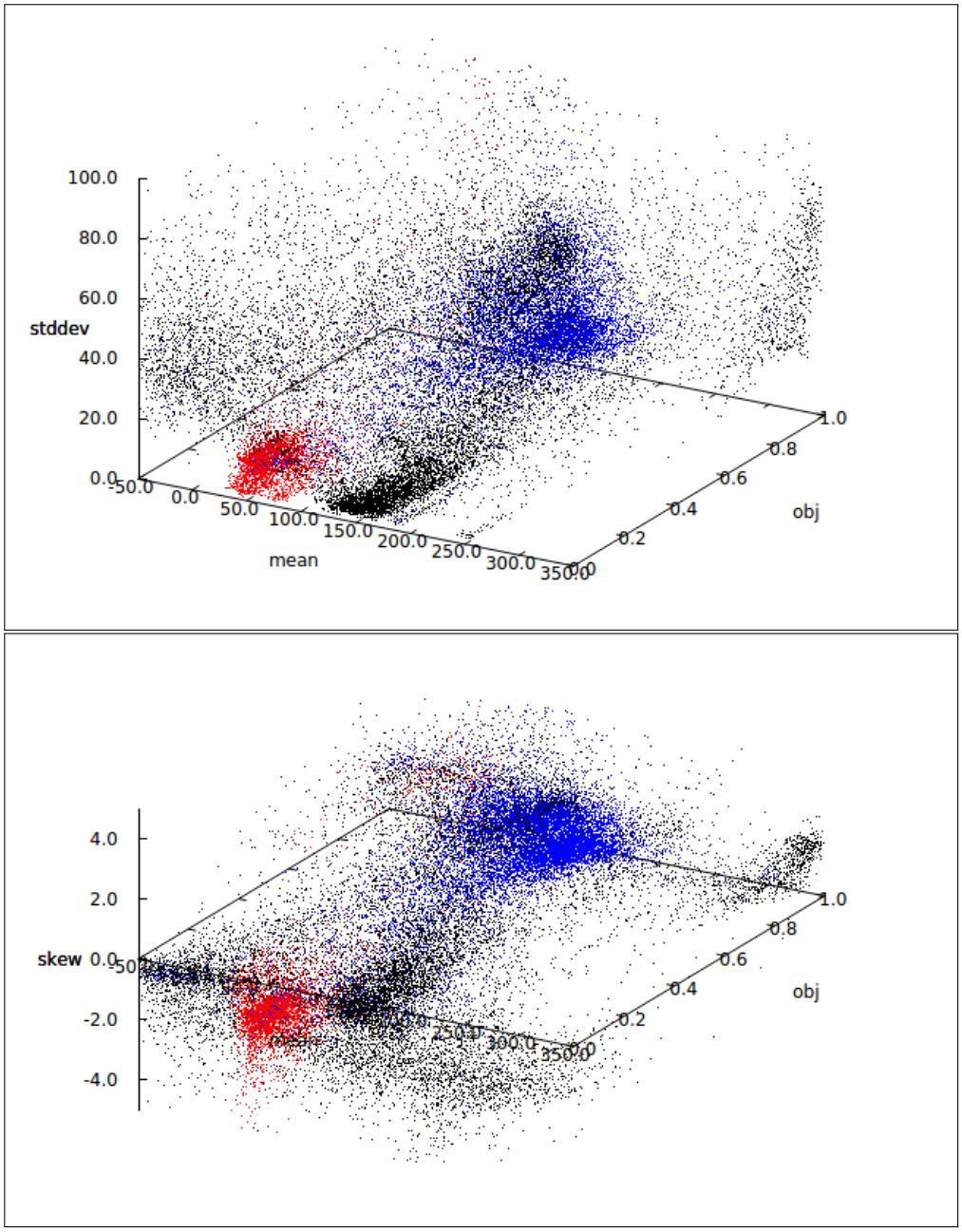

Figure 3.12: Feature space on NLM-filtered data of the training dataset. (a) Mean value × objectness × standard deviation. (b) Mean value × objectness × skewness. Red dots are regions in *lumen*, blue dots are *wall* and black dots are all other.

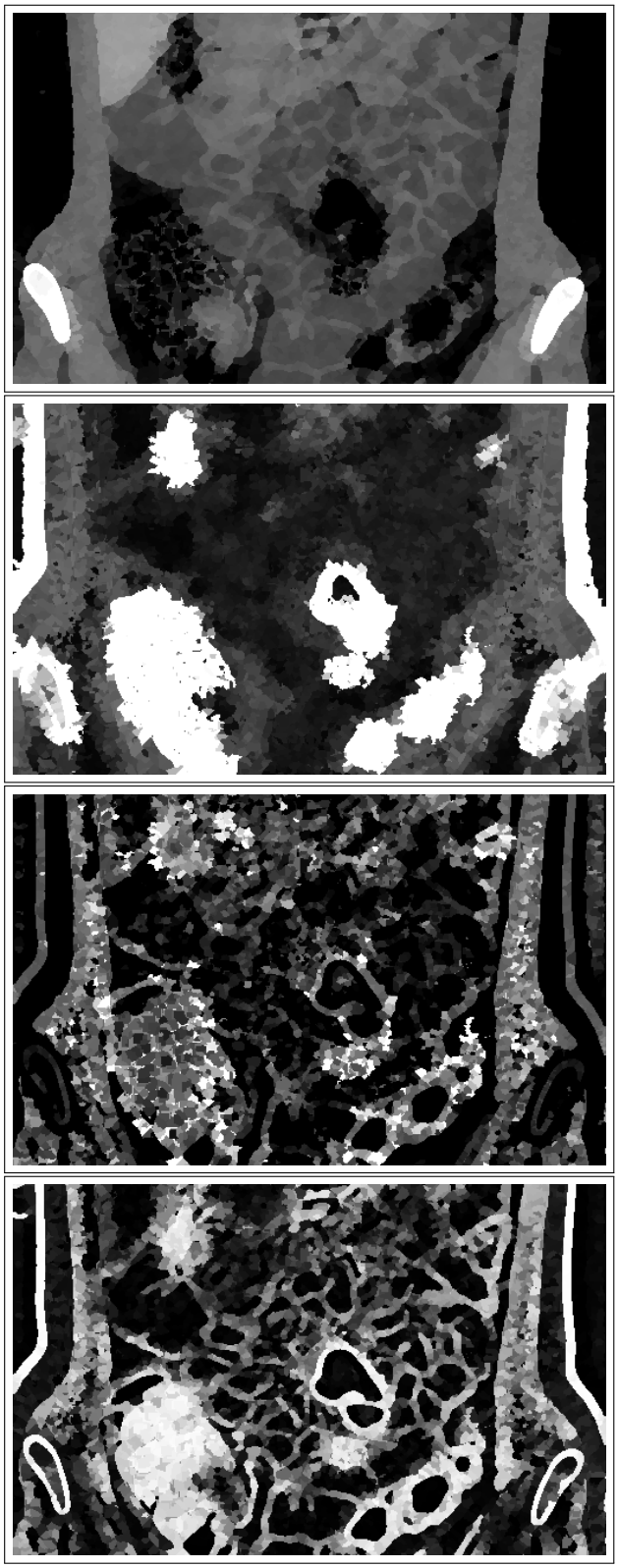

Figure 3.13: Example of features computed per region. From top: (a) Mean value. (b) Standard deviation. (c) Skewness. (d) Objectness filter.

3.4 Results

3.4.1 Preliminary evaluation

The goal of this chapter is to provide a sufficiently reliable probability function on the dataset to map the location of intestinal lumen and wall. So we performed only a preliminary test of robustness, not a full test on all available datasets. For each evaluation is necessary a full manual segmentation, which is a very time- and effort-demanding task on the side of a specialist. Quality results shown in this chapter are only given as a hint to the behavior of the probability computation method.

The final result will be the ability to track intestinal segments with results in chapter 4.

We have taken a different patient scanned with the same settings, with well cleaned and distended intestinal lumen and good saturation of intestinal wall with contrast. A part of this dataset was selected ($219 \times 180 \times 217$ voxels) and manually segmented to provide exact information about the intestinal lumen and wall.

Using the training data obtained in this chapter, we have performed similar analysis as in section 3.3.3, but only using the selected features from the training set on the new patient. The results are in table 3.5.

For comparison we have created a training set from this evaluation volume and did a second evaluation in table 3.6.

Mean + ...	Lumen	Wall	¬ Lumen	¬ Wall
Objectness	91.54	62.94	3.09	8.18
Skewness	81.16	62.62	2.32	12.31
Standard deviation	92.99	51.73	4.95	9.69
All	**94.31**	**67.6**	5.69	10

Table 3.5: Classification precision on a new (evaluation) dataset. Feature data are taken from the training dataset, evaluated dataset is a different patient scanned with the same settings. A part of the dataset with intestine has been extracted, manually segmented and compared with thresholded probability results P_{lumen} and P_{wall}.

Mean + ...	Lumen	Wall	¬ Lumen	¬ Wall
Objectness	87.48	63.77	1.77	4.97
Skewness	80.52	57.49	1.20	4.12
Standard deviation	80.37	54.73	1.80	3.49
All	**90.13**	**66.72**	1.00	3.37

Table 3.6: Classification of the evaluation dataset as in table 3.5, but with a training set extracted from the same volume.

The precision in lumen probability came as a mild surprise. The explanation seems to be that the evaluation dataset contained cleaner data and was also smaller than the original training dataset. The behavior is, however, similar in

both cases and a usable probability function can be computed using both training inputs.

A significantly better performance in false positives (¬ Lumen and ¬ Wall) can be observed on both datasets when the dataset itself is used as a training set. A training set used from different dataset causes more errors - this might be caused by the actual quality of the training set, because it is dependent on the presence of *other* regions (other than lumen and wall) from the body. It is obvious, that the fitting of training data over the present regions is much better when trained on itself.

A graph representation of the second training set (created over the evaluation patient) is shown in figure 3.14. Please compare these graphs with the original dataset in figures 3.7 and 3.12(a) to see the stability of the training set created on different patients scanned with the same settings.

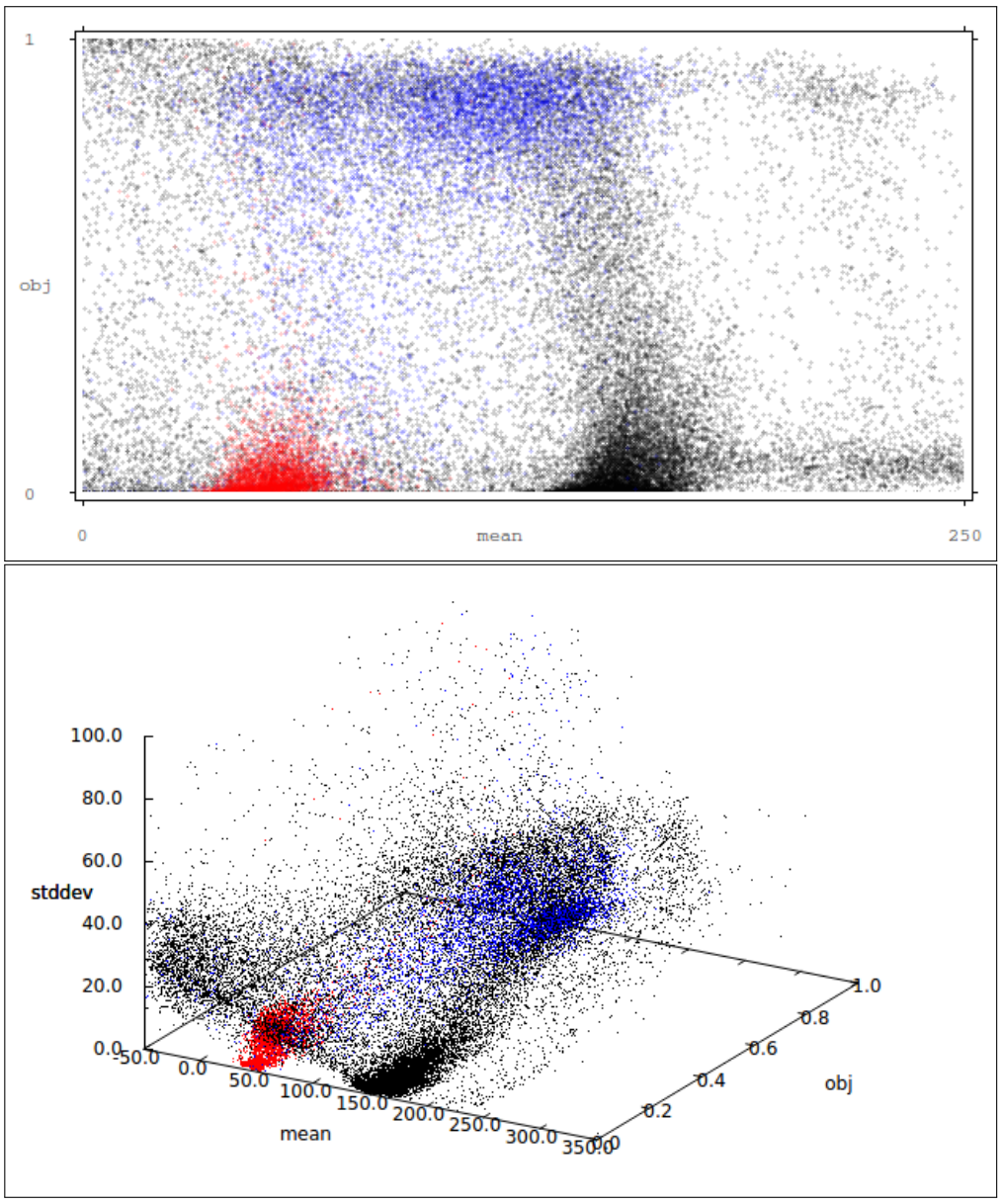

Figure 3.14: Training set created on evaluation data for table 3.6.

A side-by-side comparison is shown in section A.1.2 for all used region features.

You can compare the shape and location of matching region classes.

An interesting fact about the robustness of our approach rose from an error during dataset evaluation. We have by mistake tried to evaluate a dataset scanned with different parameters and with different scaling of values leading to similarly looking pictures to human eye, but with values not aligned with our training dataset (and significantly different characteristics in other features than region mean value). The algorithm nonetheless managed to keep a precision around $30 - 40\%$ in the lumen, $20 - 30\%$ in the wall and around $95 - 99\%$ outside. Definitely not enough for a reliable segmentation and exact evaluation has not been done as the pair training set - evaluated set was obviously wrong, but an interesting effect of the feature space adaptability. An example of the results is shown in appendix A.2.

3.4.2 Effect of denoising

We did not evaluate the original noisy data due to the problems mentioned before (see especially figure 3.4). But for a rough idea of the state of the feature data, see chapter A.1.1 where we have provided comparison graphs over original and denoised datasets.

Many of the features form an indistinguishable *blob* unsuitable for processing. Usually the resolution in any other dimension than *mean value* between the lumen and regions outside the intestine is severely lacking.

3.4.3 Parameter scaling

	Training set on itself	Training set applied on evaluation set	Evaluation set on itself
Error improvement	-0.238%	-0.049%	-0.146%

Table 3.7: By how much percent of error metric did the scaling improve the final result.

We have used several descriptors (given in section 3.3.2), but with a very different range of values. To keep the values in a similar range, we took the most "compact" set of nD (n-dimensional) values, which was the set of regions marked as lumen and multiplied the values in each dimension so that the standard deviation in each dimension is approximately the same. Due to the nature of the input data an exact fit is unnecessary, but a rough approximation always helped with the result. Small differences around the final value did not make any significant difference (i.e. had much lower impact than noise).

The results of parameter scaling can be seen in table 3.7 while applied on the evaluation of the training set from chapter 3.3.2 and on evaluation dataset from chapter 3.4.1.

3.4.4 Data size reduction

An important part of this chapter is the reduction of data for further computation. Table 3.8 gives an overview of how extensive the reduction was in the case the

	Data example	Average reduction over all data
Total number of voxels	160167497	1
Watershed on original	3637957	1:44.22
Watershed on denoised	1799773	1:87.38

Table 3.8: Data size reduction

full dataset from which was taken our training set.

3.4.5 Times

Exact time measurements of the individual steps were not taken, because their performance varies with each new piece of HW on the market and it was not the goal of this thesis. But a rough estimation is that each step takes at most a few seconds (including watershed segmentation in about 2-3 seconds in Kolomazník et al. [2012]) with the exception of denoising (about 1-5 minutes per patient, Horáček et al. [2011], in chapter 2). The complete processing can be done in general within a few minutes (under 5 minutes on commodity HW, scales well with better equipment). All steps can run automatically, so the preparation of data may proceed just right after the CT reconstruction without any interaction of the radiologist or other personnel. This is a great advantage.

3.4.6 Proposal of training set matching

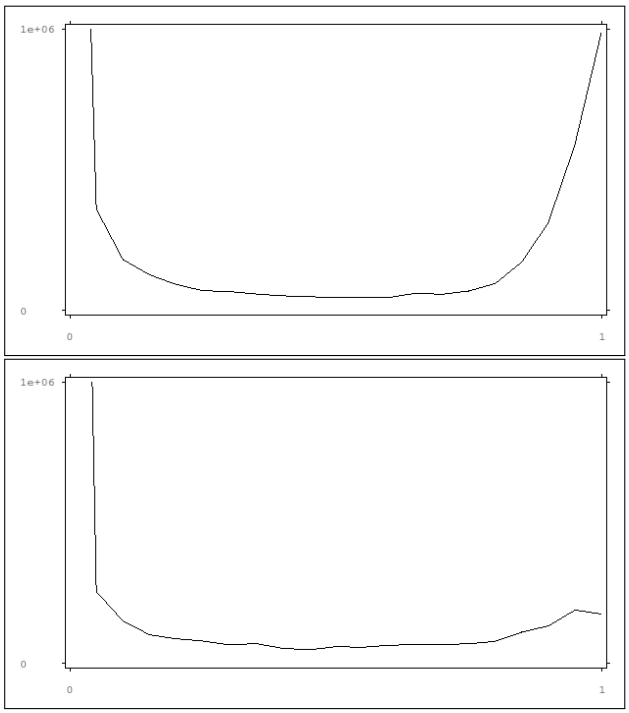

Figure 3.15: Example of histogram computed over probability function P_{lumen}. Top: Probabilities computed with correct training set. Bottom: Probabilities computed with a wrong training set belonging to different scanner settings.

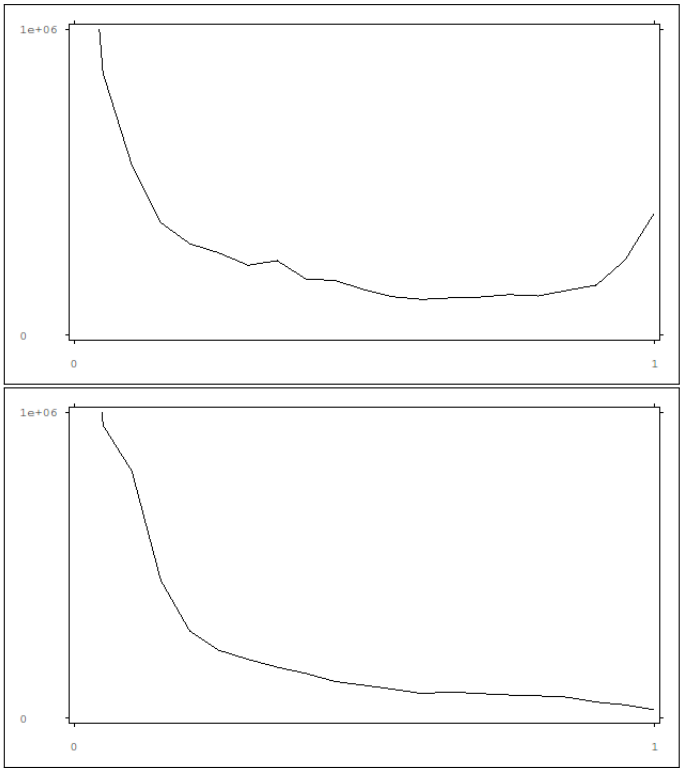

Figure 3.16: Example of histogram computed over probability function P_{wall}. (a) Probabilities computed with correct training set. (b) Probabilities computed with a wrong training set belonging to different scanner settings.

With the main preparation work done only once, the evaluation of the dataset and computation of probabilities P_{lumen} and P_{wal} is done in mere seconds. Thus there is room for optimization by using multiple training datasets for multiple scan types. If a robust-enough technique for the automatic evaluation of the probabilities is used, then the algorithm might try multiple training sets in case the probability is evaluated as not good enough.

One such method that comes to mind is an analysis of a histogram computed over functions P_{lumen} and P_{wall}. Figure 3.15 and 3.16 show such histograms, one extracted from correctly computed probabilities with matching training set and the other taken from a wrongly selected training set. The number of voxels stays the same, so a comparison of histogram values near 1 can give us at least an information about a clearly failed matching.

We do not have available a large enough amount of datasets with various scanner settings. Performing several different scans on patients just for the sake of this research raises some ethical questions about excessive use of ionizing radiation. Each patient is scanned to the best of knowledge of the radiologist to provide ideal scanning performance, no experimentation is currently feasible without the usage of a phantom. The analysis of training set fitness is thus left as a hint for future research when a larger set of scans with different settings is obtained.

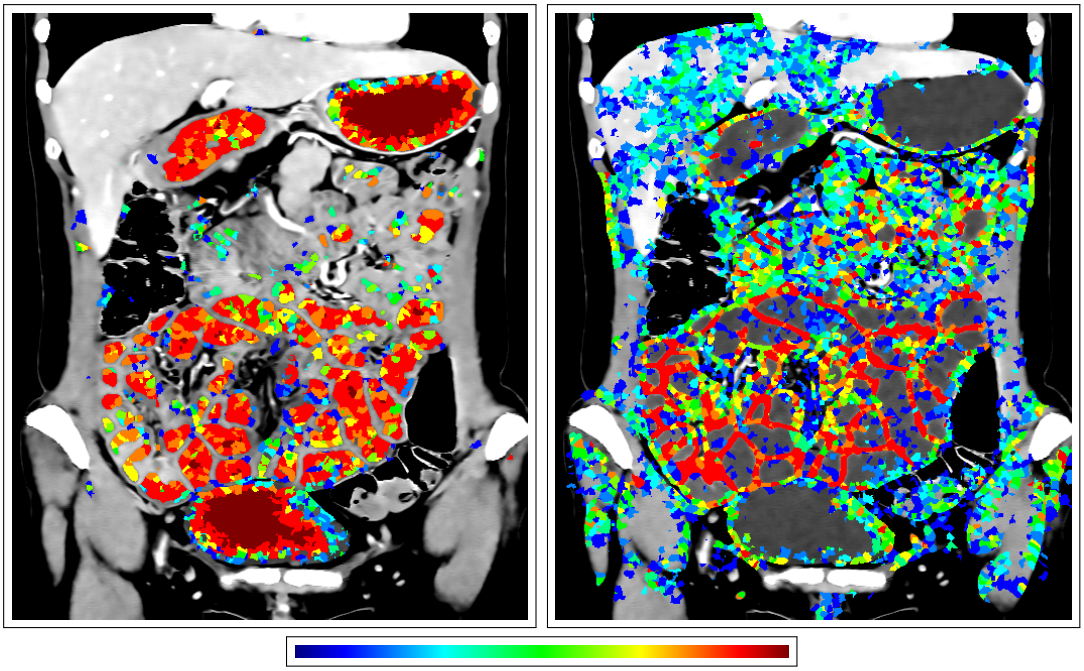

Figure 3.17: Example of computed probability of (a) lumen and (b) wall on patient that was used as a training set. Probability goes from 0 on transparent/blue parts, over green, yellow, to red with dark red being probability 1.

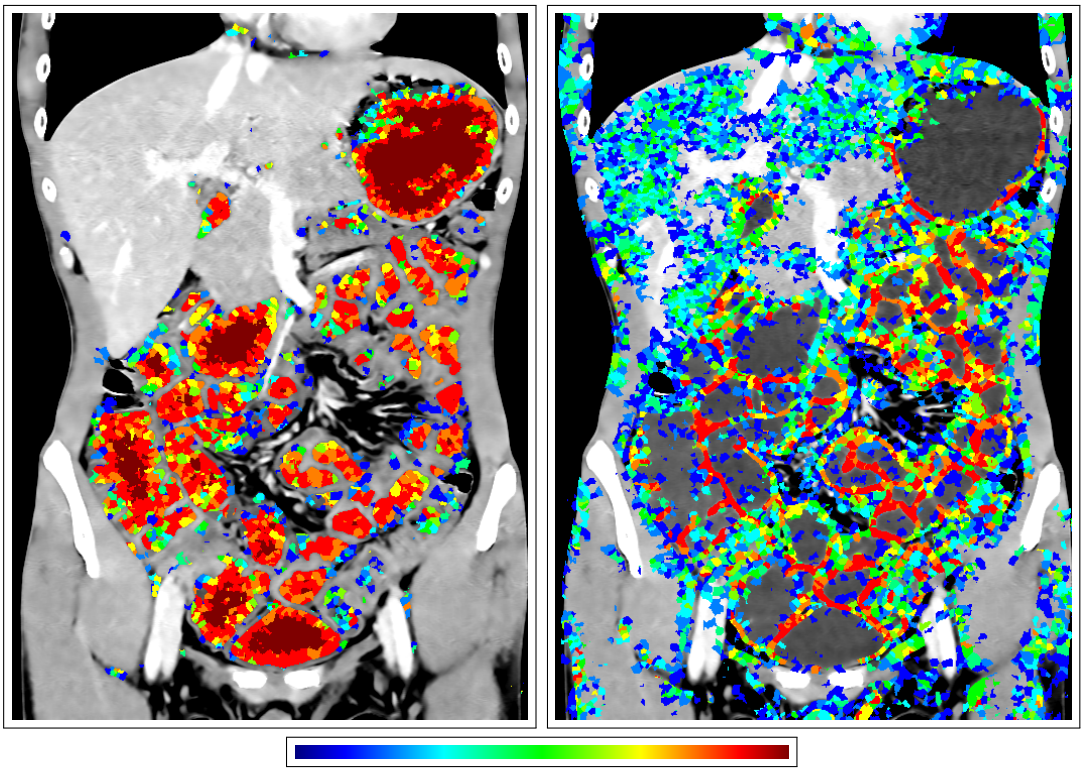

Figure 3.18: Example of computed probability of (a) lumen and (b) wall on patient that was used as an evaluation set. Probability goes from 0 on transparent/blue parts, over green, yellow, to red with dark red being probability 1.

3.5 Conclusion

We have shown, that a number of very simple automatic steps can provide a relatively reliable probability map of intestinal lumen and wall (figures 3.17 and 3.18). These might subsequently enable the use of much more complex segmentation algorithms, even on very large datasets. All these steps together take just a few minutes and can be executed without any human intervention. The result can be a map of probabilities (one scalar value) of where the intestinal lumen is most probably found. This is a great foundation for any further segmentation building on this input parameter - be it any kind of region growing, level sets or different optimization approach. A drastic reduction in input data size (around 1:80) helps in choosing from a much larger pool of algorithms otherwise discarded for their time/space requirements.

We do not change the asymptotic algorithm complexity, just reduce the input size. From a radiologist's point of view an algorithm that gives result in two hours is unusable. But the same algorithm giving its result in half a minute is already worth the wait, if it significantly helps with diagnosis.

A very important thing to notice is that using a state-of-the-art denoising algorithm is a must in cases like CT enterography. Even watershed segmentation on such data performs an order of magnitude worse and manual segmentation over such created watershed regions is practically unusable due to the large amount of noise.

Chapter 4

Tracking of intestinal segments

4.1 Introduction

Small bowel is spatially a very complex organ and thus the manual inspection of its path from the 3D CT enterography data is a lengthy process requiring a great deal of concentration and time of a radiology specialist to identify and diagnose possible inflammation, obstructions and other problems.

Our goal is to improve and simplify the inspection of the CT enterography data by tracking the intestinal path and provide the radiologist with a clear cross section in each segmented part. See fig. 4.1 and fig. 4.2.

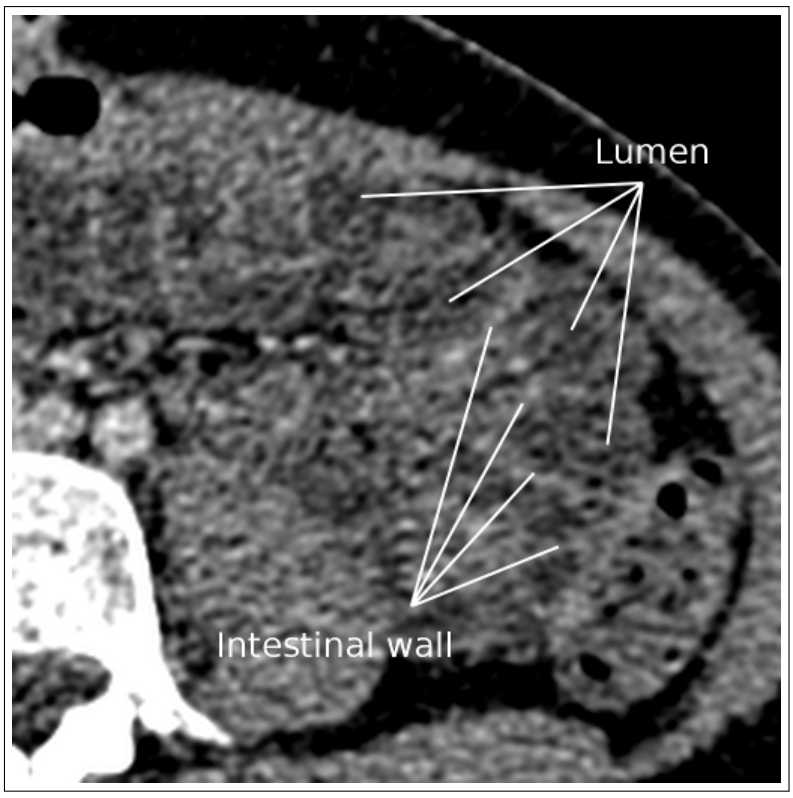

Figure 4.1: Axial slice from raw input data.

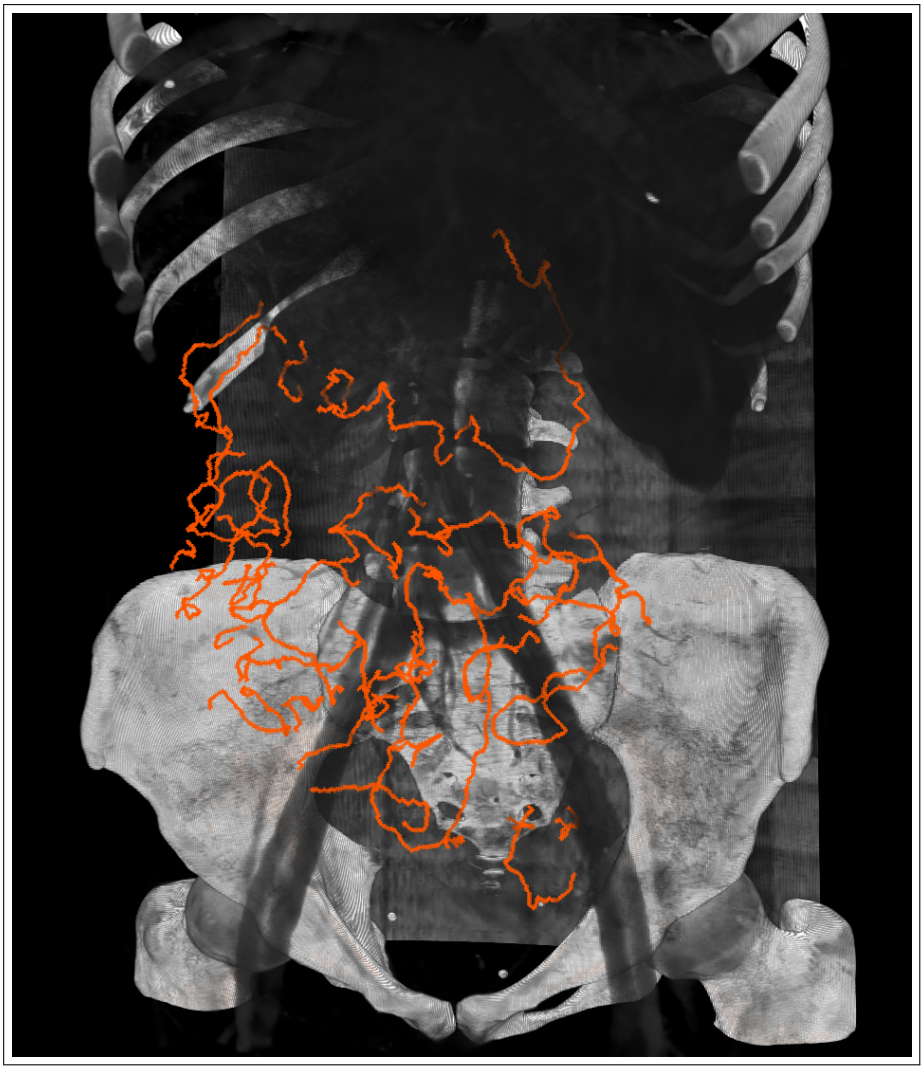

Figure 4.2: DVR projection of the desired result with segmented paths of small intestine segments.

4.2 Algorithm overview

Our algorithm consists of several main steps:

1. Denoise input data (chapter 2)
2. Watershed transformation (section 3.2.2)
3. Statistical features computation on each watershed region (section 3.3.2)
4. Probability of region belonging to lumen/wall/other (section 3.3.3)
5. Segmentation of lumen (section 4.3)
6. Intestine path tracking (section 4.4)
7. Lumen border segmentation on tracked data (chapter 5)
8. Visualization (chapter 6)

In this chapter we will deal with the segmentation of lumen from probability maps and the tracking of individual intestinal segments. An example of tracked segments is shown in figure 4.2. The golden goal would be a complete tracking of the small intestine. That is, however, still not possible due to the quality of the data from current medical-grade CT scanners, various pathologies, imperfect distention and cleaning in case of obstructions or narrow segments due to inflammation.

We present here our algorithm that is able to track a significant part of the small intestine by following several individual intestinal segments.

4.2.1 Symbols used

We will be using the following symbols throughout the rest of this chapter (some of them have been used also in chapter 3 and we will keep it consistent here):

- s ... one whole watershed region
- A ... a class of regions (i.e. lumen, wall, ...)
- $P_A(s)$ ... a probability that the region s belongs to A
- $N(s)$ ... a set of regions directly touching the region s
- v ... a voxel in a 3D dataset
- $N(v)$ ... a set of voxels in a direct neighborhood of v
- $M(v)$ ... a binary mask with values $0, 1$
- $D_M(v)$ ... a distance function computed on mask M indicating the euclidean distance in volume space to the closest voxel $w : M(w) = 0$
- $pos(v)$ ... a position of voxel v in the volume space, in millimeters
- $\tau_{decision}$ ... a threshold for an algorithmic decision, all thresholds in this chapter must be optimized to input data acquisition parameters

4.3 Segmentation

The segmentation step creates a binary mask M_{lumen} over the whole 3D volume, value 1 indicating where the intestinal lumen is and 0 elsewhere.

Segmentation is done by the union of sufficiently large areas obtained from adaptive growing from any potential lumen region S.

4.3.1 Probability clean-up

Several regions can have their probability wrongly assigned. There are many reasons for this, such as: an improper clean-up of the lumen, partial volume artifacts and also noise, both in the input data and in the training dataset.

Because the intestine usually spans over many regions, we can improve the local probability homogeneity by checking the local neighborhood. Isolated regions are usually an error.

Let's denote the cleaned probability of region S belonging to a given class A $P'_A(S)$.

$$P'_A(S) = \begin{cases} 1 & \forall n \in N(S) : P_A(n) \geq \tau_{accept} \\ 0 & \forall n \in N(S) : P_A(n) \leq \tau_{reject} \\ P_A(S) & \text{otherwise} \end{cases} \tag{4.1}$$

This way we compute $P'_{lumen}(s)$ and $P'_{wall}(s)$. Both thresholds τ_{accept} and τ_{reject} are from interval $\langle 0, 1 \rangle$ and control how much cleaning is being done.

4.3.2 Adaptive region growing

Three thresholds are needed for the process (all must be $\in \langle 0, 1 \rangle$):

- τ_{loose} for benevolent lumen probability inside lumen
- τ_{strict} for strict lumen probability near the intestinal wall, $\tau_{strict} > \tau_{loose}$
- τ_{wall} for wall probability

A starting region s must belong to the lumen class, i.e. $P'_{lumen}(s) \geq \tau_{strict}$.

Lets denote the currently processed watershed region s. Algorithm is recursively performing:

a) if $(\forall n : n \in N(s) \rightarrow P'_{wall}(n) < \tau_{wall})$: grow into all regions $m \in N(s) \wedge P'_{lumen}(m) \geq \tau_{loose}$

b) if $(\exists n : n \in N(s) \wedge P'_{wall}(n) \geq \tau_{wall})$: grow into all regions $m \in N(s) \wedge P'_{lumen}(m) \geq \tau_{strict}$

This approach prevents leaking into unclear areas where one or a small amount of wall regions are wrongly assigned a middle-to-high lumen probability (that happens when the wall is not saturated enough with the positive contrast agent).

4.3.3 Mask computation

Potential candidates usable as a starting point of adaptive growing are all regions with high lumen probability, $P'_{lumen} >= \tau_{strict}$. We go through all these regions in sequence and run the adaptive region growing algorithm. If the segmented area is large enough (the number of voxels is larger than $\tau_{minvolume}$), mark all these voxels in the result mask M_{lumen} as 1.

An optimization is to skip all already visited regions when selecting the next adaptive growing start.

The result mask is free from all isolated regions from other parts of the body and the lumen is sufficiently and compactly marked.

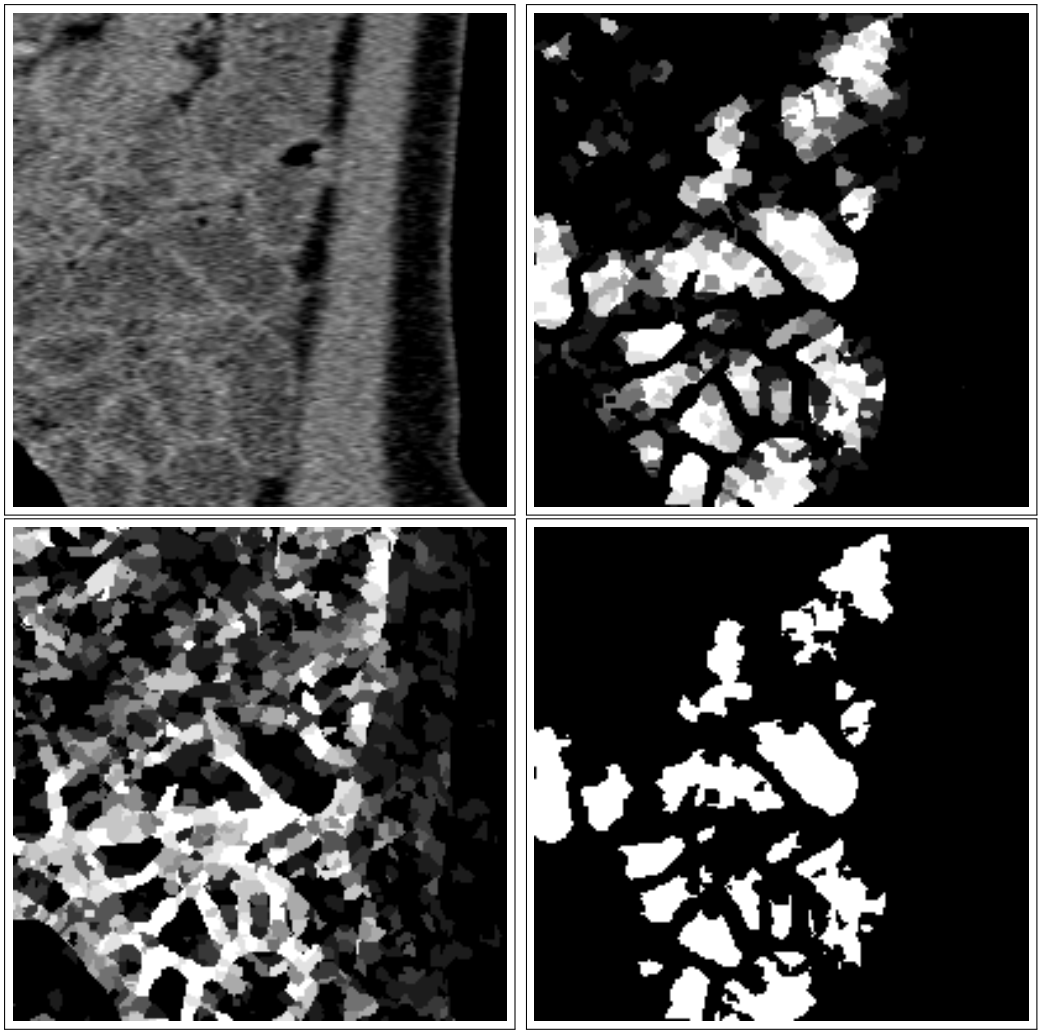

Figure 4.3: Result of probability functions. (a) Original data. (b) Lumen probability P_{lumen}. (c) Wall probability P_{wall}. (d) Binary cleaned mask M_{lumen}.

4.4 Tracking

The idea behind the tracking algorithm is that the segmented part of the intestine should be an elongated object without branches.

We use a prioritized flood fill with backtracking and validity updates for the path search. The algorithm is constructed to track the middle of the lumen while avoiding the most common pitfalls of imperfect input data - small holes in the mask through which simple voxel-based algorithms *leak*.

A set of candidate points is selected through which may lead a potential path and then a path tracking algorithm is issued through all these points to obtain a set of intestinal segments.

4.4.1 Single path tracking

The input for our path tracking algorithm is a distance map D_M computed over the lumen mask M_{lumen} computed in section 4.3.3. Each voxel in D_M contains the euclidean distance (measured in voxel coordinates space) to the closest voxel w which has $M_{lumen}(w) = 0$. (That also implies that all voxels in D_M outside

lumen are equal to 0).

If we want to track the center of the lumen, a good guess at its position would be to search for points far from the wall, i.e. in the center of the lumen. Thus, a potential starting voxel v should be a local maximum of the distance function D_M.

Prioritized flood fill algorithm grows first into voxels with higher D_M. All voxels remember their parent and thus every voxel can be backtracked to the starting voxel v. Let's denote the backtracked path (as set of voxels) from voxel w as $BTPath(w)$.

A path validity check is then performed on each newly discovered longest valid path: a path is backtracked and marked as invalid with the exception of the last segment of the maximal length of $\tau_{validpath}$ from the end. All voxels in the queue that are currently valid are kept as valid only if they share at least one common predecessor from the new longest path marked as valid, all other voxels in queue are marked as invalid for path search. Although they are processed without difference, they only distribute the (in)validity flag to all future children and are not considered for the longest path search.

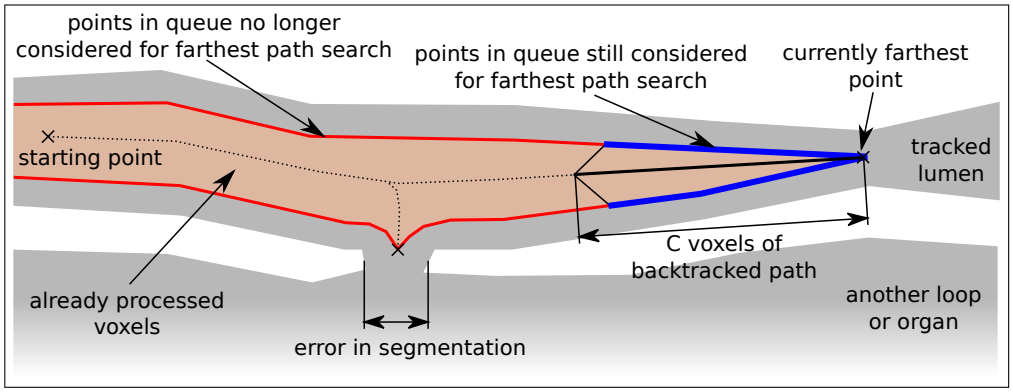

Figure 4.4: A schema of the tracking algorithm. White voxels (descendants of the thick line) are still considered for the longest tracked path computation, gray voxels (descendants of the dotted line) are not considered for the tracking anymore, but are still used for prioritized flood fill. Black voxels are not visited yet. Letter C represents threshold $\tau_{validpath}$.

The algorithm is then as follows (a schema is shown in fig. 4.4):

1. define a function V indicating that a voxel is still considered valid for the longest path search, set it to *unvisited* everywhere
2. define a function *len* indicating the backtracked path length from the starting point to any visited voxel, set it to *undefined* everywhere
3. add v to the queue Q of unprocessed voxels and set

$$V(v) = valid \tag{4.2}$$
$$len(v) = 0 \tag{4.3}$$

4. remove w from Q such that

$$\forall u \in Q : D_M(u) \leq D_M(w) \tag{4.4}$$

5. parse neighbours

$$\forall u \in N(w) \wedge V(u) = \textit{unvisited} \wedge D_M(u) > 0 : \tag{4.5}$$

   (a) add u to Q

   (b) store the parent of voxel u so that backtracking is possible from any visited voxel (add w to $BTPath(u)$)

   (c) set validity and distance

$$V(u) = V(w) \tag{4.6}$$
$$len(u) = len(w) + \parallel pos(u) - pos(w) \parallel \tag{4.7}$$

   (d) if $V(u) = \textit{valid} \wedge \forall x \neq u : len(u) > len(x)$ (found new longest valid path), then perform validity update on current path and update V (here $\textit{valid} = \textit{true}$ and $\textit{invalid} = \textit{false}$, *unvisited* does not appear):
   $\forall x \in BTPath(u) :$
   $V_{new}(x) = V_{old}(x) \wedge (\parallel dist(u) - dist(x) \parallel \leq \tau_{validpath})$
   $\forall x \in Q \wedge x \notin BTPath(u) :$
   $V_{new}(x) = V_{old}(x) \wedge (\exists a \in BTPath(x) : a \in BTPath(u) \wedge V(a) = \textit{valid})$

6. continue with step 4 until Q is empty

Lets denote the farthest point on the longest valid path as e_1. It is then used as the starting point for a second run of the algorithm (lower index 2 will denote variables computed during the second run). This time we get the farthest valid point e_2. Path correctness is then decided on the following condition:

$$v \in BTPath_2(e_2) \tag{4.8}$$

There are some things to emphasize about how this algorithm copes with errors in segmented mask:

- Prioritizing voxels with high D_M value forces the flooding to follow the center of the lumen and avoids *shortcuts* in the same intestinal segment (*shortcut* is a small hole in the mask between two parts of the same intestinal segment, thus its values of D_M will be small and processed later than the central part of lumen on both sides).

- *Path validation* is there because of similar *holes* in the mask as *shortcuts*, but between different segments of the intestine. The flooding will visit also these holes and neighboring segments, but if it first follows the original segment farther than $\tau_{validpath}$ from these *holes*, then they are flooded only for the filling completeness and not considered valid for backtracking (and thus *leaking*).

4.4.2 All paths tracking

As it was already mentioned, a set of potential candidates for points lying on a path is the set of local maxima of the function D_M. These points are sequentially

processed, each is used as a starting point for a single path tracking algorithm. When a viable path is found, all remaining candidate points in close vicinity to the new path are removed from the queue to prevent unnecessary duplication of paths.

The result is a set of isolated path segments of various lengths viable for continuously creating cross-sectional images for a good intestinal wall thickness and lumen distension analysis.

We have not found a reliable method to connect them to a single continuous run, because the discontinuities might (and usually are) caused by not only small local errors, but also longer segments of intestine that are either collapsed or filled with something else than the neutral contrast agent.

In many cases we have concluded that the real world input data is in such a state so that the necessary information needed is not present - especially badly emptied intestine or several collapsed intestinal segments touching each other looking as a single continuous (almost flat or uniformly *noisy*) area with no chance to distinguish between individual segments. Thus using only the topology of the segment ends to reconnect them may lead to significantly erroneous paths.

4.5 Results and Conclusion

We have performed a testing segmentation on 34 patient datasets scanned with CTE with diagnosed or suspected Crohn's disease or other inflammatory small bowel disease or obstruction.

We have not performed any new scans just for the purpose of this research. Our research data consist of available CT enterography scans that were taken during routine small intestine examination in one hospital (Hospital Na Homolce, Prague, Czech Republic) and the patients voluntarily signed an agreement on participating in our research.

We are aware that our testing group was currently limited. We are continuously increasing the size of our database.

4.5.1 Threshold selection

A number of thresholds are used throughout this chapter. Their selection was done either empirically or by an automated search for optimal value. The number of neighbors for kNN probability was set to 10, so all probabilities had a precision of 0.1.

An optimal value was selected for $\tau_{accept}(0.6)$ and $\tau_{reject}(0.3)$. These values were obtained by doing the best "blind" thresholding of the training dataset, comparing the manual dataset to automatically assigned values and minimizing the number of false positives and false negatives on assigning to both wall and lumen classes. A full test was done over all possible values. Most of the cleaning using these two thresholds is done inside the lumen, so a optimal value can be found as causing minimal number of errors in both lumen and wall.

A more complicated situation is in the case of adaptive growing algorithm (section 4.3.2) and its parameters τ_{loose}, τ_{strict} and τ_{wall}. Our initial guess was $\tau_{loose}(0.4)$, $\tau_{strict}(0.6)$ and $\tau_{wall}(0.6)$. We have tested values around these initial estimates, they are shown in table 4.1 and figure 4.5. The result of adaptive

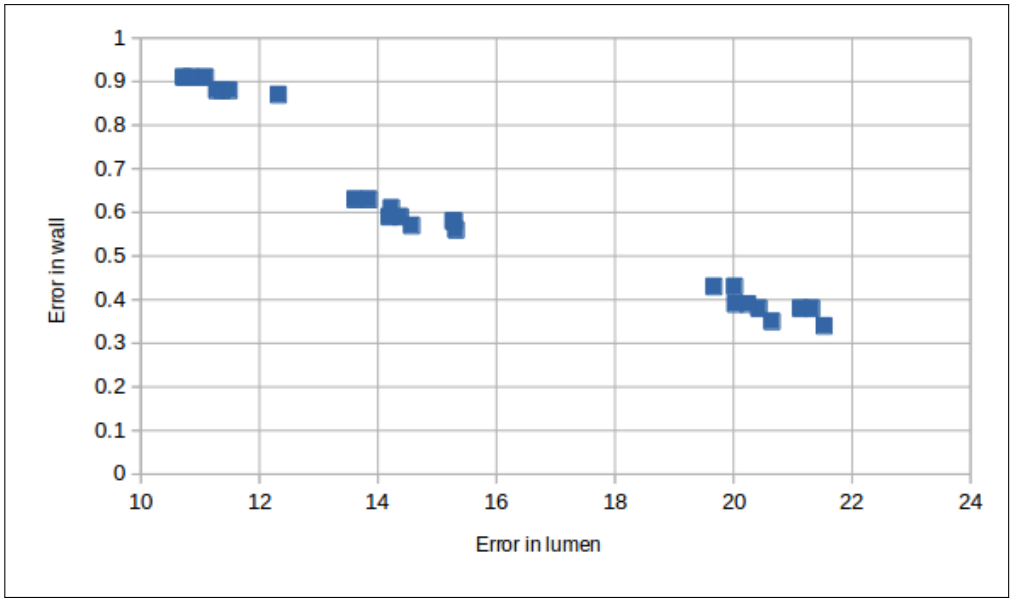

Figure 4.5: Dependency of error values during the testing of parameters τ_{strict}, τ_{loose} and τ_{wall} from table 4.1. Tested values with the same value of τ_{loose} form visible clusters (see values in table 4.1 for input values).

growing in chapter is a binary mask indicating lumen location. We have compared it with a manual segmentation. The error evaluation for wall and lumen was done by counting false negatives of result mask in manual lumen segmentation and false positives in manually segmented wall. Results with the same τ_{strict} form clusters with similar performance (i.e. τ_{strict} causes the largest variance in the result). Errors in wall cause "leaking" between segments, errors in lumen cause incorrect following of the lumen center. Leaking is a more serious problem, so the error rate in wall areas was more closely observed. Error rate over 20% in lumen was already too large to have smooth center line, error rate nearing 1% in wall caused more leaks in thin areas with bad distention. All values from the cluster around our initial guess created segmentations manually evaluated as good, so we have used it for all tracking evaluation within this chapter.

Strictly empirical values were selected for $\tau_{minvolume}$(2000 voxels) (containing $2 * 10^3$ voxels, i.e. $250mm^3$) and $\tau_{validpath}$(20 voxels). These were chosen based on our estimation of the the geometrical properties of the intestine and voxel dimensions. A range of values around these thresholds work similarly, we have not found an optimality criterion for their selection other than manual inspection of the results.

4.5.2 Tracking results

The total length of the correctly tracked segments is given in tab. 4.2. An individual segments analysis is given in tab. 4.3.

These results were cleaned manually by removing erroneously marked regions. Due to the stochastic nature of the computation of underlying functions P_{lumen} and P_{wall}, some non-lumen regions were marked in the mask M_{lumen} as lumen due to the fact that the only difference in the watershed regions was its anatomical location. The method of M_{lumen} computation was not designed to handle these

τ_{loose}	τ_{strict}	τ_{wall}	Error in lumen	Error in wall
0.3	0.5	0.5	11.09%	0.91%
0.3	0.5	0.6	10.85%	0.91%
0.3	0.5	0.7	10.72%	0.91%
0.3	0.6	0.5	14.23%	0.61%
0.3	0.6	0.6	13.85%	0.63%
0.3	0.6	0.7	13.61%	0.63%
0.3	0.7	0.5	20.43%	0.38%
0.3	0.7	0.6	20.02%	0.43%
0.3	0.7	0.7	19.67%	0.43%
0.4	0.5	0.5	11.49%	0.88%
0.4	0.5	0.6	11.37%	0.88%
0.4	0.5	0.7	11.28%	0.88%
0.4	0.6	0.5	14.57%	0.57%
0.4	**0.6**	**0.6**	14.38%	0.59%
0.4	0.6	0.7	14.19%	0.59%
0.4	0.7	0.5	20.65%	0.35%
0.4	0.7	0.6	20.24%	0.39%
0.4	0.7	0.7	20.03%	0.39%
0.5	0.5	0.5	12.32%	0.87%
0.5	0.5	0.6	12.32%	0.87%
0.5	0.5	0.7	12.32%	0.87%
0.5	0.6	0.5	15.32%	0.56%
0.5	0.6	0.6	15.29%	0.58%
0.5	0.6	0.7	15.27%	0.58%
0.5	0.7	0.5	21.53%	0.34%
0.5	0.7	0.6	21.32%	0.38%
0.5	0.7	0.7	21.13%	0.38%

Table 4.1: Testing different parameters for adaptive region growing. Bold line is our initial guess.

regions differently. So a post-process cleaning is necessary.

Most of the datasets had a false positive error in stomach, bladder and colon. Stomach was marked due to the oral administration of neutral contrast agent, thus filling the intestine through stomach. Bladder was marked for the same reason of being filled with liquid very similar in radio-density to the neutral contrast agent. Colon was marked only in case it was well cleaned and also filled with contrast liquid.

An automatic method for removing these regions might be relatively simple in case of stomach and bladder, possibly by introducing topological information about the body as these two organs are reliably placed in similar positions, their structure is simple and easy to track.

Another situation is in case of colon as it is very similar to small bowel on CTE scans. We cannot rely on wall thickness or lumen width, because we are dealing with patients suffering from inflammatory or obstruction problems. Also the anatomical location is only somewhat reliable, but much more varying than in

Dataset name	# of segments	Length in voxels	Length in mm	Training set	Note
02	18	2650	2247	04	
04	35	5604	4898	04	A
07	30	3569	2867	04	
08	57	6178	5052	04	B
09	17	1784	1577	04	
10	30	3587	2975	04	
11a	16	2007	1775	04	C
11b	21	2052	1611	04	C
12	17	2612	2128	04	
13a	37	4549	3895	04	D
13b	30	5433	4816	04	D
14	30	5253	4460	04	
15	15	2345	1920	04	
17	47	5722	5421	04	E
18	44	5599	4488	04	
19	13	1156	1088	04	
20	20	3188	2465	04	
22	7	922	700	04	F
23	29	3930	3016	04	G
24	36	5996	4949	04	H
25	21	2871	2377	04	
26	40	5543	4550	04	I
27	42	4839	3487	04	J
28	15	2263	2051	04	K
29	35	5024	4083	04	L
31	6	670	486	04	M
32	17	2902	2581	04	N
35	33	4697	4139	04	
38	8	1750	1523	04	O
B1	28	4409	3262	B1	P
B2	21	2470	1870	B1	P
B3	30	3788	2715	B1	P
B4	28	3433	2493	B1	P
B5	29	4186	3284	B1	P

Table 4.2: Tracking results per patient - total length analysis. Notes in the last column point to table 4.4 with a detailed explanation of various interesting facts. Dataset names are taken from intermediate marking during our research and do not resemble any other property than a unique name.

case of other organs (like stomach, liver, bladder, kidneys, etc.). The cleansing of colon during CTE is not often perfect, thus remaining feces wildly vary the radiodensity. Imperfect cleansing is partially solved by not marking it as intestinal lumen, thus discarding these segments from the evaluation altogether. But the remaining parts filled with similar liquid as small bowel cannot be easily connected

Dataset name	Average length of segment in voxels	Longest segment length in voxels	Longest segment length in mm
02	125	476	411
04	140	567	492
07	95	328	266
08	89	356	296
09	93	218	191
10	99	255	206
11a	111	268	234
11b	77	276	220
12	125	341	283
13a	105	434	385
13b	161	797	711
14	149	660	559
15	128	463	378
17	96	466	364
18	102	418	338
19	84	205	197
20	123	532	416
22	100	363	280
23	104	294	224
24	138	494	413
25	113	286	241
26	114	500	416
27	83	284	201
28	137	356	323
29	117	521	423
31	81	202	148
32	152	483	427
35	125	394	350
38	190	895	777
B1	117	480	357
B2	89	183	140
B3	91	262	190
B4	89	223	163
B5	113	297	237

Table 4.3: Tracking results per patient - individual segments analysis.

together.

We have left the removal of colon region entirely on the user. On the other hand it can be tracked relatively easily by hand and so a manual fixing of paths found within colon is a brief task.

In many datasets the duodenum was segmented/tracked alongside the rest of the small intestine. We have left it included in the results, because it technically belongs to the small intestine, looks similar to other parts of the intestine and it

Note name	Dataset name	Description
A	04	• part of this dataset used for **training set** and the same training set was used for processing it • good distention and contrast saturation
B	08	• good lumen distention and contrast saturation • **best segmentation** in our database, shown in fig. 4.7 • good tracking even in jejunum
C	11	• two scans of the same patient taken at different times • relatively similar performance on both scans
D	13	• same situation as in C
E	17	• this dataset used as **evaluation** dataset in section 3.4.1 • good distention and contrast saturation • good segmentation and tracking, shown in figure 4.2
F	22	• bad segmentation • very large amount of **inflammation** on intestinal walls • lumen too narrow
G	23	• one segment leaked in **bladder**: whole segment has 197 voxels, 63 leaked into bladder the rest is good, we have left it included in results for the examination of the intestinal part
H	24	• **liver** had similar radio-density like liquid in lumen • tracked segments in liver were manually removed
I	26	• very well segmented and tracked
J	27	• enlarged **gallbladder** found in segmentation • manually removed
K	28	• large amount of body **fat** • bad lumen distention • overall bad segmentation
L	29	• **bad cleaning** of lumen • many air bubbles and feces (see figure 4.8) • tracked only through water filled part of lumen
M	31	• extremely bad segmentation • most of abdomen region **filled with colon** (figure 4.9) • small bowel with very bad distention • unsuitable for automatic tracking
N	32	• large amount of body **fat** (see figure 4.10) • bad lumen distention
O	38	• bad lumen distention, also probably due to body fat
P	B1-B5	• older datasets B1-B5, different scanner • different training set necessary (created from B1) • all have visibly higher noise level (standard deviation in lumen around 50-55HU) • apparent beam hardening artifacts in several places • needed stronger noise removal

Table 4.4: Notes on tracking from table 4.2 and 4.3.

can be used for diagnosis as well.

Visualization of one complete result displayed in DVR rendering of the source dataset is in fig. 4.2.

One slice shown in fig. 4.7 displays how well the path tracking keeps to the middle of the lumen. Note that white dots represent only those places that are crossed through this slice with the path, nothing behind or in front of this slice is displayed.

Overall the method works best on well-distended ileum and duodenum, relatively reliably also on well-distended jejunum. Collapsed regions pose a problem, especially when multiple bends are collapsed on one place. But that is problematic also for a human operator to distinguish the intestinal direction or even distinguish between individual bends.

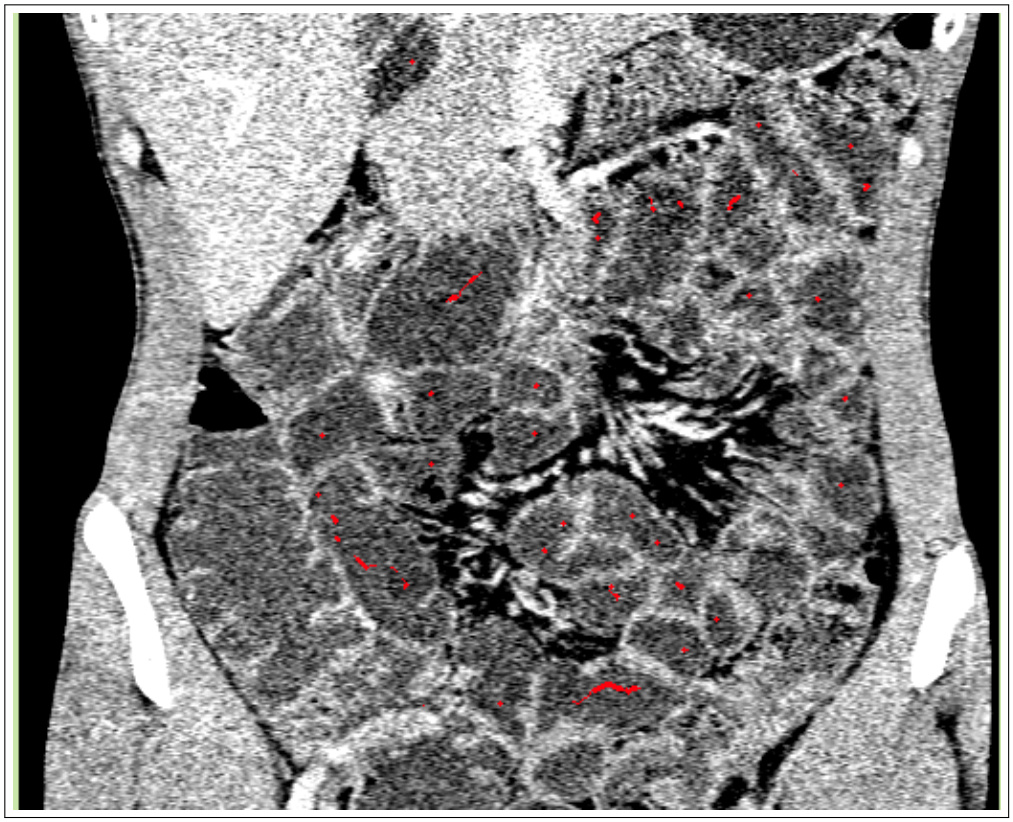

Figure 4.6: Patient 17 with good tracking. DVR rendering shown in figure 4.2.

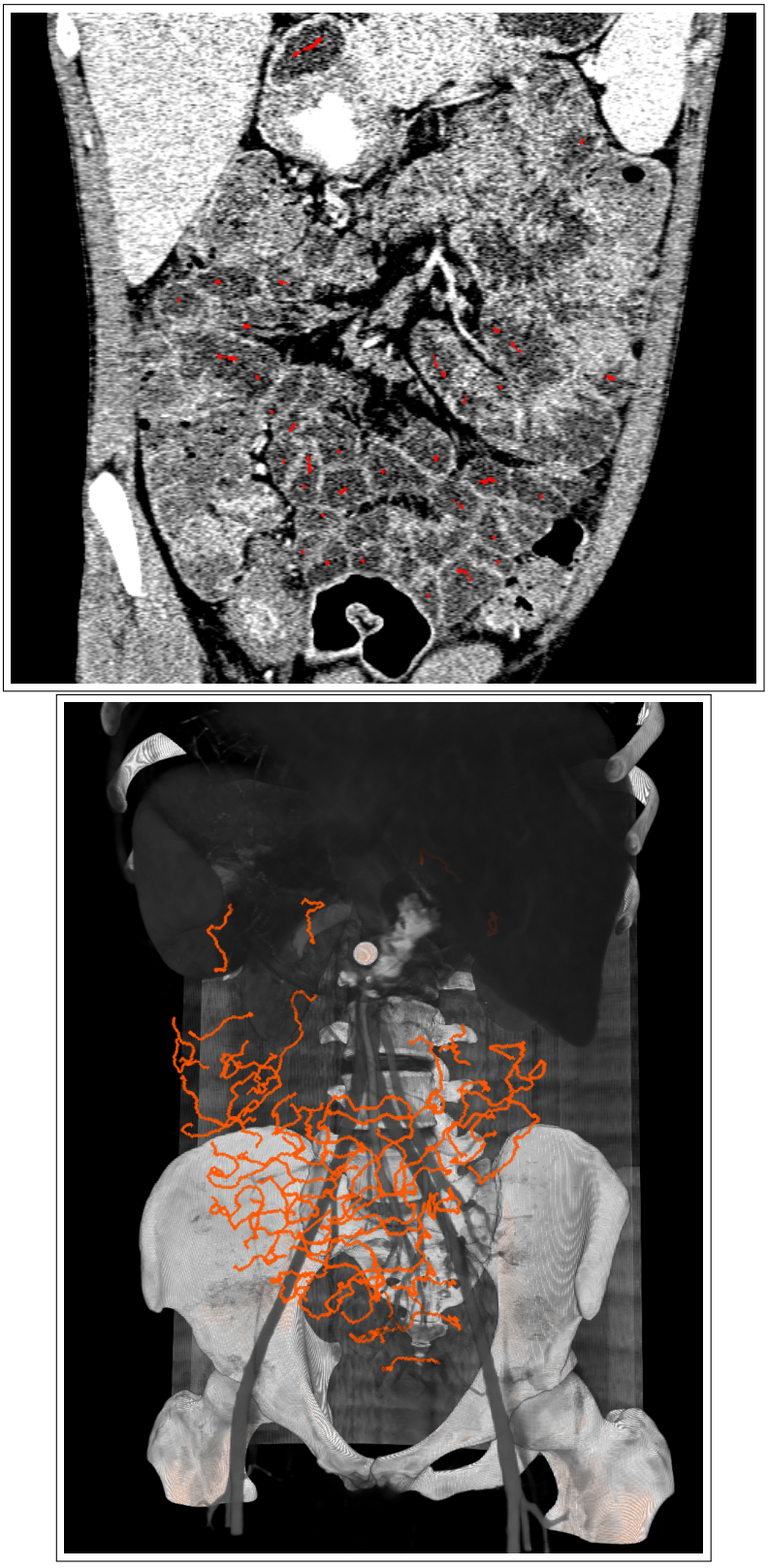

Figure 4.7: Patient 08 with the best segmentation and tracking in our database. Voxel-wise longest paths. Most of jejunum segmented well. Ileum segmented almost perfectly.

Figure 4.8: Patient 29 with imperfectly cleaned lumen. Tracking was possible, but only in water-filled regions. Parts filled with air bubbles and feces were not traceable.

Figure 4.9: Patient 31 with abnormal abdominal structure. Most of the abdominal region was filled with colon, small bowel found only collapsed in small regions. No paths visible in this slice.

Figure 4.10: Patient 32. Body fat reduces the lumen distention and thus decreases the probability for a good segmentation of intestinal lumen.

4.5.3 Time performance results

Our implementation was far from optimized, there were several unnecessary data exchanges through disk and almost all steps may be optimized and accelerated. Approximate times for individual computation parts were as follows:

- Probability cleaning and mask computation: around 30-40 seconds
- Distance map computation: around 10 seconds
- Paths tracking for the whole volume: around 1-5 minutes, depending on dataset

The path tracking step was implemented only as single threaded. Each major part is parallelizable, a significant (our guess is: almost linear) speed-up should be achievable when running on multiple CPUs.

Even in this unoptimized state the algorithm ran on a modern Core i7 processor in under 6 minutes (with a potential prospect of around 2 minutes or less), completely automatically.

Several simple optimizations might bring the total necessary time down to only a few minutes, which is an acceptable delay for a diagnosis that itself should then take much shorter time compared to a standard diagnosis using only raw slices. The subsequent diagnosis has all tracked segments readily available for real-time processing.

4.5.4 Conclusion and future work

We have presented an automatic algorithm for CT enterography small intestine tracking consisting of several known methods and adding a new algorithm based on region growing for the actual path tracking. The main purpose of this chapter was to center on a robust and reliable data preparation process and an automatic segmentation of individual segments of the intestine. We have shown that those segments correctly track the center of the lumen and can be used with a visualization process such as the ones stated in Oda et al. [2010]. The whole process is fully automatic without the need of any intervention.

Our future work will be centered on these areas that still need improvements: improving segmentation of hard-to-segment datasets by further incorporating *a-priori biological information* to the segmentation process and finding additional methods for the *topological analysis* to interconnect individual intestine segments into longer paths.

Chapter 5

Segmentation of lumen in tracked sections

5.1 Introduction

5.1.1 Goal

Having a good tracking method of individual segments of the small intestine presents us with a unique opportunity to use it for a precise per-voxel segmentation of lumen. This enables various methods of visualization and analysis of the intestine. Some methods will be mentioned in chapter 6. Segmentation of the lumen was largely inspired by our previous work in Horáček et al. [2010], although it was altered to fit the geometry of intestine and the segmentation itself was done iteratively instead of using dynamic programming.

5.1.2 Algorithm overview

The algorithm has an input dataset as the source of values and a set of paths from our tracking algorithm. The data is spatially transformed twice, segmented and back-projected:

1. Computation of slices perpendicular to the path.
2. Polar transformation.
3. Segmentation of the lumen border with an iterative algorithm.
4. Inverse polar transformation to perpendicular slices.
5. Inverse transformation to the original dataset.

5.2 Input data

We need two data sources for a precise segmentation.

1. A set of paths indicating multiple tracked intestinal segments (computed in chapter 4.4) which is a set of connected voxel coordinates.

2. Suitable data with the same or better resolution as the original dataset to perform a per-voxel selection.

5.2.1 Paths

Paths are provided by our previously mentioned tracking algorithm. Each path has a start and an end and a set of in-between voxels that are direct neighbors. Thus the distance between neighboring path voxels is between 1 and $\sqrt{3}$ voxels. We will be using these explicit voxel locations as a base for the position and some further filtering will be applied for other computations (such as path binormal and tangent computation). This will be mentioned later in this chapter.

5.2.2 Source volume

Source volume must have several properties to enable a good segmentation of the lumen.

- Good resolution
- Good difference between lumen and wall

We already have a selection of source and filtered data from the computation in chapter 3. A number of possible sources comes to the mind:

- Denoised source data (chapter 2, figure 5.1a)
- Result of objectness filter (see section 3.3.2, figure 5.1b)
- Other results computed per watershed

Original denoised source data were not used, because the only difference between a wall and a lumen is the difference between their values. As already discussed in chapters 2 and 4, this value alone is not a very reliable source.

All datasets computed on watersheds were discarded, because even though they represent the intestinal lumen relatively well, they are largely dependent on the local gradient magnitude (due to the watershed segmentation step) and are very sensitive to noise.

The per-voxel computed objectness filter has a good ability to differentiate between the wall and the lumen, much better than the source (denoised) data. Also it is by definition searching objects of a specified size and shape - in our case an intestinal wall of thickness around 1.5 ± 0.5mm (Cronin et al. [2010], Fernandes et al. [2014], see figure 5.2). The result of this filter is equal to or near zero in flat areas and a positive value on positively enhanced wall like structures. This is perfect for our case as this enables us to segment all near-zero values near the path voxels. Also the sensitivity to noise is much lower than in the watershed-segmented datasets while having a better resolution in complex areas (it is not "compressed" into watershed regions).

No further comparison and tests between different types of data were performed. Objectness filter was selected for its properties and simplicity of interpretation of the values and homogeneity in the lumen area. It is possible that a better data source might be found.

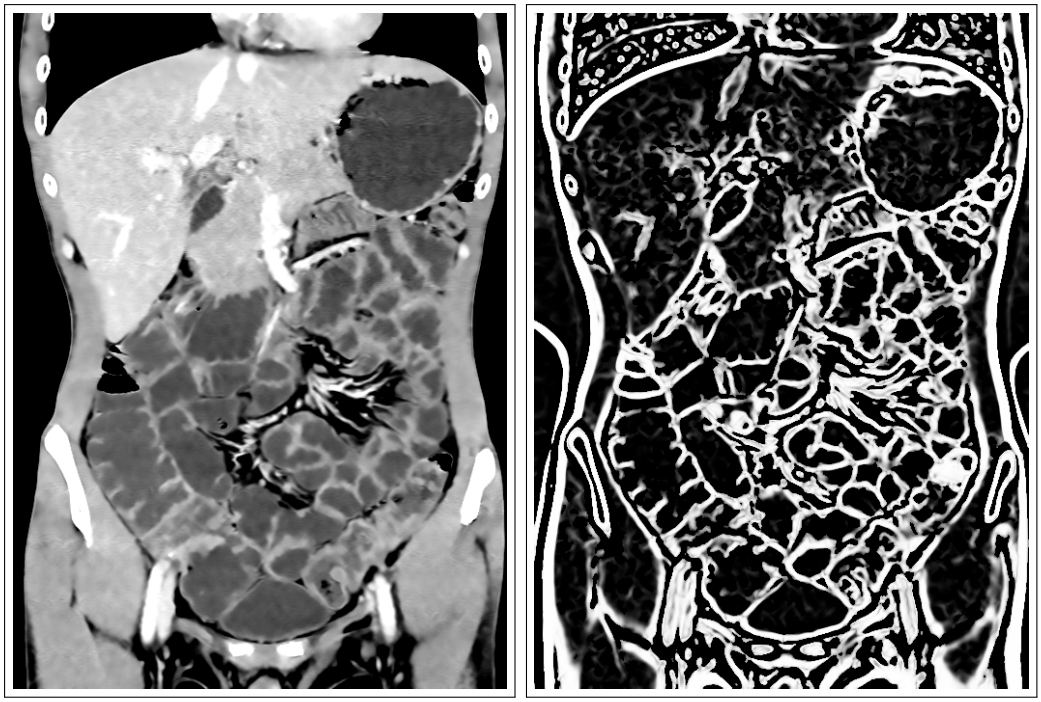

Figure 5.1: Frontal slice through (a) denoised source and (b) per-voxel computed objectness filter.

5.3 Perpendicular slices

We have estimated that the best position for obtaining a 2D slice with a good visibility of the intestinal wall would be a slice perpendicular to the path - using the Frenet frame (discovered by Frenet [1847]) vectors normal and binormal as a base.

The path might bend relatively quickly, so slices are computed not only for each path voxel, but also in-between. A good selection for the number of slices is between 4-10 slices per path step. Larger number of slices might be preferred in areas with high curvature. Selecting regular intervals has proven to have sufficient precision for our cause - we will be using exclusively regular sampling. Adaptive interval might improve the precision while lowering computational cost. However the segmentation algorithm would then need to be adjusted accordingly.

The **origin** (central pixel) of each slice is taken by linear interpolation between neighboring path voxels. Better interpolation or curve fitting is unnecessary, because comparing it to the linearly interpolated positions would create difference less than 1 voxel and that would have no effect on the algorithm, because the total precision of the segmentation is 1 voxel.

The computation of the slice orientation is done using the Frenet-Serret frame vectors tangent, normal and binormal. A set of steps is performed per original path voxels:

1. A tangent vector is estimated by central difference of path voxels.
2. Tangent vector is filtered with Gauss filtering.
3. Normals are computed by central difference of filtered tangent vectors.

Figure 5.2: Typical intestinal wall thickness in well distended segment. Two walls are next to each other, each around 1.5mm thick.

4. Binormals are computed by cross product of normals and filtered tangents.

Binormals and normals are then linearly interpolated for each slice and normalized to make an orthonormal base of each slice.

The reason for Gauss filtering of tangents is to have smoother changes in slice orientation. The position of the voxels needs to be precise, so only linear interpolation between neighboring voxels is used. However, the orientation is supposed to cover as much intestinal wall on the slices as possible, so a smooth transition is desirable. Using directly the result of central difference estimation would limit the number of possible orientations too much. Filtering with neighboring voxels makes smoother not only the tangent, but also the normal/binormal (ie. direction of the "up" and "right" vector). In our case we have used filtering with standard deviation $\sigma = 10.0$ voxels.

An example of a slice perpendicular to the tracked intestinal segment is shown in figure 5.3.

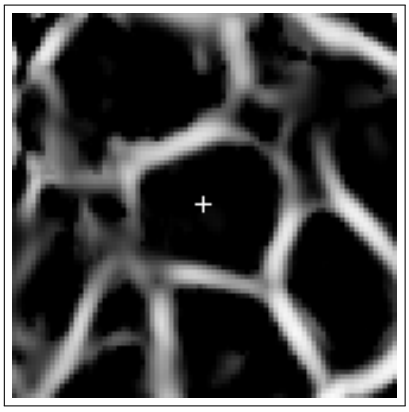

Figure 5.3: A slice of input data perpendicular to the path direction.

The shape of intestinal cross-section usually resembles a deformed circle, so polar transformation is used to unwind it into a form more suitable for segmen-

tation. An example is shown in figure 5.4. Our polar transformation uses the following notion in the text:

- r is the distance from the slice center
- α is the angle
- u, v are coordinates in the source slice with origin in the center

The resolution of the polar transformation, i.e. the number of possible values for r and α is estimated so that it approximately represents all source pixels in the input data. For a typical diameter of the small intestine around $20mm$ (Cronin et al. [2010]) and voxel size $0.5mm$, we recommend using radius of at least $2\times$ of the typical intestinal radius:

$$2 \times \frac{10mm}{0.5mm} = 40px \tag{5.1}$$

For the number of angles we recommend to estimate the size of pixel in polar coordinates in the position at least around $1.5\times$ the typical intestinal radius with the size of the voxel in the original data:

$$2\pi \times 1.5 \times \frac{10mm}{0.5mm} \approx 190px \tag{5.2}$$

Figure 5.4: Polar transformation of the slice from fig. 5.3.

Please note that the resolution and dimensions of the images shown here are not exactly computed as explained, they are slightly adjusted to better display the topic.

Of course, these two transformations can be combined together, and instead of sampling the data twice with all the sampling problems that come from that, we can sample directly the input data:

$$\begin{aligned} x &= origin_x(d) + r \cdot (binormal_x(d) \cdot cos(\alpha) + normal_x(d) \cdot sin(\alpha)) \\ y &= origin_y(d) + r \cdot (binormal_y(d) \cdot cos(\alpha) + normal_y(d) \cdot sin(\alpha)) \\ z &= origin_z(d) + r \cdot (binormal_z(d) \cdot cos(\alpha) + normal_z(d) \cdot sin(\alpha)) \end{aligned} \tag{5.3}$$

where *origin* is the position of the slice center in the input data and *binormal* and *normal* are normalized perpendicular vectors computed earlier in this chapter and d is the distance along the tracked path.

5.4 Iterative segmentation

5.4.1 Data shape

A segmentation of the intestinal lumen in one slice perpendicular to the path (figure 5.5) represented in polar coordinates (figure 5.6) would be a curve winding along the α axis from one side of the polar slice to the other.

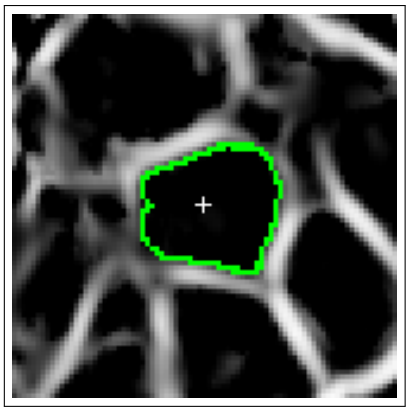

Figure 5.5: Segmentation of lumen in a slice from figure 5.3.

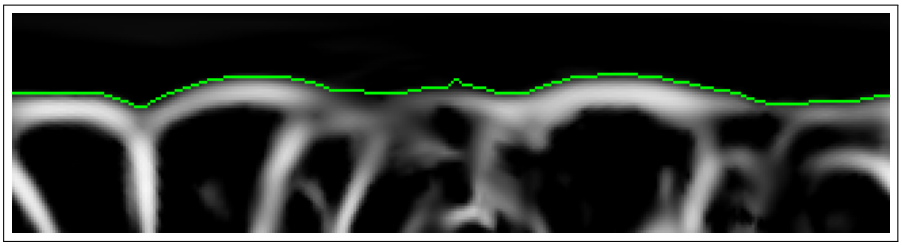

Figure 5.6: Polar transformation of the slice from fig. 5.5.

For the simplicity of explanation let's use the directions as on figures 5.4 and 5.6. We can see that the center of the lumen is in the upper part of this polar representation. Also the value in input data (Objectness filter) are 0 or near it for lumen.

If we place the polar transformations of all slices behind each other, we will obtain a 3D volume. Let's call its axes α, r and d (representing directions *right*, *down* and *forward* on figure 5.6). An example of such volume from a single tracked intestinal segment can be seen in 5.7.

Values with the coordinate $r = 0$ are very near 0 and the surface of the intestinal wall is represented as the top grey mountainous surface in the DVR visualization. We will try to find this surface in our segmentation algorithm.

Some of the large transparent or almost empty "holes" in the DVR representation are caused by visualization, but the actual values are nonzero even in these areas. However due to noise or other CT image acquisition problems some of these "holes" might be true near 0 valued areas in the intestinal wall. We need an algorithm that resembles the whole surface area as closely as possible, but at the same time does not leak into those mentioned holes.

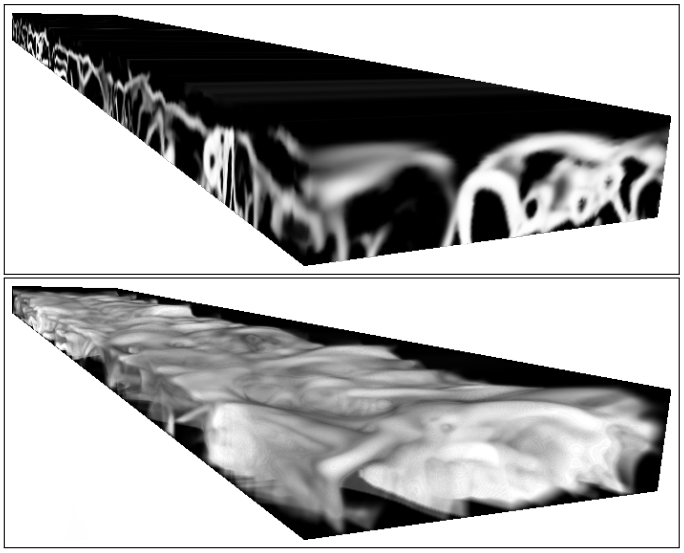

Figure 5.7: Direct value and DVR representation of a volume composed of polar slices.

5.4.2 Lumen border approximation with heightmap

We will approximate the surface with a height-map - i.e. a function representing the distance of lumen/wall border from the center for a given angle:

$$f_{hm}(\alpha, d) = r_{border} \tag{5.4}$$

where α is the angle coordinate in the polar slice, d is the distance along the tracked path and r_{border} is the distance of the lumen border from the center in a perpendicular slice at the (α, d) location.

This height-map can be stored as a 2D discrete rectangular array of values with the same resolution as we were sampling the angle α and traced path d. It contains integer values, which correspond to multiples of the voxel size along the r axis in the polar slices volume.

The algorithm will try to match the surface of the lumen by gradually moving the height-map in its direction:

1. Initialize this height-map with 0. That represent a "tube" around the central path with diameter 0:

$$\forall\alpha\forall d : f^0_{hm}(\alpha, d) = 0 \tag{5.5}$$

2. In each iteration step go through all height-map pixels and increase their value only if neighbors have the same value or greater:

$$\forall\alpha\forall d : f^{n+1}_{hm}(\alpha, d) = \begin{cases} f^n_{hm}(\alpha, d) + 1, & \text{if } f^n_{hm}(\alpha \pm 1, d) \geq f^n_{hm}(\alpha, d) \wedge \\ & \wedge f^n_{hm}(\alpha, d \pm 1) \geq f^n_{hm}(\alpha, d) \\ f^n_{hm}(\alpha, d), & otherwise \end{cases} \tag{5.6}$$

3. End if no height-map value was changed during the last iteration.

Values outside the definition of the height-map are handled by repeating the function, i.e.

$$\begin{aligned} f^n_{hm}(\alpha_{max}+1, d) &= f^n_{hm}(0, d) \\ f^n_{hm}(\alpha, d_{max}+1) &= f^n_{hm}(\alpha, 0) \\ f^n_{hm}(-1, d) &= f^n_{hm}(\alpha_{max}, d) \\ f^n_{hm}(\alpha, -1) &= f^n_{hm}(\alpha, d_{max}) \end{aligned} \tag{5.7}$$

The algorithm has at most r_{max} iterations, where r_{max} is the resolution of the polar slices along the r axis. So this is effectively less or equivalent to the computational complexity of a simple linear filter on the volume of polar slices.

An example of a result height-map can be seen in figure 5.8 and 5.9.

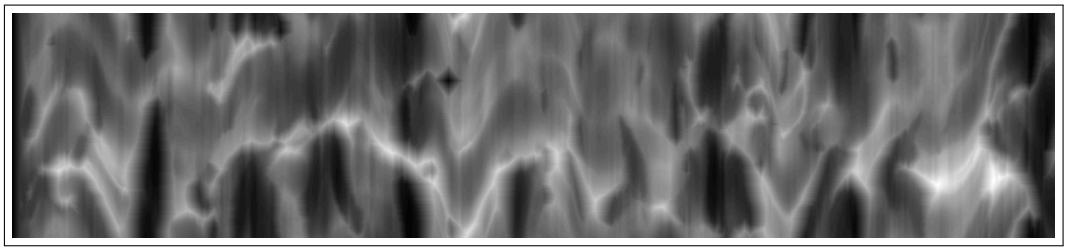

Figure 5.8: Height-map computed in our iterative lumen approximation algorithm.

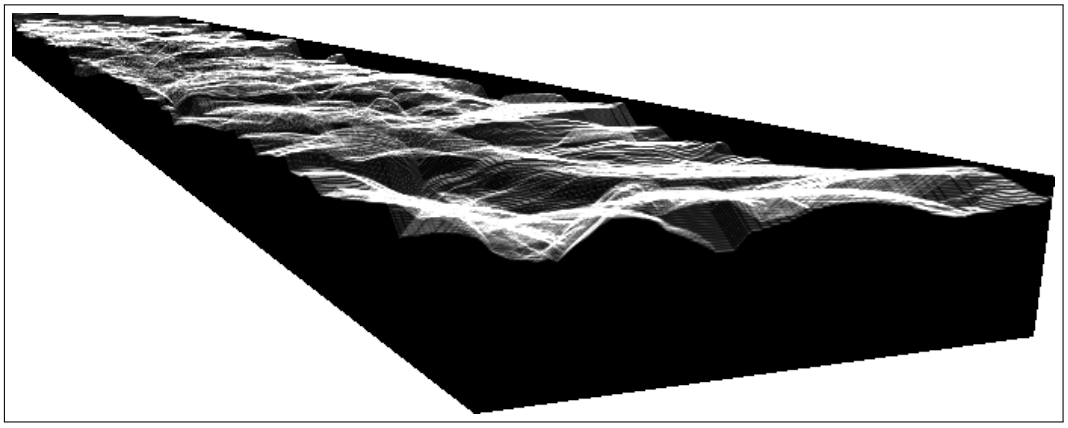

Figure 5.9: 3D rendering of the height-map from figure 5.8. (Note that $f_{hm}(\alpha, d) = 0$ means low distance from the center of lumen, which is located on the upper plane in the 3D rendering. So dark areas in fig. 5.8 are high peaks in this figure).

5.5 Inverse spatial transformation

The last step remaining in lumen border segmentation is projection of the height-map $f_hm(\alpha, d)$ back to the input dataset. As the input dataset has the same or better resolution than the original CT dataset (requirement in section 5.2), the result of the segmentation might be used as a segmentation in the original dataset as well.

The transformation from the height-map to the original dataset space is done similarly to the polar coordinates transformation. By substituting the value of height-map into equation 5.3 we get the following transformation equation:

$$\begin{aligned} x =& origin_x(f_{hm}(\alpha, d))+ \\ & + r \cdot (binormal_x(f_{hm}(\alpha, d)) \cdot cos(\alpha) + normal_x(f_{hm}(\alpha, d)) \cdot sin(\alpha)) \\ y =& origin_y(f_{hm}(\alpha, d))+ \\ & + r \cdot (binormal_y(f_{hm}(\alpha, d)) \cdot cos(\alpha) + normal_y(f_{hm}(\alpha, d)) \cdot sin(\alpha)) \\ z =& origin_z(f_{hm}(\alpha, d))+ \\ & + r \cdot (binormal_z(f_{hm}(\alpha, d)) \cdot cos(\alpha) + normal_z(f_{hm}(\alpha, d)) \cdot sin(\alpha)) \end{aligned} \tag{5.8}$$

A result of transforming the whole height-field into the original dataset is shown in figure 6.1.

Chapter 6

Visualization

This chapter gives several hints for good visualization of results obtained throughout this thesis. The main focus is understandably the intestinal lumen, intestinal wall, their properties and topological relations within the body. We report several techniques that we have used during our research and proved to be useful for this case.

6.1 Brief DVR introduction

3D visualization of CT data (Drebin et al. [1988], Levoy [1988], Weiskopf [2007]) is best achieved with DVR (direct volume rendering). This method is used for false color projection of the volumetric data to a 2D plane. The algorithm computes volume orientation in respect to the camera, generates rays from the camera through the volume and samples and accumulates values along the ray. Sampling might be in regular intervals, jittered intervals, by DDA (digital differential analyzer) algorithm through all voxels, combined with anti-aliasing etc. Sampled values do not have to be interpolated or can be interpolated with trilinear or more complex interpolation.

Sampled values are then transformed with a transfer function LUT (look-up table) that projects a value in HU (in our case from range -1000 to +3000) to a color value and transparency. Custom LUT can be used to generate remarkable and easy to interpret results. Many images throughout this thesis are rendered using a DVR algorithm.

There are many variants, especially in recent years when the computer hardware advancement allowed for real-time rendering of volumetric data or off-line rendering with global illumination. We will deal only with a simple DVR with a 1D RGBA custom transfer function. That is sufficient for presentation of our datasets.

There are also other methods for volumetric data projection, such as MIP (maximum intensity projection), iso-surface rendering, X-ray simulation, etc. We will not deal with those in this thesis.

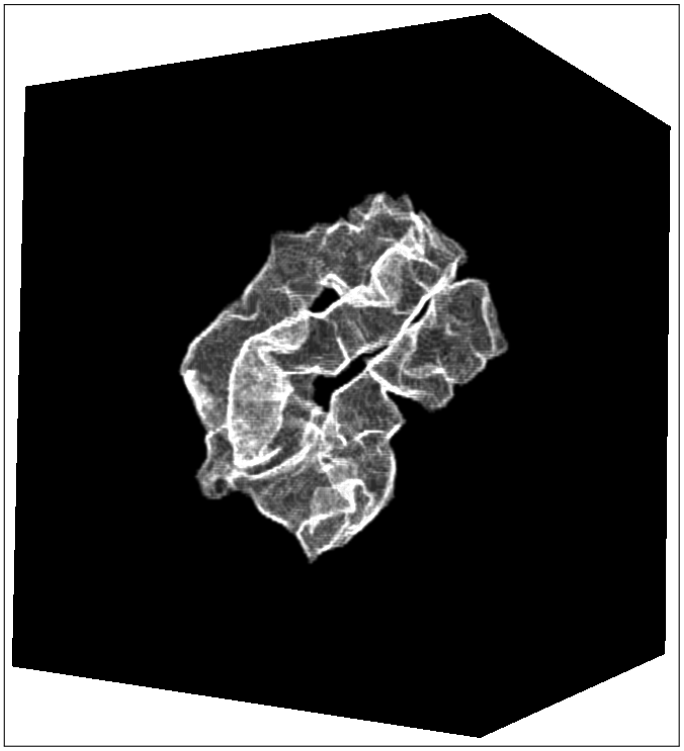

Figure 6.1: Visualization of the lumen border in one tracked segment of intestine.

6.2 Topological view of a single spatially complex segment

The single lumen border obtained from a segmentation in chapter 5 can be visualized in 3D to enable quick overview of the width of the small intestine. Also even though we try to make every step as robust as possible, we are working on imperfect real-world data, there might be errors and this is a very quick overview if any such error is present. A simple DVR projection of a binary segmentation mask is shown in figure 6.1.

We can also use the segmentation as an overlay mask over the original dataset to enable the radiologist operator evaluate the original data and have the information about the lumen at the same time. An example is shown in figure 6.2.

Another method to display a single intestinal segment is to project the segmented height-map on a straight line and thus effectively straighten the intestine. An example is shown in figure 6.3. This view is suitable for analysis of bowel obstructions.

6.3 Tracking slice view for wall analysis

We have tried several methods for a more thorough analysis of intestinal distention, wall saturation and thickness. The first method does not need further preprocessing, it only displays tracked slices along the intestinal path. An example is shown in figure 6.4.

The radiologist can move simultaneously on 4 slices through the path of the intestine. One slice is perpendicular to the tracked path, other three represent

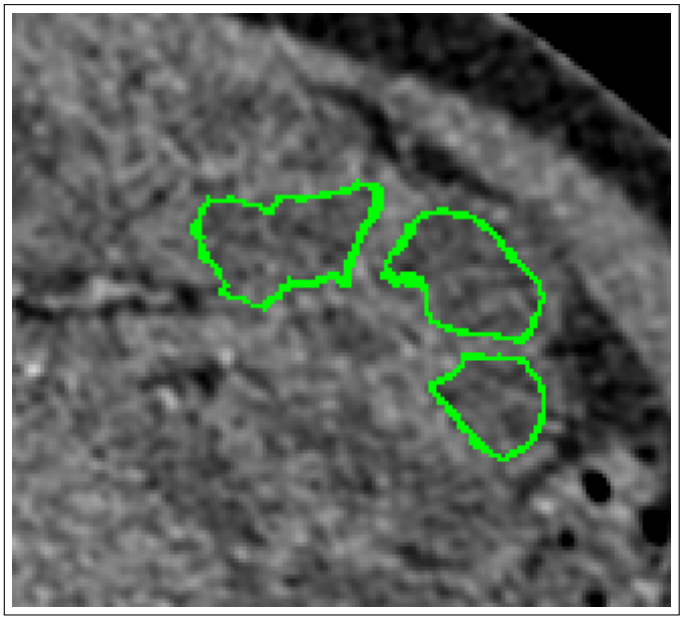

Figure 6.2: Axial slice of the original CT enterography data with an overlay of the lumen border segmentation.

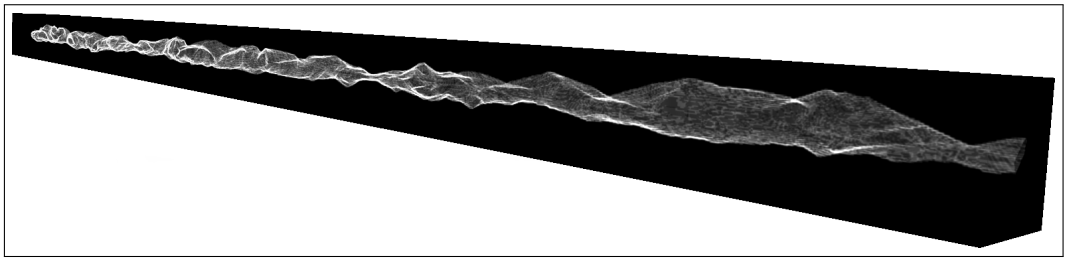

Figure 6.3: Lumen border from figure 6.1 projected onto a straight line.

standard orthogonal slices for CT visualization. Simultaneous movement allows for wall thickness assessment in complicated areas.

Figure 6.5 shows another possibility - to display a slice following the whole path direction. We have been inspired by CPR (in Kanitsar et al. [2002]) and its variants. In our case we have a path with greatly varying direction. We have thus generated images along the path - one axis in the result cut is a selection of slice along the path (axis d), the second axis is projected onto a rotated vector in the Frenet frame plane $N \times B$ (normal and binormal) in each slice computed in section 5.3. Rotation angle stays the same for one projection.

This way the radiologist has a complete overview along the intestinal segment. All walls can be examined by rotating this slice along the axis. This projection has the same source data as in figure 6.4. The inflammation can be observed in fig.6.5, top right section (about $\frac{1}{3}$ along the complete path length).

All such slice-based views in figures 6.4 and 6.5 can be overlaid by lumen border indication, such as in figure 5.6 and 5.5. The *height-map* data stored in f_{hm} (see chapter 5.4.2) are available for this task without problems. We have omitted here the overlay to better display the wall/lumen interface.

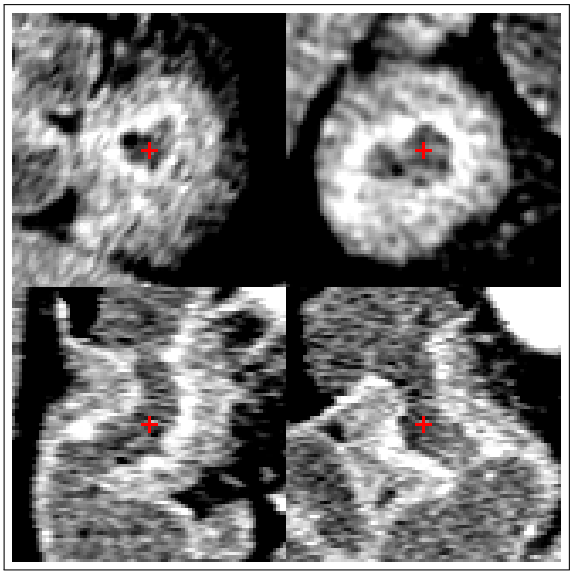

Figure 6.4: Slices through the currently examined path point along the tracked intestine. Top left is a slice perpendicular to the path, other three represent axial, frontal and sagittal planes. This example shows an inflammation in the intestinal wall.

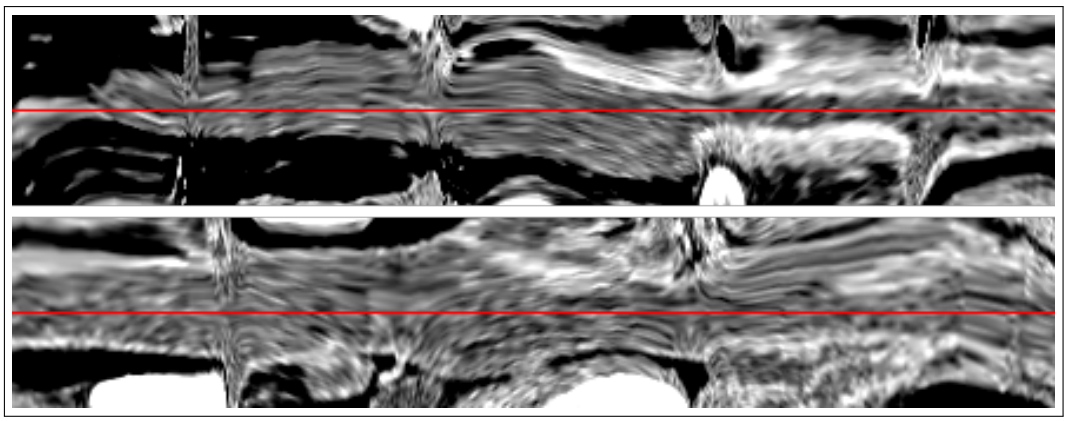

Figure 6.5: A slice following the intestinal path. This slice is parallel to the path direction and can be rotated along the axis. For visualization purposes it was divided into two parts - top right end continues on bottom left border.

6.4 Advanced wall visualization

We have also prepared methods for advanced wall analysis. For this we need perpendicular direction to the intestinal wall. In chapter 5 we have computed the location of lumen border for each step along the intestinal path. To estimate the perpendicular direction through the wall from the lumen border we need gradient estimation. Sampling the wall at a different angle creates errors in wall thickness.

Gradient estimation is best computed over objectness function computed in section 3.3.2. This way we get a clean gradient exactly along the lumen border. By sampling this gradient at the lumen border and unwrapping it along the path and polar coordinates, we get an estimation of perpendicular direction through the intestinal wall.

An example of sampled and unwrapped gradient along the whole tracked

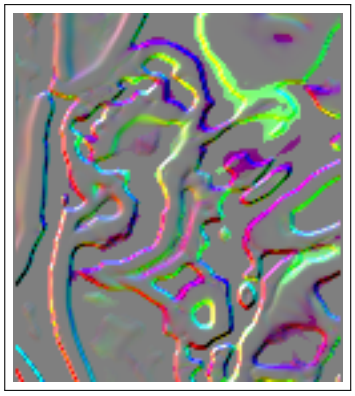

Figure 6.6: Gradient computed over objectness-filtered data. This slice roughly corresponds to frontal slice in figure 6.4 - bottom-left.

intestinal segment is shown in figure 6.7. We did not include this image only for its aesthetic appeal, but to show the irregularities of the gradient along the intestinal track (horizontal axis). If the gradient computation were not necessary, it would appear much smoother. By using the measured gradient we are significantly correcting the sampling direction for the intestinal wall.

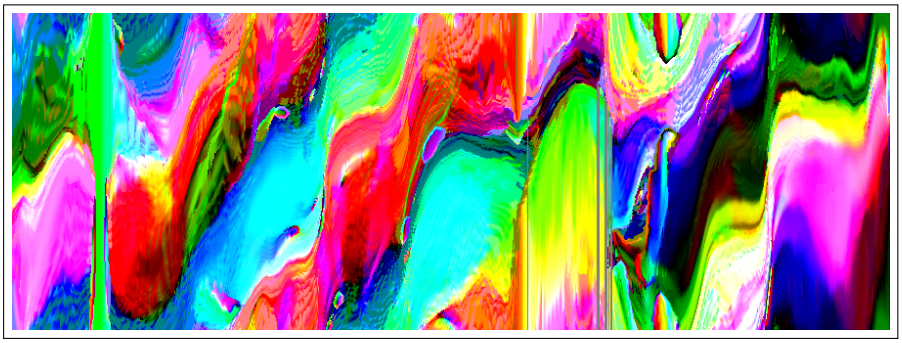

Figure 6.7: Gradient sampled along the path and at the lumen border. Axes on this image correspond to axes on figure 5.8 (although it is a different intestinal segment from figure 6.4). Colors R,G,B represent normalized gradient coordinates X, Y, Z.

We will proceed to sampling of the intestinal wall in the perpendicular direction on each lumen/border point. We need information as close to the original data as possible, without the noise. So we will be sampling the denoised original dataset. Samples must be dense enough to hit all voxels.

After we have samples, we try to fit a Gaussian curve to the intestinal wall. Its parameter σ gives us an information about the width of the wall. The peak μ of a fitted curve gives information about the contrast saturation of the intestinal wall. These numbers roughly characterize the wall within the neighborhood. It is not a completely reliable technique, but handles many situations well and brings attention to potentially important places. An example of this fitting is shown in figure 6.8. Gaussian curve is not strictly necessary, we have selected it for its ease of use. Any other fitting that can express the wall width and saturation might be applicable (see for example Näppi et al. [2010]).

The information about saturation is another important value that might reveal

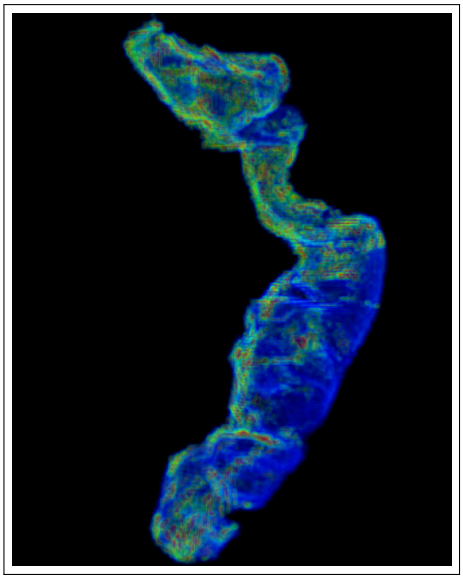

Figure 6.8: DVR visualization of the σ parameter fitted on wall, projected on the segmented lumen border. We have used *jet* ramp as a color LUT. Blue values indicate thin wall, red values indicate thick wall. Yellow/red part in the narrow segment corresponds to position over inflammation in figure 6.4.

potential problems. We have used a sample from the denoised original dataset at the μ position of the fitted Gaussian curve and applied it similarly to the segmented lumen border. An example of the same dataset (this time with the saturation information) is in figure 6.9. This example is even more revealing the inflammation of the intestinal wall.

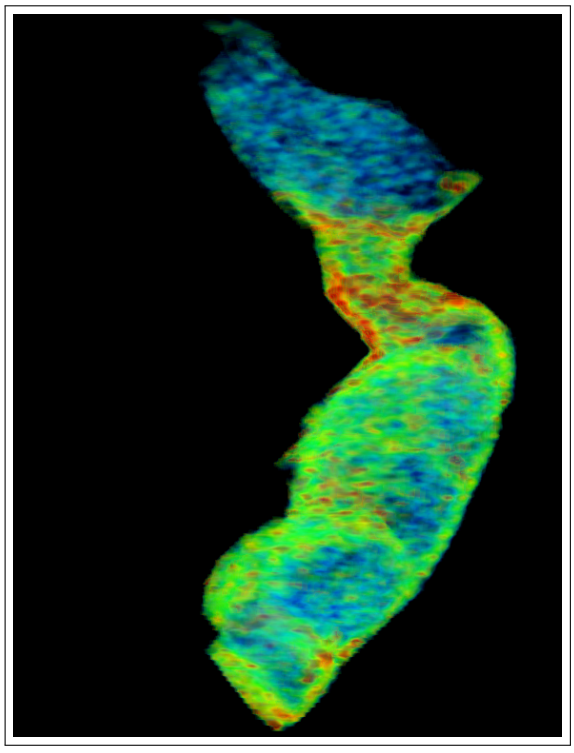

Figure 6.9: DVR visualization of the contrast saturation estimation of the intestinal wall.

A more complex method for intestinal wall analysis has been given in Näppi et al. [2012]. Similarly to previous visualization a line perpendicular to the wall is projected through the wall. A polynomial is fitted and a support vector machine is used for Crohn's disease prediction. This method would ideally complement the whole dataset visualization.

6.5 Complete dataset visualization

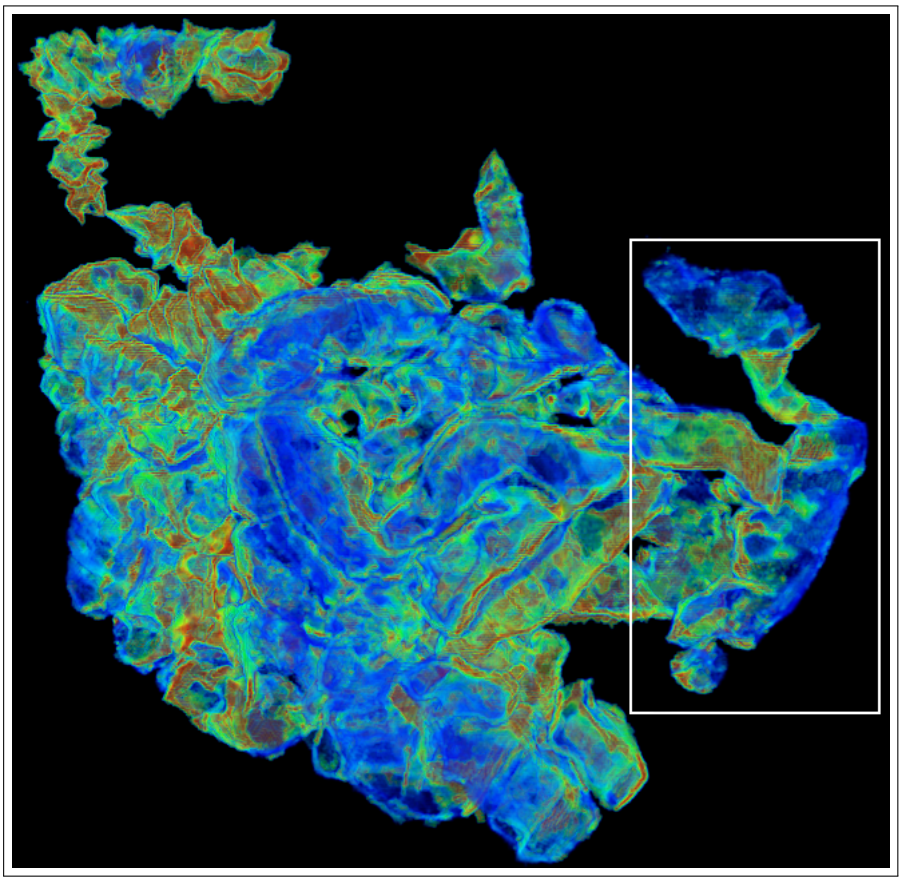

Figure 6.10: DVR visualization of the σ parameter fitted on wall, projected on the segmented lumen border. The intestinal segment from chapter 6.4 can be seen highlighted on the right side.

Methods for displaying individual segments might be generalized for the whole patient. An example of wall width estimation is shown in 6.10.

The next figure 6.11 shows the maximal contrast saturation of the intestinal wall - the inflammation region is here much more apparent as a red section.

We have also prepared for completeness a rendering of the whole body, including the complete segmented intestine - figure 6.12. Such view might be important for a surgeon before any surgery of the small intestine takes place. A knowledge about the actual anatomy shown with a good overview makes the surgery preparation stage much simpler.

This rendering shows the whole bulk of the intestine, but of course any combination of the techniques mentioned in this whole chapter is possible. Including

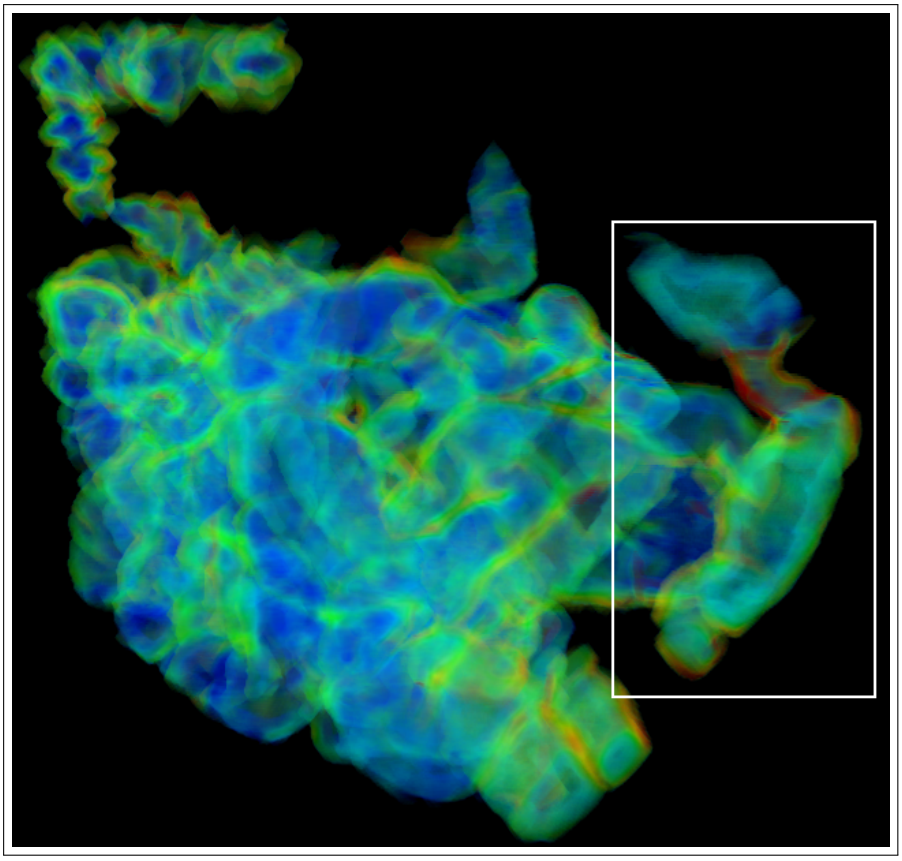

Figure 6.11: DVR visualization of the contrast saturation estimation of the intestinal wall applied to the whole segmented intestine. The segment from figure 6.9 is highighted on the right side.

visualization of individual segments, combined visualization of wall properties estimation with whole body rendering, etc.

From the technical point of view the current computer hardware is at a point, when a dataset of size roughly 512^3 can be displayed in real-time or even animated without any problems.

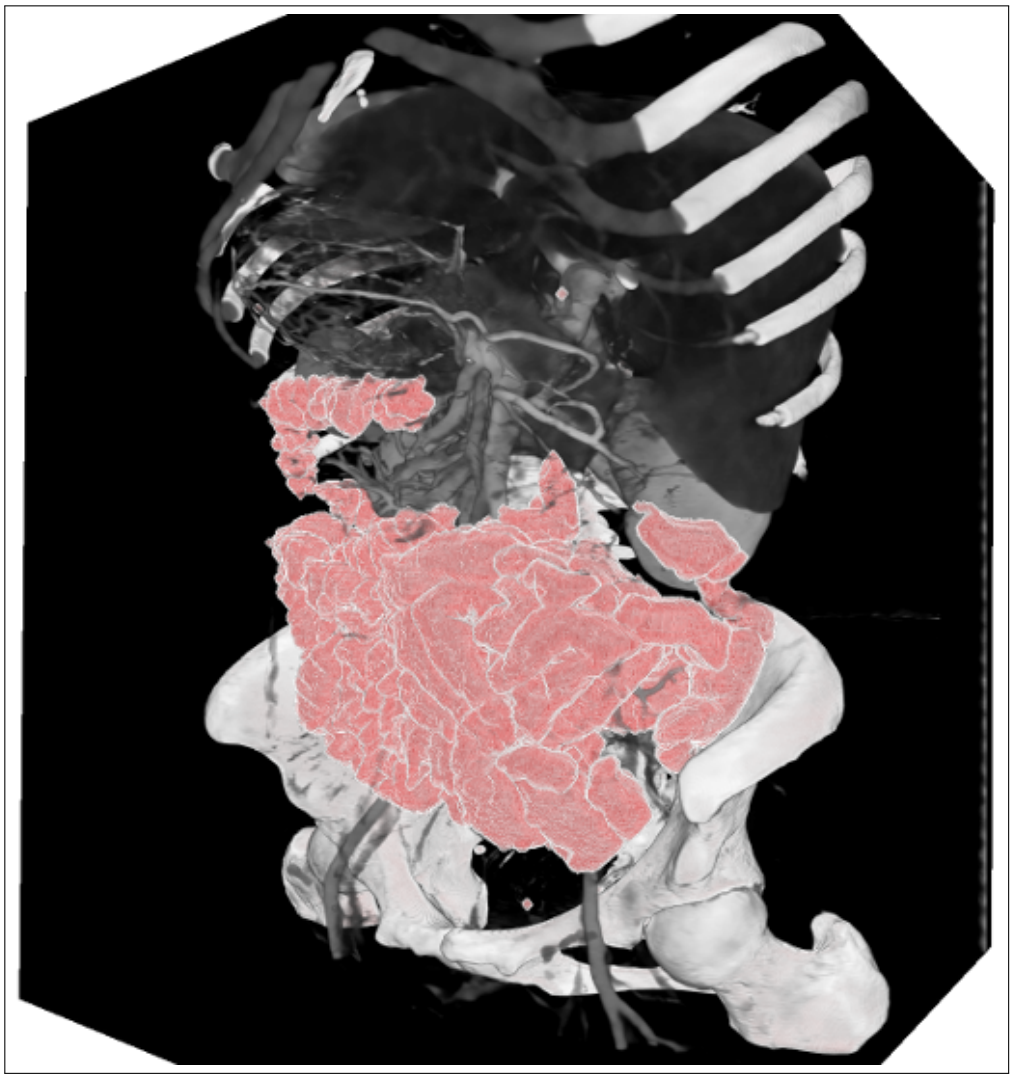

Figure 6.12: DVR visualization of the whole dataset with strongly highlighted segmented intestinal lumen.

Chapter 7

Implementation overview

7.1 Workflow pipeline

A schema of the first part of the whole pipeline described throughout this thesis is shown in figure 7.1. All major steps of the computation for a daily practice are shown. The only manual step before visualization is a potential manual cleaning of segments erroneously shown as intestine. We have created a simple GUI tool visualizing the location of each segment by the intersection with current slice and the length of a selected segment. User can remove said segment with one click.

Training set creation is not included in the schema, only the actual training data in yellow input cells. Training dataset is created by manual marking of regions belonging into lumen or wall and extracting the necessary feature values from per-region datasets computed over the training input dataset.

7.2 Implementation details

All implementation was done in C++, Cuda, OpenCL, GLSL and bash.

Due to the research nature of the process it was subdivided into several easy to manage applications. The main parts were:

1. Utilities for extracting data from DICOM format
2. Denoising utility based on OpenCL
3. Cuda-based utility for computing Hessian-based objectness filter
4. Commandline C++ utilities for processing simple operations over the datasets. Built-in support for text-based scripts for performing a long sequence of operations.
5. Qt-based multiplatform viewer for volumetric data. Support for .rek (.raw), .mrc and custom built-in formats. Basic volumetric operations, masking, multiple datasets, basic morphologic operations, evaluations and segmentation algorithms that need user input.
6. Qt-based multiplatform 3D ray-casting viewer for volumetric data. Support for .rek (.raw), .mrc and custom built-in formats. Ray-casting and sampling

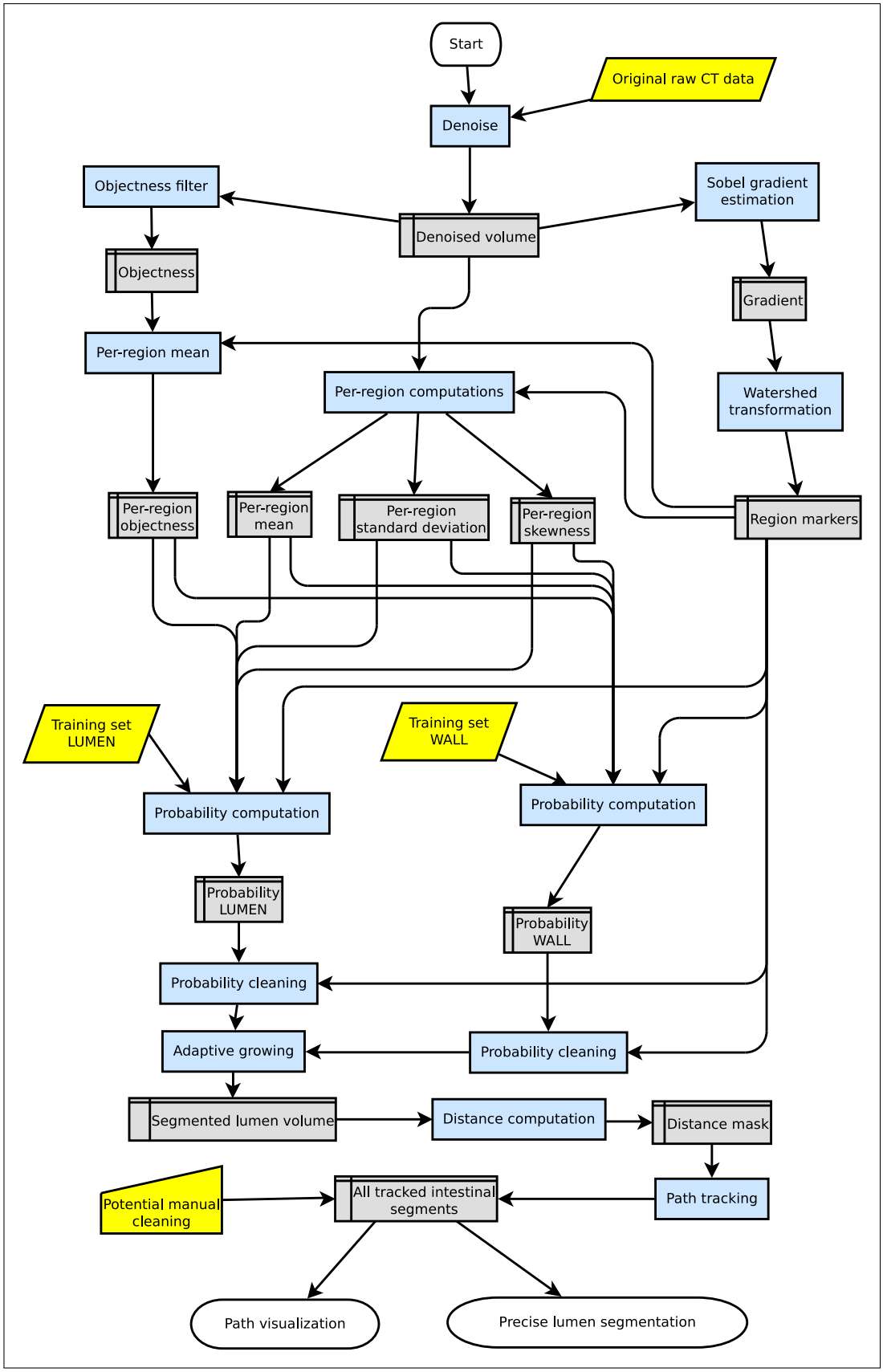

Figure 7.1: Workflow pipeline from source data to tracked intestinal segments.

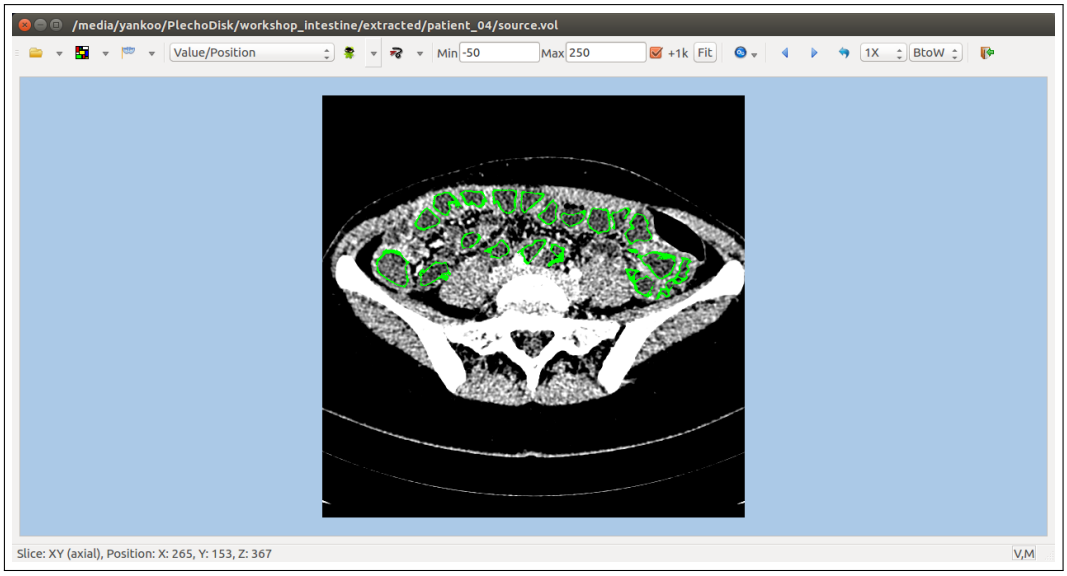

Figure 7.2: Qt-based multiplatform viewer of volumetric data.

done in real-time based on OpenGL Shading Language. Rendering modes include DVR, DVR+light, X-ray simulation, MIP. Support for stereo rendering and full-screen mode. User editable 1D transfer functions for DVR rendering.

All implementation is scriptable. User input is needed only during two situations - DICOM data extraction to select correct dataset and cleanup of wrong paths after tracking is finished. Everything else runs automatically.

All algorithms and applications are created so that compatibility between different operating systems is seamless. Applications were developed and used on all three major desktop systems - Windows, Linux and MacOS X.

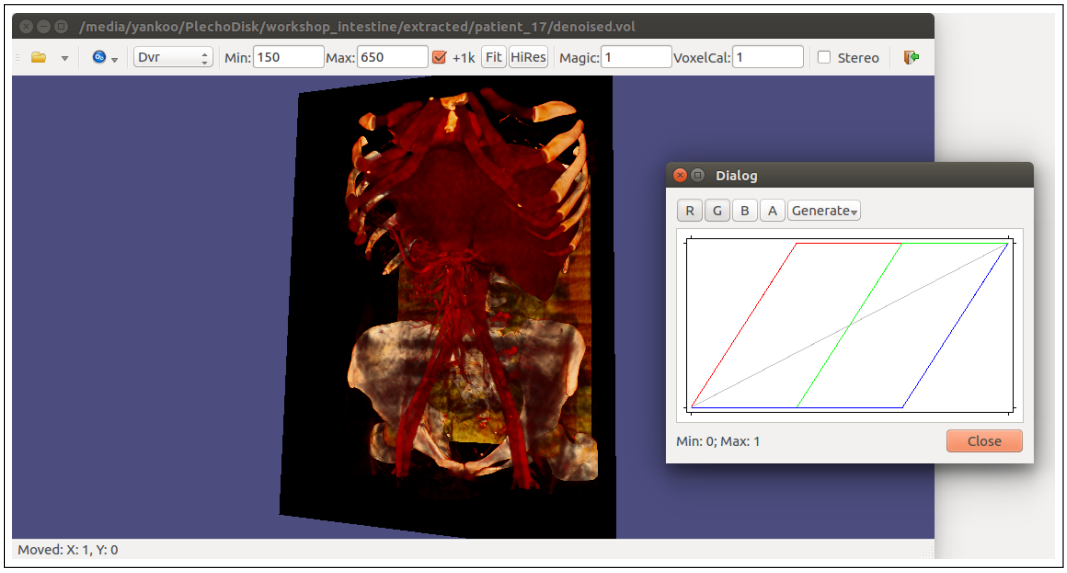

Figure 7.3: Qt-based multiplatform ray-casting 3D viewer of volumetric data.

Chapter 8

Conclusion

We have succeeded in designing a process to enable segmentation, tracking and uncluttered visualization of thin-slice CT enterography data. We have designed several new algorithms and validated our results on real data. To our knowledge we have not found another published work that would manage to give results matching ours.

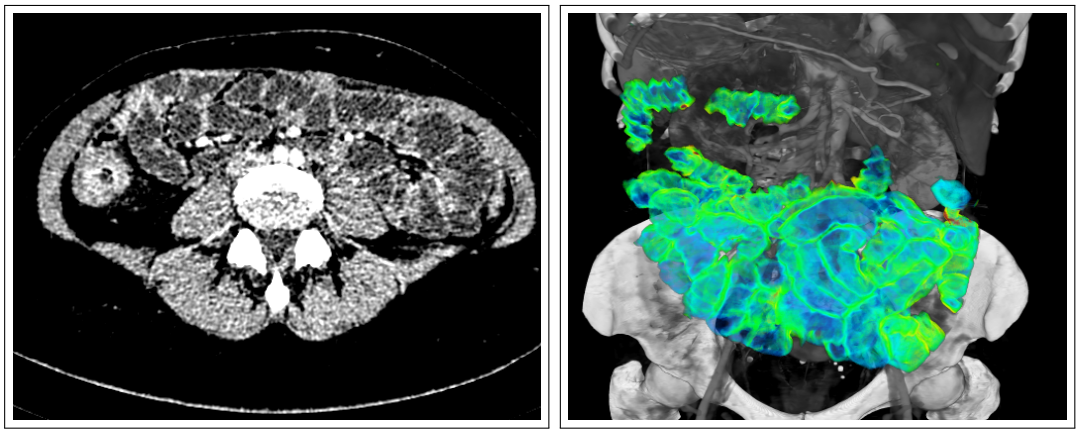

Figure 8.1: We have achieved to process the CTE data into a segmented and tracked small intestine information.

The whole process is subdivided into several parts analyzed and discussed in this thesis:

1. Fast, efficient and high-quality volumetric denoising of the input data
2. Watershed segmentation, processing of watershed regions, computation of region feature descriptors and classification
3. Segmentation of the bulk of small intestine organ by filtering the classification results
4. Robust tracking of intestinal segments
5. Segmentation of intestinal lumen on tracked data
6. Visualization and analysis

We have contributed to these steps and published our work in international journals and conferences. This thesis reports on our results and follows the same logical order of operations as the actual practical algorithm. All steps in our processing pipeline are fully automatic with several possibilities of user input to correct or improve the process.

From the practical point of view we have optimized all algorithms to keep the total time requirements of the pipeline within several minutes on a commodity hardware with good scaling possibilities on higher-end hardware. This is an important fact to enable widespread use of our methods.

8.1 Denoising

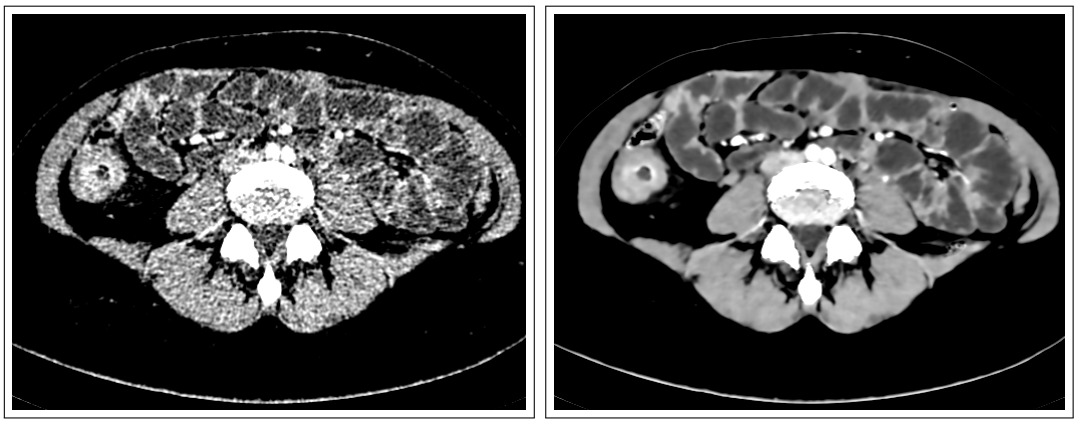

Figure 8.2: Denoising result example.

High quality denoising of volumetric data is not computationally cheap. We have presented an algorithm to compute a proven nonlocal means algorithm in its volumetric variant on one abdominal CT scan in under one minute on a modern GPU. The same algorithm on a CPU would take hours to compute.

This step was crucial in processing high resolution thin-slice CT enterography data. Otherwise the data were unfit for any reasonable processing approaches.

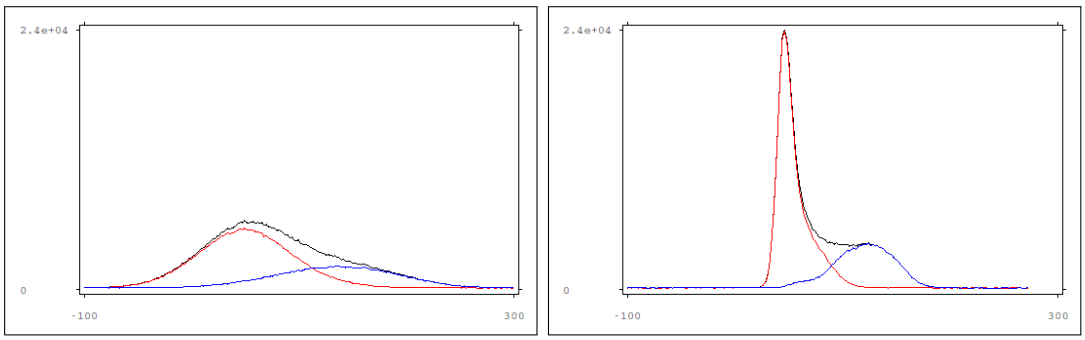

Figure 8.3: Histogram of the small bowel area on a CT enterography scan. (a) Raw CTE scan. (b) Denoised dataset. Red histogram indicates lumen, blue indicates intestinal wall and black is a sum of all voxels within small intestine.

The denoising process significantly improved the results of CTE data processing. It increased the possibility to recognize and segment important parts of abdominal CTE scans.

8.2 Watershed processing

We have presented a behavior analysis of watershed-segmentation applied on CT enterography data and selected a number of descriptors of watershed regions. These selected descriptors enable a relatively reliable classification of the input data. We have also given a discussion of different descriptors and their efficiency at classification.

We have managed to create probability maps indicating the location of intestinal lumen and wall within the body without any a-priori anatomical information.

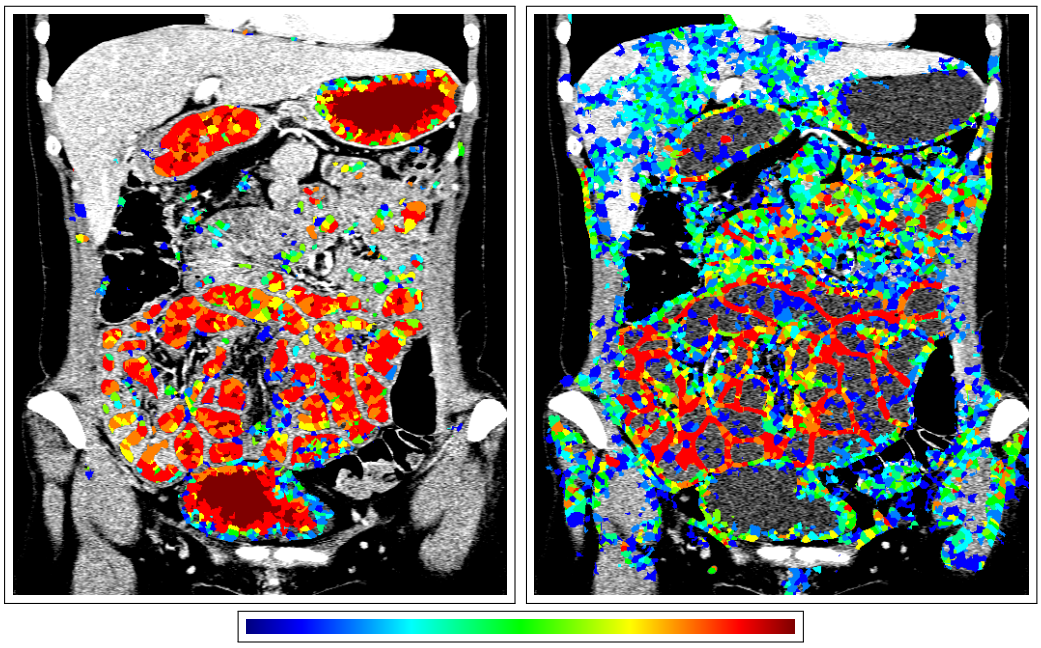

Figure 8.4: Computed probability maps on watershed regions. These maps indicate the probability of lumen and wall respectively.

Different settings during scanning create slightly different datasets. We have hinted at an idea to automatically match various classification training sets on the given data.

8.3 Tracking of the intestine

We have used the probability maps to compute a segmentation of the intestine. An approach of combining intestinal lumen and intestinal wall probabilities creates a binary segmentation of the CT dataset that provides the location of small intestine within a patient's body.

A new algorithm for robust tracking of the segmented lumen was presented. A two-iteration prioritized flood-fill with backtracking was designed from the ground up to solve the most problematic aspects of CT enterography data tracking.

We have overcome the problems of leaking due to errors in data and the algorithm is able to track even very complex paths of the small intestine.

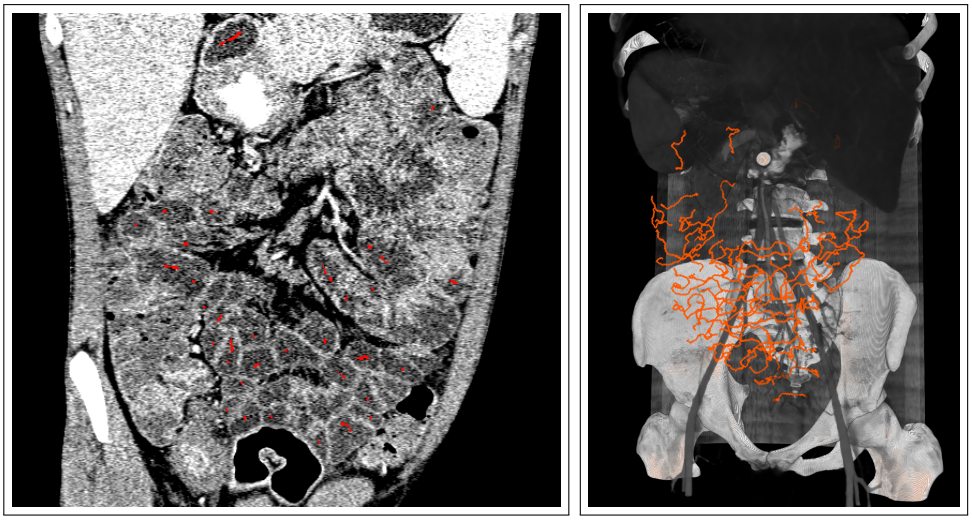

Figure 8.5: A result of our algorithm designed for tracking the small intestine. Frontal slice with intersecting paths and a DVR rendering of the whole dataset.

Figure 8.6: Result lumen border segmentation overlay on an axial slice.

8.4 Segmentation of tracked lumen

We have adapted one of our previous algorithms (previously used for precise segmentation of femoral head corticallis) for use on tracked intestinal data. It provides us with an exact location of border between the lumen and wall.

A cross-section of the previously tracked intestinal path is used to generate an unwrapped continuous polar transformation of the small intestine segment. An

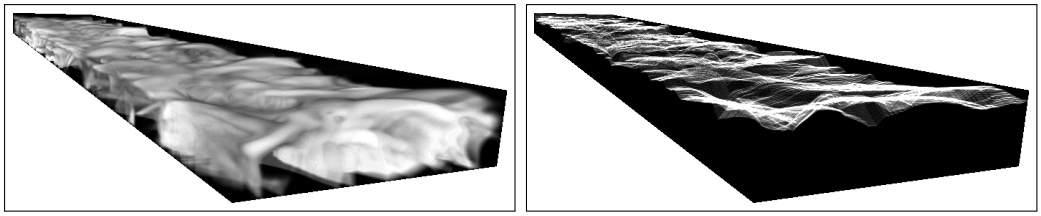

Figure 8.7: Intermediate *unwrapped* part of one segment of tracked intestine in polar coordinates and the result segmentation computed over it.

iterative algorithm is then used to descend on the border of lumen and wall and backward transformation to volume space creates a nice continuous wrapping of the intestinal lumen.

8.5 Visualization

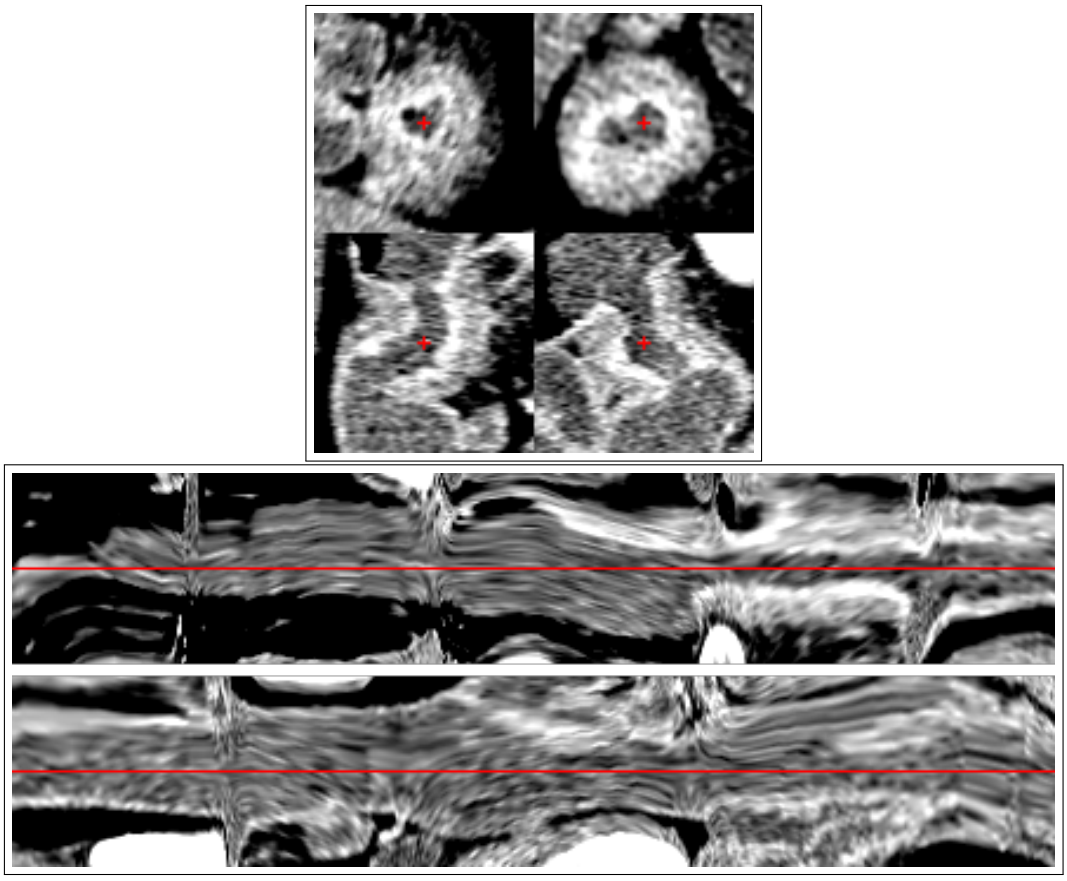

Figure 8.8: Visualization in 2D slices through the intestine.

The ultimate goal of the work on CTE data processing was to simplify and quicken the analysis of the data, the assessment and diagnosis of inflammatory diseases in small bowel, obstructions and other problems. We present several ideas how the results of intestine segmentation might be visualized and analyzed.

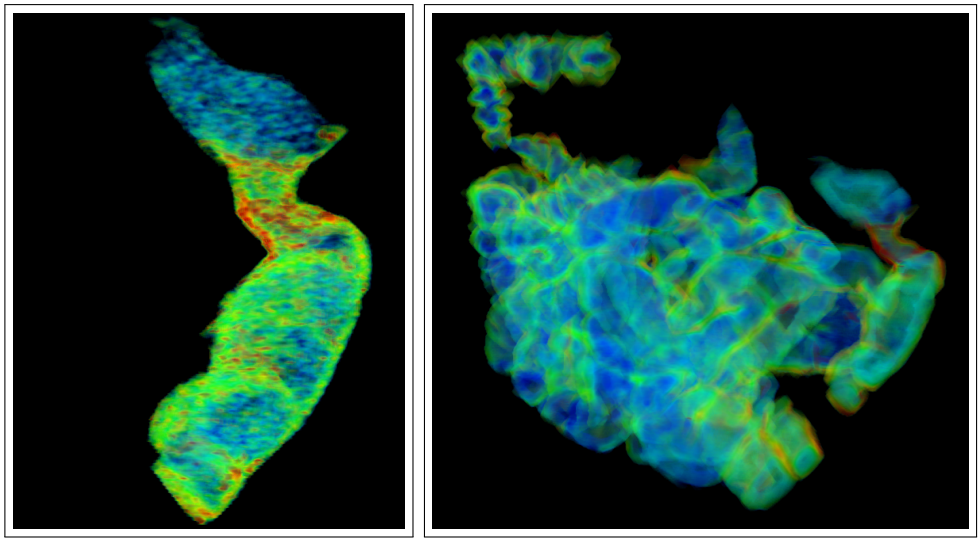

Figure 8.9: 3D visualization of one intestinal segment with inflammation and the whole segmented bulk of small intestine.

8.6 Diagnosis improvement results

A significant subjective relief and a better concentration on diagnosis was observed even on our small set of data. The reduction in time can be observed when all already displayed paths are marked in the dataset, so that the radiologist concentrates only on parts not visited yet.

It is hard to quantify the exact diagnosis improvement as every patient is unique and every diagnosis takes significantly varying amount of time. However, our approach brings several meters of tracked intestinal data ready for a fast overview through healthy parts and enables the inflammations to be viewed without disturbances emitting from a constant orientation in the dataset.

However a more in-depth analysis of the diagnosis time and conclusions can be done in the future and on a larger set of datasets - ideally parallel to traditional techniques to have a direct comparison.

8.7 System automation

All steps of the system from the loading of the original raw CT dataset up to individual segment tracking are **fully automatic**. There are basically only three moments of user interaction and only two of them are performed regularly:

1. **Manual creation of a training set** for a given CT enterography setting. This is done only once per CTE setting. That means each tuple (machine, power, current, exposition, set of contrast agents, contrast agent administration process) should have one training set to guarantee good performance. However, many settings might be compatible. As mentioned in section 3.4.6 obtaining scans with different settings just for the sake of our research is not currently feasible, so the actual degree of compatibility is still partly an open question.

2. **Removal of segments** in other areas than small intestine that have a similar appearance. This happens usually in stomach, colon and bladder, sometimes also the gallbladder or other parts. This is caused by the stochastic nature of the probability computation and similar or even the same contrast agents present in organs with a locally similar shape to the intestine. A simple GUI tool with a one-click interaction is enough for mistaken segments removal. This removal is necessary only for complete dataset visualization. Individual segments analysis can be performed even without this removal step, because wrong segments are simply not selected.

3. **Actual analysis/diagnosis** composes of manual inspection of our visualization results shown in chapter 6. The decision making is still kept on the radiology specialist or surgeon (in case of a pre-surgery inspection for topological information).

8.8 Future work

Our future work will be centered on several areas:

- Incorporating a-priori information into the stochastic part of the dataset evaluation and probability computation. However, advancing and incorporating the actual shape for an organ such as small intestine is a challenging task and a more promising path seems to be an automatic detection of segments not belonging to the small intestine. Another approach might be based on a statistical model by extending the work in Kolomazník et al. [2015].

- An automatic system for selecting a matching training set from a selection of provided ones. That would enable a preparation of several training sets with a selection of commonly used parameters that can be provided with the system. That would eliminate the need for creating a training set manually at each workplace.

- Topological analysis of individual segments and their relative positions to obtain longer segments from disjoint parts.

- Improve the analysis of intestinal wall. Current methods are not robust enough for real CTE data and are used mostly as a hint for a radiologist. A fully automatic process would further speed up the diagnosis. However, a high degree of robustness would be necessary to allow the radiologist to rely on such result. Any overlooked problem might cause harm to the patient.

Bibliography

A.H. Afifi and M.I. Kassem. Crohn's disease: Activity, complications and treatment. Evaluation using MDCT enterography and endoscopy. *The Egyptian Journal of Radiology and Nuclear Medicine*, 43(4):507–517, 2012.

J. Als-Nielsen and D. McMorrow. *Elements of Modern X-ray Physics*. Wiley, 2011. ISBN 9781119970156. URL `https://books.google.cz/books?id=rlqlboWlTRMC`.

A.H. Andersen and A.C. Kak. Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm. *Ultrasonic imaging*, 6(1):81–94, 1984.

L. Antiga. Generalizing vesselness with respect to dimensionality and shape. *The Insight Journal*, December 2007.

M. Bazalova and F. Verhaegen. Monte Carlo simulation of a computed tomography x-ray tube. *Physics in Medicine and Biology*, 52(19):5945, 2007. URL `http://stacks.iop.org/0031-9155/52/i=19/a=015`.

F. Booya, J.G. Fletcher, J.E. Huprich, J.M. Barlow, C.D. Johnson, J.L. Fidler, C.A. Solem, W.J. Sandborn, E.V. Loftus Jr, and W.S. Harmsen. Active Crohn disease: CT findings and interobserver agreement for enteric phase CT enterography 1. *Radiology*, 241(3):787–795, 2006.

A. Buades, B. Coll, and J.M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

D. Choi, S.J. Lee, Y.A. Cho, H.K. Lim, S.H. Kim, W.J. Lee, J.H. Lim, H. Park, and Y.R. Lee. Bowel wall thickening in patients with Crohn's disease: CT patterns and correlation with inflammatory activity. *Clinical radiology*, 58(1): 68–74, 2003.

R.D. Cohen. *Inflammatory Bowel Disease: Diagnosis and Therapeutics*. Clinical Gastroenterology. Humana Press, 2011. ISBN 9781603274333. URL `https://books.google.cz/books?id=V-5fPE0mHg8C`.

P. Coupe, P. Yger, S. Prima, P. Hellier, C. Kervrann, and C. Barillot. An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images. *Medical Imaging, IEEE Transactions on*, 22, 2008.

C.G. Cronin, E. Delappe, D.G. Lohan, C. Roche, and J.M. Murphy. Normal small bowel wall characteristics on MR enterography. *European Journal of Radiology*, 75, 2010.

S. Cuomo, P. De Michele, and F. Piccialli. 3D data denoising via nonlocal means filter by using parallel GPU strategies. *Computational and mathematical methods in medicine*, 2014, 2014.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3D filtering. In *Electronic Imaging 2006*, pages 606414–606414. International Society for Optics and Photonics, 2006.

J. Darbon, A. Cunha, T.F. Chan, S. Osher, and G.J. Jensen. Fast nonlocal filtering applied to electron cryomicroscopy. In *ISBI'08*, pages 1331–1334, 2008.

S.R. Deans. *The Radon Transform and Some of Its Applications*. A Wiley-Interscience Publication. John Wiley & Sons Inc., New York, 1983. ISBN 0-471-89804-X. Reprinted by Dover in 2007.

J.J. DeMarco, C.H. Cagnon, D.D. Cody, D.M. Stevens, C.H. McCollough, J. O'Daniel, and M.F. McNitt-Gray. A Monte Carlo based method to estimate radiation dose from multidetector CT (MDCT): cylindrical and anthropomorphic phantoms. *Physics in Medicine and Biology*, 50(17):3989, 2005. URL `http://stacks.iop.org/0031-9155/50/i=17/a=005`.

R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *SIGGRAPH Comput. Graph.*, 22(4):65–74, June 1988. ISSN 0097-8930. doi: 10.1145/378456.378484. URL `http://doi.acm.org/10.1145/378456.378484`.

A. Eklund, M. Andersson, and H. Knutsson. True 4D image denoising on the GPU. *International Journal of Biomedical Imaging*, 2011, 2011.

M.P. Federle. CT of the small intestine: Enterography and angiography. *Applied Radiology*, November 2007.

T. Fernandes, M.I. Oliviera, R. Castro, B. Araújo, B. Viamonte, and R. Cunha. Bowel wall thickening at CT: Simplifying the diagnosis. *Insights into imaging*, April 2014.

J.L. Fidler, L. Guimaraes, and D.M. Einstein. MR imaging of the small bowel. *RadioGraphics*, 29(6):1811–1825, 2009.

J.F. Frenet. *Sur les fonctions qui servent à déterminer l'attraction des sphéroides quelconques. Programme d'une thèse sur quelque propriétés des courbes à double courbure*. Doctoral thesis, University of Toulouse, 1847.

N. Gallagher and G. Wise. A theoretical analysis of the properties of median filters. *IEEE Transactions on Acoustic, Speech and Signal Processing*, ASSP-29(6), December 1981.

P. Gilbert. Iterative methods for the three-dimensional reconstruction of an object from projections. *Journal of theoretical biology*, 36(1):105–117, 1972.

L.W. Goldman. Principles of CT: multislice CT. *Journal of nuclear medicine technology*, 36(2):57–68, 2008.

R. Gordon, R. Bender, and G.T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *Journal of theoretical Biology*, 29(3):471–481, 1970.

N. Gourtsoyiannis, N. Papanikolaou, and A. Karantanas. Magnetic resonance imaging evaluation of small intestinal Crohn's disease. *Best Practice and Research Clinical Gastroenterology*, 20(1):137–56, 2006.

J. Gu, L. Zhang, G. Yu, Y. Xing, and Z. Chen. X-ray CT metal artifacts reduction through curvature based sinogram inpainting. *Journal of X-Ray Science and Technology*, 14(2):73–82, 2006.

H. Herlinger, D. Maglinte, and B.A. Birnbaum. *Clinical Imaging of the Small Intestine*. Springer, 2001.

G.T. Herman. *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press, New York, 1980.

G.T. Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer, 2009.

D. Holmes, J. Huprich, J. Fidler, R. Robb, and J. Fletcher. Feasibility of developing interactive small bowel segmentation from MR enterography. In *Proceedings of MICCAI 2010 Workshop: Virtual Colonoscopy and Abdominal Imaging*, page 105, 2010.

J. Horacek, M. Horak, J. Kolomaznik, and J. Pelikan. An automatic algorithm for tracking small intestine in CT enterography. In *Digital Image Computing: Techniques and Applications (DICTA), 2015 International Conference on*, pages 1–8, Nov 2015. doi: 10.1109/DICTA.2015.7371228.

J. Horáček, L. Maršálek, M. Horák, P. Slusallek, and J. Pelikán. Segmentation of femoral head from CT after femoral neck fracture. In *Proceedings of IAPR Conference on Machine Vision Applications - MVA 2009*, 2009.

J. Horáček, M. Horák, and J. Pelikán. An improved algorithm for fractured femoral head segmentation from CT using dynamic programming. In *Proceedings of The Eleventh IASTED International Conference on Computer Graphics and Imaging - CGIM 2010*, 2010.

J. Horáček, J. Kolomazník, M. Horák, and J. Pelikán. Denoising volumetric data on GPU. In *Proceedings of IMAGAPP 2011*, 2011.

G.N. Hounsfield. A method and apparatus for examination of a body by radiation such as x or gamma radiation, Patent specification 1283915. *The Patent Office, London, England*, 1972.

Armin Kanitsar, Dominik Fleischmann, Rainer Wegenkittl, Petr Felkel, and Eduard Groller. Cpr-curved planar reformation. In *Visualization, 2002. VIS 2002. IEEE*, pages 37–44. IEEE, 2002.

A. Kharlamov and V. Podlozhnyuk. Image denoising. *NVIDIA Corporation, June*, 2007.

Khronos OpenCL Working Group. The OpenCL specification. *The Khronos Group, July*, 2009.

J. Kolomazník. Cuda generic image processing library. `https://github.com/JanKolomaznik/cugip/`, 2013-2016.

J. Kolomazník, J. Horáček, V. Krajíček, and J. Pelikán. Implementing interactive 3D segmentation on CUDA using graph-cuts and watershed transformation. In *Proceedings of WSCG 2012*. Václav Skala-UNION Agency, 2012.

J. Kolomazník, J. Horáček, and J. Pelikán. Low level statistical models for initialization of interactive 2D/3D segmentation algorithms. In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications - Volume 1: VISAPP, (VISIGRAPP 2015)*, pages 686–692, 2015. ISBN 978-989-758-089-5. doi: 10.5220/0005361506860692.

M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8, May 1988.

E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. NVIDIA Tesla: A unified graphics and computing architecture. *IEEE Micro*, 28(2):39–55, March 2008. ISSN 0272-1732. doi: 10.1109/MM.2008.31.

A. Macovski. *Medical Imaging Systems*. Prentice-Hall information and system sciences series. Prentice-Hall, 1983. ISBN 9780135726853. URL `https://books.google.cz/books?id=G74eAQAAIAAJ`.

A.J. Madureira. The comb sign. *Radiology*, 230(3):783–784, 2004.

M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE transactions on image processing*, 22(1):119–133, 2013.

D.D.T. Maglinte, K. Sandrasegaran, J.C. Lappas, and M. Chiorean. CT enteroclysis. *Radiology*, 245(3):661–671, 2007. doi: 10.1148/radiol.2453060798. URL `http://dx.doi.org/10.1148/radiol.2453060798`. PMID: 18024448.

M. Maisl, L. Marsalek, Ch. Schorr, J. Horacek, and P. Slusallek. GPU-accelerated computed laminography with application to non-destructive testing. *European Conference on Nondestructive Testing*, 2014. URL `http://www.ndt.net/events/ECNDT2014/app/content/Paper/329_Schorr_Rev2.pdf`.

S. Mazzeo, D. Caramella, L. Battolla, L. Melai, P. Masolino, M. Bertoni, P. Giusti, C. Cappelli, and C. Bartolozzi. Crohn disease of the small bowel: spiral CT evaluation after oral hyperhydration with isotonic solution. *Journal of computer assisted tomography*, 25(4):612–616, 2001.

D. Mery. *Computer Vision for X-Ray Testing*. Springer, 2015.

J. Näppi, J.G. Lee, J.G. Fletcher, and H. Yoshida. Computer-assisted diagnosis for quantitative image-based analysis of Crohn's disease in CT enterography. In *Proceedings of MICCAI 2010 Workshop: Virtual Colonoscopy and Abdominal Imaging*, 2010.

J. Näppi, D.V. Sahani, J.G. Fletcher, and H. Yoshida. Automated detection and diagnosis of Crohn's disease in CT enterography. In *Proceedings of the Third International Conference on Abdominal Imaging: Computational and Clinical Applications*, MICCAI'11, pages 84–90, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-28556-1. doi: 10.1007/978-3-642-28557-8_11. URL `http://dx.doi.org/10.1007/978-3-642-28557-8_11`.

NVIDIA Corporation. Technical brief: NVIDIA GeForce GTX 200 GPU Architectural overview. May 2008a.

NVIDIA Corporation. OpenCL Programming guide. *The Khronos Group, March/April*, March/April 2008b.

NVIDIA Corporation. OpenCL Best practices guide. *The Khronos Group, August*, August 2009.

NVIDIA Corporation. Maxwell tuning guide. `http://docs.nvidia.com/cuda/maxwell-tuning-guide/index.html`, 2016.

M. Oda, T. Kitasaka, K. Furukawa, O. Watanabe, T. Ando, H. Goto, and K. Mori. Development of CAD prototype system for Crohn's disease. In *SPIE Medical Imaging*, pages 76241U–76241U. International Society for Optics and Photonics, 2010.

S.R. Paulsen, J.E. Huprich, J.G. Fletcher, F. Booya, B.M. Young, J.L. Fidler, C.D. Johnson, J.M. Barlow, and F. Earnest IV. CT enterography as a diagnostic tool in evaluating small bowel disorders: Review of clinical experience with over 700 cases. *RadioGraphics*, 26(3):641–657, May-June 2006.

Z. Peter, V. Bousson, C. Bergot, and F. Peyrin. A constrained region growing approach based on watershed for the segmentation of low contrast structures in bone micro-CT images. *Pattern Recognition*, 41(7):2358–2368, 2008.

J. Radon. über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. *Berichte über die Verhandlungen der Königlich-Sächsischen Akademie der Wissenschaften zu Leipzig*, 69, 1917.

J. Radon. On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging*, 5(4):170–176, Dec 1986. ISSN 0278-0062. doi: 10.1109/TMI.1986.4307775.

L. Rudin and S. Osher. Total variation based image restoration with free local constraints. *IEEE Transactions on Image Processing*, 1:31–35, 1981.

L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60, 1992.

R. Sostegni, M. Daperno, N. Scaglione, A. Lavagna, R. Rocca, and A. Pera. Crohn's disease: monitoring disease activity. *Alimentary Pharmacology & Therapeutics*, 17:11–17, 2003. ISSN 1365-2036. doi: 10.1046/j.1365-2036.17.s2.17.x. URL `http://dx.doi.org/10.1046/j.1365-2036.17.s2.17.x`.

J. Wang, H. Lu, Z. Liang, D. Eremina, G. Zhang, S. Wang, J. Chen, and J. Manzione. An experimental study on the noise properties of X-ray CT sinogram data in Radon space. *Physics in medicine and biology*, 53(12):3327, 2008.

D. Weiskopf. *GPU-Based Interactive Visualization Techniques*. Springer, 2007.

M. Zalis and A.K. Singh. Imaging of inflammatory bowel disease: CT and MR. *Digestive Diseases*, 22(1):56–62, 2004.

W. Zhang, J. Liu, J. Yao, A. Louie, T.B. Nguyen, S. Wank, W.L. Nowinski, and R.M. Summers. Mesenteric vasculature-guided small bowel segmentation on 3-D CT. *IEEE transactions on medical imaging*, 32(11):2006–2021, 2013.

List of Tables

List of Figures

Chapter A

Attachments

A.1 Feature space

A.1.1 Original and denoised comparison

Graphs in this section are taken from a training set computed over original source data and denoised data. Each figure represents a matching pair to provide easy comparison of the denoising process effect on our training sets.

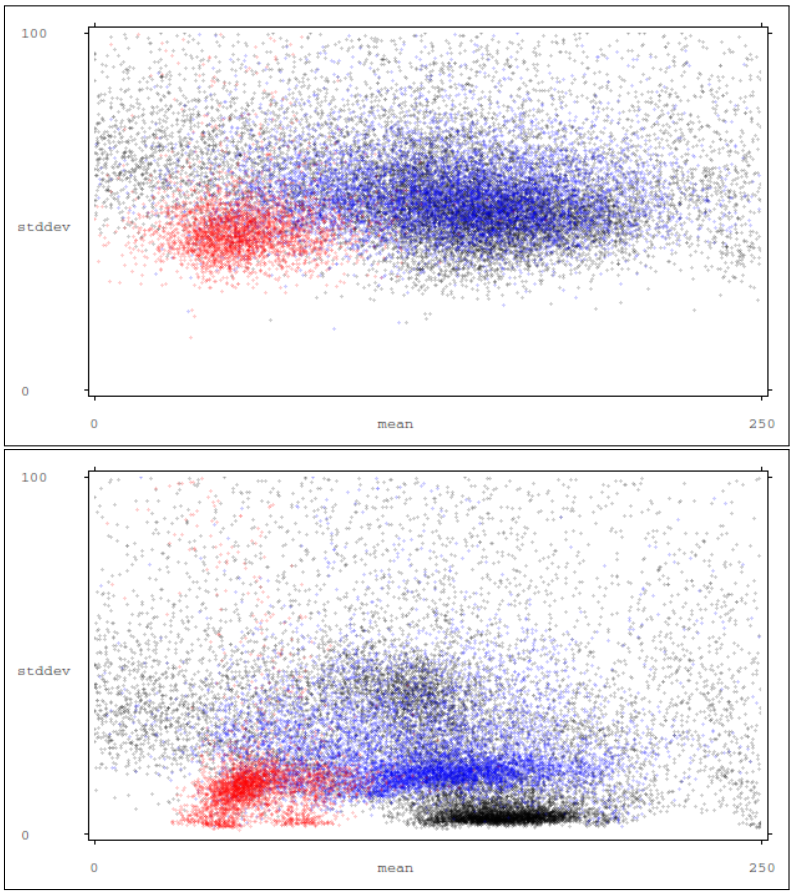

Figure A.1: Comparison of (a) original and (b) denoised features - mean value vs **standard deviation** per region on denoised data. Red is lumen, blue wall, black other regions.

Please note that all graphs are shown within the same range along matching axis types. Important difference is the shape of the blobs and the significant overlap in case of raw CT noisy data (causing bad discrimination of classes) and also the overall location (causing incompatibility of training data of denoised data with original source).

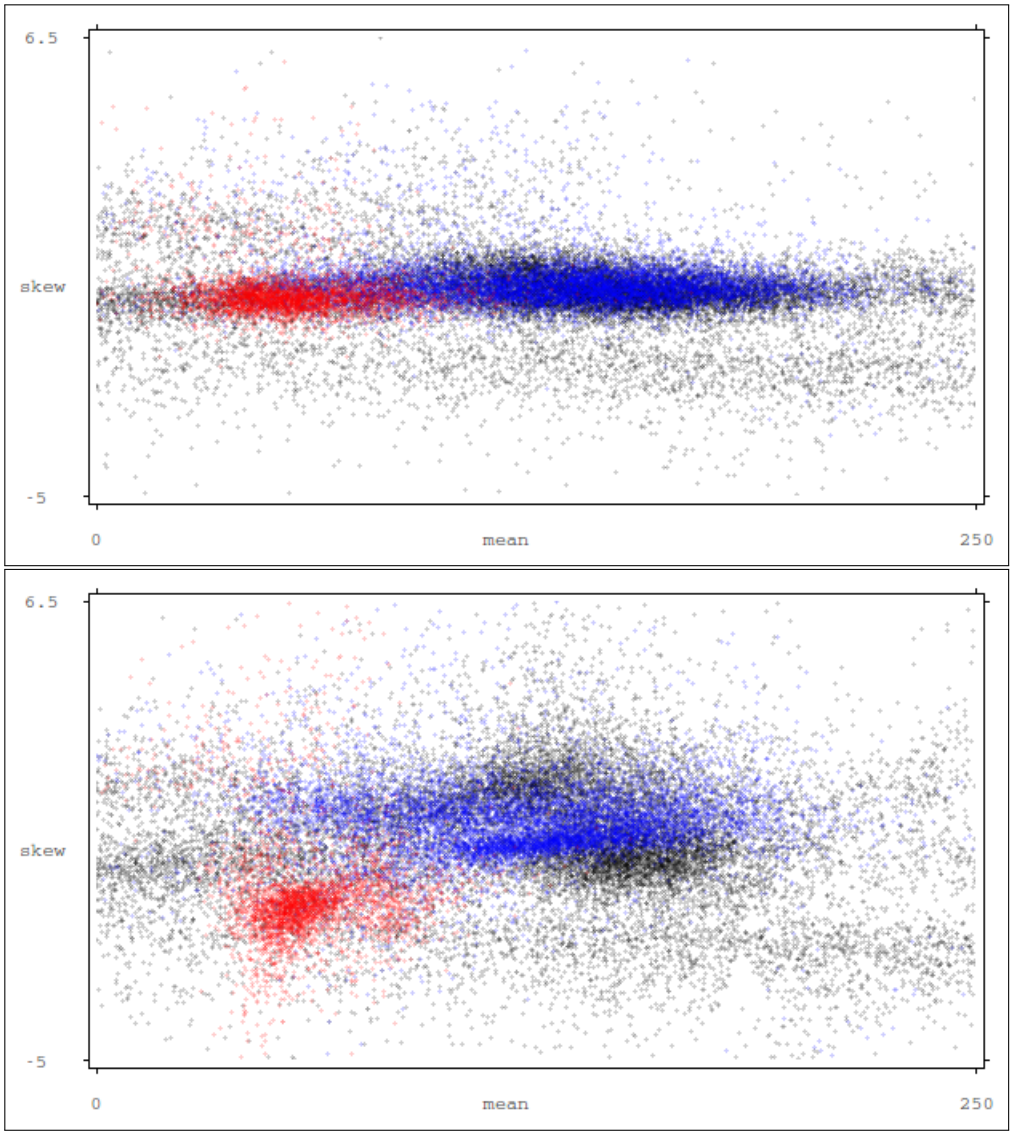

Figure A.2: Comparison of (a) original and (b) denoised features - mean value vs **skewness** per region on denoised data. Red is lumen, blue wall, black other regions.

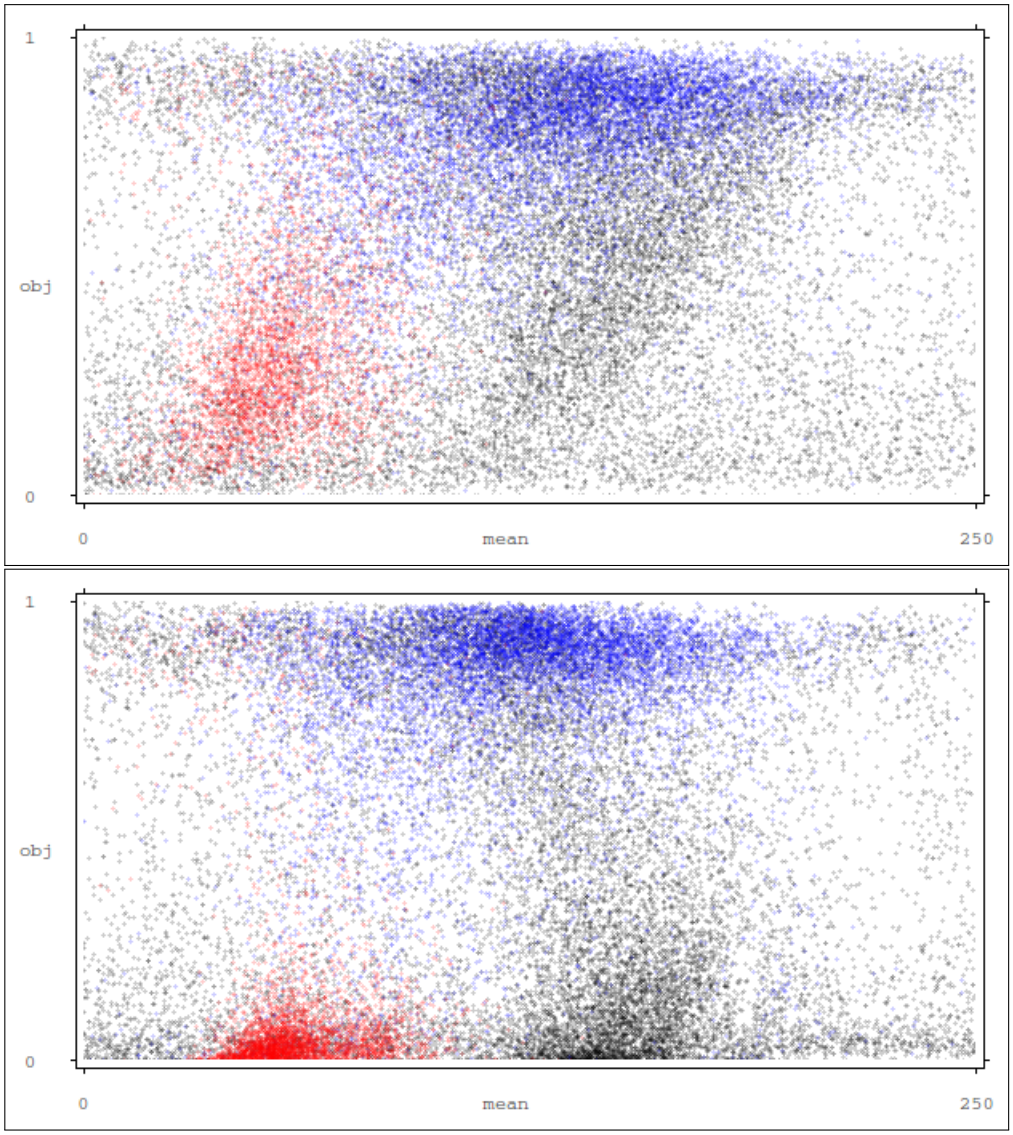

Figure A.3: Comparison of (a) original and (b) denoised features - mean value vs **objectness filter** per region on denoised data. Red is lumen, blue wall, black other regions.

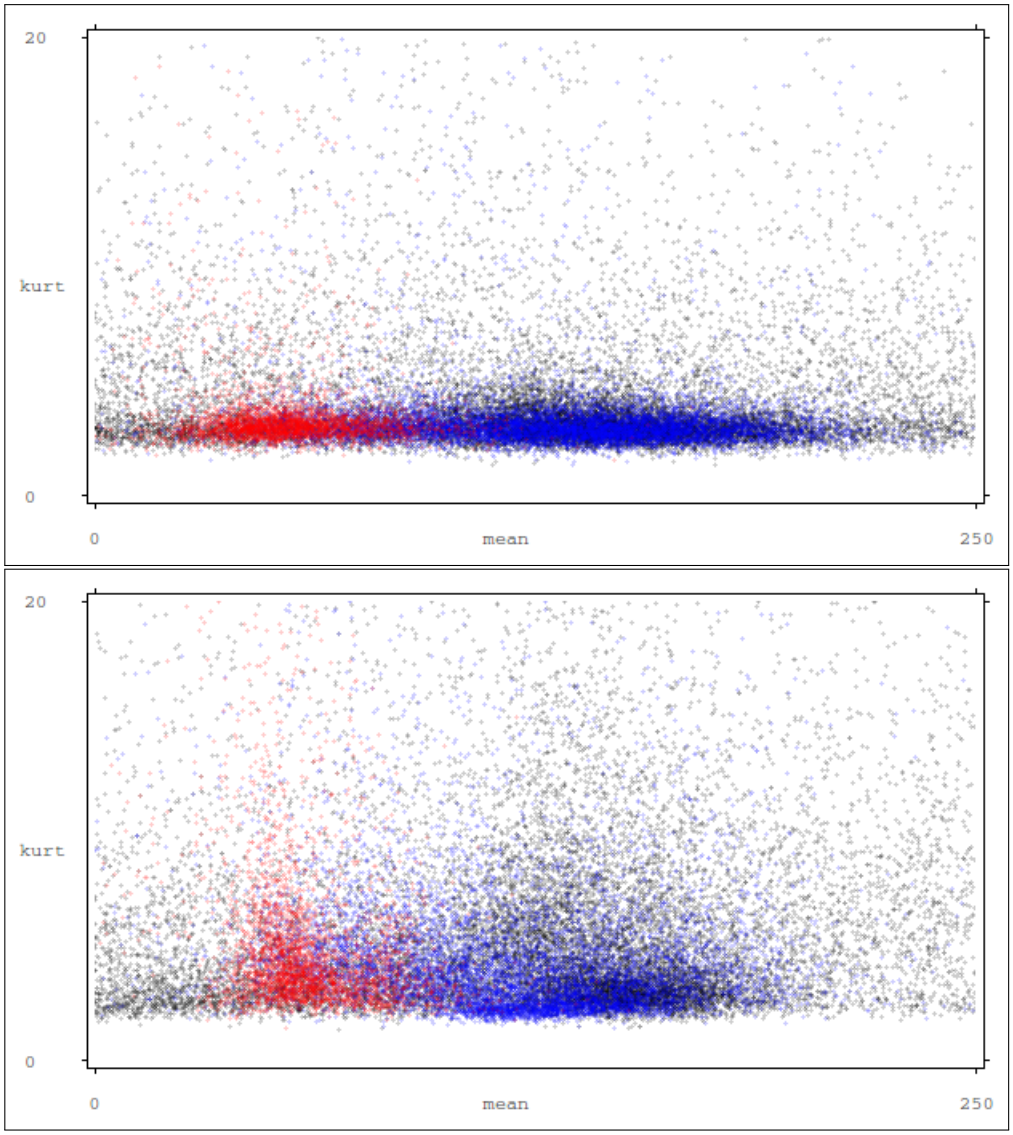

Figure A.4: Comparison of (a) original and (b) denoised features - mean value vs **kurtosis** per region on denoised data. Red is lumen, blue wall, black other regions.

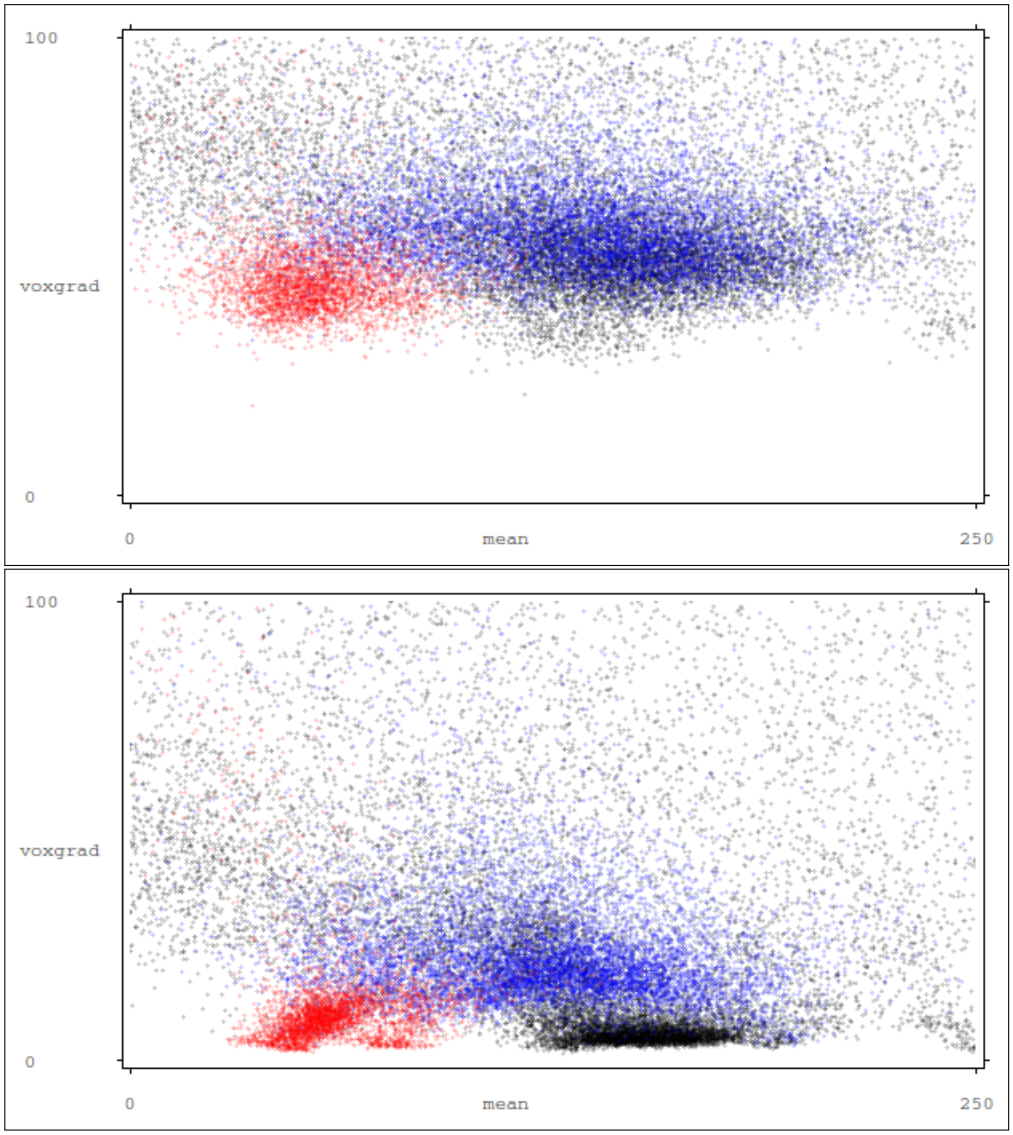

Figure A.5: Comparison of (a) original and (b) denoised features - mean value vs **average size of gradient** per region on denoised data. Red is lumen, blue wall, black other regions.

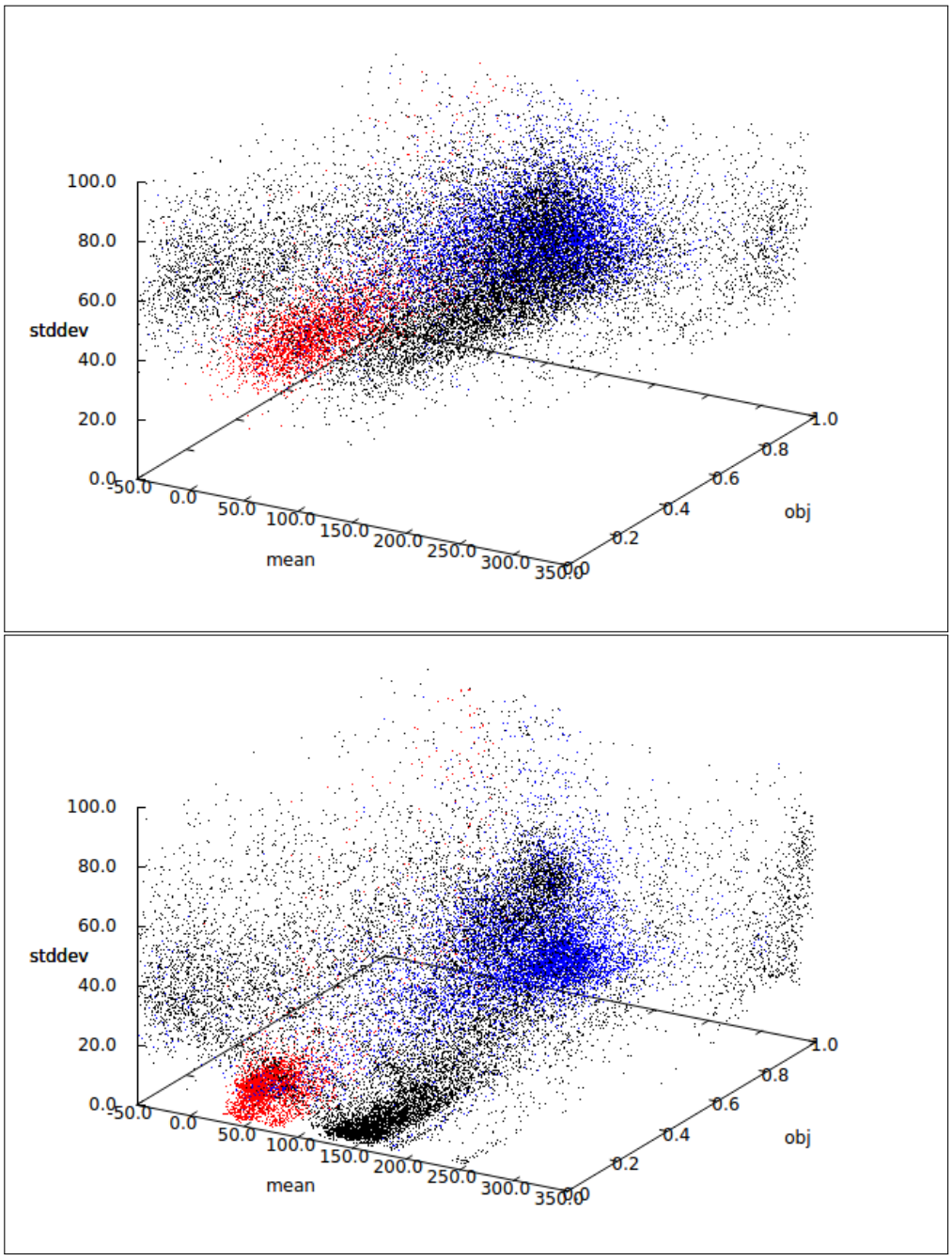

Figure A.6: Comparison of (a) original and (b) denoised features - mean value vs **objectness** vs **standard deviation**. Red is lumen, blue wall, black other regions.

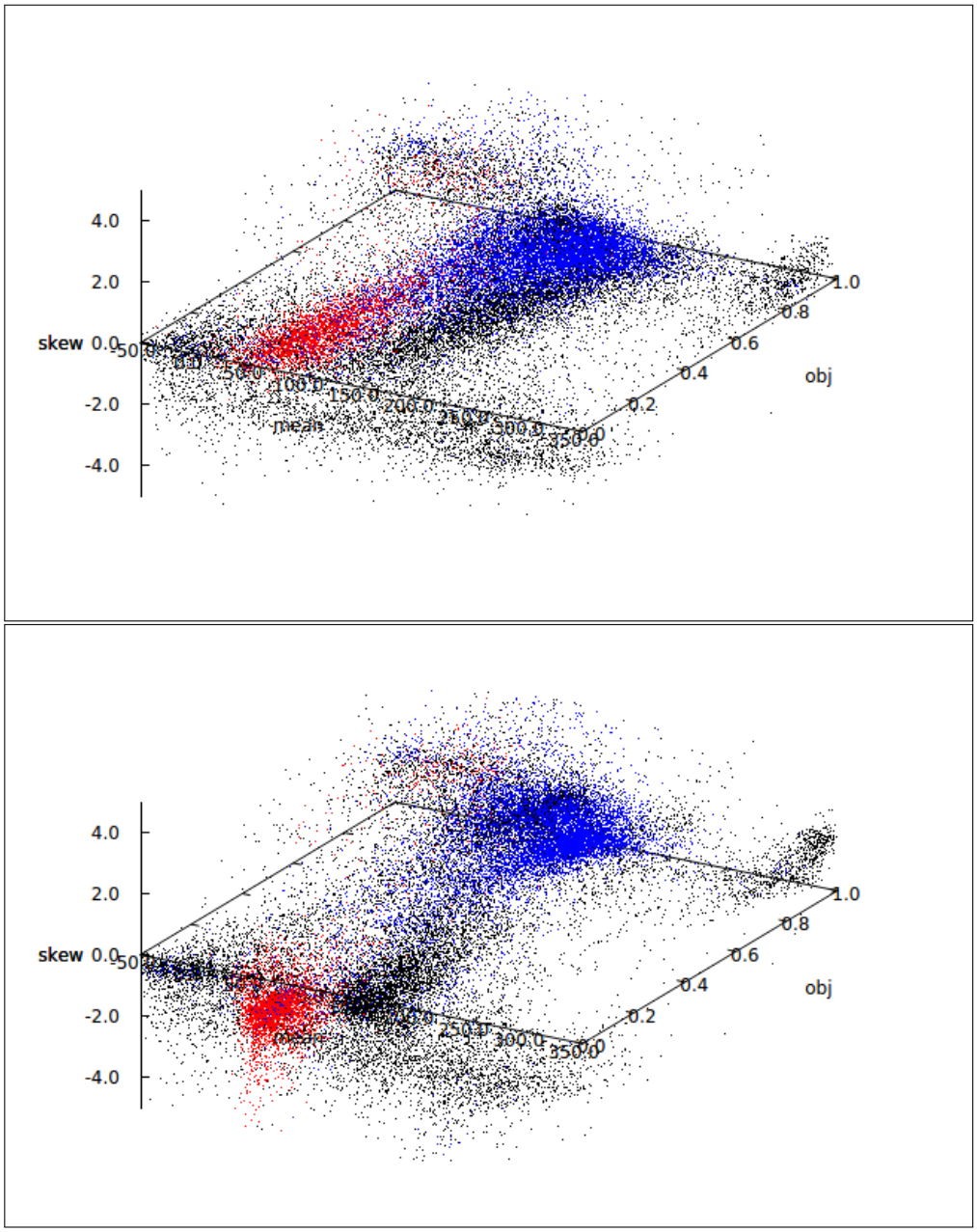

Figure A.7: Comparison of (a) original and (b) denoised features - mean value vs **objectness** vs **skewness**. Red is lumen, blue wall, black other regions.

A.1.2 Alternative training set comparison

A comparison of two different training set is given in this section. We have selected two different patients and used similar criteria for training set generation - the region in body was left part of abdominal region with well distended lumen, reaching approximately to the spine and being around $100 \times 100 \times 100mm$ in size.

Both training sets are created manually by marking lumen/wall regions in the said part of dataset.

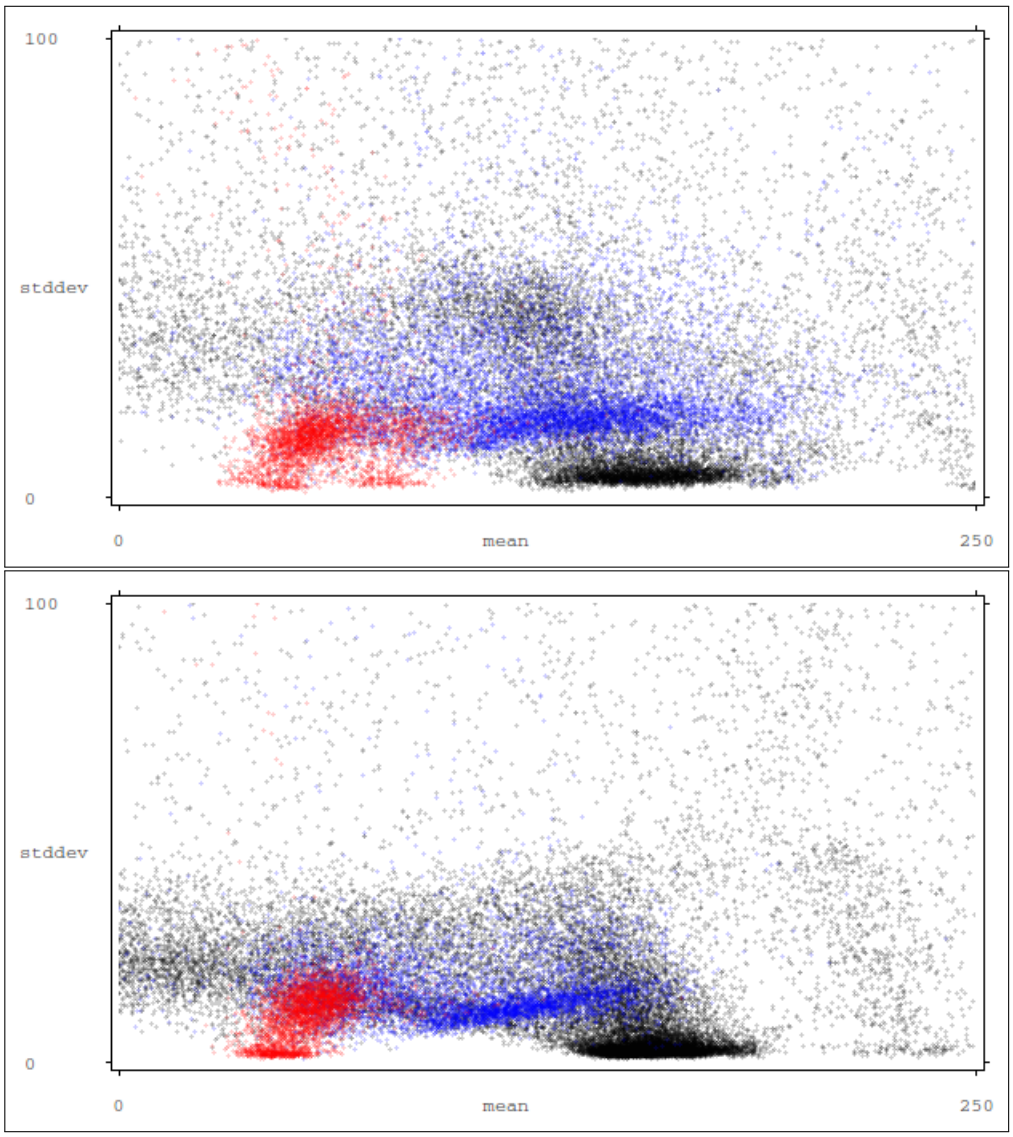

Figure A.8: Comparison of training set created over two different datasets - mean value vs **standard deviation** per region on denoised data. Red is lumen, blue wall, black other regions.

The scanning parameters are the same in both sets. The goal was to show the similarity and stability of training sets taken on different patients. Scatter graphs are shown always in a pair, where the first graph is the training set generated over dataset 04 (used in most of our evaluations) and second is dataset 17 (used in comparisons and stability tests) - see table 4.2.

Relatively large differences in *other* regions (in all graphs with black color) are understandable, because these regions strongly depend on the selection of the training source and the visible presence of other organs.

The rest of the regions (lumen = red and wall = blue) should form *blobs* with similar shape and location, differing mostly only in the density of the represented regions - again caused by the size and selection of the training set.

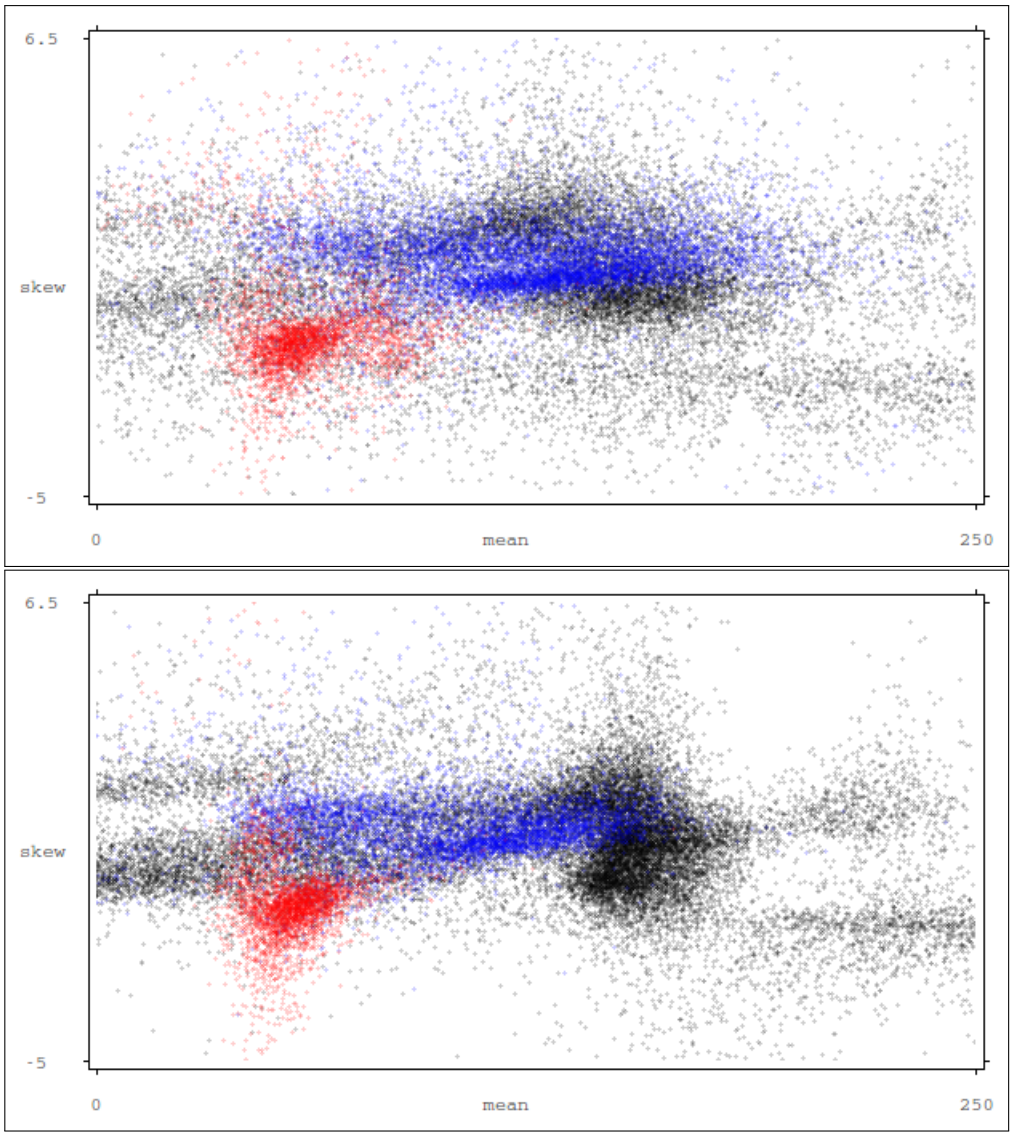

Figure A.9: Comparison of training set created over two different datasets - mean value vs **skewness** per region on denoised data. Red is lumen, blue wall, black other regions.

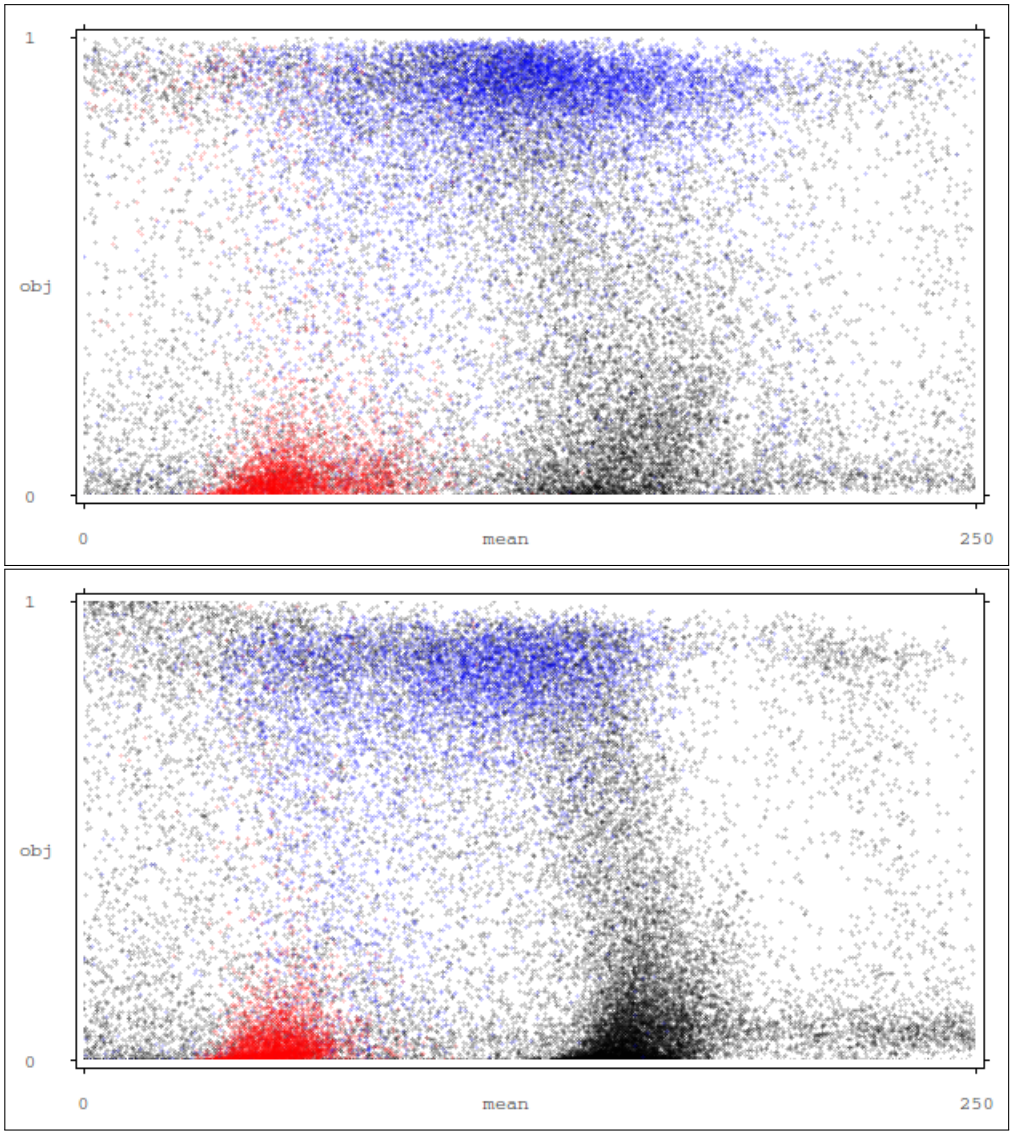

Figure A.10: Comparison of training set created over two different datasets - mean value vs **objectness filter** per region on denoised data. Red is lumen, blue wall, black other regions.

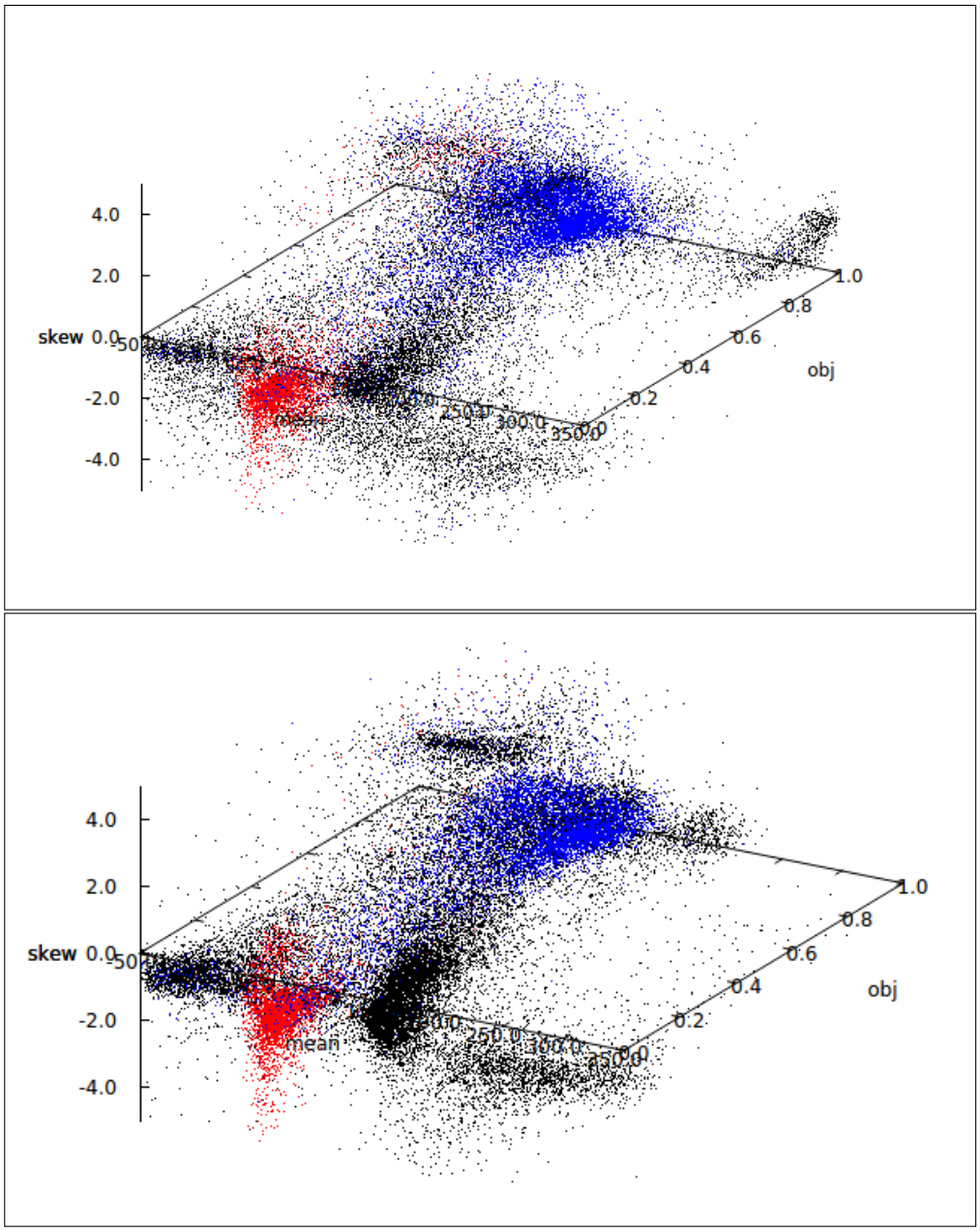

Figure A.11: Comparison of training sets created over two different datasets - mean value vs **objectness** vs **skewness**. Red is lumen, blue wall, black other regions.

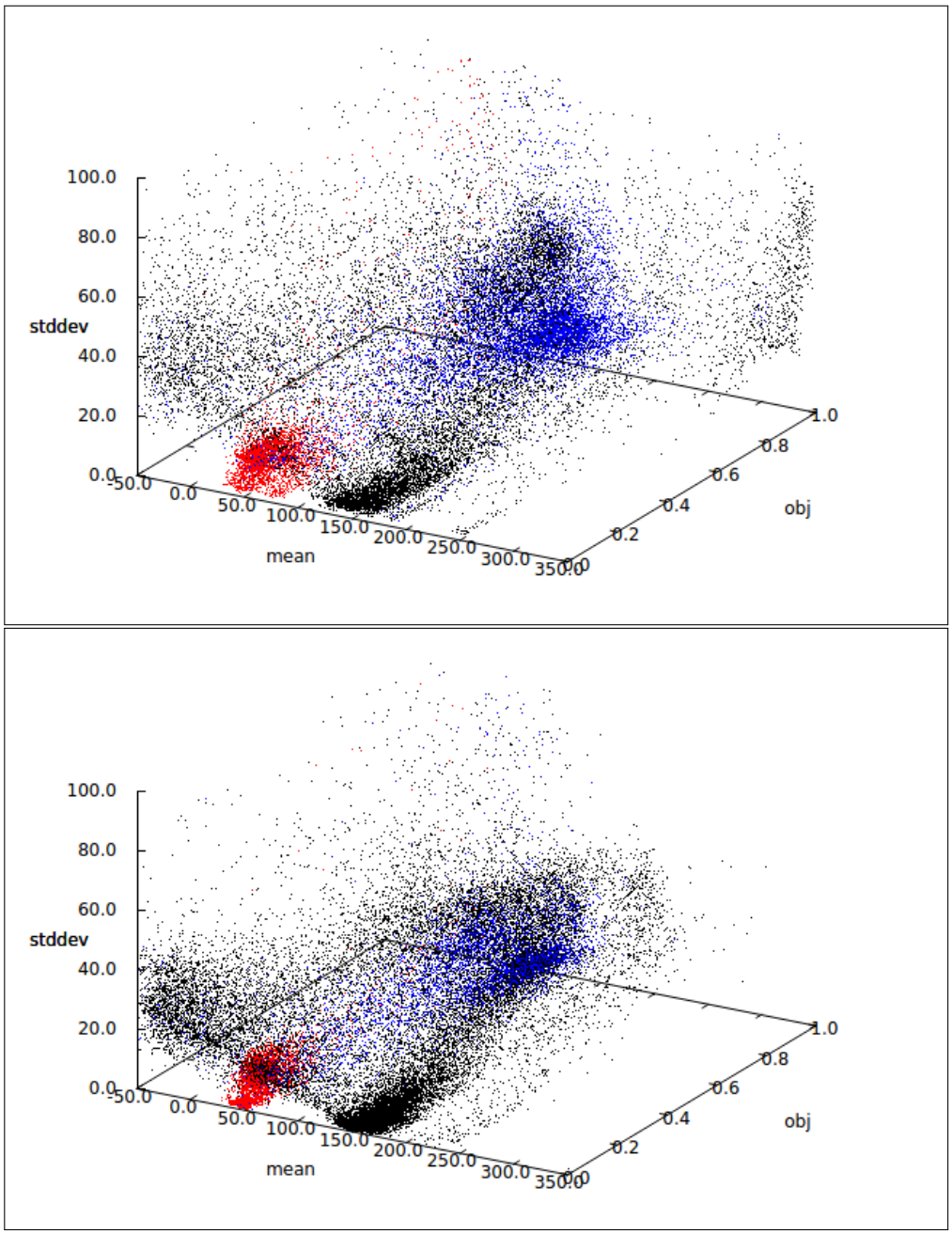

Figure A.12: Comparison of training sets created over two different datasets - mean value vs **objectness** vs **standard deviation**. Red is lumen, blue wall, black other regions.

A.2 A note on probability robustness: wrong training set

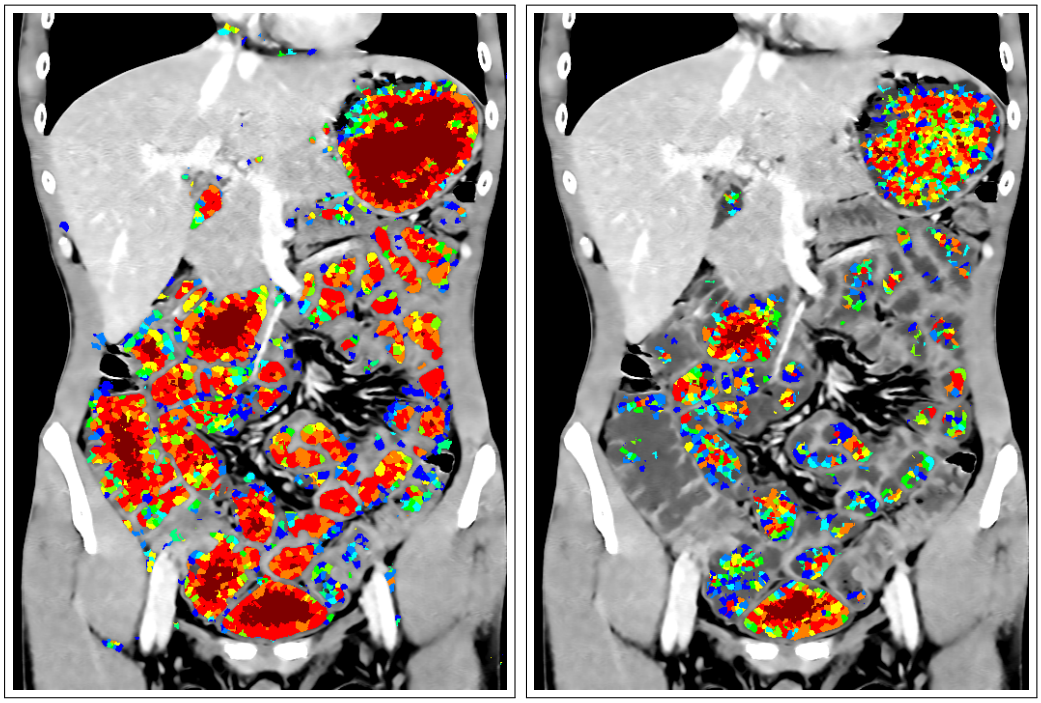

Figure A.13: Comparison of lumen probability P_{lumen} computed with (a) correct and (b) wrong training set.

Results in this chapter are a result of an interesting mistake that happened during the evaluation of our datasets. We have not done a full evaluation, but we found the effect interesting enough to give it a short explanation.

We had two major groups of datasets with different scanning parameters. Both groups have been CT enterography scans with similar preparation and scanning procedure. For one group we had full information about the scanning procedure and all patients were scanned on the same machine with the same settings. For the second group (slightly older) the scan parameters were lost during the anonymization process. The only information we had about this group was that the scans were CTE and all patients within the group were scanned with the same settings.

Obviously the datasets looked similar under casual view, but with evident quantifiable differences. There was a shift in the dataset values, around 20HU in neutral-contrast agent filled lumen and more than twice that difference in intestinal wall. We do not know whether this was due to the scanning procedure (voltage, exposition, ...) or due to the contrast agent preparation (type, saturation, time of scan, ...). This information is currently no more retrievable and we would have to do an in-depth analysis with the database stored in hospital. That is currently not practically viable. However the data were consistent within each group.

You can see in the examples in figures A.13 and A.14 that even though the training set was wrong and useless for a segmentation and tracking, the probability followed the data in a relatively consistent way. This leads us to a conclusion,

that the sensitivity to the training set fitness is not extremely narrow. So a preparation of multiple training sets and an automatic algorithm for selecting the right one should be possible.

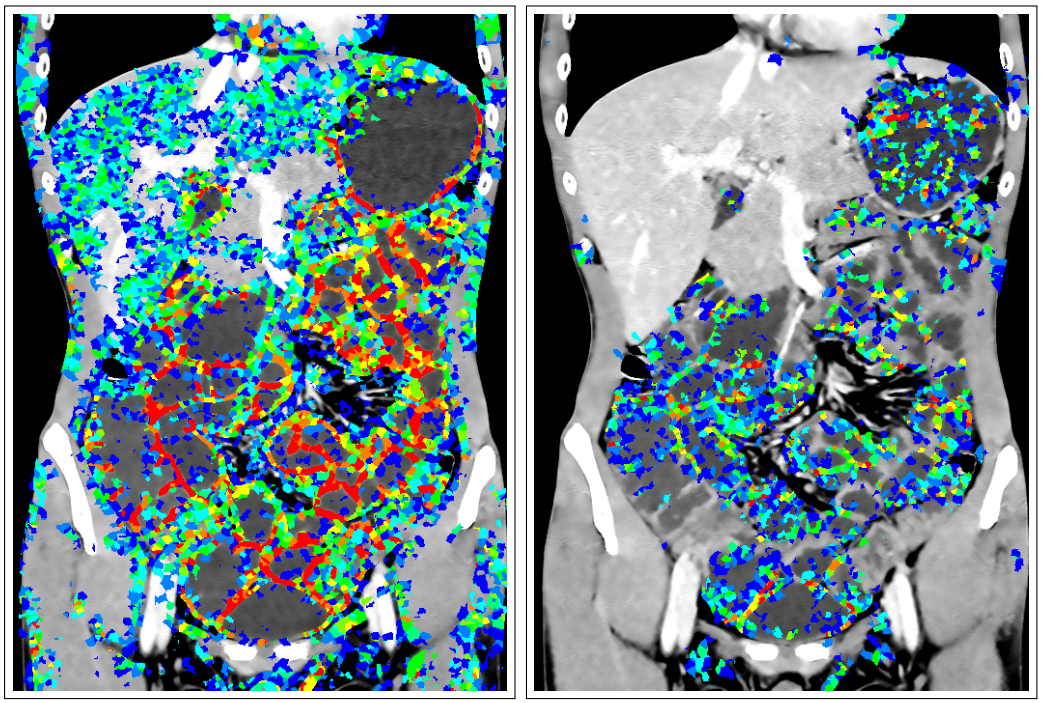

Figure A.14: Comparison of wall probability P_{wall} computed with (a) correct and (b) wrong training set.