UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MEDICAL INFORMATICS

From the Institute of Medical Informatics
of the University of Lübeck
Director: Prof. Dr. rer. nat. habil. Heinz Handels

Direct Volume Rendering Methods for Needle Insertion Simulation

Dissertation for Fulfillment of Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by
Dirk Fortmeier

from Braunschweig
Lübeck 2016

First referee: Prof. Dr. rer. nat. habil. H. Handels, Institute of Medical Informatics
Second referee: Prof. Dr.-Ing. habil. A. Mertins, Institute for Signal Processing
Date of oral examination: May 18, 2016
Approved for printing. Lübeck, October 21, 2016

Abstract

This thesis presents a simulation framework and newly developed methods for visuo-haptic rendering of a virtual liver puncture intervention. Visuo-haptics include both visual representation and haptic rendering using a haptic input device for force display. The framework includes methods and algorithms to simulate a medical intervention that consists of palpation, ultrasound probing and X-ray imaging. Using these components, a percutaneous transhepatic cholangiodrainage can be performed in virtual reality with the aim of providing an environment for training and planning.

The central aspect distinguishing the innovative approaches from comparable state of the art methodology is the focus on methods that use direct volume rendering of medical computed tomography image data. Therefore, no creation of an intermediate surface representation of organ tissues has to be performed. These surface representations are needed in comparable approaches in order to perform visualization, force feedback computation and soft tissue simulation. To create these surface representations, it is necessary to perform a time-consuming segmentation process. This process took more than 60 hours in the framework's predecessor. It is essential to reduce this time in case a patient specific scenario should be prepared based on new image data.

In the presented framework, the segmentation is reduced to structures that are central to the intervention. The resulting partial segmentations and patient image data are then rendered visually and haptically during run-time by adapted and newly developed rendering approaches. Visual direct volume rendering is realized by ray casting of the volume data. A soft tissue simulation component is included into the framework by computation of local deformations on the regular grid of the volume data. The resulting deformed image is then considered in the volume rendering process. Furthermore, respiratory motion can be visualized by the framework and is integrated into the haptic algorithms. This is done by using 4D CT image data and creation and application of resulting motion models. For achieving real-time rendering capability, the visualization methods are implemented using Nvidia CUDA.

First of all, this thesis summarizes the newly developed methods and obtained results and then analyzes the components of the framework. It is shown that the methods and their implementation fulfill the requirements with regard to the reduction of segmentation effort, real-time capabilities and plausibility of visuo-haptic rendering.

Zusammenfassung

In dieser Arbeit werden ein Simulationsframework sowie neu entwickelte Methoden zum visuo-haptischen Rendering von virtuellen Leberpunktionen präsentiert. Visuo-haptische Verfahren enthalten sowohl visuelles Rendering als auch haptisches Rendering, wobei Kräfte mittels eines haptischen Eingabegeräts wiedergegeben werden. Das Framework enthält Methoden und Algorithmen zur Durchführung eines virtuellen Eingriffs mittels Palpation, Ultraschallsondierung und Röntgendurchleuchtung. Unter Zuhilfenahme dieser Komponenten kann eine virtuelle perkutane transhepatische Cholangiodrainage durchgeführt werden. Ziel ist die Bereitstellung einer Umgebung zu Training und Planung.

Der zentrale Aspekt ist die Fokussierung der Entwicklung von Methoden, die auf direktes Volumenredering von medizinischen CT-Daten zurückgreifen, ohne den Umweg über eine Oberflächenrepräsentation zu gehen. Diese sind in vergleichbaren Ansätzen des State-of-the-Art Grundlage für Visualisierung, Berechnung von haptischer Kraftrückgabe und Weichteilsimulation. Um die Oberflächenrepresentation bereitzustellen, wird in der Regel eine aufwändige Segmentierung der Bilddaten vorausgesetzt. Eine solche Segmentierung benötigte im Vorgängerprojekt mehr als 60 Stunden pro Patient. Es ist daher essentiell, diese Zeit zu verkürzen, um ausgehend von einem neuen Bilddatensatz eine patientenindividuelle Simulation bereitzustellen.

In dem vorgestellten Framework wird der Segmentierungsbedarf reduziert auf Gewebe und Organstrukturen, welche für den Eingriff von zentraler Bedeutung sind. Die entstehenden Segmentierungsmasken und nicht segmentierten Strukturen in den Bilddaten werden zur Laufzeit mittels angepasseter sowie neuer, innovativer Renderingverfahren visuo-haptisch dargestellt. Visuelles direktes Volumenrendering wird durch Ray-Casting-Verfahren auf Grundlage der CT-Daten durchgeführt. Für die Berechnung von lokalen Deformationen auf dem regulären Gitter der Bilddaten ist für das Framework eine neue Methode entwickelt worden. Die dabei entstehenden Deformationen finden im Ray-Casting-Verfahren Beachtung. Weiterhin kann die Atembewegung eines virtuellen Patienten visualisiert und haptisch gerendert werden. Dies geschieht durch den Rückgriff auf 4D-CT-Sequenzen und die Erstellung und Nutzung daraus resultierender Bewegungsmodelle. Um Echtzeitfähigkeit zu erreichen wurden die Visualisierungsmethoden in Nvidia CUDA implementiert.

Diese Arbeit beschreibt zum einen die entwickelten Methoden und die mit ihnen erzielten Resultate und analysiert weiterhin die Komponenten des Frameworks. Es wird gezeigt, dass die entwickelten Methoden und ihre Umsetzung im Simulationsframework die an sie gestellten Anforderungen hinsichtlich Reduktion des Segmentierungsaufwands, Echtzeitfähingkeit und Plausibilität des visuo-haptischen Renderings erfüllen.

Danksagung

Zunächst danke ich Prof. Dr. rer. na. habil. Heinz Handels für die Vergabe und Betreuung des interessanten und vielseitigen Promotionsthemas sowie für die Möglichkeit, dieses am Institut für Medizinische Informatik zu bearbeiten.

Weiterhin gilt mein Dank allen Kollegen und Kolleginnen des Instituts für Medizinische Informatik für wertvolle Ratschläge, Anregungen und für die angenehme und lehrreiche Zeit.

Mein besonderer Dank gilt den Kollegen und Freuden Oskar Maier, Dr. rer. biol. hum. Andre Mastmeyer, Matthias Wilms, Christoph Schröder und Martin Barron für die Anregungen zur Verbesserung des Manuskripts.

Abschließend danke ich meiner Lebenspartnerin Miriam Karstens für die fortwährende Unterstützung in der Zeit des Verfassens dieser Dissertation, sowie meinen Eltern und Großeltern.

Contents

1. Introduction **1**

1.1. Surgery Simulation - State of the Art 2
1.2. Motivation and Objectives 5
1.3. Scientific Contributions 6
1.4. Overview 7

2. Background **9**

2.1. Medical Background: Percutaneous Transhepatic Cholangiodrainage 9
2.2. Nvidia CUDA: Using the General Purpose Graphics Processing Unit 11

3. Direct Visuo-haptic Volume Rendering Algorithms **15**

3.1. Input Data and Notations 15
3.2. Ray Casting based Volume Rendering Methods 17
3.2.1. Ray Casting 17
3.2.2. Clipping and Tagging 18
3.2.3. Shadow Rendering 19
3.2.4. Combined Volume Rendering and Rasterization 22
3.3. X-Ray Rendering and Contrast Agent Diffusion 22
3.4. Simulated Ultrasound Imaging 24
3.5. A Bendable Needle 25
3.5.1. Updating the Insertion Path 26
3.5.2. Simulation of Needle and Tissue Coupling 28
3.5.3. Force Computation 31
3.6. Palpation and Ultrasound Probing 33
3.7. Experiments & Results 37
3.7.1. Ray Casting 37
3.7.2. X-Ray and Ultrasound Simulation 40
3.7.3. Needle Algorithm 42
3.7.4. Palpation & US-probing Algorithm 44

3.8. Discussion 45

4. Visuo-haptic Rendering with Local Deformations 47

4.1. Background/Related Work 48
4.1.1. Finite Element Method 48
4.1.2. ChainMail 49
4.1.3. Other Methods 50
4.1.4. Image Registration 50
4.2. Real-time Image-based Deformations 51
4.2.1. A Priori Known Deformations 52
4.2.2. Variational Formulation 54
4.2.3. Regularization Terms 56
4.2.4. Discretization using Finite-Differences 57
4.2.5. Finding a Minimal Solution 58
4.3. Implementation 59
4.3.1. Algorithm Overview 59
4.3.2. Blocks and Threads 61
4.3.3. The Kernel 62
4.4. Optimizations 62
4.4.1. Region-of-Interest Pyramid Approach 62
4.4.2. Multigrid Approach 64
4.4.3. ChainMail 65
4.4.4. Fast Explicit Diffusion 65
4.5. Evaluation Framework with In-silico Ground Truth 65
4.6. Experiments & Results 67
4.7. Discussion 68

5. Visuo-haptic Rendering using Respiratory Motion Models 71

5.1. Related work 72
5.2. Respiratory Motion as a Transformation 72
5.3. Respiratory Motion Models 73
5.3.1. Key Frame Approach with a Single Respiratory Cycle 75
5.3.2. Key Frame Approach using a Half Cycle 78
5.3.3. Model using Surrogate Signals 78
5.4. Direct Visuo-haptic Rendering using Motion Fields 80
5.4.1. On-the-fly Inversion of the Displacement Field 80

5.4.2. Surface based Rendering . . . 82
5.4.3. Haptic Rendering . . . 82
5.4.4. Modifications to Haptic Rendering of Needle Insertion . . . 82
5.5. Implementation . . . 83
5.5.1. Remarks on Visualization Implementation on the GPU . . . 83
5.5.2. Remarks on CPU Implementation for Haptics . . . 84
5.6. Input data and Preparation of the Virtual Patient Model . . . 84
5.7. Experiments & Results . . . 85
5.8. Discussion . . . 88

6. A Framework for Image-based Puncture Simulation 91
6.1. Puncture Atlases with Partially Segmented Data . . . 91
6.1.1. Creation of Partially Segmented Data . . . 92
6.1.2. Label Estimation Heuristic . . . 92
6.1.3. Property Tree . . . 93
6.1.4. Parameter Evaluation . . . 94
6.2. Haptic Workbench Hardware-Setup . . . 95
6.2.1. Workbench Constraints . . . 95
6.2.2. User Interface . . . 96
6.3. Software Architecture . . . 97
6.4. Discussion . . . 100

7. Summary & Discussion 103

8. Outlook & Conclusion 107
8.1. Future Work . . . 107
8.2. Conclusion . . . 109

List of Symbols 111

A. Appendix 113
A.1. Deflection Measurements for Two Needles . . . 113
A.2. List of Tissue and Organ Structures Needed for PTCD . . . 114

Bibliography 115

List of Publications by the Author 129

1. Introduction

Virtual reality surgery simulation aims to provide a training and planning environment for physicians. The main reason to use such a tool is the prospect that a trainee can practice without high risk and costs. Traditionally, the method for training of surgeons was "learning by doing" summarized in the motto "see one, do one, teach one" [VHRG04]. Obviously, this method is not optimal and in the worst case mistakes by the trainee can be painful, dangerous or even fatal for the patient. Alternatives to practicing on a patient are mannequins, models and cadavers. These surrogates are often expensive, not reusable or ethically questionable. Virtual reality (VR) is a well researched field with applications in entertainment and simulation. Even if VR is ubiquitously present in everyday life, it is beneficial to explicitly state its essential nature: The key idea is to provide an interactive environment that behaves and acts similar to reality, but that has no physical existence in itself. Abstractly speaking, computers produce synthetic sensory data and display it to the sensory system of a user. Furthermore, the user can react to this input and interact with the system through input devices. Most surgery simulation systems use a stereoscopic monitor and a haptic device, making them capable of visuo-haptic rendering, see Fig. 1.1. In such a system, the monitor is used to display visual representations of the virtual environment. For haptic rendering two aspects have to be considered: The haptic device displays forces to the user, but also the user can manipulate the haptic device and thus can interact with the simulation. This creates a feedback loop, for which the development requires special attention.

This thesis presents a VR simulation system that uses visuo-haptic rendering for needle insertion simulation. The chapter covers four key areas: Firstly, an overview of the current state of surgical VR simulation with a focus on needle insertion interventions is given. Secondly, the next section motivates and summarizes the objectives addressed in this dissertation. The third section summarizes the scientific contributions that are subject of this thesis and lists the associated publications. Finally, the last section of this chapter gives an overview of the structure of the following chapters.

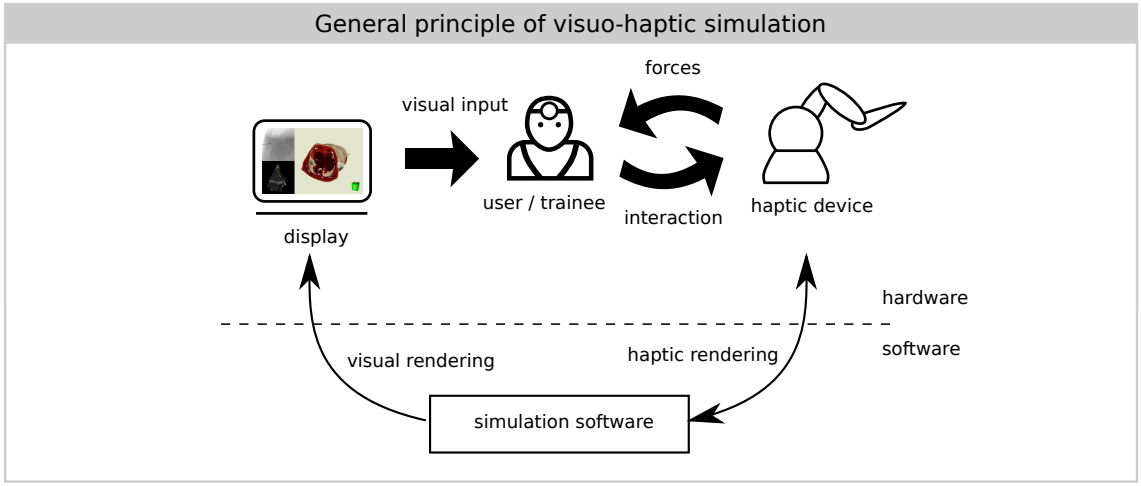

Figure 1.1.: Visuo-haptic VR systems use visual and haptic rendering to display visual output and provide haptic interaction. In comparison to visual rendering, the haptic rendering forms a (force-)feedback loop.

1.1. Surgery Simulation - State of the Art

Research on interactive virtual reality surgery simulation began in the early 1990s and has been continually refined since then, especially the areas of realism of visual display and haptic interaction. Key technological advances for this are 3D graphics hardware and available haptic devices. The first simulators included custom made prototypes, whereas nowadays haptic devices are commercially available by manufacturers such as Geomagic (formerly Sensable), Novint or ForceDimension. Depending on the number of degrees of freedom, working space and maximally displayable force output, the costs of these devices range from around two hundred Euros (Novint Falcon) to several tens of thousands of Euros, making them the central cost factor for visuo-haptic simulation systems.

A high number of different surgical techniques exists that are based on very different approaches including open surgery, minimal invasive surgery (laparoscopic surgery) and needle insertion techniques. The simulation of these different techniques has to emphasize different aspects and requires special attention, especially for the interaction between tools and soft tissue.

In this thesis, the focus will be on needle insertion, for which the most recent and relevant developments will be presented in the following. Needle scenarios that have been simulated in the past include lumbar puncture, regional anesthesia, liver (vessel) puncture, brachytherapy and biopsy. Noticeable examples are given in Fig. 1.2 and are compared in the following list and table 1.1.

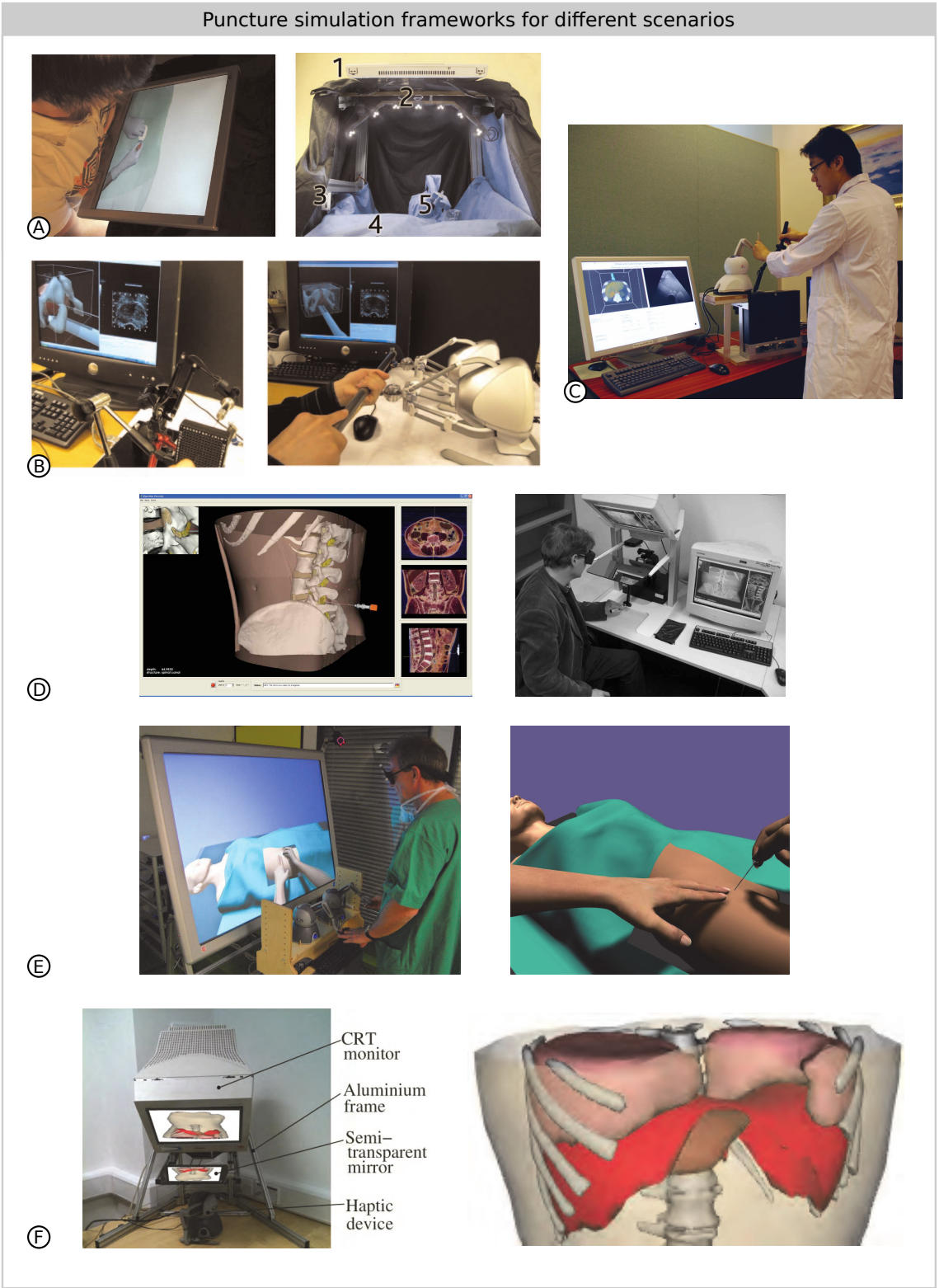

Figure 1.2.: Puncture simulation frameworks with different application scenarios: (A) Regional anesthesia simulator with augmented reality setup [CJGC11] (B) Brachytherapy simulator with simulated needle insertion and TRUS probing [GSS11] (C) Liver biopsy simulator [DWJ10] (D) Lumber puncture simulator [FHGH09] (E) Regional anesthesia simulator with two haptic devices [UIK12] (F) Liver puncture simulator [VBBG11]. All copyrights by the resp. authors.

auth./name	appl.	visuals	haptics	soft body simulation
PalpSim	regional anesthesia	surf.	surf.	mesh based
Ullrich et al.	regional anesthesia	surf.	surf.	mesh based
Goksel et al.	brachytherapy	surf.	surf.	mesh based
ImaGINe-S	liver puncture	surf.	surf./vol.	Chain-Mail
Ni et al.	liver puncture	vol./surf.	vol.	geometric (US only)
Acus-VR	lumbar puncture	surf.	vol.	-
(*)	liver puncture	vol.	vol.	finite differences

Table 1.1.: Recent needle insertion simulation systems in comparison to the one presented in this thesis (*). Different applications are targeted and visual and haptic rendering differs in usage of volume or surface based methods. The simulation of soft tissue is also handled in different ways.

PalpSim Needle insertion for regional anesthesia into the femoral artery was addressed in [CJGC11]. This intervention requires a palpation of the femoral artery before performing the actual needle insertion. In [CJGC11] this step is simulated by a gel pad tissue phantom that can be palpated by the user's finger. To provide immersion, the palpation process is presented to the trainee by augmented reality (AR) techniques: The working environment of the setup including the hands of the trainee and the haptic device are filmed by a camera and are displayed on a screen. Using chroma key methods, the green background captured by the camera is replaced by a virtual scene that includes the virtual patient. Using a modified Phantom Omni, the user is also able to insert a virtual needle in the AR environment.

Ulrich et al. In contrast to this, in [UIK12] a setup using two Phantom Omnis is presented also for simulation of regional anesthesia. In this setup one device is used for palpation, the other one for needle insertion. For the computation of palpation forces a surface model of the virtual patient and internal structures is used. This work includes a rigorous evaluation by means of a user study that highly supports the applicability of visuo-haptic rendering for surgery simulation.

Goksel et al. The work of Goksel et al. [GSMS13, GS09, GSS11] contain the first complete brachytherapy simulation system. It includes virtual transcrectal ultrasound probing (TRUS) with visual and haptic simulation using linear-strain quasi-static finite element simulation. The tetrahedral mesh representation of soft tissue consists of around 10,000 tetrahedra. Insertion of flexible and bevel-tipped needle together with placement of radioactive seeds can be performed.

ImaGINe-S The "ImaGINe-S" system, which deals with a similar intervention as the one in this theses, is presented in [VVA+13, VBBG11] and [VJHG08, VVH+09, VVL12, BBG+09]. It was designed for needle insertion into liver for biopsy and percutaneous transhepatic

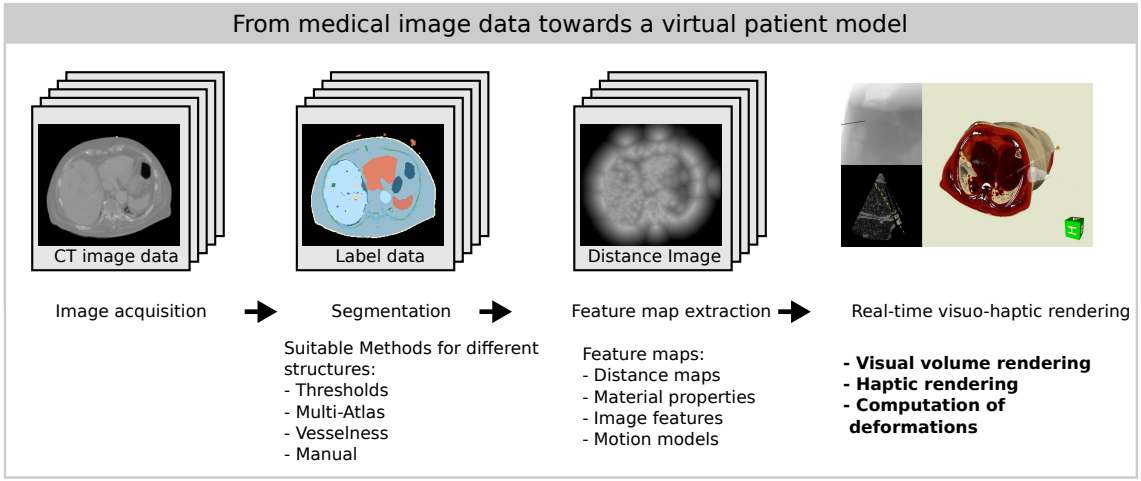

Figure 1.3.: Overview of a conceptual pipeline for image-based patient specific surgery simulation. New algorithms for visuo-haptic rendering are the focus of this thesis.

cholangiography and features a respiration simulation component. Deformations induced by the breathing are computed using a Generalized ChainMail [LB03] approach and visualization uses a surface based approach.

Ni et al. The system presented by [DWJ10] also features a simulation system for ultrasound guided training of biopsy of lesions in the liver. Its main contributions are the fusion of multiple 3D ultrasound images with segmented CT data and visuo-haptic rendering of these. A simple respiratory motion model is used together with deformations of a needle to introduce visible deformations into the rendering. The used haptic parameters have been estimated by the help of expert radiologists and are given for the relevant structures: skin, muscle and liver. It also includes a user study to evaluate the performance of trainees vs. the performance of expert radiologist with and without the simulation of respiratory motion.

Acus-VR The predecessor of the framework here is the Acus-VR [FHGH09] simulator. The scenario of application is lumbar puncture for which static patient models were created based on CT data or colored RGB volume data from the Visible Human data set [SASW96].

1.2. Motivation and Objectives

For visual and haptic rendering, most of the simulation frameworks given in the previous section rely on triangular surface data and mesh models. To produce a sophisticated virtual patient a surface creation process has to be used that is based on real medical image data. For instance medical image data is voxelized 3D image data obtained by a medical imaging process such as CT or MRT imaging. Furthermore, each structure that has to appear in

the simulation needs a segmentation mask, i.e., an assignment of a distinct label to each voxel that is part of the concerned anatomical structure. This segmentation can then be used for the creation of surfaces by the Marching Cubes algorithm [LC87] and follow up post-processing steps [Fär09, Dal14].

The segmentation process is a time consuming task and often involves the manual interaction of a user. For the preparation of a virtual patient for lumbar puncture in Acus-VR [Fär09], a time consuming process, that takes more than 60 h, was needed. Specifically, time measurements given by [Dal14] are 20 h for bone structures, 4 h for ligaments, 3–4 h for intervertebral disks, 7 h for spinal nerves, 3 h for liver, 14–15 h for muscular structures, 4 h for skin and 5–6 h for fatty tissue. Therefore, in case that a physician needs to create a simulation for a specific patient in a fast and inexpensive way, i.e., patient specific virtual reality simulation [WAVH$^+$12], it is desirable to circumvent the segmentation and surface creation steps. Computing all visual and haptic rendering directly based on the image data is the alternative that is the central concern of this thesis.

In Fig. 1.3, a conceptual process for image-based patient specific simulation is proposed. This process consists of four major steps: (1) the acquisition of new patient image data by medical imaging, (2) segmentation of the image data, (3) extraction and generation of additional feature images and finally (4) the real-time simulation of an intervention based on the previous steps. In this thesis the last parts of this pipeline are addressed with a clear focus on the real-time visuo-haptic simulation algorithms and software with the following objectives:

- A) Segmentation effort should be low to keep the time needed for preparation of a new patient specific simulation in a reasonable time frame.
- B) To provide an interactive environment it is necessary that the implemented components exhibit real-time capability.
- C) The virtual environment created by the visuo-haptic rendering should be convincing with regard to the displayed content and produced force feedback.

As a field of application, needle insertion into the liver with special focus on to the percutaneous transhepatic cholangiodrainage (PTCD) has been chosen since this intervention contains various interesting aspects and a training environment is highly desirable as explained in the next chapter.

1.3. Scientific Contributions

The result present in this dissertation is a new “Framework for Image-based Puncture Simulation”, which includes innovative algorithms for visuo-haptic simulation. In contrast to

the state of the art, these methods do not rely on mesh models for visualization, soft tissue deformation simulation or respiratory motion but can work on virtual patient image data defined on regular grids. This way, most segmentations and meshing procedures can be avoided. Overall, the main contributions are:

- **Direct visuo-haptic rendering** for needle insertion, ultrasound probing and palpation. This includes visualization of a virtual patient by ray casting, ultrasound rendering and X-ray simulation, summarized in [FMSH16]. Also, a new haptic algorithm for needle insertion [FWMH15] is presented. For simulation of palpation and ultrasound probing, methods have been developed [FMH14, FMH13a] that use distance images for collision detection and force computation.
- Computation and visualization of **deformation** of soft tissue represented by image data, as occurring in a needle insertion and palpation scenario, can be performed by the simulation framework [FMH12, FMH13a, FMH13b, FMH13c]. It is worth mentioning that the approaches developed for this framework differ from existing methods by using a displacement field with a high resolution in comparison to existing frameworks and methods that generally rely on a simplified unstructured tetrahedral mesh representation of internal structures.
- Introduction of **respiratory motion** (breathing) into this simulation framework was published in [FWMH15, WFMH15]. Using a 4D CT data set and a resulting displacement function or motion model, a high-fidelity simulation of motion caused by the breathing of a virtual patient was developed. Its innovation lies in the fact that the displacement function is defined for each element of the image data of the simulated virtual patient. Its generality is demonstrated by the straight-forward integration into the rendering components of the framework.
- The framework is able to provide a training simulation of a liver puncture scenario by only using **partial segmentations** of important structures of the virtual patient and **labeling heuristics**, also published in [FMSH16]. This way, the time needed for segmentation is largely reduced.

1.4. Overview

The thesis is structured as follows: First in chapter 2 background information is given on the intervention that is simulated. Various design decisions were made during the development of suitable algorithms concerning their parallelization. Since the parallelization has been performed using Nvidia CUDA, an introduction to this language is given as well. Then in the second chapter, general visuo-haptic rendering methods are introduced, consisting of

the visual and haptic algorithms for components such as ray casting and needle insertion. It also includes an algorithm for the visuo-haptic simulation of palpations and ultrasound probing. Chapter 4 presents methods and algorithms for the computation of local deformations caused by tool interaction of the user. Chapter 5 presents a method to introduce global deformations caused by breathing motion into the simulation framework using motion models. The complete PTCD simulator is presented in chapter 6. It incorporates the methods from the previous chapters, additional components for tool selection, description of the visuo-haptic simulation workbench and it also describes the implementation of the overall framework. After summarizing and discussing the overall results in chapter 7, the thesis concludes and suggests further work in chapter 8.

2. Background

Before newly developed methods are presented in the next chapters required background knowledge is given. Firstly, the medical background is summarized, consisting of the description of the intervention that can be trained by the proposed system. Secondly, a short introduction to the CUDA programming language is given, which is considered helpful for the reader to understand various design decisions made for the implementation and runtime analysis of the visualization techniques presented in the following chapters.

2.1. Medical Background: Percutaneous Transhepatic Cholangiodrainage

Percutaneous transhepatic cholangiodrainage (PTCD) is an extension of the percutaneous transhepatic cholangiography (PTC), which is a medical intervention to visualize the biliary tract [DBI11]. The idea of PTC is to insert a needle through the skin (percutaneous) into the liver (transhepatic) and finally into the bile ducts. In this target structure, a contrast agent is injected making the biliary tract visible under fluoroscopy using X-rays similar to angiographic procedures. Initially, this intervention has been developed for diagnostic purposes only and nowadays is replaced by methods such as endoscopic retrograde cholangio(pancreato)graphy (ERP resp. ERCP) [Cot77] or endosonography-guided biliary drainage (EUS-BD) [IDY14]. Modern medical imaging techniques as such as MRI or CT, also enable 3D imaging and visualization of the liver and hepatic ducts.

The PTC intervention was developed further [DBI11] to perform a therapeutic drainage of the biliary vessels in case of obstructed bile ducts. An obstruction is commonly caused by a lesion or gallstone, see Fig. 2.1. As for PTC, needle insertion in the biliary vessels is performed first and afterwards a drainage is placed by insertion of a catheter. For the needle insertion, the surgeon first identifies a suitable insertion site which depends on the location of the target vessel. This location is either in the right or left lobe of the liver, which determines either an intercostal (generally between 10th and 11th rip, Fig. 2.1) or epigasteric access respectively. After having determined the entry location by ultrasound probing and palpation, a small incision is made at the insertion site. Through this incision, a needle

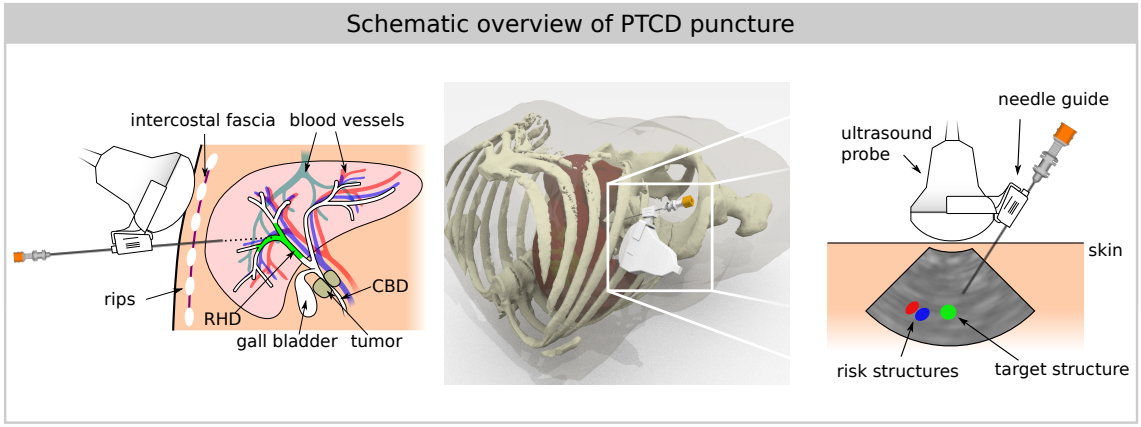

Figure 2.1.: Puncturing the hepatic ducts (green target region) is performed by percutaneous transhepatic insertion of a needle through the intercostal spaces. Ultrasound probing with attached needle guide can be used to ease the procedure. Here, the right hepatic duct (RHD) is punctured since a present tumor obstructs the common bile duct (CBD).

with removable stylet is inserted into the patient's body. The needle diameter can be chosen between 18 to 22 gauge (around 1.2 cm down to 0.7 cm), which influences the flexibility of the needle noticeable (see section A.1 in the appendix for the bending behavior of a 16 resp. 20 gauge needle). Larger needle diameters cause more trauma and thus thinner needles are preferable. Using ultrasound imaging and a needle guide attached to the ultrasound probe can reduce the number of needed repositionings of the needle inside the patient. To check the successful insertion of the needle into the bile ducts, the stylet is then removed and the needle is checked for outflow of bile. Injection of contrast agent and fluoroscopy can also be applied, which shows the contrasted bile ducts in the case that puncture was successful. In cases of failures, the operator can slowly retract the needle and reperform the puncture with a changed insertion direction. Overall, it is desirable to keep the number of trials, and thus the radiation exposure, low, for which training with a VR simulator is considered to be beneficial.

After needle insertion, the PTCD is completed by insertion of the drainage. This is performed by the Seldinger technique, which is applied in many other areas (e.g. angiography) as well: Through the needle, once the stylet is removed, a guide wire is inserted into the biliary vessels. The needle is then removed while the guide wire remains in the patient. Now, the catheter can then be inserted by pushing it over the guide wire and finally the wire is removed. Before inserting the catheter it might be required to widen the insertion channel by a dilator.

PTC by itself is performed less frequently today as there are now better imaging modalities available for the sole purpose of visualizing the bile ducts (see above). PTCD is normally

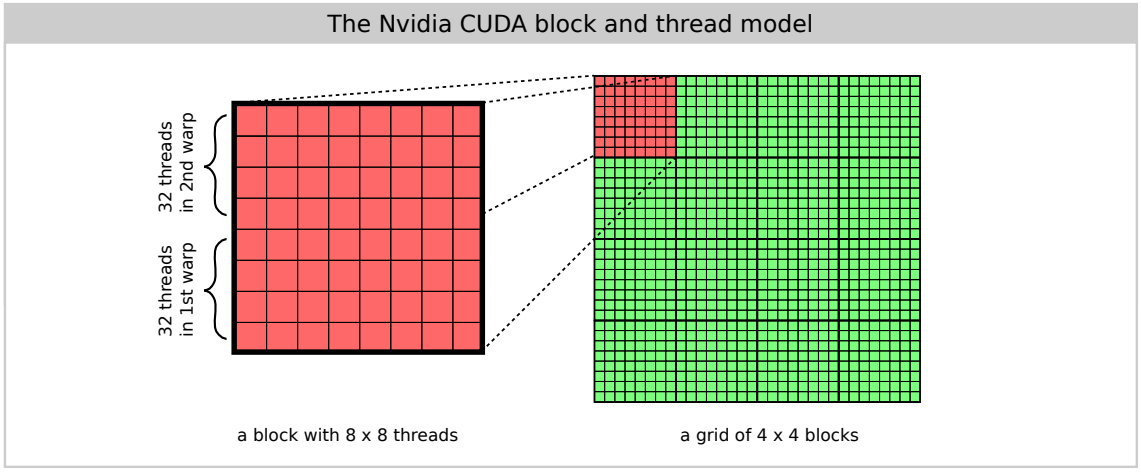

Figure 2.2.: Exemplary block and thread distribution for the computation of the elements of a 32×32 pixel image. The image is subdivided into 4×4 blocks. Each of the blocks then is computed by the multiprocessors of the GPU, whereas for each block 8×8 threads are executed in "warps" of 32 threads.

```
1 __global__
2 void cuDeformKernel( float4* A, float4* B, float4* C )
3 {
4   unsigned int x = blockIdx.x*8 + threadIdx.x;
5   unsigned int y = blockIdx.y*8 + threadIdx.y;
6   unsigned int index = x + y*32;
7   C[index] = A[index] + B[index];
8 }
```

Figure 2.3.: CUDA kernel source code for adding two 2D arrays A and B and storing the result in array C for a 32 × 32 pixel image.

only the second choice if ERCP is not possible or has failed [DBI11]. These facts make it desirable to have a simulation tool at hand for regular training or planning of this intervention. It is also noteworthy that with ultrasound contrast agents a completely radiation free PTCD could be performed making the sonographic part of the intervention even more interesting for visuo-haptic simulation.

2.2. Nvidia CUDA: Using the General Purpose Graphics Processing Unit

Nvidia CUDA is a superset of the C++ programming language giving the developer the possibility to design and implement highly parallelized algorithms [SK10]. The resulting program can be run on general graphics hardware and is highly suitable for algorithms in which the same instructions can be applied to a high amount of input data simultaneously.

Examples for application of CUDA are manyfold. In the medical context these include visualization, simulation and image registration [PX11, SSKH10].

CUDA is based on the "single instruction multiple thread" (SIMT) paradigm. This is slightly different to the "single instruction multiple data" (SIMD) paradigm in so far as SIMT programming allows simultaneous computation with branching. Using SIMD only, the programmer is limited to apply exactly the same instructions on a large amount of data. With SIMT it is possible to write a single piece of code executed in parallel that also contains branching instructions, e.g. the IF statement, leading to different execution paths for each parallel thread.

Abstractly speaking, a program written in CUDA consists of a single set of instructions (the "kernel"), which is distributed onto "blocks" and in each of the blocks, several threads run concurrently. This distribution scheme is exemplary demonstrated for the computation of the elements of a 32×32 pixel image in Fig. 2.2. Fig. 2.3 also lists the code for a simple kernel that adds two gray scale 32×32 pixel images. In each thread running this code the location in the block and thread layout is known, enabling the thread to compute and store the desired image values.

The actual distribution and execution by the CUDA cores is best understood by considering the hardware architecture of a modern general purpose GPU. Here, this is explained for the GPU that is installed in the Nvidia Geforce GTX680 [Nvi12c] graphics card, which has been used as the target architecture for design, implementation and evaluation of the visualization algorithms in this thesis. The GTX680 uses one GK104 GPU from the Kepler family, which is fabricated using a 28 nm process. As illustrated in Fig. 2.4, this GPU consists of four Graphic Processing Clusters, each divided into two Next-Generation Streaming Multiprocessor (SMX) units. Each SMX itself comprises of 192 CUDA cores, memory (instruction cache, registers, shared memory) and further functional units (PolyMorph 2.0 engine, warp schedulers, see [Nvi12c]).

With a total number of 1536 cores, a very high number of threads can be executed simultaneously. In comparison, CPUs have a much lower number of cores available. As an example, the Intel Core i7 CPU used for benchmarking has six distinct cores. During execution of a kernel, a group of 32 CUDA cores each processes 32 threads at the same time. Together, the 32 threads form what is called a warp. If a warp waits for an instruction to complete for the 32 parallel threads, stalling occurs. When this happens, the SMX can switch the context of the cores executing the warp to a different warp. This switching is handled by one of the four "warp schedulers" on the SMX. A high number of available registers makes it possible to perform the context switch without having to save and restore the registers for

the threads. Depending on the number of registers used by the kernel, registers for up to 64 active warps are available on a single SMX [Nvi12b]. For the programmer it is essential to maximize performance by ensuring that a high number of warps are active on the SMX. This is measured by occupancy, which is the actual number of active warps divided by the maximum number of active warps for a streaming multiprocessor [Nvi12a].

The programmer has the possibility to use 64 KB of shared memory per SMX (L1 cache) exclusively accessible by the threads run on the SMX. Apart from this shared memory, global memory is available for read and write access by all threads. It is not possible for threads that are not executed on the same SMX to communicate with each other without using the global memory. However, the order of execution of blocks and threads is not specified and may vary for different calls of a kernel. In the worst case, threads might wait for the execution of other threads for which processing will not start until the resources already blocked are freed, deadlocking the whole execution. Knowledge of this fact and the resulting implications is crucial for the successful implementation of parallelized algorithms in CUDA. For instance, performing a reduction i.e., the computation of a single value based on the results of all threads generally needs a separate kernel call and also a special strategy, see for example [Lui14]. Avoiding algorithms that rely on reductions are thus better suited for parallelization.

Another critical aspect is the avoidance of "thread divergence", which occurs if the execution path of the threads of a single warp diverge by conditional execution, i.e., different branches of an if-then-else conditional construct are taken for different threads. To resolve this the SMX then has to stop the concurrent execution of members of the warp and has to handle each thread separately, slowing down the execution. Different execution paths for different warps are not problematic in this context.

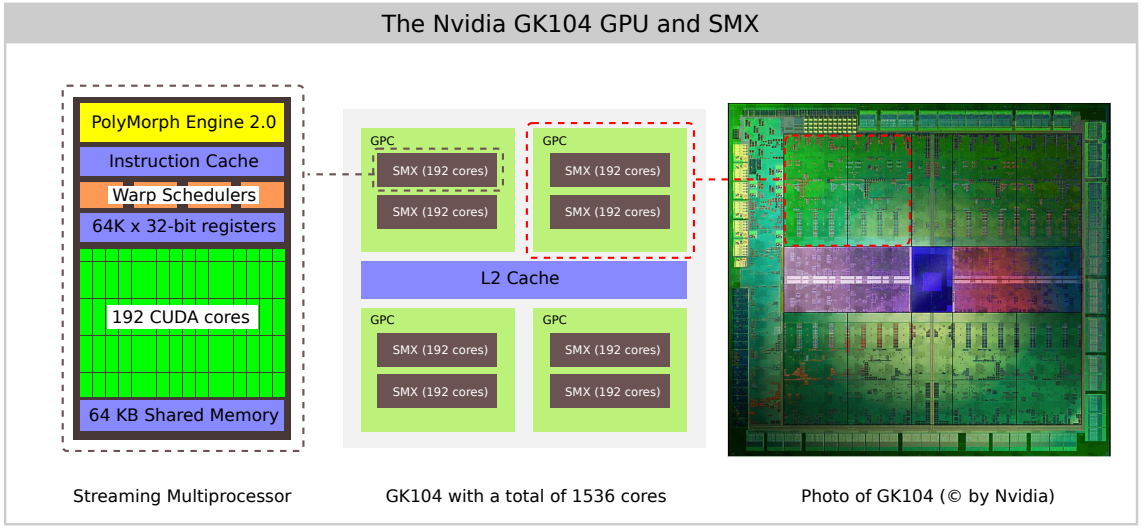

Figure 2.4.: Schematic hardware architecture of the Nvidia GK104 GPU and photo of the actual die. The chip contains four Graphic Processing Clusters (GPC), consisting of two Streaming Multiprocessors (SMX) each. Altogether, the GPU facilitates 1536 CUDA cores capable of processing the same number of threads simultaneously.

3. Direct Visuo-haptic Volume Rendering Algorithms

The following chapter will lay the foundation for chapters 4, 5 and 6. It explains how a CT image can be used to represent a virtual patient in a visuo-haptic simulation without surface or volumetric mesh representations being necessary for visual and haptic rendering. In the preceding lumbar puncture simulation framework [FHGH09], surface meshes had to be created and thus indirect volume rendering techniques were used (Marching Cubes [LC87]). Here, the mesh creation prepossessing steps are omitted and rendering is performed *directly* on the image data for both visualization and haptics. This is a unique property of the simulation system, distinguishing it from previous available puncture simulation systems. It builds on haptic volume rendering algorithms given in [LYG02], which were adapted for needle insertion simulation in the lumber puncture framework [FHGH09, MFH12].

First, the chapter presents new methods for realistic visual volume rendering based on ray casting. Its novelties include the combined rendering of a voxel image and a deformed sub-region and comprehends shadow rendering and different rendering modes. Furthermore, a fluoroscopy simulation with a novel contrast agent propagation algorithm based on diffusion is included. It can be used for simulated angiography of the biliary ducts. These methods have been mainly published in [FMSH16].

Apart from the visual rendering, haptic rendering of a bendable needle is presented that is capable of puncturing non-linear paths, distinguishing it from preceding versions. It was published in [FWMH15], with additionally taking into account respiratory motion. Here, it will first be explained for a static patient model, which is detailed later in chapter 5 for included respiratory motion. Also, a new haptic algorithm for palpation and ultrasound probing is presented, published previously in [FMH13a] and [FMH14].

3.1. Input Data and Notations

The following methods and algorithms for visual and haptic rendering are depending on volumetric image data and are influenced by interaction using a haptic device. The used

notations for the image data and data from the haptic device thus shall shortly be defined: The CT image is stored in a voxelated fashion, which means that image values are only stored for elements on a regular grid $\dot{\Omega} \subset \mathbb{R}^3$. For rendering this image, image values in the space in between the grid points are of interest, making it necessary to have a definition for the image that is defined on a continuous domain $\Omega_I \subset \mathbb{R}^3$. This definition of the image will be denoted by the function $I : \Omega_I \to \mathbb{R}$ in the following. In case of evaluation of $I(\mathbf{x}_i)$ at positions $\mathbf{x}_i \in \Omega_I \setminus \dot{\Omega}$ that are not lying on the image grid $\dot{\Omega}$, trilinear interpolation is chosen since it is fast to evaluate. Apart from the function I, a function $J : \Omega_J \to \mathbb{R}$ with $\Omega_J \subset \Omega_I$ represents a small deformed part of the image, for which the next chapter will present computational methods. The visualization algorithms uses a combination of both image functions by the following sampling function

$$s(\mathbf{x}_i) = \begin{cases} I(\mathbf{x}_i) & \text{if } \mathbf{x}_i \in \Omega_I \setminus \Omega_J \\ J(\mathbf{x}_i) & \text{if } \mathbf{x}_i \in \Omega_J \\ 0 & \text{else} \end{cases} \tag{3.1.1}$$

Based on segmentation masks or heuristics, a labeling function $l : \mathbb{R}^3 \to L \subset \mathbb{N}^0$ will be used that yields a distinct label for a given location and 0 representing air resp. no segmentation available. The subsets L_{risk} and L_{target} of L are intervention specific. See A.2 in the appendix for a list of labels and associated structures for the PTCD intervention. Similar to the sampling function $s(\mathbf{x}_i)$, the labeling function uses a combination of a deformed and an undeformed image function but uses nearest neighbor instead of trilinear interpolation. For this chapter, this definition is sufficient but chapter 6 will provide an extended definition based on a combination of partial segmentations and heuristics.

The position of the haptic device in the virtual scene will be denoted by $\mathbf{x} \in \mathbb{R}^3$. Its orientation is encoded in the unit quaternion $\mathbf{q} \in \mathbb{H}, \|\mathbf{q}\| = 1$. Quaternions are a more compact form for representation of orientation than rotation matrices [Kui02]. Here, the corresponding rotation matrix will be denoted as $\mathbf{Q} \in \mathbb{R}^{3\times3}$. Often, only the direction $\mathbf{q}_z = \mathbf{q} \cdot (0,0,1)^\top \cdot \mathbf{q}^* = \mathbf{Q} \cdot (0,0,1)^\top, \mathbf{q}_z \in \mathbb{R}^3, \|\mathbf{q}_z\| = 1$ will be needed in the following. It represents the z-axis of the haptic device handle, i.e. the direction the handle is pointing.

For the haptic algorithms, and as well as in following chapters, properties updated in an iteration of an algorithm are marked by $^+$. For instance, $\mathbf{x}^+$ is the updated value of $\mathbf{x}$ for a single iteration of an algorithm. It is simply a more convenient form of writing $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$, respectively.

3.2. Ray Casting based Volume Rendering Methods

Visualization of a voxel image can be performed in several ways. One way is ray casting, which is extensively explained in [EHK+06]. It can be used for realistic rendering of the image data with the purpose of presenting a virtual environment with high immersion. In contrast to means of realistic and immersive visual rendering to provide a plausible and convincing setting, it is also of interest to use the visualization in a functional way. This means, it should be possible to inspect and explore the image data for educational and planning purposes as it would not be possible with an actual patient. Both realistic and functional rendering aspects are included in the ray casting presented in the following.

3.2.1. Ray Casting

Abstractly speaking, ray casting assumes the volumetric image to consists of a gas of particles that has certain light emitting and absorbing properties. It can be evaluated for viewing rays that pass through the volume by the rendering integral [EHK+06]. In simple terms, a ray is traversed for each pixel of the output image through the volume and color values are sampled and accumulated along this ray. This ray $R \subset \mathbb{R}^3$ consists of equidistantly placed sampling points $\mathbf{x}_i$. Here, the RGBA color value $v(\mathbf{x}_i) \in [0,1]^4$ at each sample point is determined by the Hadamard product of the value of a transfer function and a mixed shading and shadowing model

$$v(\mathbf{x}_i) = c(s(\mathbf{x}_i)) \circ f_{\text{shade}}\left(s(\mathbf{x}_i)\right) \tag{3.2.1}$$

The transfer function $c : \mathbb{R} \to [0,1]^4$ relates the Hounsfield units of the CT image to a RGBA color tuple. In the implemented framework, n transfer functions, each defined for a single organ or tissue class, can be combined into a single one by

$$c(v) = \sum_{1 \le i \le n} \alpha_i a_i(v) \begin{pmatrix} r_i(v) \\ g_i(v) \\ b_i(v) \\ 1 \end{pmatrix} \tag{3.2.2}$$

where $\alpha_i \in [0,1]$ is a weight and $r_i, g_{i,}, b_i, a_i$ are the individual channels of each of the combined transfer functions. Alternatively, the function can be replaced by a piecewise linear windowing ramp, which is also used in the multiplanar reformations already included in the predecessor of the framework [Fär09].

Shading is performed by Phong-Shading based on the sum of the components for ambient, diffuse and specular lighting. A detailed explanation of Phong-Shading is included in [EHK+06, pp. 114]. Using the lighting components, the shading function can be written as

$$f_{\text{shade}}\left(s(\mathbf{x}_i)\right) = \begin{pmatrix} \text{ambient} + (\text{diffuse} + \text{specular}) \cdot f_{\text{shadow}}(s(\mathbf{x}_i)) \\ \text{ambient} + (\text{diffuse} + \text{specular}) \cdot f_{\text{shadow}}(s(\mathbf{x}_i)) \\ \text{ambient} + (\text{diffuse} + \text{specular}) \cdot f_{\text{shadow}}(s(\mathbf{x}_i)) \\ 1 \end{pmatrix} \tag{3.2.3}$$

The terms of diffusive and specular lighting are linearly influenced by the shadowing term, which is explained in section 3.2.3.

Currently available visualization frameworks that include volume rendering implementations only support static volume images. This fact made it necessary to implement a custom rendering framework that also supports the visualization of deformed parts of the image, i.e. that is capable of using the sampling function defined by Eq. 3.1.1.

To increase the speed of rendering, the number of potential sampling points is decreased by first limiting the rays to the boundaries of the image by using ray-box intersection [KK86]. Further improvement is achieved by leaping over empty space [AW87] that is represented by a low resolution binary mask. Also, the ray casting is terminated early in case the resulting color accumulator is fully saturated. These optimizations are illustrated in the left part of Fig. 3.1; the sampling start is set at position $\mathbf{x}_{\text{start}}$, caused by the ray-box intersection and empty space leaping. Sampling ends as soon $\mathbf{x}_{\text{end}}$ is reached or the color accumulator is saturated.

3.2.2. Clipping and Tagging

In a training environment for educational purposes it is desirable to be able to easily inspect the internal parts of the virtual patient although this is obviously not possible in a real intervention. The same applies for planning application of the framework. To provide inspection of internal structures, a "clipping" mode that uses the tool or haptic device orientation and position to hide one half of the virtual patient is part of the ray casting, which is illustrated in the right part of Fig. 3.1. For elements of this plane, shading and shadowing is disabled by setting $f_{\text{shade}} = (1, 1, 1, 1)^\top$. Also, the elements on this clipping plane can be augmented by various tagging methods by replacing the transfer function term in Eq. 3.2.1. These tagging methods are:

1. No tagging and shading. Simply use the transfer function to determine the color value of the elements on the clipping plane: $v_1(\mathbf{x}_i) = c(s(\mathbf{x}_i))$

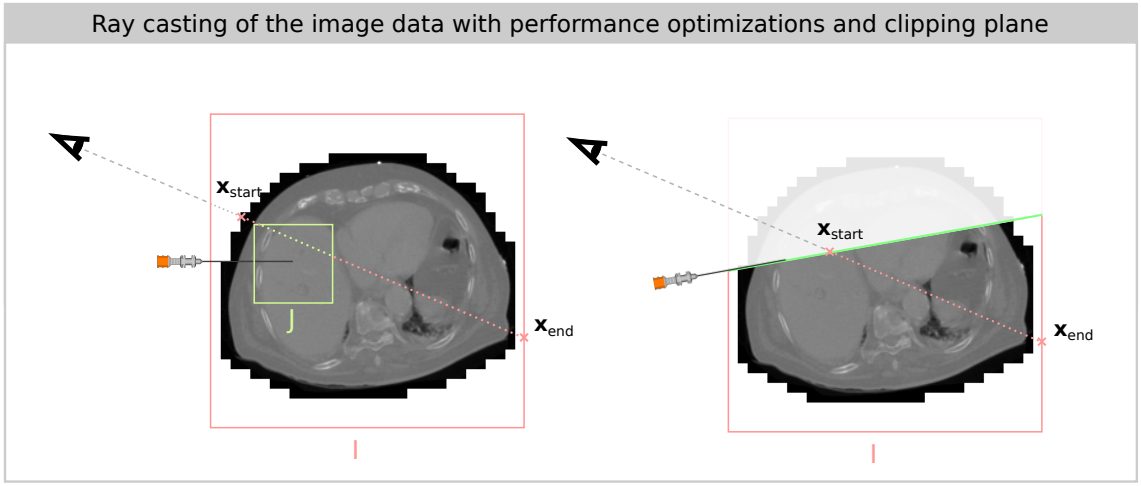

Figure 3.1.: Overview of ray casting along a single ray. Left: After ray-box intersection (gray dashed line) and empty space leaping (gray dotted line), sampling starts at $\mathbf{x}_{start}$ until the accumulated color value is fully saturated or $\mathbf{x}_{end}$ is reached. If a sampling point lies within $\mathbf{J}$, the deformed volume is sampled, otherwise $\mathbf{I}$ is used. Right: To inspect internal structures of a virtual patient, parts of the patient can be hidden by a clipping plane. Rendering starts on this clippling plane to reduce the number of sampling points.

2. Target and risk structure overlay. Mix the color value determined by the transfer function with an overlay color representing target structures or risk structures. The color depends on the label associated with the element $\mathbf{x}_i$

$$v_2(\mathbf{x}_i) = 0.5 \cdot c(s(\mathbf{x}_i)) + \begin{cases} (0, 0.5, 0, 0.5)^\top & \text{if } l(\mathbf{x}_i) \in L_{\text{target}} \subset L \\ (0.5, 0, 0.2, 0.5)^\top & \text{if } l(\mathbf{x}_i) \in L_{\text{risk}} \subset L \\ 0.5 \cdot c(s(\mathbf{x}_i)) & \text{else} \end{cases} \tag{3.2.4}$$

3. Label color overlay. Use a function $c_{\text{label}} : L \to [0,1]^4$ that yields a color tuple for a given label. Set the color of the element to the color associated with the label of $\mathbf{x}_i$

$$v_3(\mathbf{x}_i) = c_{\text{label}}(l(\mathbf{x}_i)) \tag{3.2.5}$$

3.2.3. Shadow Rendering

Depth perception of a virtual scene is important for immersion. As for example in painting, a method to create the illusion of depth is to use perspective and perspective projection. Naturally, depth is perceived by the human mind by fusing the images of the two eyes. Stereoscopic display, that is the display of two images taken from two different points of views, is already a standard technique of modern multi-media. Another property that indicates

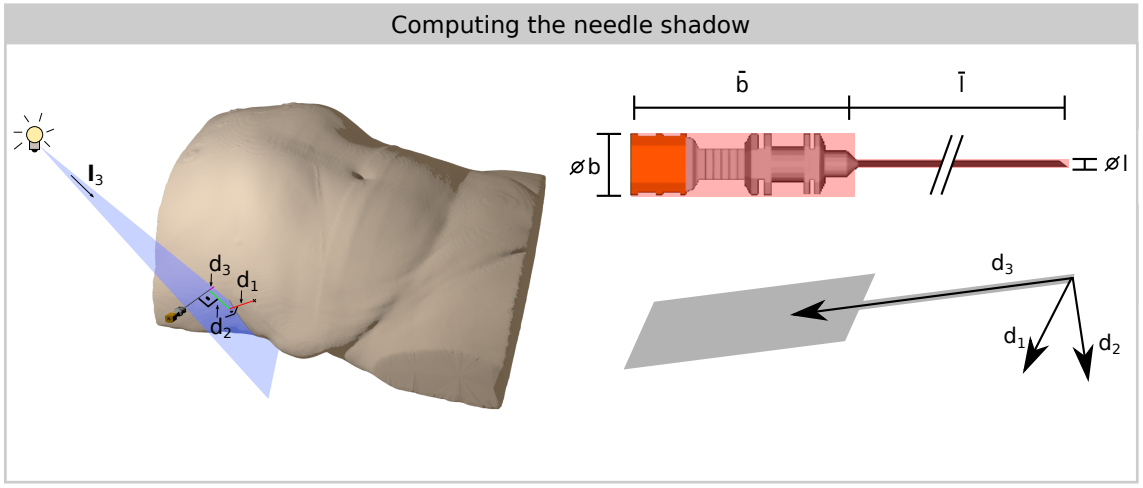

Figure 3.2.: Shadow rendering is based on the light direction **l** and pose of the needle. The needle shadow is approximated by the red overlay on the right side.

depths and spacial relations is shadowing and shading. For rendering of triangular surfaces by rasterization, this is often implemented by shadow mapping [Wil78]. The basic idea is this: Before rendering the scene from the position of the camera, it is first rendered from the position of a light source and the resulting depth buffer is stored. It is then used during the actual rendering pass to determine if the currently rendered fragment is in shadow.

This approach requires a first rendering pass from the position of the light source and also an additional texture lookup for each sample position on a ray during volume rendering. For means of needle insertion and basic tool interaction, it is only necessary to indicate to the user the distance towards the skin of the virtual patient. Therefore, the shadow cast by the tools or virtual finger is reduced to a suitable simplification and the resulting shadow is represented implicitly in form of an algorithmic function. For a directed sun light source, i.e. a light source with only direction but no explicit position (infinitely far away), this is illustrated in Fig. 3.2. For a single sampling point $\mathbf{x}_i$, a given light direction **l**, needle direction **n**, needle tip position **x** and needle length $\bar{l}$, the values d_1, d_2 and d_3 are calculated and used in algorithm 3.1 to determine if the sampling point is within the shadow of the needle, giving a value for the shadowing term $f_{\text{shadow}}(s(\mathbf{x}_i))$ in Eq. 3.2.3. The values returned by the algorithm are constants. The value $f_{\text{shadow}}(s(\mathbf{x}_i)) = \text{f}_{\text{out}} = 1$ indicates that the sampling point is not in the shadow and thus the calculated color is not affected for the sampling point. The second case, i.e. $f_{\text{shadow}}(s(\mathbf{x}_i)) = \text{f}_{\text{in}} < 1$ indicating that the sampling point is within the shadows and the shading terms for diffusive and specular lighting are affected. For a shadow that is cast by a sphere, algorithm 3.2 is used. The shadows cast by the palpating finger and the ultrasound probe are approximated this way.

Algorithm 3.1 Algorithm for determination if a sampling point $\mathbf{x}_i$ is withing the simplified needle shadow.

input: $\mathbf{x} \leftarrow$ needle tip position
input: $\mathbf{x}_i \leftarrow$ sampling point position
input: $\mathbf{l} \leftarrow$ normalized light direction
input: $\mathbf{n} \leftarrow$ normalized needle direction
input: $\bar{l} \leftarrow$ needle length
input: $\bar{b} \leftarrow$ needle base length
input: $\varnothing l \leftarrow$ needle diameter
input: $\varnothing b \leftarrow$ needle base diameter
output: shadowing function value $f_{\text{shadow}}(s(\mathbf{x}_i))$
$\mathbf{q} \leftarrow \frac{\mathbf{l}-\mathbf{n}(\mathbf{n}\cdot\mathbf{l})}{\|\mathbf{l}-\mathbf{n}(\mathbf{n}\cdot\mathbf{l})\|}$
$\mathbf{d} \leftarrow \mathbf{x}_i - \mathbf{x}$
$\mathbf{p} \leftarrow \mathbf{l} \times \mathbf{n}$
$\mathbf{j} \leftarrow \mathbf{p} \times \mathbf{l}$
$d_1 \leftarrow \frac{\mathbf{d}\cdot\mathbf{p}}{\mathbf{p}}$
$d_2 \leftarrow \mathbf{q} \cdot \mathbf{d}$
$d_3 \leftarrow \frac{\mathbf{j}\cdot\mathbf{d}}{\mathbf{j}\cdot\mathbf{n}}$
if $|d_1| < \frac{1}{2}\varnothing l$ **and** $d_2 > 0$ **and** $0 < d_3 < \bar{l}$ **then**
    return f_{in}
else if $|d_1| < \frac{1}{2}\varnothing b$ **and** $d_2 > 0$ **and** $\bar{l} < d_3 < \bar{l} + \bar{b}$ **then**
    return f_{in}
end if
return f_{out}

Algorithm 3.2 Algorithms for shadowing of a simple sphere shaped object.

input: $\mathbf{x} \leftarrow$ needle tip position
input: $\mathbf{x}_i \leftarrow$ sampling point position
input: $\mathbf{l} \leftarrow$ normalized light direction
input: $\mathbf{n} \leftarrow$ normalized needle direction
output: shadowing function value $f_{\text{shadow}}(s(\mathbf{x}_i))$
$r \leftarrow$sphere radius
$k \leftarrow \mathbf{l} \cdot (\mathbf{x}_i - \mathbf{x})$
$d \leftarrow \|(\mathbf{x}_i - \mathbf{x}) - k\mathbf{l}\|$
if $k > -r$ and $d < r$ **then**
    return f_{in}
end if
return f_{out}

3.2.4. Combined Volume Rendering and Rasterization

Surface based rendering methods used in the framework are implemented using VTK, which relies on OpenGL for rasterization of surface meshes. It is possible to interface OpenGL and CUDA without having to copy data back to the host system's memory. Without this interface, each result computed using CUDA would have to be copied from device (GPU) memory to host (CPU) main memory and back to GPU memory, which is a costly operation. In practice, this interoperation is performed by OpenGL pixel buffer objects (PBO) and is well supported by the CUDA API. For the combination of a ray casting of volume data performed in CUDA and rasterized objects (such as for example the triangle mesh of the virtual needle) in the OpenGL framebuffer, a shared depth buffer is needed. This depth buffer is necessary to perform either hidden surface removal for the rasterization or for early ray termination in the volume rendering. Otherwise, one of either the volume rendering or rasterization would be simply rendered in front of the other without proper occlusions. In terms of rendering a combined volumetric CT image and a needle mesh, without proper occlusions, no visual penetration effect of the needle into the virtual patient model could be created.

Since it is not possible to interface the OpenGL framebuffer or depth buffer directly from CUDA, two options arise: One is to perform ray casting first, and store the resulting depth values in a separate texture. Then, this depth image can be written to the depth buffer by drawing a viewport filling square and a custom GLSL shader that only writes the texture values to the depth buffer. The second option is to first perform the rasterization, followed by copying the resulting depth buffer to a CUDA array. This is not trivial, since the OpenGL function that provides the copy is very slow when copying from the standard depth buffer. To perform this with reasonable performance, the rasterization process has to be performed on a frame buffer object (FBO), which is not part of the standard VTK visualization pipeline. It is possible to realize by using VTK render passes and implementing a custom render pass that initializes rendering to a frame buffer object. Both options have been implemented but the latter was finally chosen, because this way, rasterization is performed first, followed by the volume rendering. This is important for rendering highly translucent volumes without artifacts.

3.3. X-Ray Rendering and Contrast Agent Diffusion

The previous generation of the framework (ACUS-VR) featured a CPU-based implementation of fluoroscopy simulation [Fär09], that did neither include deformations nor the spread-

ing of contrast agent in the bile ducts. In PTCD, it is necessary to check for contrast agent in the bile ducts to confirm a successful puncture of the biliary vessels before placement of a catheter. To this aim, the ray casting from the previous sections is adapted for simulated X-ray. The simulation of X-ray and angiography has been performed previously: This includes digitally reconstructed radiography based on CT data [RRM+05, SNC90], angiography simulation [WAC07] and training environments for brachytherapy [GSMS13] or C-arm based fluoroscopy [WDL+12]. The contribution here is a simulation that incorporates contrast agent and simulates the distribution in the bile ducts by a diffusive model on a regular grid.

As stated above, the approach is based on the ray casting methods from the previous sections. Instead of using a color transfer function, the attenuation a along a ray is used to compute each pixel of the output image [RRM+05]. The attenuation along a ray is included in the Beer-Lambert law $I_{\text{out}}/I_{\text{in}} = e^{-a}$ with radiation influx I_{in} and outflow I_{out}. To compute the total attenuation, the ray can be subdivided into segments separated by the sampling points. For each sampling point $\mathbf{x}_i \in R$ on the ray R with sampling distance Δx, the attenuation is $a_i = \mu(\mathbf{x}_i)\Delta x$ with a given attenuation coefficient $\mu(\mathbf{x}_i)$. This gives a total attenuation of $a = \sum_{\mathbf{x}_i \in R} \mu(\mathbf{x}_i)\Delta x$ [RRM+05]. Using the linear equation $\mu(\mathbf{x}_i) = \frac{s(\mathbf{x}_i)\cdot\mu_{\text{water}}}{1000} + \mu_{\text{water}}$ from [SNC90], the Hounsfield units given by the sampling function $s(\mathbf{x}_i)$, which combines I and J, can be used to calculate the attenuation coefficient for a sampling point. The attenuation coefficient of water is depending on the energy of the photons emitted by the X-ray device. Values for attenuation coefficients for different energies can be found in [HS04]. In the implementation presented here, a value of 0.1 MeV has been chosen, given an attenuation coefficient of $\mu_{\text{water}} = 0.17\,\text{m}^{-1}$.

Apart from the attenuation caused by the tissue represented in the CT data, the contrast agent also attenuates the amount of photons passing through the virtual patient. For fluoroscopy, iodine is an often used contrast agent. For the given energy of 0.1 MeV, iodine has a high attenuation of $\mu_c(\mathbf{x}_i) = 1.942\frac{\text{cm}^2}{\text{g}}\rho(\mathbf{x}_i)$ for a given density $\rho(\mathbf{x}_i)$. The overall attenuation for a single ray then can be modeled by $a = \Delta x \sum_{\mathbf{x}_i \in R}(\mu(\mathbf{x}_i) + \mu_c(\mathbf{x}_i))$.

Given the segmentation of the bile ducts, the propagation of contrast agent can be simulated by a diffusion process. The diffusion process is defined similar to heat diffusion using explicit Euler integration

$$\rho^+ = \rho + \tau\Delta\rho + \tau\iota \tag{3.3.1}$$

with τ being the step width of the process, Δ the Laplace operator and ι the inflow of contrast agent. To simulate contrast agent injection at the needle tip, the inflow is set to a constant value at the tip and is zero everywhere else. Since the bile ducts are modeled by a grid, this is performed by increasing the contrast agent density of the grid element that is the nearest

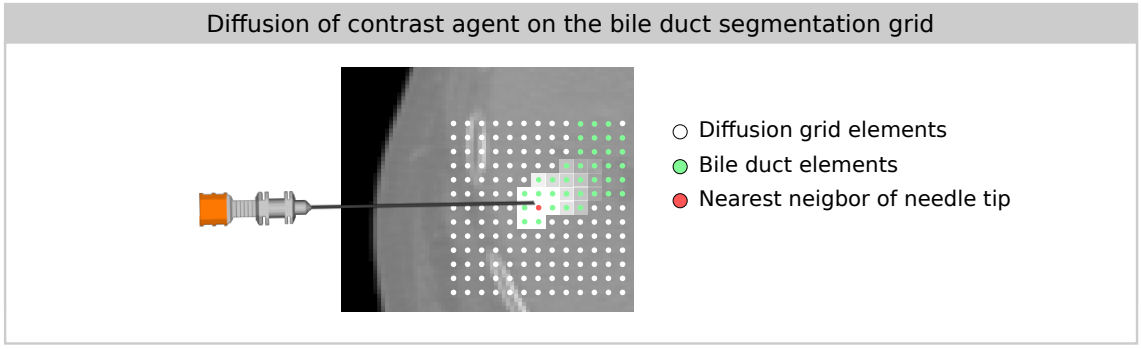

Figure 3.3.: Overview of the diffusion process of contrast agent, which takes place on the grid given by the bile duct segmentation. Contrast agent flows in at the element closest to the needle tip.

neighbor of the needle tip, see Fig. 3.3. In the implementation, this process takes place on a cubic grid with a fixed number of elements much lower than the grid of the image data to decrease the time needed for computation. The grid is placed centered on the puncture location of the bile ducts during run-time. Using CUDA, the process can be implemented efficiently by first adding the inflow to the density value of the nearest neighbor grid element and then by computing the density for each grid element i by a single thread

$$\rho_i^+ = \rho_i + \begin{cases} \sum_{j \in N} \rho_j - |N| \, \rho_i & \text{if } l(\mathbf{x}_i) = L_{\text{ducts}} \\ 0 & \text{else} \end{cases} \tag{3.3.2}$$

with N being the grid neighbors of element i that are part of the bile duct segmentation mask L_{ducts}. The term $\sum_{j \in N} \rho_j - |N| \, \rho_i$ approximates the Laplace-operator without conditions at the boundaries. For instance, if an element is part of a thin tubular structure, i.e. it only has two neighbors on opposite sides, only these two neighbors will be used for computation of the Laplace operator and all other neighbors are discarded. Since the elements of the diffusion grid that are part of the bile duct segmentation will be sparse, only a subset of elements will be changed by Eq. 3.3.2, reducing the number for which the evaluation of the Laplace operator has to be performed.

3.4. Simulated Ultrasound Imaging

As depicted in the background chapter, PTCD is often supported by ultrasound probing with the purpose of reducing the number of needle repositionings [DBI11]. In comparison to only using fluoroscopy for performing the PTCD, the patient is not exposed to X-ray radiation by the ultrasound probing and thus providing a training scenario that supports

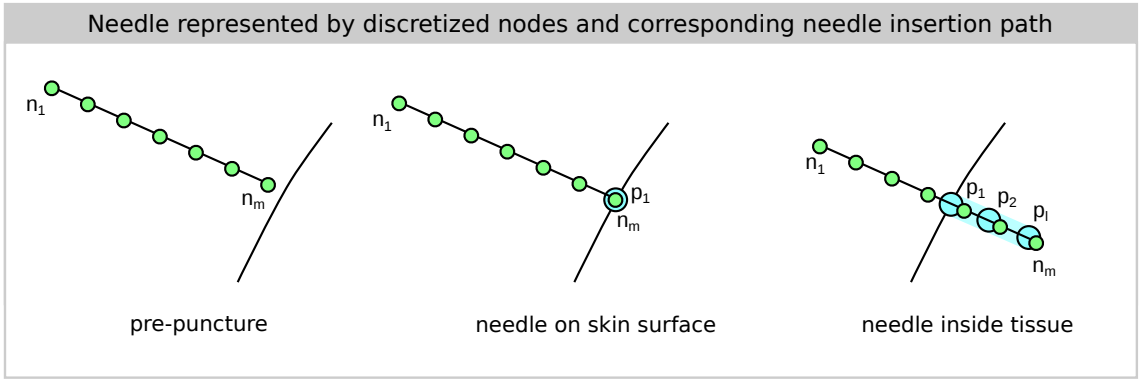

Figure 3.4.: Example of the needle nodes $\mathbf{n}_j$ and path nodes $\mathbf{p}_i$ for three phases during needle puncture. The needle is always discretized by the same number of nodes. The path nodes form the insertion channel and their number depends on the puncture phase and insertion depth of the needle.

ultrasound simulation may reduce the overall X-ray exposure of patient and operator in real interventions.

The implementation used here [SMFH14] is based on the methods of [RPAS09], [KKWN09] and [WBK+08], which use patient CT image data to estimate the acoustic properties of the patient's soft tissue. An addition is the use of partial segmentations. Also, tissue and organ boundaries can be visualized, and also a prototypical Doppler mode simulation is included.

To compute the ultrasound image, several rays originating at the head of the transducer are cast through the image volume. In a first step, the image values of a fixed number of equidistant points are sampled. Then, these are converted to acoustic properties for the elements of the resulting simulated ultrasound image. Liver blood vessels are visualized by a Doppler-mode simulation that assumes that from known blood pressure $p(t)$, the magnitute of the blood's flow velocity can be estimated by $v(t) = \sqrt{2 \cdot (p(t)/\rho_{\text{blood}})}$. Given this value, arteries and veins are visualized by a red-to-yellow resp. blue-to-cyan color gradient.

3.5. A Bendable Needle

The predecessor of the presented simulation system included a simulated bendable needle. However, it was limited in the following aspects:

- Puncturing of non-linear paths was not possible, the direction of insertion was fixed from the moment of initial puncturing of the skin of the virtual patient.
- The elements of the needle are connected to multiple proxies and in an iteration of the algorithms, the value of the image I and the segmentation mask have to be evaluated at each proxy position, which is a time consuming process.

To mitigate these issues, the following needle algorithm has been developed. Its major difference is the fact that two sets of needle nodes are used. This is similar to finite element needle insertion simulation as presented in [CAR+09]. One set represents the actual discretized needle and is also used to visually represent the needle, see [FDBH09]. The other set represents the path of the needle in the virtual patient. This insertion path does not have to be linear and these sets are defined as

- $N = \{\mathbf{n}_j \in \mathbb{R}^3\}_{1 \leq j \leq m}$: nodes representing the discretized needle
- $P = \{\mathbf{p}_i \in \mathbb{R}^3\}_{1 \leq i \leq l}$: nodes representing the needle insertion path

In Fig. 3.4, the behavior of these two sets are demonstrated for three phases of a needle insertion: As long as the needle is outside the virtual patient, the needle path set P is empty. As soon as the needle tip gets in contact with the virtual patient's skin, a first node $\mathbf{p}_1$ is placed at the insertion position. During further insertion, new elements are added to the set while the needle is advancing and elements are removed while the needle is retracted. In summary, the cardinality of P indicates the state of the needle:

- $|P| = 0$: the needle is outside the body,
- $|P| = 1$: the needle tip is on the skin surface of the patient,
- $|P| \geq 2$: the needle is inserted into the patient.

The set P is updated in each iteration of the haptic algorithm based on the position $\mathbf{x}$ of the haptic device. A desirable property of this algorithm is that the positions of the equidistantly distributed path nodes do not change with exception of the path tip node, making it possible to only update the values of material properties and material label at this tip node. Otherwise, this would be necessary for each node along the path, which would increase computational time.

The haptic algorithm is summarized in Alg. 3.3. It contains the update of path nodes, physics simulation for the needle nodes and force feedback computation. These will be explained now in detail.

3.5.1. Updating the Insertion Path

Associated to each of the nodes is a normalized direction vector. For the path nodes, this is denoted by $\mathbf{d}_i \in \mathbb{R}^3$ and for the needle nodes by $\mathbf{r}_j \in \mathbb{R}^3$. 3.4. To include a steering effect into the needle path, the tangential direction $\mathbf{d}_l$ in which the tip path node $\mathbf{p}_l$ advances into the tissue is set to the needle node direction vector $\mathbf{r}_m$ as illustrated in Fig. 3.5. This update step is denoted as $\mathbf{d}_l \leftarrow$ `calcTipDirection`$(...)$ in Alg. 3.3. All the tangents $\mathbf{r}_j$ and needle node positions $\mathbf{n}_j$ are updated in each iteration of the haptic algorithm by a physical needle model in case the needle is inside tissue, as described later in the next section.

Algorithm 3.3 One iteration of the haptic loop. It computes new node positions P^+ and N^+ as well as the haptic force **f** and torque **t**.

input: $\mathbf{x} \leftarrow$ haptic device position
input: $\mathbf{q} \leftarrow$ haptic device orientation
input: $P \leftarrow$ set of path nodes from prev. step
input: $N \leftarrow$ set of needle nodes from prev. step
output: $N^+, P^+, \mathbf{f}, \mathbf{t}$
if $|P| \neq 0$ **then**
  $\mathbf{d}_l \leftarrow$ `calcTipDirection(...)` // update tangential direction of path tip
end if
$P^+ \leftarrow$ `updatePathNodes(...)` // update path nodes
if $|P^+| \neq 0$ **then** // if needle is inside of patient model
  $N^+ \leftarrow$ `updateNNodes`$(P^+, N, \mathbf{x}, \mathbf{q})$ // use physics simulation to update needle nodes
  $\mathbf{f} \leftarrow$ `computeForces`$(P^+, N^+, \mathbf{x})$ // compute force feedback
  $\mathbf{t} \leftarrow$ `computeTorque`$(P^+, N^+, \mathbf{x})$
else // if needle is outside of patient model
  $N^+ \leftarrow$ `simpleNNodePlacement`$(\mathbf{x}, \mathbf{q})$ // place nodes on a line/no forces
  $\mathbf{f} \leftarrow 0$
  $\mathbf{t} \leftarrow 0$
end if

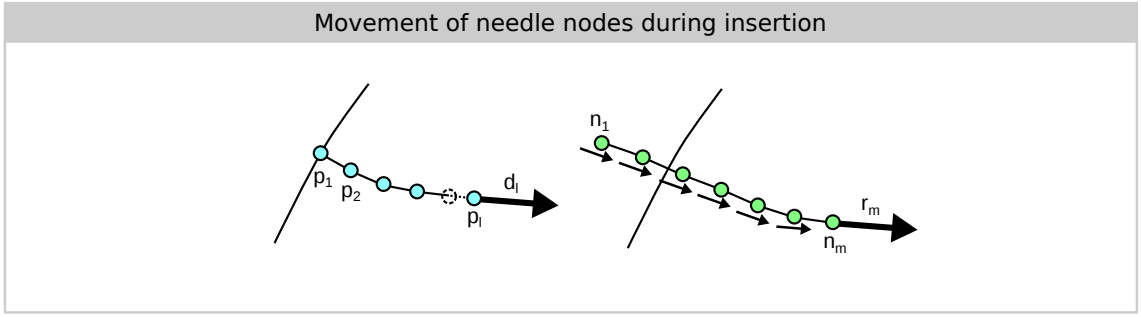

Figure 3.5.: Path Nodes $\mathbf{p}_i$ are placed during insertion and constitute the insertion channel. Here, it is illustrated that only the path tip node is moving dynamically for the path node set and it is moving in direction of the needle tip node.

Alg. 3.4 summarizes the update procedure `updatePathNodes(...)` of the node positions P, for which the execution path is dependent of the current number of path nodes. The algorithm includes the update of the tip node by calculation of a restricted movement. (`compPTipMov(...)`, Alg. 3.5). This enables the unrestricted backwards movement of the tip node in direction towards the preceding node $\mathbf{p}_{l-1}$ and also the restricted forward movement in tangential direction $\mathbf{d}_l$. The restriction limits the insertion depth of the needle to the overall length $\bar{l}$. Also, it is simulating a cutting force threshold that has to be exceeded to permit advancing of the needle into the tissue. The force is modeled by a non-linear quadratic spring with spring parameters $f_{k_1}(\mathbf{p}_l)$ and $f_{k_2}(\mathbf{p}_l)$ and the force has to exceed the threshold $f_{\text{cutthresh}}(\mathbf{p}_l)$. These material properties are functions that vary and are defined for each element of the image domain Ω_I and can take into account the image I and the label image L. The method for evaluation of the material property functions based on the partially segmented data will be given in chapter 6.1.

Based on the new position of the needle path tip node and the resulting distance $\Delta l = (\mathbf{p}_l - \mathbf{p}_{l-1}) \cdot \mathbf{d}_{l-1}$ one of the following is performed. A new node is inserted between $\mathbf{p}_l$ and $\mathbf{p}_{l-1}$ in case the distance is larger than 1.5 times the fixed needle spacing Δp. In case $\Delta l < 0.5 \cdot \Delta p$ and the needle path consisting of more than two nodes, the tip node is approaching the node after the tip, resulting in the removal of the latter from the set. If only two node are remaining on the path, and the needle is withdrawn further, it is possible that the distance Δl is becoming negative, which indicates that the needle was withdrawn from the patient's body, which is processed by emptying the path node set.

3.5.2. Simulation of Needle and Tissue Coupling

The needle model is implemented using the Bullet Physics library [Cou14]. In the physics simulation, each needle node $\mathbf{n}_j$ is synchronized with a rigid body object and adjacent nodes are connected by ball-and-socket joints and a rotational spring. Additionally, to each path node and corresponding rigid body inside the tissue, a virtual spring is attached (Fig. 3.6, left side). These springs have a stiffness as defined by the material stiffness $f_k(\mathbf{p}_i)$ at the position of the needle nodes. The purpose of these springs is to model the force acting on the needle nodes perpendicular to the needle. This lead to the design decision of neglecting the forces acting on the rigid body elements of the needle in tangential direction by projecting the force on the plane perpendicular to the needle tangent, with exception of the tip node.

Furthermore, the base of the Bullet needle model is connected by a spring to a position $\mathbf{v}$, which induces bending to the needle in case the user angulates it. The bending of the needle will influence the orientation of the needle tip node and thus will affect the insertion paht.

Algorithm 3.4 New needle path node positions P^+ are computed based on the movement of the haptic device.

input: $\mathbf{x} \leftarrow$ haptic device position
input: $P \leftarrow$ set of path nodes from prev. step
input: $\mathbf{d}_{1...l} \leftarrow$ needle tangents at path nodes $1...l$
output: new set of path nodes P^+
if $|P| = 0$ **then** // if needle is not yet in tissue
  if $l(\mathbf{x}) \neq \emptyset$ **then** // and needle tip touches skin surface
    $P^+ \leftarrow \{\mathbf{x}\}$ // add first path node on skin surface
  end if
else if $|P| = 1$ **then** // if needle is on tissue surface
  $\Delta d \leftarrow (\mathbf{x} - \mathbf{p}_l) \cdot \mathbf{d}_l$
  if $\Delta d > 0$ **then** // if needle is inserting
    $P^+ \leftarrow \{\mathbf{p}_1\} \cup \{(\mathbf{p}_1 + \mathbf{d}_1 \cdot \Delta d)\}$ // add a second path node
  else if $\Delta d < 0$ **then** // or if needle is retracting
    $P^+ \leftarrow \emptyset$ // needle leaves body
  end if
else // needle is moving in the tissue
  $\mathbf{p}_l^+ \leftarrow \mathbf{p}_l + \texttt{compPTipMov}(...)$
  $\Delta l \leftarrow (\mathbf{p}_l^+ - \mathbf{p}_{l-1}) \cdot \mathbf{d}_{l-1}$
  if $\Delta l > 1.5 \cdot \Delta p$ **then** // if distance between tip and next path node is too large
    $P^+ \leftarrow P \cup \{(\mathbf{p}_{l-1} + \Delta p \cdot \mathbf{d}_{l-1})\}$ // insert new node behind tip
  else if $\Delta l < 0.5 \cdot \Delta p \wedge |P| > 2$ **then** // of if distance is too small
    $P^+ \leftarrow \{\mathbf{p}_1, ..., \mathbf{p}_{l-2}\} \cup \{\mathbf{p}_l^+\}$ // remove node behind tip
  else if $\Delta l < 0$ **then** // if only two nodes, distance can be decreased until
    $P^+ \leftarrow \emptyset$ // the needle is withdrawn fully
  else // in all other cases
    $P^+ \leftarrow P$ // no changes to needle set
  end if
end if

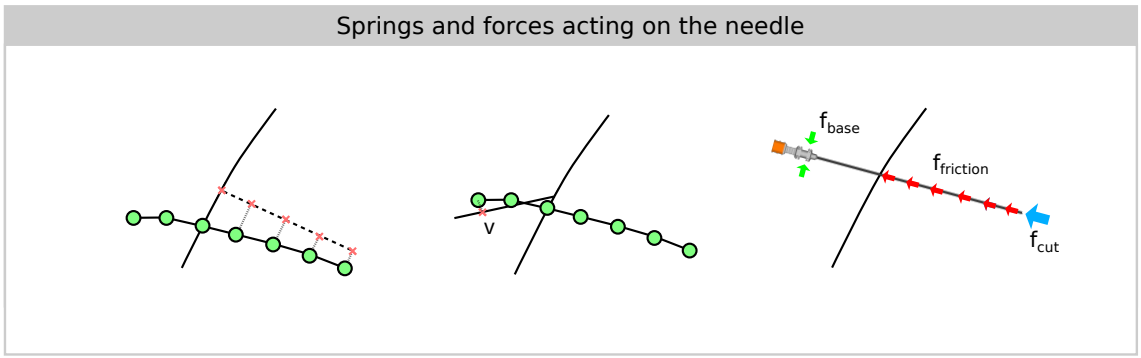

Figure 3.6.: Left: The nodes of the discretized needle that are inside tissue are connected to it by virtual springs. Middle: The base of the needle is connected to $\mathbf{v}$. Right: Different forces acting on the needle are simulated and accumulated to compute the force given to the user.

Algorithm 3.5 Computation of path tip node movement `compPTipMov(...)`. The movement of the tip node is restricted in its tangential direction $\mathbf{d}_l$ and also by the needle length $\bar{l}$ and simulation of cutting resistance.

input: $\mathbf{x} \leftarrow$ haptic device position
input: $\mathbf{q}_z \leftarrow$ haptic device direction
input: $\mathbf{d}_l \leftarrow$ needle path tip node direction
input: $P \leftarrow$ set of path nodes from prev. step
output: movement of path tip node $\Delta\mathbf{p}_l$
$\Delta d \leftarrow (\mathbf{x} - \mathbf{p}_l) \cdot \mathbf{d}_l$
if $\Delta d \geq 0$ **then** // if needle moves forward
  $f \leftarrow f_{k_2}(\mathbf{p}_l) \cdot \Delta d^2 + f_{k_1}(\mathbf{p}_l) \cdot \Delta d$ // compute cutting force magnitude
  if $f > f_{\text{cutthresh}}(\mathbf{p}_l) \wedge \sum_{i=2}^{l} \|\mathbf{p}_i - \mathbf{p}_{i-1}\| < \bar{l}$ **then** // if force exceeds threshold and needle is not fully inserted
    if $f_{k_2}(\mathbf{p}_l) = 0$ **then** // if using a linear spring
      $\Delta\mathbf{p}_l \leftarrow \left(\Delta d - \frac{f_{\text{cut}}(\mathbf{p}_l)}{f_{k_1}(\mathbf{p}_l)}\right) \mathbf{d}_l$ // set movement using linear spring
    else // if using a non-linear spring
      $a \leftarrow 0.5 \cdot \frac{f_{k_1}(\mathbf{p}_l)}{f_{k_2}(\mathbf{p}_l)}$
      $b \leftarrow \frac{f_{\text{cut}}(\mathbf{p}_l)}{f_{k_2}(\mathbf{p}_l)}$
      $\Delta\mathbf{p}_l \leftarrow \left(\Delta d + a - \sqrt{a^2 + b}\right) \mathbf{d}_l$ // set movement using non-linear spring
    end if
  else // or if force to small or needle fully inserted
    $\Delta\mathbf{p}_l \leftarrow 0$ // prohibit all movement
  end if
else // unrestricted backward movement
  $\Delta\mathbf{p}_l \leftarrow \Delta d \frac{\mathbf{p}_l - \mathbf{p}_{l-1}}{\|\mathbf{p}_l - \mathbf{p}_{l-1}\|}$ // move towards next node, not in tip direction
end if

The position of $\mathbf{v}$ is based on the first path node position $\mathbf{p}_1$, the haptic device direction $\mathbf{q}_z$ and the length of the part of the needle that still remains outside of the body

$$\mathbf{v} = \mathbf{p}_1 - \left(\bar{l} - \sum_{i=2}^{l} \|\mathbf{p}_i - \mathbf{p}_{i-1}\| \right) \mathbf{q}_z \tag{3.5.1}$$

3.5.3. Force Computation

After having computed new node positions N^+ and P^+, force and torque for display by the haptic device are computed. In case the needle is moved freely without tissue contact yet, no needle forces are computed and the needle nodes are placed along the ray defined by haptic device position $\mathbf{x}$ and direction $\mathbf{q}_z$ (`simpleNNodePlacement`). The force is divided in three parts: A cutting resistance force, a friction force and a base force (see right side of Fig. 3.6). These forces are assumed to act accumulated on the base of the needle, which represents the interface between the user and the needle.

1. The cutting force is defined as the spring force of a quadratic spring with an extension of $\Delta d = (\mathbf{x} - \mathbf{p}_l) \cdot \mathbf{d}_l$, i.e. the projection of the difference between haptic device tip and the last needle path node $\mathbf{p}_l$ onto the cutting direction

$$\mathbf{f}_{\text{cut}} = -\left(f_{k_2}(\mathbf{p}_l) \cdot \Delta d^2 + f_{k_1}(\mathbf{p}_l) \cdot \Delta d \right) \cdot \mathbf{d}_l \tag{3.5.2}$$

   Note that the magnitude of this force is also used in Alg. 3.5 to restrict tip node movement. Only in case the user applies a force large enough, the needle advances into the tissue.
2. Apart from the cutting force, friction along the shaft has to take into account the material at each point along the needle. Here, it is modeled for each node on the insertion path by a simple friction law [HA00], that uses a spring-like force based on an offset and material properties for maximum friction force $f_R(\mathbf{p}_i)$ and stiffness $f_k(\mathbf{p}_i)$. For the friction, it is assumed that only the relative movement in tangential needle direction is relevant, which is approximated by the relative movement $\Delta x = (\mathbf{x} - \mathbf{x}^-) \cdot \mathbf{d}_l$ of the haptic device in needle insertion direction, based on the current haptic device position and position from the previous iteration. The relative offset for each segment then is $\Delta x_i \in \mathbb{R}$ and it is updated in each iteration as

$$\Delta x_i^+ = \begin{cases} \Delta x_i + \Delta x & \text{if } |\Delta x_i + \Delta x| < \frac{f_R(\mathbf{p}_i)}{f_k(\mathbf{p}_i)} \\ \frac{f_R(\mathbf{p}_i)}{f_k(\mathbf{p}_i)} \operatorname{sgn}(\Delta x_i + \Delta x) & \end{cases} \tag{3.5.3}$$

Based on all offsets and the corresponding stiffness values, the total frictional force along the elements of the needle path is

$$f^{+} = \sum_{2 \leq i \leq l} f_k(\mathbf{p}_i) \frac{\|\mathbf{p}_i - \mathbf{p}_{i-1}\|}{\Delta p} \Delta x_i \tag{3.5.4}$$

with $\frac{\|\mathbf{p}_i - \mathbf{p}_{i-1}\|}{\Delta p}$ being a weighting term that is 1 for all segments except the last. Spring based haptic algorithms can exhibit unstable force calculation due to overshooting when using high stiffness values for the springs. For example, if a constant stiffness is assumed, the total stiffness of f increases linearly with the number of path nodes, which can exceed the maximum stiffness of a stable simulation while inserting the needle. This maximum stiffness value $k_{\max}$ is known for the haptic devices and is used to clamp the change of total friction force

$$\Delta f = \max\left(-\|k_{\max}\Delta x\|, \min\left((f^{+} - f), \|k_{\max}\Delta x\|\right)\right) \tag{3.5.5}$$

finally yielding the frictional force as

$$\mathbf{f}_{\text{fric}} = (f + \Delta f)\mathbf{d}_l \tag{3.5.6}$$

3. Finally, a force that keeps the haptic device handle on the insertion vector is computed. It takes the insertion direction at the first path node $\mathbf{d}_1$ as a plane normal and uses the projection of the difference of the needle base $\mathbf{n}_1$ and the haptic device position $\mathbf{x}$

$$\mathbf{f}_{\text{base}} = \kappa \cdot [(\mathbf{n}_1 - \mathbf{x}) - (\mathbf{n}_1 - \mathbf{x}) \cdot \mathbf{d}_1] \tag{3.5.7}$$

with κ being the stiffness coefficient of this spring. This coefficient has to be high enough to facilitate compliance of the haptic device to the virtual needle base while not causing oscillation of the haptic device handle.

Torque computation uses a quaternion based rotational spring. For this aim, a quaternion $\mathbf{q}_{\text{dest}}$ is defined as the quaternion for which the local z-axis is aligned with direction $\mathbf{d}_1$ and the x- and y-axis aligned with the local axis $\mathbf{q}_x$ and $\mathbf{q}_y$ of the haptic device. Aligning it in x- and y-axis with the haptic device will remove twisting torque so the needle is freely rotatable around its shaft. Given $\mathbf{q}_{\text{dest}}$, a difference quaternion $\Delta\mathbf{q} = \mathbf{q}_{\text{dest}}^{-1} \cdot \mathbf{q}$ is computed and converted to Euler-angles $\mathbf{e} \in \mathbb{R}^3$. Based on these, the torque in the space of the haptic device is

$$\mathbf{t} = \mathbf{q} \cdot (k_q \mathbf{e}) \cdot \mathbf{q}^{*} \tag{3.5.8}$$

with a spring coefficient $k_q = -300 - \sum_{i=2}^{l} \|\mathbf{p}_i - \mathbf{p}_{i-1}\| \cdot 50.0$ that takes the insertion depth of the needle into account.

3.6. Palpation and Ultrasound Probing

Palpation and ultrasound probing is common practice for many interventions making it an interesting aspect to be featured in a visuo-haptic simulation. For example, for needle insertion into the femoral artery with the aim of regional anesthesia or preparation of the Seldinger technique, the artery has to be located first. This, for example, is included in the simulation frameworks presented in [CJGC11] and [UIK12]. In the former, a sophisticated hardware setup is used to give the user the possibility to palpate a tissue phantom with his own fingers in a mixed-reality simulation environment. In contrast to this, in [UIK12] a Phantom Omni haptic device is used. Algorithmically, the haptic device steers a single interaction point, which uses surface models for handling contacts. Another approach for virtual palpation is presented in [YS13]. There, organs are represented by mathematical distance functions, which are used to render haptic force output for palpation. Haptic interaction with a virtual patient using a simulated ultrasound probe is also included in the framework of [VVH+09, BBG+09]. In general, interaction with volumetric data through the use of haptic devices is presented for example in [LYG02, PCY07] and using distance maps in [BG00].

Insertion of a needle in between the ribs for liver puncture includes the palpation of the intercostal spaces to find a suitable insertion site. Furthermore, ultrasound probing is used to guide the needle into the target structure. For the simulation of virtual palpation the methods presented here include a newly developed algorithm [FMH13b, FMH14] that is based on distance maps of segmentations of skin and bone. It uses multiple contact points distributed on the virtual palpating finger or ultrasound probe and can make use of the 6 degrees of freedom of force output of the Phantom Premium haptic device. The distance maps are computed in a preprocessing step based on the available CT image data and only require the determination of reasonable thresholds for bone and skin surface classification, making it suitable for patient-specific simulation. These thresholds can be set by the scenario designer or estimated using the gray value distribution of a reference patient.

More precisely, in the preprocessing step, Euclidean distance maps are computed from the image data as illustrated in Fig. 3.7. The image data is first thresholded by segmentation thresholds $t_{\text{air}}^{\text{skin}}$ and $t_{\text{fat}}^{\text{bone}}$ to obtain segmentation masks. Afterwards, region growing is applied to the skin segmentation mask to remove air cavities from the segmentation. In the same fashion as for the image data, the distance maps are defined on the

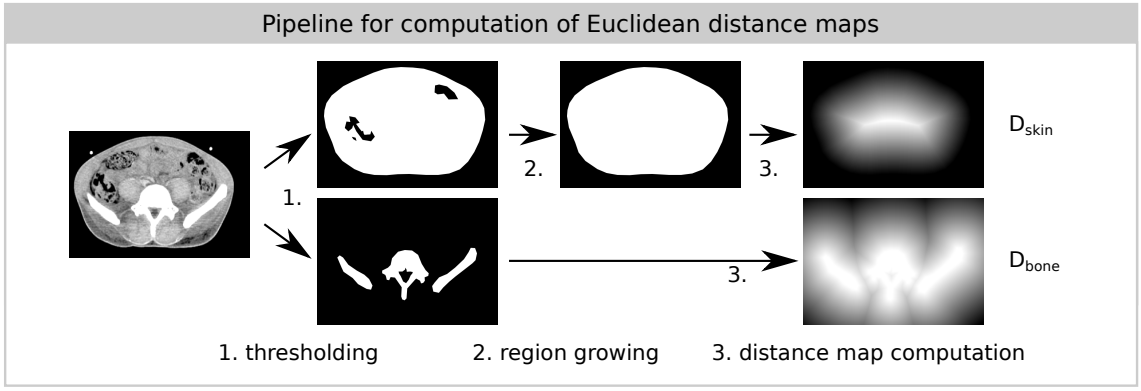

Figure 3.7.: Creation of Euclidean distance maps for simulation of virtual palpation and ultrasound probing.

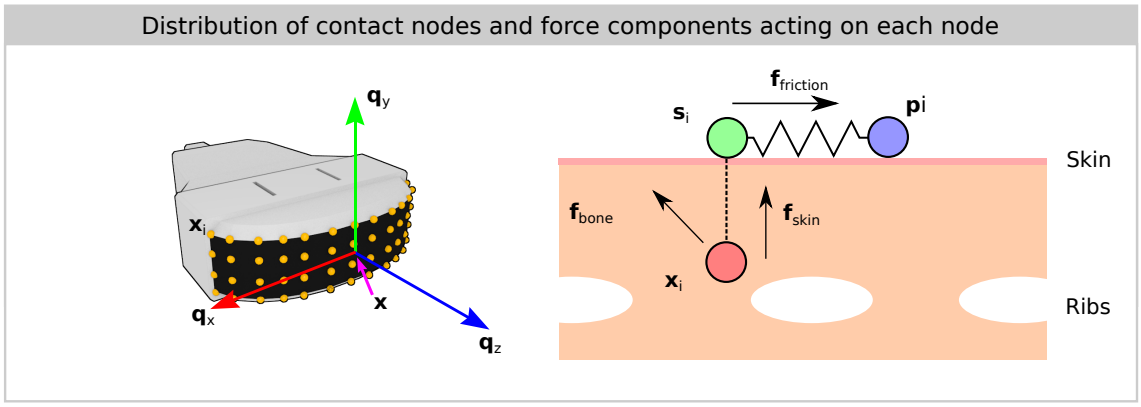

Figure 3.8.: Left: Distribution of contact nodes on a virtual ultrasound probe in the local frame of the haptic device. Right: Force components for haptic rendering of palpation and ultrasound probing for a single contact node.

image grid, but using trilinear interpolation they can be represented as distance functions $D_{\{\text{skin,bone}\}} : \Omega_I \rightarrow \mathbb{R}$ on the continuous image domain. Each of D_{skin} and D_{bone} yield the signed distance for a given point to the closest element that represent the surface of the patient's skin or bone respectively. A positive sign indicates that the given point is inside the structure, whereas a negative sign indicates that the point is outside the structure. Calculation of the gradient $\nabla D_{\{\text{skin,bone}\}}(\mathbf{x}_i)$ gives the direction of the shortest paths towards each of the surfaces.

The algorithms use several contact points with proxies connected to them. Without loss of generality, this is first explained for a single point on the surface of the interacting finger or ultrasound probe. This point has a static location $\mathbf{x}_i^0$ in the reference frame of the haptic device resp. ultrasound probe or palpating finger given by the rotation matrix $\mathbf{Q} = (\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z)$. In the world space, the location is given by the vector $\mathbf{x}_i = \mathbf{x} + \mathbf{Q} \cdot \mathbf{x}_i^0$, see left side of Fig. 3.8.

As shown also in Fig. 3.8, the force is subdivided into three components, namely two

penalty forces $\mathbf{f}_{\text{skin}}$ and $\mathbf{f}_{\text{bone}}$ and a proxy spring force $\mathbf{f}_{\text{friction}}$. The direction of the penalty forces is based on the gradient of the distance maps. For the skin force component, the direction is

$$\mathbf{v}_i^s = -\frac{\nabla D_{\text{skin}}(\mathbf{x}_i)}{\|\nabla D_{\text{skin}}(\mathbf{x}_i)\|} \tag{3.6.1}$$

Based on this direction and a force function $f_{\text{skin}} : \mathbb{R} \to \mathbb{R}$, the force then is defined as

$$\mathbf{f}_{\text{skin}} = f_{\text{skin}}(D_{\text{skin}}(\mathbf{x}_i)) \cdot \mathbf{v}_i^s \tag{3.6.2}$$

The force function relates the indentation of the skin to a force magnitude. For the case of femoral artery palpation, such a force function is given in [CJGC11]. Since the measurements have been performed for a large region of nearly homogenous soft tissue, it can be assumed to be applicable for other regions as well. Using a curve fitting algorithm for the data from [CJGC11] yielded the following cubic function

$$f_{\text{skin}}(x) = \begin{cases} 1.78 \cdot 10^{-2}x^3 + 3.82 \cdot 10^{-3}x^2 + 2.28 \cdot 10^{-4}x & \text{if } x > 0 \\ 0 & \text{else} \end{cases} \tag{3.6.3}$$

In theory, this approach could suffer from "pop-thru" effects in case the gradient of the distance map flips its direction while the user is increasing the indentation of the skin. Since the virtual patient is large in comparison to the possible indentation, this is not a problem in reality.

The bone penalty force is calculated similar but incorporates a heuristic to adjust the direction of the force in case the contact point is past the intercostal spaces, which would cause the already mentioned "pop-thru". Again, the base force direction is calculated as

$$\mathbf{v}_i^b = -\frac{\nabla D_{\text{bone}}(\mathbf{x}_i)}{\|\nabla D_{\text{bone}}(\mathbf{x}_i)\|} \tag{3.6.4}$$

but it is corrected towards the surface force direction by

$$\mathbf{d}_{\text{bone}} = \begin{cases} \mathbf{v}_i^b & \text{if } \mathbf{v}_i^b \mathbf{v}_i^s > 0 \\ \mathbf{v}_i^b - 4 \cdot \mathbf{v}_i^s(\mathbf{v}_i^b \mathbf{v}_i^s) & \text{otherwise} \end{cases} \tag{3.6.5}$$

in case the deviation of $\mathbf{v}_i^b$ is larger than 90° towards $\mathbf{v}_i^s$.

As a force function for this component, the following function has been used

$$f_{\text{bone}}(x) = \begin{cases} 20 \cdot \left(\frac{x}{5}\right)^{-3} & \text{if } x > 0 \\ 0 & \text{else} \end{cases} \tag{3.6.6}$$

based on experiences with the system. Overall, this gives the total bone force

$$\mathbf{f}_{\text{bone}} = f_{\text{bone}}(-D_{\text{bone}}(\mathbf{x}_i)) \frac{\mathbf{d}_{\text{bone}}}{\|\mathbf{d}_{\text{bone}}\|} \tag{3.6.7}$$

This force increases the closer the location $\mathbf{x}_i$ is to the bone surface. In comparison, the force representing the skin surface is increasing for locations that have already passed the skin surface.

Apart from these penalty forces, the palpation and ultrasound probing haptic algorithm includes a proxy-based component. The idea of a proxy-based haptic algorithm is that the haptic device is connected to a proxy by a virtual spring, and the rendered object interacts with the proxy. This rendered object can be represented by surface models [RKK97] or volume models [LYG02]. Here, a surface position $\mathbf{s}_i = \mathbf{x}_i - D_{\text{surf}}(x) \cdot \mathbf{v}_i^s$ corresponding to the contact node at position $\mathbf{x}_i$ (see Fig. 3.8) is connected to a virtual proxy $\mathbf{p}_i$. The behavior of the proxy is then defined as follows: In each iteration, the movement of the proxy is restricted by using the friction law of [HA00]

$$\mathbf{p}_i^+ = \begin{cases} \mathbf{s}_i^+ - \frac{\mathbf{d}}{\|\mathbf{d}\|} z_{\max} & \text{if } \alpha(\|\mathbf{d}\|)\, \|\mathbf{d}\| > 1 \\ \mathbf{p}_i + \left|\mathbf{s}_i^+ - \mathbf{s}_i\right| \alpha(\|\mathbf{d}\|)(\mathbf{d}) & \text{otherwise} \end{cases} \tag{3.6.8}$$

with $\mathbf{d} = \mathbf{s}_i^+ - \mathbf{p}_i$ and $\mathbf{s}_i^+$ being the newly computed surface position. Furthermore, as suggested in [HA00], the term $\alpha(z) = \frac{1}{z_{\max}} \frac{z^8}{z_{\text{stick}}^8 + z^8}$ is used. To model friction based on the force acting on the contact node, the values of $z_{\max}$ and z_{stick} are adjusted:

$$\mathbf{f}_{\text{n}} = (\mathbf{f}_{\text{skin}} + \mathbf{f}_{\text{bone}}) \tag{3.6.9}$$

$$z_{\max} = \begin{cases} c_{\text{fric}}\, \mathbf{f}_{\text{n}} \cdot \mathbf{v}_i^s + 1 & \text{if } \mathbf{f}_{\text{n}} \cdot \mathbf{v}_i^s < 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.6.10}$$

$$z_{\text{stick}} = z_{\max} - 1 \tag{3.6.11}$$

Since proxy based rendering uses a spring, Hooke's law then can be used to calculate the resulting spring force $\mathbf{f}_{\text{slide}} = k_{\text{fric}}\, (\mathbf{p}_i - \mathbf{s}_i)$. The friction model uses two constants c_{fric} and k_{fric} and can be adjusted to reflect the different behavior of palpating fingers and ultrasound

probing with a lubricant that reduces friction. Values of $c_{\text{fric}} = 2.0$ and $k_{\text{fric}} = 0.4$ can be used for palpation with a finger. The reduced friction caused by the lubricant is reflected by values of $c_{\text{fric}} = 0.4$ and $k_{\text{fric}} = 0.3$ [FMH14]. Also, higher values of c_{fric} cause characteristic tissue dragging, i.e. the contact nodes' proxies have a higher stick threshold, increasing the maximal distance between proxy and contact node.

The torque action on the haptic device is given by the cross product of the lever arm and the total force $\mathbf{f}_i = \mathbf{f}_{\text{skin}} + \mathbf{f}_{\text{slide}} + \mathbf{f}_{\text{bone}}$ acting on the contact node

$$\mathbf{t}_i = (\mathbf{x}_i - \mathbf{x}) \times \mathbf{f}_i \tag{3.6.12}$$

Having computed the force and torque for a single contact position, now the final force and torque can be computed given n contact nodes that are in contact with the virtual body as

$$\mathbf{f}_{\text{device}} = \frac{1}{n} \sum_{i=0}^{n} \mathbf{f}_i \tag{3.6.13}$$

$$\mathbf{t}_{\text{device}} = \frac{1}{n} \sum_{i=0}^{n} \mathbf{t}_i \tag{3.6.14}$$

Averaging of forces and torques is done to make these independent of the size of the contact area and density of contact node distribution.

3.7. Experiments & Results

To demonstrate the realism of the presented methods, screenshots of the ray casting based volume rendering, the X-ray simulation and ultrasound simulation have been taken. The patient CT image data used for the experiments has a resolution of $256 \times 256 \times 234$ voxels and a voxel spacing of 1.55 mm $\times$ 1.55 mm $\times$ 2.00 mm.

Also, to confirm the applicability of implemented methods and their improvements, they have to be analyzed for their run-time behavior on a Nvidia Geforce GTX680 graphics card. Especially the ray casting based volume rendering is demanding in terms of computational resources, for which an in-depth description follows.

3.7.1. Ray Casting

To analyze the run-time behavior of the volume rendering, a standard scene setup has been defined, which corresponds to the setup shown in Fig. 3.9. All following experiments use the camera pose defined for this standard scene to ensure reproducibility and comparability. Also, a scripted needle insertion into the bile ducts is performed during the measurements,

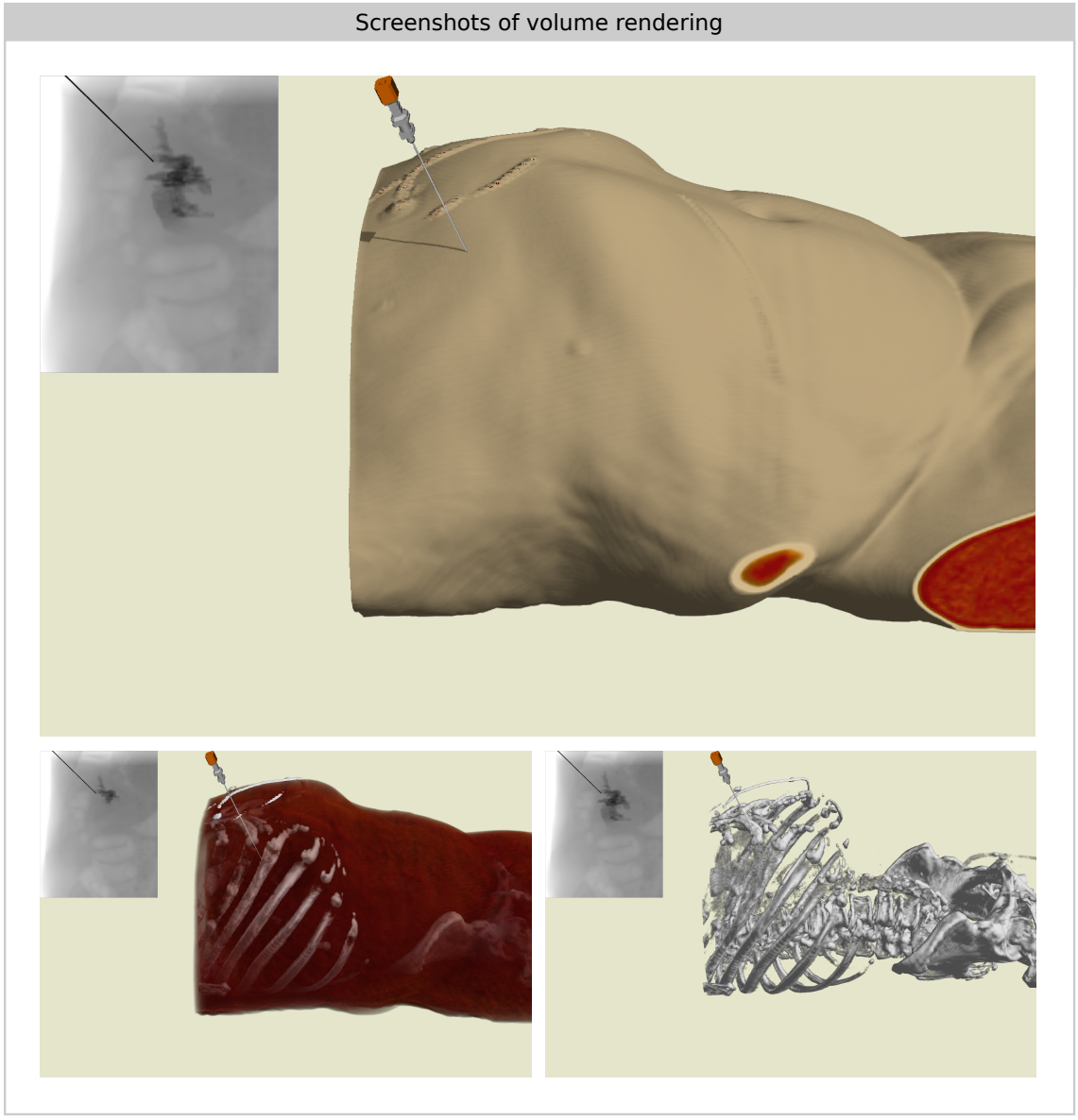

Figure 3.9.: Results of volume rendering using different combination of transfer functions. The results show the setup used for time measurments. Upper: $\alpha_{\text{skin}} = \alpha_{\text{soft}} = \alpha_{\text{bone}} = 1.0$ Lower left: $\alpha_{\text{skin}} = 0$, $\alpha_{\text{soft}} = 0.03$ and $\alpha_{\text{bone}} = 1.0$ Lower right: $\alpha_{\text{skin}} = \alpha_{\text{soft}} = 0$ and $\alpha_{\text{bone}} = 1.0$

with a total duration of 10 seconds. For each experiment, the processing time of the volume rendering CUDA kernels is measured during this insertion sequence. Thus, the processing time directly influences the sampling rate. For volume rendering, three trapezoid shaped transfer functions for rendering voxels with gray values representing the skin surface, soft tissue and bone resp. have been defined. These are combined using Eq. 3.2.2 with weights $\alpha_{\text{skin}} = \alpha_{\text{soft}} = \alpha_{\text{bone}} = 1.0$ if not stated otherwise. As a default, the resolution of the rendered image is set to 1225×1016 pixels. Table 3.1 shows the measured times for which a detailed description follows.

First, the performance of the kernel when using no improvements were measured (no. 1). For the ray casting, two major speed improvements were implement: Ray-box intersection and empty space leaping. In a second experiment, the unoptimized volume rendering is compared to ray-box intersection (no. 2) and empty space leaping (no. 3-6) for the normal 1D transfer function based rendering. The empty space leaping is performed for a volumetric mask consisting of 16, 32, 64 and 128 elements in each spatial direction. Decreasing the resolution of the rendering viewport to 612×508 and 306×254 pixels was measured in experiments no. 7 and 8. As expected, the transfer function has a major influence on the processing time: In experiment no. 9, a combination of transfer functions with weights $\alpha_{\text{skin}} = 0$, $\alpha_{\text{soft}} = 0.03$ and $\alpha_{\text{bone}} = 1.0$ was chosen. The visual result is also shown in Fig. 3.9. In the same fashion, no. 10 uses the weights $\alpha_{\text{skin}} = \alpha_{\text{soft}} = 0$ and $\alpha_{\text{bone}} = 1.0$, see also Fig. 3.9. For these two experiments, ray-box intersection and empty space leaping was also activated.

To confirm the effects of different block and thread layouts (see section 2.2), blocks with different number of threads have been tested (8×8, 16×16, 32×32) in experiments no. 11-13. For each of this configurations, the Nvidia profiling tool Nsight states a theoretical occupancy of 50% based on a number of 57 used registers per thread.

Furthermore, it is worth to analyze the contribution of single elements to the overall run-time. This has been done for the shadowing term (no. 14) and the Blinn-Phong shading with costly gradient computation (no. 15) by setting the result of the respective function values to a constant value. Experiment no. 16 additionally uses a different transfer function. Finally, it is worth to analyze the effect of thread divergence on the total run-time. Analyzing this is tricky, so thread divergence is removed by using the same ray configuration for each thread in a block (no. 17-18). This results in a "pixelated" render image (each element of a block computes the same result) but gives a good hint on the effect of thread-divergence.

As can be seen, the transfer function used has a major influence on the number of samples that have to be taken until the color accumulator is fully saturated and thus also a severe

Table 3.1.: Processing times in *ms* for the ray casting implemented in CUDA using a resolution of 1225 × 1016 pixel (with exception of experiment 7 and 8). For each experiment, mean processing time, standard deviation and number of samples are given.

no.		comment	mean	stddev	N
1	Unimproved		542.20	1.60	11
2	+Ray-box-intersect.		63.12	0.98	79
3	+Empty-space-leap.	16^3 elements.	28.04	0.34	110
4		32^3 elements	19.27	0.31	123
5		64^3 elements	14.98	0.21	128
6		128^3 elements	14.08	0.19	122
7	Lower resolution	$\frac{1}{2}$ resolution	4.55	0.23	160
8		$\frac{1}{4}$ resolution	2.06	0.15	188
9	Different tfx	$\alpha_{\text{skin,soft,bone}} = (0.0, 0.03, 1.0)$	113.32	0.54	67
10		$\alpha_{\text{skin,soft,bone}} = (0.0, 0.0, 1.0)$	54.28	0.36	110
11	Blocksize	8×8	14.21	0.35	195
12		16×16	14.69	0.26	192
13		32×32	19.74	0.34	181
14	w/o Shadowing	$f_{\text{shadow}} = 1.0$	13.56	0.31	196
15	w/o Shading	$f_{\text{shade}} = 1.0$	13.45	0.26	192
16		$\alpha_{\text{skin,soft,bone}} = (0.0, 0.03, 1.0)$	87.90	0.75	80
17	-Thread-divergence		11.15	0.42	195
18		$\alpha_{\text{skin,soft,bone}} = (0.0, 0.03, 1.0)$	98.21	1.04	77

effect on the rendering times. This is also valid for the experiments without shading and with removed thread divergence: Overall, the measured times are lower for comparable measurements (no. 4 vs no. 15+17 and no. 9 vs no. 16+18).

Apart from the simple 1D transfer function volume rendering, the clipping mode is demonstrated in Fig. 3.10, including the target-and-risk-structure overlay, label color overlay and windowing mode.

3.7.2. X-Ray and Ultrasound Simulation

Output of the X-ray simulation is included in Fig. 3.9. The processing speed of the X-ray simulation component has been tested for two different resolutions of the output image (256 × 256 pixel, 512 × 512 pixel) and a diffusion grid with 64^3 elements. For the lower resolution, two additional timing tests have been performed. One without rendering of contrast agent, i.e. setting $\mu_c = 0$, and one with an increased size of 128^3 elements of the diffusion grid. The results are given in Tab. 3.2.

The performance of the contrast agent diffusion process was tested separately using four different resolutions of 64^3, 96^3, 128^3 and 160^3 elements resp. with results in Tab. 3.3.

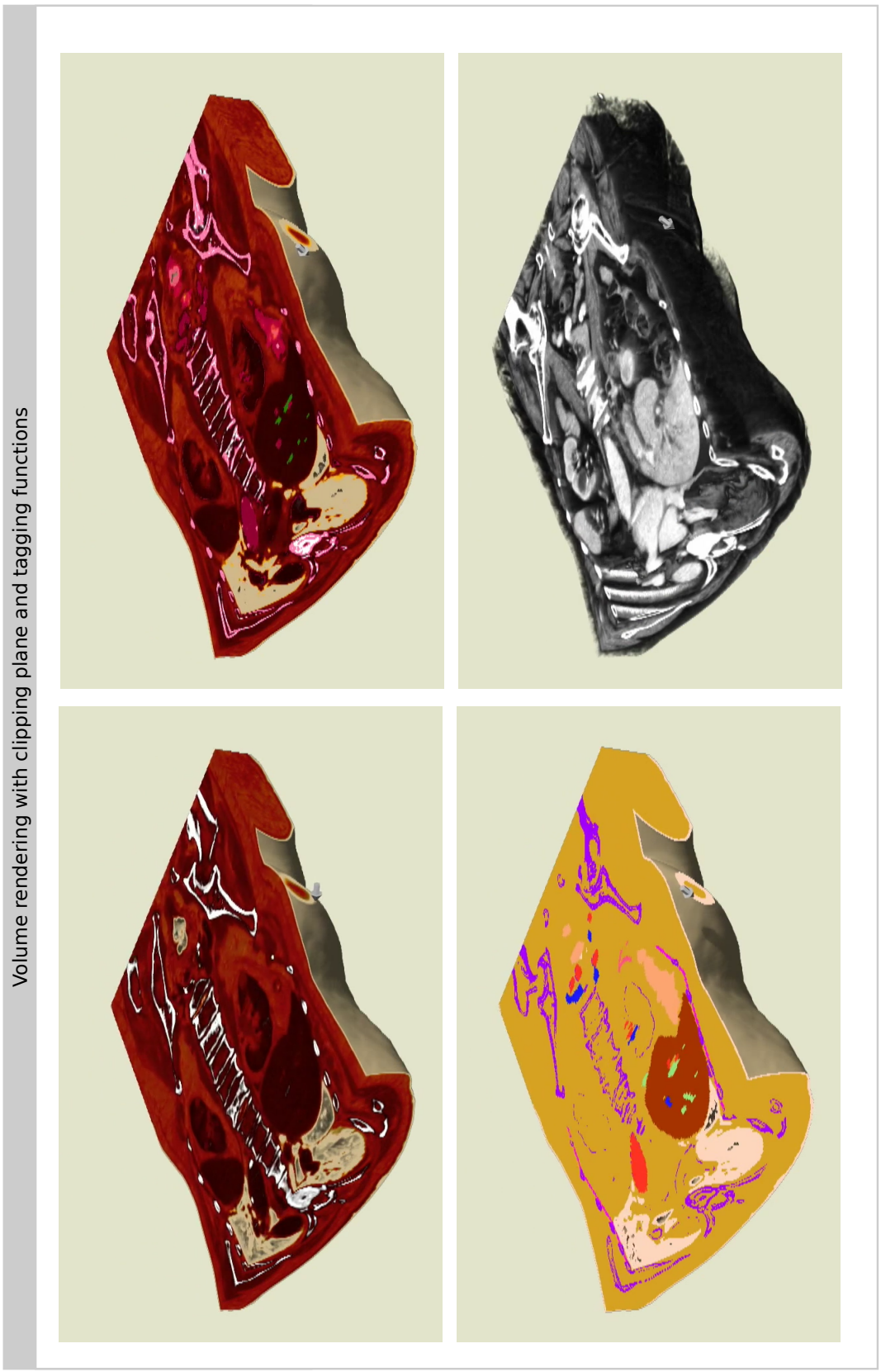

Figure 3.10.: Different rendering modes for clipped volume rendering.

Table 3.2.: Processing times in *ms* for the X-ray simulation with mean, standard deviation and number of samples.

no.		resolution	mean	stddev	N
1	Normal	256×256	12.83	0.10	130
2		512×512	48.92	0.13	88
3	w/o contrast agent	256×256	13.56	0.12	130
4	with increased desity field	256×256	12.83	0.11	130

Table 3.3.: Processing times in *ms* for the contrast agent propagation simulation with mean, standard deviation and number of samples for different sizes of the grid.

no.	resolution	mean	stddev	N
1	64	0.33	0.06	110
2	96	0.60	0.05	110
3	128	1.00	0.06	105
4	160	1.55	0.07	90

Fig. 3.11 contains a screenshot of a simulated ultrasound image with highlighting of the target structure bile ducts.

3.7.3. Needle Algorithm

In contrast to algorithms that produce visual results, i.e. images, the output of haptic algorithms can only be presented by force plots. Fig. 3.12 shows such a plot for the presented needle algorithms during a scripted sequence of bile duct puncture. Instead of using a physical haptic device, a software implementation is used that steers the haptic device position $\mathbf{x}$ along a predefined trajectory starting outside of the patient, into the target structure bile duct and back to the starting point over a period of 20 s. The force plot shows a distinct puncture event of the intercostal fascia at ca. 7 s and puncture of the bile ducts at ca. 12 s. By splitting the force into the components cutting force $\mathbf{f}_{\text{cut}}$ and friction force $\mathbf{f}_{\text{fric}}$, it can be shown that the algorithm works as expected: Cutting force depends on the material at the needle tip and is zero when retracting the needle, the friction force increases linearly with insertion depth and changes its sign when retracted.

Additionally, Fig. 3.13 shows the capability of the algorithm to puncture curved paths. A slight bend of the needle is visible in the upper right part showing the needle nodes. In the lower right part, the path nodes are shown which are placed in the tissue on a curved insertion path. This was caused by angulation of the needle during insertion.

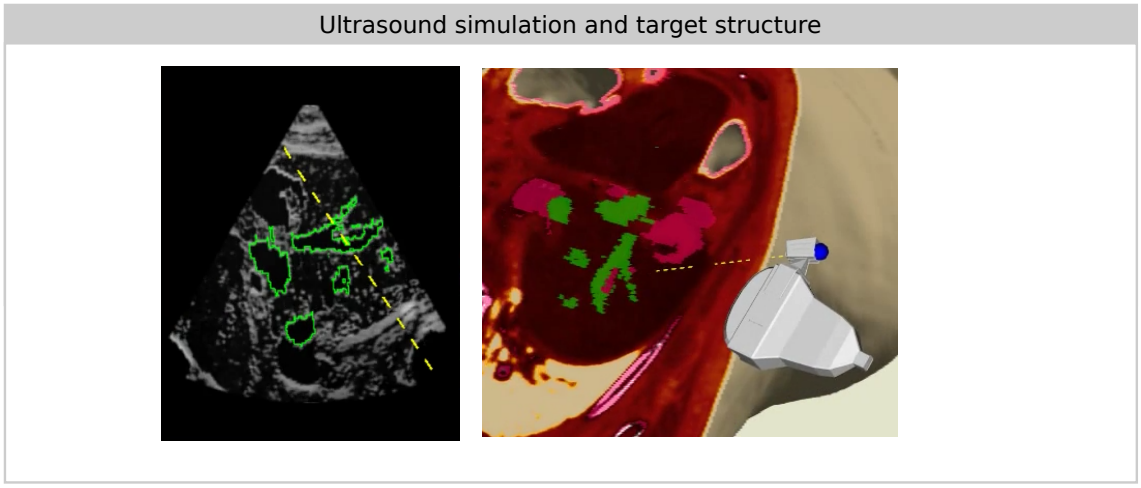

Figure 3.11.: Left: Screenshot of the results of the ultrasound simulation with outlining of the target structure bile ducts (green) and needle path when using the needle guide. Right: Screenshot of simulated scene with ultrasound probe with attached needle guide. The clipping plane is fitted to the orientation of the ultrasound probe. Notice the correspondence of the target structure in both images.

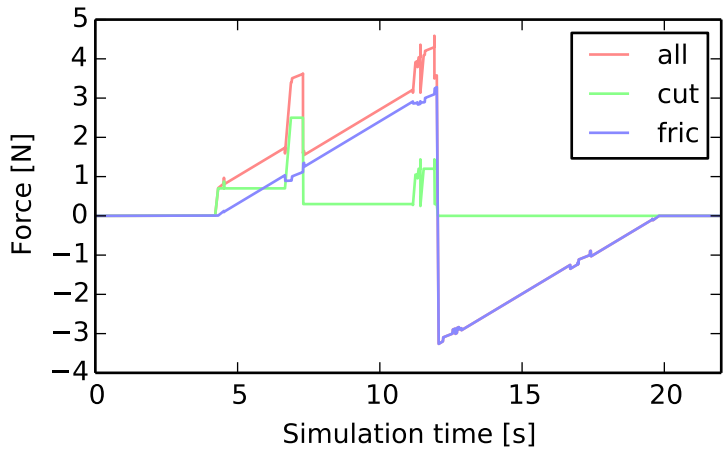

Figure 3.12.: Force plot for a scripted needle insertion into the bile ducts. The force total force in insertion direction $\mathbf{f}$ is divided into the components of cutting force $\mathbf{f}_{\text{cut}}$ and friction force $\mathbf{f}_{\text{fric}}$. The insertion consists of skin puncture (ca, 4.5 s), fascia puncture (ca. 7 s), bile duct puncture (ca. 12 s) and needle retraction.

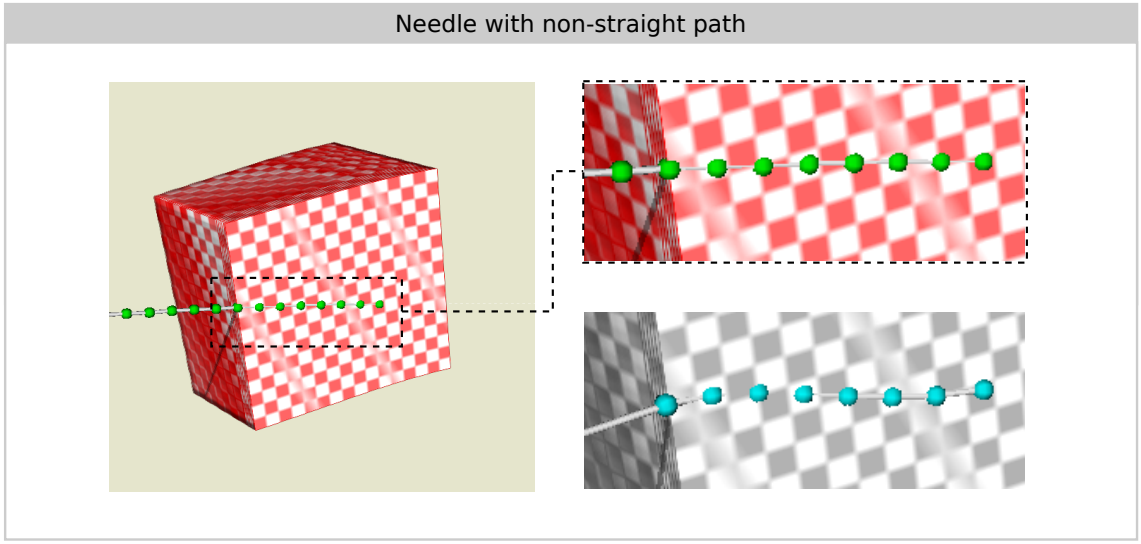

Figure 3.13.: The needle was inserted perpendicular to the surface of a virtual tissue phantom and has been angulated after a few centimeters. In the upper right, the needle nodes N are shown (green) and in the lower right, the needle path nodes P are shown (cyan).

3.7.4. Palpation & US-probing Algorithm

The major purpose of simulated ultrasound probing and palpation is to find a suitable insertion site for the needle. For PTCD, the needle has to be inserted between the ribs, which makes the localization of the intercostal spaces by palpation a feature that has to be part of the simulation. To demonstrate that the presented method is capable of simulating these, the following experiment has been performed: For palpation, the virtual finger was placed on the skin surface of the virtual patient at the chest and was moved in cranial-caudal direction by manual steering of the haptic device by a user. To press the virtual finger onto the patient's skin surface in a controlled manner, the haptic device was programmed to output a constant force of 1 N in distal-proximal direction, i.e. in this case in direction of the y-axis of the world space. This way, it is simulated that the user of the haptic devices presses the virtual finger with a constant force onto the skin, which cannot be performed by a human user. Otherwise, this would have to be performed by another device capable of generating a constant force. It is expected that by applying a constant force, the finger will be able to further indent the skin surface between the rips and less when directly palpating on the rips. In Fig. 3.14 (left) a resulting experimental curve is given. Distinct peaks can be recognized that represent the expected behavior on and between the ribs. The difference between adjacent peaks is in the range of around 50 mm, which is easily recognizable by an observer steering the haptic device. The rightmost part with a high indentation, i.e., high y-values, reflects

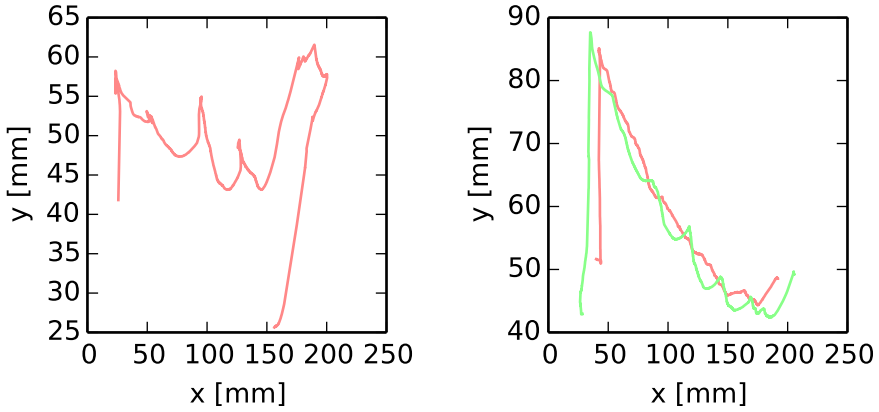

Figure 3.14.: Plots of position of the virtual finger (left) and ultrasound probe (right) in cranial-caudal (x-axis) and distal-proximal (y-axis) directions during a measurement sequence (time is omitted). For probing with the virtual ultrasound device, two curves with different orientations of the probe are considered: The green curve with visible indentation events represents parallel orientation of the probe to the intercostal spaces.

the palpation of soft tissue after the last rib has been passed. In this area, the resistance to indentation caused by the ribs is not present, and thus the soft tissue yields.

A similar experiment has been performed for ultrasound probing. In contrast to palpation, the ultrasound probe uses a pattern of contact nodes that reflect its typical shape (Fig. 3.8). This will have an influence onto the resulting force feedback when the user tries to align the probe with the intercostal spaces. In case that the probe is not aligned, the intercostal space should not be easily palpable or distinguishable. If probe and intercostal spaces are parallel to each other, the probe can 'sink' into the spaces as the finger does. This behavior can be measured very well and is shown for two device trajectories with and without alignment also in Fig. 3.14.

3.8. Discussion

In this chapter, it has been shown that direct rendering of medical image data is valid approach to visuo-haptic rendering. As intended, the methods presented here do not rely on polygonal surface meshes. For visualization and haptic rendering, a labeling function is needed that can either rely on a full segmentation or partial segmentations and heuristics. Nonetheless, the methods are independent of the actual implementation of the function, which will be detailed later in chapter 6. The algorithm for virtual palpation and ultrasound probing haptics use distance maps, which need threshold based segmentations. These only

need minimal user effort when setting these manually and can be created quickly. Also, the high gradient at skin surface and between soft tissue and bones reduce the importance of setting this value with high precision.

Concerning the real-time capabilities, it has been experimentally validated that visual rendering using ray casting is capable of producing realistic and functional visualization within reasonable time frames. Also the choice of transfer function was identified as an important source of influence on the rendering time. It furthermore has been validated that additional X-ray simulation is also possible within a few milliseconds. In [SMFH14], the rendering times for ultrasound simulation are measured to be around 8 ms. To provide interactive simulation, the total processing time of a single frame should not exceed 100 ms (10 Hz). Summing the processing times of the three main visualization components normal rendering, X-ray rendering and ultrasound simulation, a total minimal time frame of around 35 ms is possible depending on the choice of transfer function. This corresponds to a frame rate of around 28.6 Hz, which is well in the range of real-time requirements. Validiation of sufficiently high rendering rates of the haptic algorithms can be found in [FMH14, FMSH16] and [FWMH15].

Force and motion plots have been given for the needle insertion and palpation algorithms to support the claim of plausible haptic rendering. Also, the user study published in [FMSH16] includes positive reception of the direct visuo-haptic rendering components. The validated good performance and applicability of these components forms the basis for further refinements in the next chapters.

4. Visuo-haptic Rendering with Local Deformations

In the previous chapter it was explained how a volumetric CT image can directly be rendered visually and haptically to the user. It was also outlined how a small deformed sub-image J can be rendered visually in combination with the complete image I, but it was not specified how this sub-image actually is computed. This chapter describes and analyzes the process that is used to compute the image J based on the interaction of interactive virtual tools and the virtual patient's body, using the state of the haptic algorithms and the a priori known deformations resulting from this.

The methods presented here diverge from the typical approach of the state of the art of using mesh based methods for the computation of deformations. Instead, all deformations are computed on the grid of the image data, which makes it obsolete to create a mesh in the first place. An initial version of the algorithm was published in [FMH12] for deformations occurring in needle puncture, and further improvements and comparison of the results to ground truth computed by the finite element methods were given in [FMH13c, FMH13b]. Additionally, the methods were extended and improved for palpation and ultrasound probing in [FMH13a, FMH14, FMSH16].

First of all, background information and related work from the literature are given, followed by a description of how certain deformations can be considered as known a priori for interaction of the needle, ultrasound probe and a palpating finger. The finite differences approach embodied in the framework is then described in detail. For visuo-haptic rendering it is especially important that the used methods are capable of rendering in real-time. Solving this equation for a convergent solution is expensive in terms of computational cost even on a small grid, making it necessary to use optimizations. Thus, several methods that have been employed to achieve better performance are proposed. An evaluation framework is presented that was used to compare the general difference of a diffusive and the linear-elastic approach. The speed up achieved by the different methods is also analyzed by this framework. Lastly, visual examples and time measurement results are presented.

4.1. Background/Related Work

For realistic surgery simulation, it is essential to simulate the behavior of soft tissues under deformation. These deformations can be caused by interaction of tools and tissue or by physiological movement of organs. A computer simulation for surgery simulation of soft tissue deformations has to take into account the large deformations that can occur, the varying material parameters and the computational complexity of such a simulation. The following presents different approaches to soft tissue simulation used in the literature.

4.1.1. Finite Element Method

The finite element method is an approach to solve differential equations describing the behavior of a continous medium [MWTT98]. To do this, the continuum is discretized into a mesh of finite elements. Often, a discretization consisting of triangles (in two dimensions) or tetrahedrons (in three dimensions) is used. The differential equation is then solved for the finite elements by using basis functions. Apart from being used in engineering by using offline computations, which can take up hours or days to solve, it is nowadays a part of real-time computer graphics. Early application of finite element based approaches to computer graphics include the methods presented in [TPBF87]. The work presented in [MG04] uses finite element method with linear elasticity and a plasticity model that is capable of interactive rendering of objects under large deformation and rotation. A stable and fast implementation of another FE simulation is presented in [PO09], which was first used in the 2008 computer game "The Force Unleashed" and is now available as a commercial product. With the advent of General Purpose Graphics Processing Units (GPGPU), fast implementations of FE models have been published, for example in [DGW10]. The use of these techniques also have been researched for application in surgery simulation. One of the first was presented in [BNC96, BN98]. Other work on interactive soft tissue simulation include [CDA99][NMP+05]. In [MJLW07], a method especially suited for soft tissue is presented. Recent simulation includes the work of [LZW14] and [WWD14] for simulation of deformations and cutting.

For the special application of needle insertion simulation the finite element method can be found to be used in various publications. The major challenge involved in this is the coupling of the virtual needle and the soft tissue simulation, which is performed in different ways:

1. Node positions of the soft tissue mesh are modified in a way that they conform to the needle model [DiM03, DS05]. To increase the simulation accuracy subdivision of tissue

around the needle tip can take place.

2. Coupling of tetrahedra and the nodes of the needle by constraining their barycentric coordinates [DGM+09, PND+11].
3. Remeshing at the needle tip [Gok09, CAR+09]. This is different from pure subdivision in the aspect that the elements at the tip are rearranged to better conform to the needle path.

Needle insertion simulation can be used solely for the purpose of computation or prediction of a needle path. The other application is of course visuo-haptic simulation, i.e., the coupling of the virtual needle with a steerable haptic device. Since the finite element simulation is generally performed at low update rates due to its computational demands, and haptic interaction requires very high update rates, this is another major issue which is solved differently in the various approaches. In [PND+11] this problem is focused on with special attention.

For simulating and visualization of deformed medical image data, most surgery simulation systems rely on a surface based approach. The tetrahedral mesh used for FEM computation can then directly be applied for deforming the surface meshes: Each vertex of the surface mesh can be transform by using their barycentric coordinates in the FEM mesh. However, for direct rendering, that is rendering of the image data directly without intermediate surface representains, e.g., by ray casting, different approaches are needed. To be able to use standard ray casting approaches, the FEM mesh can be used to first resample the image under consideration of the deformations. Real-time capable resampling has been demonstarted in [MFMH15] and [ASPO15]. In [GW06], a pipeline for directed rendering without resampling was presented. The advantage of using a resampled image is that this image can be used for other visualization methods as well, for example simulated ultrasound.

4.1.2. ChainMail

The ChainMail algorithm was first published in [Gib97] and models deformations of volumetric structures by assuming a behavior resembling medieval chain mail armor, i.e., if one element is pulled or pushed, the movement and position of linked (chain) elements is affected. Similar to the rings in a chain mail, each element can move unrestricted in a certain area. This approach can be applied for elements of a voxel image [Gib97, RWE08, SBH07, WF04]. The original algorithm is of a highly iterative nature: Based on a starting element, all neighboring elements are added to lists. The elements in these lists are then checked for compliance with the chain mail constraints. In the case that an element does not comply its

position is enforced according to the constraints and the unprocessed neighbors of this element are added to the lists. The algorithm terminates as soon as all the lists are empty. For GPU-based approaches this iterative nature is not desirable therefore parallizable variants are given in [SBH07] and [RWE08]. The idea is to check the chain mail constraints iteratively for consecutive shells and parallelize the process for all the elements of a shell: Starting at the initially moved element, first, all direct neighbors (the first shell) are checked for compliance and if necessary their positions are changed. This is then repeated for the next shell, that is all neighbors of the first shell with exclusion of all the already processed elements (the current shell and the start element). Repetition of this takes place until no elements have been moved in a shell. Another method for parallelization is presented in [RLAM15].

The approach can also be extended to generalized 3D mesh structures [LB03], which is for example applied in the ImaGiNe-S framework [BBG+09] for the soft tissue simulation of liver under respiratory motion.

4.1.3. Other Methods

Besides these, other mesh based methods for computation of deformations can be found in the literature. Mass-spring networks are a popular approach that works by connecting the mesh nodes that represent the mass center of parts of the soft body by a set of springs. It was researched intensively for the simulation of clothes [Pro95] and was also applied for soft tissue simulation in medical contexts [MSN+06]. The major advantage of this method is the straight forward implementation and efficient implementation made possible by using the GPU [GW05]. The drawback is that soft tissue behavior simulated by mass-spring is not directly related to material constitutive laws and the spring parameters have to be identified by dedicated methods [LMH07]. To better resemble the properties of real soft tissue quadratic springs have been used in [SVAT12].

Furthermore, the methods presented for example in [DKS01, JJM+14] use meshless approaches by using a cloud of unconnected spheres or nodes that represent the tissue.

4.1.4. Image Registration

Non-linear image registration deals with the computation of displacement fields to warp two image onto each other [Mod04]. Smoothness of the resulting displacement field is a major concern and is often modeled by a physical process. It also deals with image data defined on regular grids and thus methods developed in this domain are candidates for real-time soft tissue simulation.

The problem to be solved by 3D image registration can be represented by a variational formulation

$$\mathcal{I}[u] = \mathcal{D}(T, R, u) + \alpha\mathcal{S}(u) \overset{u}{\rightarrow} \min \tag{4.1.1}$$

with $T : \mathbb{R}^3 \rightarrow \mathbb{R}$ being a template image that is matched onto the reference image $R : \mathbb{R}^3 \rightarrow \mathbb{R}$, $u : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ being a displacement field, $\mathcal{D}(T, R, u)$ a distance measure and $\mathcal{S}(u)$ a regularization term that is weighted by a constant $a \in \mathbb{R}$. Minimizing this functional results in a smooth displacement field that can be used to compute a warped version of the template image that matches to the reference image. Since image data generally is defined on an image grid, Eq. 4.1.1 is discretized and then solved for the displacement field at the grid locations.

In the context of this thesis, the regularization term is of special interest. It forces the resulting displacement field to be smooth, whereas smoothness can be defined by various approaches. The most researched regularization terms used for image registration are diffusion and linear elasticity, and both reflect physical behavior of matter.

Image registration is usually a computationally intensive process and not suited for real-time execution when applied to large images. To speed up the process implementation on the GPU [MOOXS08] and multi-resolution approaches [HM06, Ash07] can be applied.

4.2. Real-time Image-based Deformations

For the simulation framework a finite differences approach on a regular grid similar to image registration was chosen since it can be directly applied on the grid of the image data and in contrast to the ChainMail and mass-spring approaches, it can easily incorporate formulations that are physically motivated. The following terminology is introduced for distinguishing the undeformed and deformed state of the virtual patient. The undeformed state is the configuration of the virtual patient at time of image acquisition, meaning the CT image data represented in the image I captures the undeformed state. A point in this will be denoted by $\mathbf{X} \in \mathbb{R}^3$ and is considered to be in *reference space*.

Interaction with the virtual patient will cause the tissue of the patient to deform, bringing it into a deformed state. It is assumed that such a deformed configuration exists for each time step t of the simulation. Likewise, for each point in reference space a corresponding point $\mathbf{x}_t$ for each time step exists. These points $\mathbf{x}_t$ will be considered to be in *world space*. This nomenclature reflects the terms Eulerian configuration and Lagrangian configuration, or world and material space often used in finite element texts. The wording world space and reference space is more general and intuitive for the application in the framework and

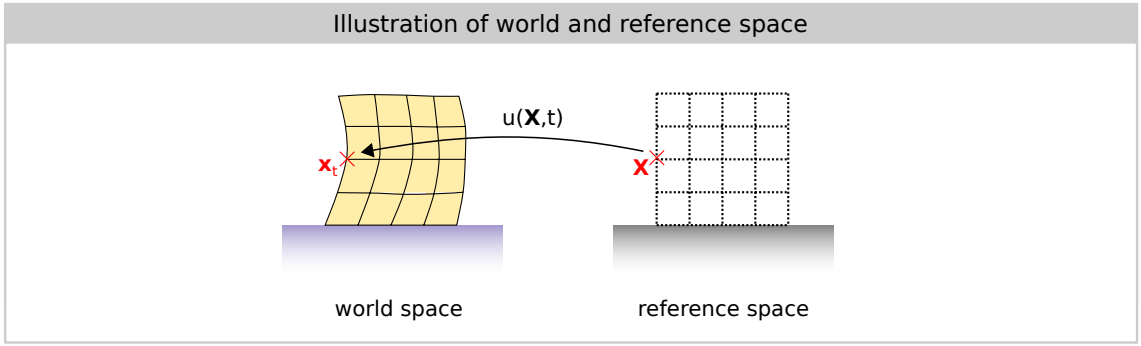

Figure 4.1.: Left: World space with deformed object and displaced point $\mathbf{x}_t$. Right: Reference space with an undeformed object and the corresponding point $\mathbf{X}$.

will also be used in chapter 5. The relation of world and reference space is illustrated[1] in Fig. 4.1. Between the reference space and the world space a transformation exists, which assigns to each point $\mathbf{X}$ a displacement vector $u(\mathbf{X}, t)$ such that

$$\mathbf{x}_t = \mathbf{X} + u(\mathbf{X}, t) \tag{4.2.1}$$

These vectors form the displacement field $u : \Omega_I \rightarrow \mathbb{R}^3$. The deformed image is then given by $J(\mathbf{x}) = I(\mathbf{X} + u(\mathbf{X})) = I(\mathbf{x} + u^{-1}(\mathbf{x}))$. As for I, the image J is defined for certain grid points $\dot{\Omega}_J$. This grid is a regular grid in the world space and resampling of the image J is thus performed for elements $\mathbf{x} \in \dot{\Omega}_J$ for which the corresponding $\mathbf{X} \in \Omega_I$ is unknown. To be able to do so, it is mandatory to use the inverse u^{-1} of the displacement field, which is either possible by performing the computations described in the following on the inverse displacement field or by inverting the resulting field in an extra step.

4.2.1. A Priori Known Deformations

The domain on which a minimization procedure will be performed is limited to the neighborhood of the tool interaction site. As illustrated in Fig. 4.2, the modeling of deformations is based on the following assumptions:

1. For elements beyond and on the border $\partial\Omega_J$ of the local sub-region around a tool, all deformations are zero.
2. Some structures represented by $\bar{\Omega}$ cannot be deformed. Namely, these are bone structures with high Hounsfield values in the CT image data.
3. The deformations $\tilde{u}(\mathbf{X})$ of certain points $\mathbf{X} \in \tilde{\Omega}$ are known a priori. For a needle,

[1]For convenience, the illustration of objects in reference space is desaturated (gray scale) in the following figures, whereas the world space is illustrated in full color.

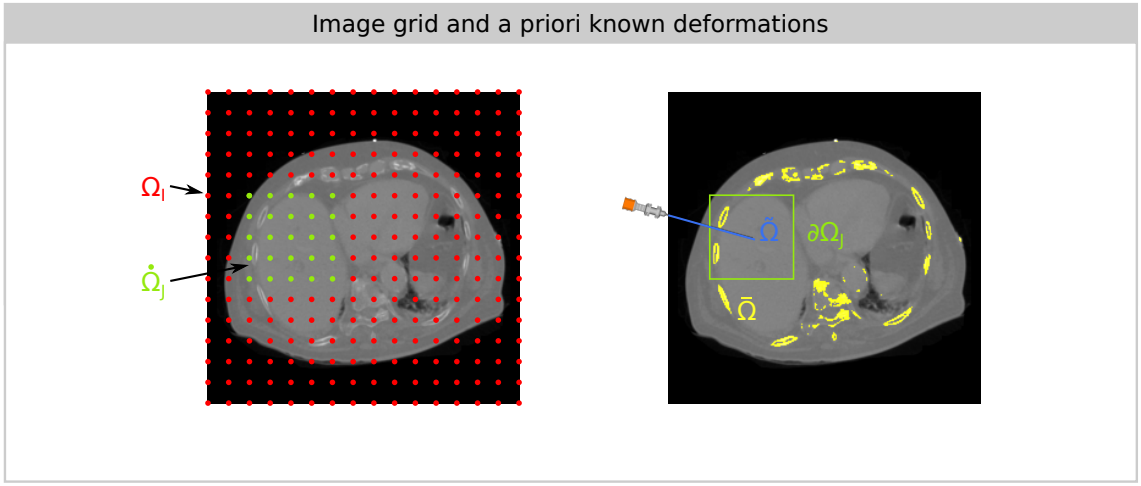

Figure 4.2.: Example of elements of the image grid and the sub image grid $\dot{\Omega}_I$ resp. $\dot{\Omega}_J$ and the known deformations at the border $\partial\Omega_J$, hard structures $\bar{\Omega}$ and points on tool surfaces $\tilde{\Omega}$.

$\tilde{\Omega}$ contains all the points that lie on the needle shaft. For palpation and ultrasound probing, these points are the contact points between the finger/US probe and the skin surface.

The last point needs closer explanation: For a needle with OpenGL $\bar{l}$, all points on the needle shaft with distance $d \in [0, \bar{l}]$ to the tip are denoted by the functions $\tilde{\mathbf{X}}_t(d)$ in the reference and $\tilde{\mathbf{x}}_t(d)$ in the world space. In the case that a stiff and bending free needle is simulated and a simple proxy based haptic algorithm is used, a reasonable choice for these functions are the linear functions [FMH13b]:

$$\tilde{\mathbf{X}}_t(d) = \mathbf{p}_t - \mathbf{r}d \tag{4.2.2}$$

$$\tilde{\mathbf{x}}_t(d) = \mathbf{x}_t - \mathbf{q}_z d \tag{4.2.3}$$

with $\mathbf{p}_t$ being the proxy position, $\mathbf{r}$ the needle insertion direction, $\mathbf{x}_t$ the haptic device position and $\mathbf{q}_z$ the haptic device direction. For the bending needle presented in section 3.5, the relation of the nodes of the needle path set P and needle node set N can be used as basis for the function and in between the nodes interpolation can take place. Likewise, this can be done for the contact nodes of a palpating finger (or ultrasound probe) as illustrated in Fig. 4.3.

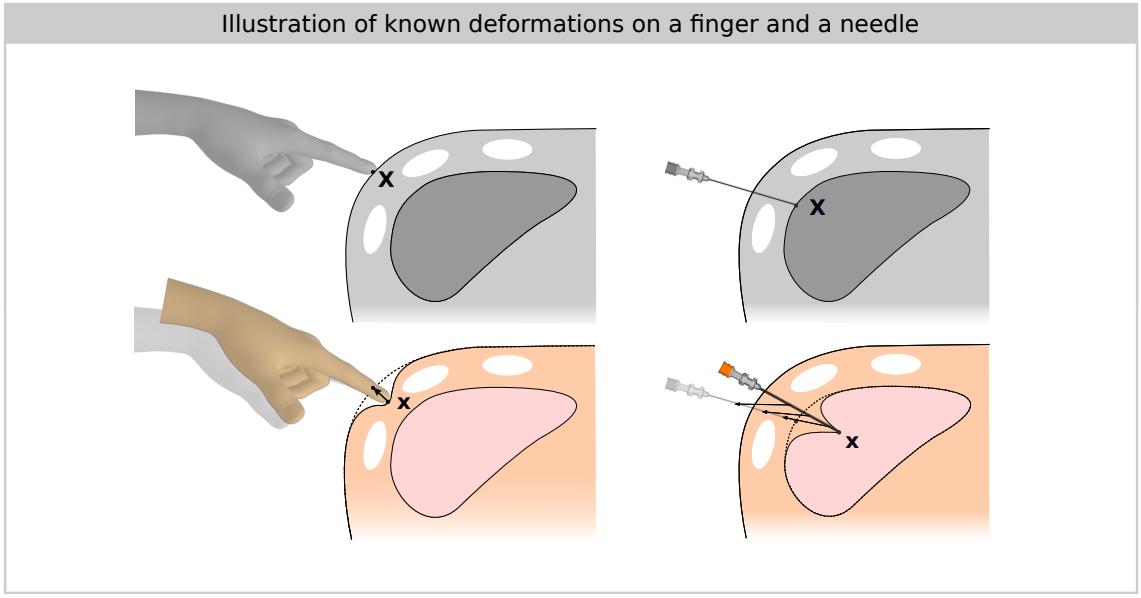

Figure 4.3.: Deformations are based on the spatial relation of the palpating finger resp. needle in reference and world space. For a palpating finger the displacements at the contact points on the surface of the finger are assumed to be known a priori. Likewise, this is assumed for the displacements along the needle shaft. The location of these points are given by the haptic algorithms.

4.2.2. Variational Formulation

For each time step t of the simulation, it assumed that a deformed static state can be computed based on the assumptions from the previous section. Based on these, the computation of the displacement field is modeled as an initial value problem with boundary conditions using a calculus of variations approach. For computing a solution that is usable in the simulation a description of the deformations in form of partial differential equations, a discretization of the continuum and a solving scheme are needed.

Similar to image registration, the problem can be written as a functional that on the one hand enforces smoothness of the displacement field u and on the other enforces boundary conditions:

$$\mathcal{J}[u,\xi] = \mathcal{S}(u) + \mathcal{B}(u,\xi) \overset{u,\xi}{\rightarrow} \min. \tag{4.2.4}$$

with $\mathcal{S}(u) = \int_{x\in\Omega_I} S(u,x)dx$ being the regularization term and $\mathcal{B}(u,\xi) = \int_{x\in\Omega_I} B(u,x)dx$ a boundary condition term using a Lagrange multiplier formulation with multipliers $\xi : \Omega_I \rightarrow \mathbb{R}^3$. This formulation differs to the one of image registration given in Eq. 4.1.1 in so far that the distance term $\mathcal{D}(T,R,u)$ is removed, and a boundary condition term is added. The boundary condition term can be derived from the constraints from the previous section and represent the known deformations. First of all, these are written as Dirichlet boundary

conditions [BHWM10, p. 231]. The boundary conditions enforce zero displacement on the border and the undeformable tissue and the deformations given by the haptic algorithms

$$u(x) = 0, x \in \partial\Omega_J \tag{4.2.5}$$
$$u(x) = 0, x \in \bar{\Omega} \tag{4.2.6}$$
$$u(x) = \tilde{u}(x), x \in \tilde{\Omega} \tag{4.2.7}$$

These conditions are then written in form of a function by rearranging Eqs. 4.2.7 and multiplication with the multipliers ξ

$$B(u, \xi, x) = \begin{cases} \xi(x)u(x) & \text{if } x \in \partial\Omega_J \cup \bar{\Omega} \\ \xi(x)u(x) - \xi(x)\tilde{u}(x) & \text{if } x \in \tilde{\Omega} \\ 0 & \text{else} \end{cases} \tag{4.2.8}$$

It is important to note, that when solving Eq. 4.2.4, the actually resulting values for the multipliers ξ are not of interest. They are only introduced to the equation to model the boundary conditions.

Calculating the Euler-Lagrange equation $\frac{\partial \mathcal{J}}{\partial u} - \frac{d}{dt}\frac{\partial \mathcal{J}}{\frac{\partial}{\partial \dot{t}}\partial u} = 0$ for Eq. 4.2.4 now gives a necessary condition for a solution of Eq. 4.2.4. Here, both $\mathcal{S}(u)$ and $\mathcal{B}(u, \xi)$ do not depend on t and can be discarded giving

$$\begin{aligned} \frac{\delta \mathcal{J}[u, \xi]}{\delta u, \xi} &= \frac{\partial \mathcal{S}(u)}{\partial u} + \frac{\partial \mathcal{B}(u, \xi)}{\partial u} + \frac{\partial \mathcal{S}(u)}{\partial \xi} + \frac{\partial \mathcal{B}(u, \xi)}{\partial \xi} \\ &= \int_{\mathbf{X} \in \Omega_J} \frac{\partial S(u, x)}{\partial u} + \frac{\partial B(u, \xi, x)}{\partial u} + 0 + \frac{\partial B(u, \xi, x)}{\partial \xi} dx \\ &= 0 \end{aligned} \tag{4.2.9}$$

For the constraints, the respective derivatives in Eq. 4.2.9 are with respect to the displacement u and multipliers ξ

$$\frac{\partial B(u, \xi, x)}{\partial u} = \begin{cases} \xi(x) & \text{if } x \in \partial\Omega_J \cup \bar{\Omega} \\ \xi(x) & \text{if } x \in \tilde{\Omega} \\ 0 & \text{else} \end{cases} \tag{4.2.10}$$

$$\frac{\partial B(u,\xi,x)}{\partial \xi} = \begin{cases} u(x) & \text{if } x \in \partial\Omega_J \cup \bar{\Omega} \\ u(x) - \tilde{u}(x) & \text{if } x \in \tilde{\Omega} \\ 0 & \text{else} \end{cases} \tag{4.2.11}$$

For the term $\mathcal{S}(u)$ describing the elastic properties of the simulated tissue, several approaches are possible. Its derivative can be interpreted as the gradient of the minimization process and resembles a differential operator and thus will be abbreviated as ∇S.

4.2.3. Regularization Terms

In image registration, diffusion is used as a regularizer to facilitate smooth deformation fields. It is also suitable for a basic method to visualize deformations. For a simple diffusive regularizer $\nabla \mathrm{S}_{\mathrm{Diff}}$ resulting from the term

$$S_{\mathrm{Diff}}(u,x) = \frac{1}{2}\langle \nabla u(x), \nabla u(x) \rangle = \frac{1}{2}\nabla \left| u(x) \right|^2 \tag{4.2.12}$$

the respective operator is given by [Mod04, p. 138] as

$$\nabla \mathrm{S}_{\mathrm{Diff}} = \frac{\partial S_{\mathrm{Diff}}(u,\lambda,x)}{\partial u} = \frac{\partial}{\partial u}\frac{1}{2}\langle \nabla u(x), \nabla u(x) \rangle = \frac{\partial}{\partial u}\frac{1}{2}\nabla u(x) u(x) \rangle = \frac{1}{2}\Delta u(x) \tag{4.2.13}$$

The diffusive approach works well for image registration, but for physical simulation of soft tissue, compression and shearing effects modeled by an elastic formulation better resemble reality. The linearized elastic potential [Mod04, p. 83] is

$$\mathcal{P}[u] = \int_\Omega \frac{\mu}{4} \sum_{j,k=1}^{3} (\partial_{x_j} u_k + \partial_{x_k} u_j)^2 + \frac{\lambda}{2}(div\, u)^2 dx \tag{4.2.14}$$

It contains the Lamé constants μ and λ responds to the compression and shearing described by the displacement field u. For this potential, the Euler-Lagrange equation are the Navier-Lamé equations [Mod04, p. 83] giving the operator

$$\nabla \mathrm{S}_{\mathrm{LE}} = \mu \Delta u + (\lambda + \mu) \nabla div\, u \tag{4.2.15}$$

The material parameters μ and λ are directly related to the material properties Young's modulus $E = \frac{\mu(2\mu+3\lambda)}{\mu+\lambda}$ and the Poisson ratio $\nu = \frac{\lambda}{2(\mu+\lambda)}$.

The diffusive approach and the linear-elastic approach with constant values for μ and λ do not distinguish the heterogeneous soft tissue present in a patient. Based on the assumption that stiff tissues feature high Hounsfield values in the corresponding CT image

data, anisotropic diffusion [PM90] can be used to introduce the image information into the diffusion process, giving

$$\frac{\partial S}{\partial u} = div(c(x)\nabla u) = c(x)\nabla^2 u + \nabla c \cdot \nabla u \tag{4.2.16}$$

with a function $c(x) : \Omega_I \rightarrow \mathbb{R}$ influencing the local effect of the process when the diffusion is applied to the displacement field u. For edge preserving behavior, it is suggested in [PM90] to set $c(x)$ to a function based on the image gradient. Here, this function should reflect the fact that image values above a certain threshold can be considerd hard tissue. The value has been set heuristically to $s = \frac{100\text{HU}+250\text{HU}}{2}$ representing the transition of soft tissue to undeformable bone structures. For values above this threshold, the function should approach 0.0 to maximally inhibit the diffusion process. Likewise for very soft tissue the function should approach 1.0, making it reasonable to use the logistic function

$$f_{\text{mat}}(x) = 1 - \frac{1}{e^{-k(x-s)}} \tag{4.2.17}$$

with $k = 0.001$ close to zero as the function $c(x)$. The operator

$$\nabla \text{S}_{\text{Diff}}^{\text{mat}} = f_{\text{mat}} \cdot \nabla \text{S}_{\text{Diff}} \tag{4.2.18}$$

thus denotes the anisotropic diffusion with $c(x) = f_{\text{mat}}(x)$.

Regarding the linear elasticity operator, a material function can be introduced similarly. To capture the general concept of a material function influencing the linear elasticity, the concerning operator will be denoted as

$$\nabla \text{S}_{\text{LE}}^{\text{mat}} = f_{\text{mat}} \cdot \nabla \text{S}_{\text{LE}} \tag{4.2.19}$$

The introduction of the material function by the dot operator can be implemented twofold: On the one hand, the general process can be inhibited as for the diffusion process, or on the other hand, the material function directly constitutes varying material parameters for different tissue types. In the context of this thesis, the former is used, whereas varying material parameters for linear elastic regularization in image registration is given in [Kab06].

4.2.4. Discretization using Finite-Differences

For computation, a discretization of the continuous formulation from the last sections is needed that is suitable for implementation on the GPU. It is possible to do this by using a finite element approach, which would be a reasonable choice. In the past, finite element

approaches have been preferred for real-time applications because they allow large homogeneous parts of the simulated object to be represented by a single element. Due to limited computational power, it was highly required to simplify the simulated objects by as few as possible elements. With modern GPUs, it is possible to perform the computation on a larger number of elements, see for example [DGW10]. As an alternative, it is now also possible to perform the computation on a high resolution regular grid, giving a finite differences approach. For fulfilling the objectives of the simulation framework by circumventing additional meshing processes, the usage of the grid of the image data is very suitable. It is also beneficial that the resulting displacement field is defined for the same elements of the image domain which will be needed to resample the deformed image.

The fixed spatial difference between two elements on the image grids $\dot{\Omega}_J$ or $\dot{\Omega}$ is $\mathbf{h} = (h_1, h_2, h_3)^\top \in \mathbb{R}^3$. For instance, the derivative of the first component of the displacement with respect to the second dimension can be approximated by the central difference

$$\frac{\partial u_1}{\partial x_2} \approx \frac{u_1(x_1, x_2 + h_2, x_3) - u_1(x_1, x_2 - h_2, x_3)}{2h_2} \tag{4.2.20}$$

All needed differentiations of an element of the grid can be approximated likewise using the values of the 26 neighbors on the grid. For elements on the border of $\dot{\Omega}_J$, forward resp. backward differences can be used.

4.2.5. Finding a Minimal Solution

To find a minimal solution for Eq. 4.2.4, the Euler-Lagrange Eq. 4.2.9 $\frac{\delta \mathcal{J}[u,\xi]}{\delta u,\xi} = 0$ has to be solved for values $u(x), x \in \dot{\Omega}_J$ and $\xi(x), x \in \dot{\Omega}_J$ on the grid. These values can be arranged by lexicographical ordering in the vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{3n}$ with $n = \left|\dot{\Omega}_J\right|$ being the number of nodes in the grid. Using $\mathbf{u}$ as the solution and a system matrix $\mathbf{A} \in \mathbb{R}^{3n \times 3n}$ containing the approximation of the differential operators, the problem now reduces to solving the set of linear equations in the matrix form $\mathbf{Au} = 0$. This of course has the trivial solution $\mathbf{u} = 0$ and thus the constraints captured in Eq. 4.2.10 and 4.2.11 can be added giving

$$\left(\begin{pmatrix} \mathbf{A} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} + \mathbf{B} \right) \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{b} \end{pmatrix} \tag{4.2.21}$$

with constraint system matrix $\mathbf{B} \in \mathbb{R}^{6n \times 6n}$. In practice, several rows and columns can be removed due to the boundary conditions. A solution for the values of the displacement field can be found by solving Eq. 4.2.21 using inversion of the combined system matrix or more

conveniently by the Conjugate Gradient (CG) method [She94]. However, the problem can also be solved by explicit Euler integration, i.e., introducing an artificial time t and setting $\xi = 0$ giving the derivative with respect to time

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla \mathcal{J}\mathbf{u} = -\nabla \mathcal{S}\mathbf{u} - \nabla \mathcal{B}\mathbf{u} \tag{4.2.22}$$

To find a solution for a single simulation step, the displacements can be initialized as

$$\mathbf{u}_0(x) = \begin{cases} \tilde{u}(x) & \text{if } x \in \tilde{\Omega} \\ 0 & \text{else} \end{cases} \tag{4.2.23}$$

which already fulfills the requirements of the boundary conditions. Eq. 4.2.4 can then be minimized by iteratively solving

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \begin{cases} 0 & \text{if } x \in \tilde{\Omega} \cup \bar{\Omega} \cup \partial\Omega_J \\ \tau \frac{\partial \mathbf{u}}{\partial t} & \text{else} \end{cases} \tag{4.2.24}$$

with a fixed time step. This formulation keeps the boundary conditions enforced by explicitly treating the time derivate for these elements as zero.

4.3. Implementation

As with the visualization methods from the previous chapter, the minimization process was implemented and parallelized using CUDA. The discrete displacement field of the sub region Ω_J is implemented as a double buffered CUDA array with a fixed size of grid elements.

4.3.1. Algorithm Overview

In each time step of the simulator the process of computation of a new displacement field $\mathbf{u}^+$ is performed, which is summarized in Alg. 4.1 and is detailed now. The process is performed on a double buffered array of displacement vectors: `uA` and `uB`. In an alternating fashion one of the buffers is used for reading whereas the other is used for writing. The result from the previous iteration of the algorithm can be used as an initial solution $\mathbf{u}_0$. This requires that in each step the region of interest, i.e., the position of the deformed image does not change. In reality, this is not always the case, so the algorithm has to reflect this by copying the appropriate values from the old grid when initializing.

Before performing the actual computations by a CUDA kernel, the content of the read only buffer at the elements as specified by Eq. 4.2.7 is set. This enforces the right values

at the boundaries and constrained elements. The algorithm then includes the application of a CUDA kernel `cuDeformKernel` that iteratively minimizes Eq. 4.2.4 on the discrete displacement field. This uses the explicit Euler approach, which has been chosen since it does not rely on reductions and can be parallelized efficiently. A description of the kernel follows below.

The general approach for minimization would be to iterate until a certain convergence criterion is reached. For instance, if the sum of squared differences $\mathrm{SSD}(\mathbf{u}, \mathbf{u}^+) = (\mathbf{u} - \mathbf{u}^+) \cdot (\mathbf{u} - \mathbf{u}^+) < \epsilon$ between the results of two iterations falls below a threshold, the loop is terminated. The problem with this kind of criterion is that it relies on a reduction of results computed from the large displacement field to a single scalar value, which is an operation that is not suited for parallel processing hardware. The maximum processing time is also limited by the real-time requirements of the simulation, which makes it necessary to limit the maximum possible time needed for computation. Both these arguments are reflected in the choice of a fixed number of iterations of the deformation algorithm without using a convergence criterion at all.

If the computation has not been performed on the inverse displacement field directly, it is inverted by an iterative fixed-point scheme [CLC+08]. Finally, the displacement field is used to sample the image values $J(\mathbf{x}), \mathbf{x} \in \dot{\Omega}_J$ representing the deformed image function J used for visual rendering as explain in Chapter 3.

Algorithm 4.1 Computation of a new displacement field u and deformed image J

1: **input:** known deformations given by Eq. 4.2.7
2: **input:** double buffered displacement arrays `uA`, `uB` from previous iteration
3: **input:** fixed number of iterations n
4: **output:** array $\dot{J}$ representing the deformed image values on the grid
5: **if** grid location has moved **then**
6: initialize `uA` with values from `uB` with offset
7: **end if**
8: **for** $i \leftarrow 1..n$ **do**
9: set `uA` as given by Eq. 4.2.7
10: apply `cuDeformKernel( uA, uB )`
11: set `uB` as given by Eq. 4.2.7
12: apply `cuDeformKernel( uB, uA )`
13: **end for**
14: set `uA` as given by Eq. 4.2.7
15: optionally invert `uA`
16: use `uA` to sample image values $J(\mathbf{x}), \mathbf{x} \in \dot{\Omega}_J$

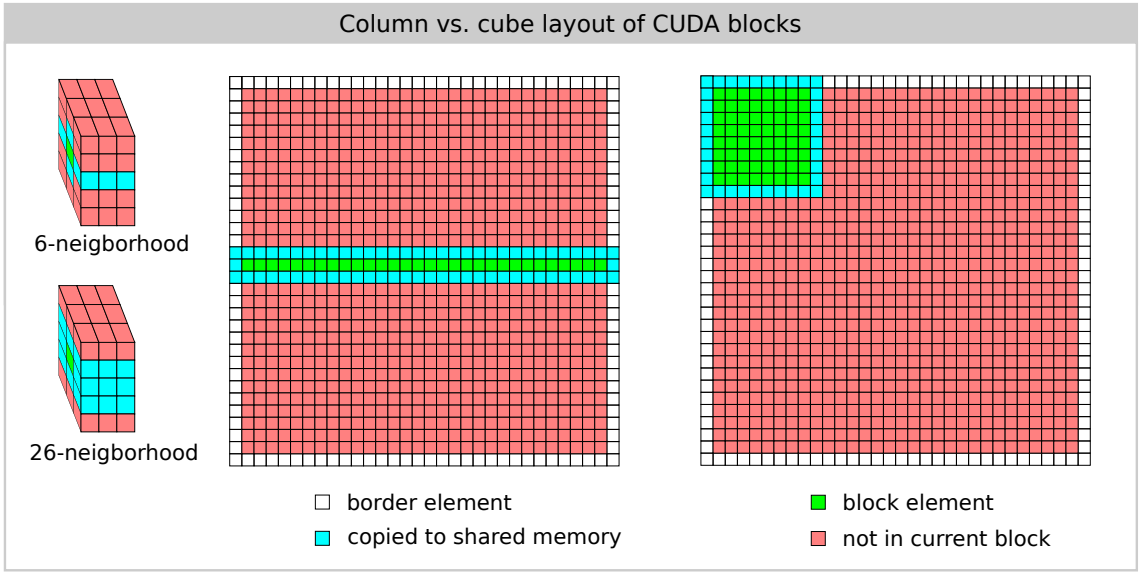

Figure 4.4.: Different block/thread setups in two dimensions with highlighting of the elements that are needed for the processing of all elements of a block. On the left: A column shaped layout uses 5 resp. 9 neighboring columns. On the right: A cube shaped layout in 2D.

4.3.2. Blocks and Threads

For implementation in CUDA, the choice of the layout of blocks and threads is an important design decision. Fig. 4.4 shows two possible block/thread layouts: one that computes a single column of the grid per block and a second one which uses a cubical region per block. In both cases, a single thread is used to compute the value for a single grid element. The main difference is in the number of neighboring elements taken into account in relation to the number of grid elements processed for a single block. It is clear that for computing one iteration of the minimization process for a single element, it is necessary to have available the value of the element and its neighbors from the previous iteration. In an efficient implementation, these values are first loaded into the shared memory of a single block for a neighborhood of elements and then shared memory is used by each thread. Additionally, the loading into shared memory should coalesce well. With a column based block layout with columns aligned to sequential memory location, it is possible to load each column into shared memory by a single coalesced copy instruction. For cube-shaped blocks, the coalescing effect might be reduced. In contrast, the ratio of memory that has to be fetched to results computed (M/R) for a column based layout is much higher than for a cube-shaped layout. For clarification, a short elaboration of these follows. For the column based layout, results for $W - 2$ elements are computed, but the memory of $5 \cdot W$ elements (or $9 \cdot W$ elements, depending on the model) has to be loaded into shared memory. In comparison, the cube-

shaped layout computes $(w-2)^3$ elements but only has to load w^3 elements. For a large grid size W, the ratio approaches $1/5$ or $1/9$ resp., for the column-shaped layout and 1 for the cube-shaped layout. The layout also has a major influence on the theoretical occupancy.

4.3.3. The Kernel

For the diffusive approach, a simplified kernel code is listed in Fig. 4.5. In the listing, the actual computation of element indexes is omitted to keep it generic and independent from the block and thread layout. The time derivative `du_dt` includes the material function, which for this implementation is multiplied with the regularization operator.

The kernel itself does not enforce the boundary conditions given in Eq. 4.2.7 explicitly. The zero displacement on the border are indirectly enforced by skipping the evaluation of the kernel for these elements. Special treatment of the elements on $\tilde{\Omega}$ is not performed and these elements are computed as all others to avoid thread divergence. Instead, the corresponding boundary conditions are enforced by setting the values of the concerning elements before and after the kernel call.

4.4. Optimizations

Under the restrictions of the limited amount of time given for the computation of the new displacement field in each rendering frame the unoptimized minimization algorithm from the previous section might not fully converge. To speed up the process several methods have been developed and applied. During development, it became clear that needle insertion and palpation/ultrasound probing have different requirements for convincing deformation results. For needle insertion, the most important aspect is the small indentation of tissue at the needle tip whereas for palpation a smooth and much larger deformation is required.

4.4.1. Region-of-Interest Pyramid Approach

A first approach is to vary the size of the region of interest during minimization. The idea behind this is that deformations in the proximity of the needle tip are the most important for visual plausibility and thus the solution of the minimization process for elements close to the tip is more important. So instead of performing the minimization process on the full grid $\dot{\Omega}_J$ in each iteration, it is first performed on a region of interest of the grid $R_2 \subset \dot{\Omega}_J$ centered around the interaction site, see left part of Fig. 4.6. Afterwards, it is performed with an increased region size R_1 and finally on the full region of interest ($R_0 = \dot{\Omega}_J$). This way, more processing time will be used on elements that are closer to the needle tip, increasing visual

```
__global__
void cuDeformKernel( float4* _fieldin, float4* _fieldout )
{
  // load values of neighborhood to shared memory
  __shared__   float4 neighborhood[ ... ];
  unsigned int index = ...;
  unsigned int nIndex = ...;
  neighborhood[ nIndex ] = _fieldin[ index ];

  // syncronize all threads of a block so
  // the shared memory is filled with all needed values
  __syncthreads();

  // alias for neighbors t, b, l, r, f, h
  const float4& t = neighborhood[ nIndex + ... ];
  const float4& b = neighborhood[ nIndex + ... ];
  ...

  // alias for center element
  const float4& c = neighborhood[ nIndex ];

  // compute result of laplace operator
  float4 lapl = c * -6 + t + b + l + r + f + h;

  float f = getMaterialParameter( c );

  float du_dt = f * lapl

  // update value if not on border
  // (omitting zero displacement boundary elements)
  if( isBorder( index ) == false )
    // perform one step in forward Euler integration
    _fieldout[...] = c + tau * du_dt;
}
```

Figure 4.5.: Simplified source code for a basic deformation kernel using diffusive regularization and explicit Euler integration.

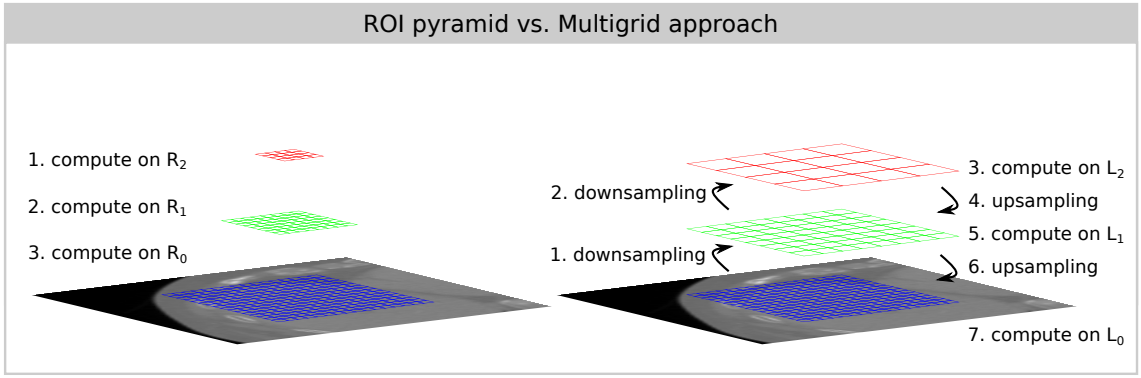

Figure 4.6.: Illustrations for optimization schemes. Left: The ROI pyramid applies the deformation process in different areas of the region of interest. Right: In contrast, the multigrid approach also computes on differently sized grids but these grids cover the same area.

plausibility in a smaller time frame. Performance gains of this and various settings of the size and number of iterations can be found in [FMH13c].

4.4.2. Multigrid Approach

In practice, the ROI pyramid approach has been applied successfully for the simulation of deformations of skin surface and organ capsules inflicted by the needle. With regard to palpation, this approach did not converge fast enough to capture the large deformations created by palpation. As for image registration [HM06, Ash07] or finite element simulation [DGW10] a multigrid approach can be use. In contrast to the ROI pyramid approach, not only is a subset of the grid is included in the computation, but grids with lower resolution are used and the results computed on these are used as an initial solution for the grids with higher resolution. As published in [FMSH16], three grids L_0, L_1, and L_2 are used with the resolutions of 64^3, 32^3 and resp. 16^3 elements are used. To find a solution, first the solution from the previous time step is downsapled twice to the grid with the lowest resolution (L_2). The downsampling can be defined recursively as

$$L_i(x,y,z) = L_{i-1}(2x,2y,2z), x,y,z \in \{1,...,w_i\} \tag{4.4.1}$$

with i being the grid level and w_i the associated grid dimension. An intermediate solution can now be computed on L_2 as given by Alg. 4.1. This solution is then upsampled to the grid L_1 by

$$L_i(x,y,z) = L_{i+1}\left(\left\lfloor\frac{x}{2}\right\rfloor, \left\lfloor\frac{y}{2}\right\rfloor, \left\lfloor\frac{z}{2}\right\rfloor\right), x,y,z \in \{1,...,w_i\} \tag{4.4.2}$$

with $i = 1$. The process is the repeated for L_1 and finally, a solution is computed on L_0.

4.4.3. ChainMail

Since a ChainMail implementation on the GPU [SBH07, Rößl09] is also a promising candidate for computation of deformations in a surgery simulation setting, it has been applied in [FMH13a, FMH14] for palpation simulation. It is used to find an initial solution for the minimization process already described.

Using a single contact node as in [FMH13a] is straight forward to realize with the ChainMail algorithm. The element in the displacement field corresponding to the contact node is used as the initially moved element. For the multiproxy scheme either the initially moved element can be set to the grid location closest to the average of the position of the multiproxies or each contact node position can be used to create a set of initially moved elements. In the latter case, the hull including all the initially moved elements has to be used as the starting point for the iterative application of the ChainMail algorithm.

Deformations computed by the ChainMail method on a regular grid show a diamond shaped regularity that is visually unconvincing, see [FMSH16], and the time needed for the algorithm to terminate is highly influenced by the size of the computed deformation that has to be computed.

4.4.4. Fast Explicit Diffusion

Fast Explicit Diffusion [GW10] is a time stepping scheme for explicit Euler integration. It introduces a sequence of variable step OpenGL into the diffusion process for which some steps are larger than the theoretically stable step OpenGL. It can be applied in diffusion based regularization in image registration [SREWH12] and also in the soft tissue simulation framework presented here. In [FMH13c], it has been shown that this approach decreases the time needed for finding a stable solution. As a downside, this approach can only be applied to diffusion and not to the approach based on linear elasticity.

4.5. Evaluation Framework with In-silico Ground Truth

Evaluating the performance and accuracy of the methods for computation of deformations can be done by comparison to ground truth. Obtaining reasonable ground truth is of major importance for the evaluation. Ideally, data based on in-vivo experiments is desired that includes a series of volumetric images and force measurements for a needle insertion or palpation. Image registration then could be used to create ground truth displacement fields. Obviously, it is not easy to acquire such data. As a substitute, in-silico ground truth has been generated in [FMH13c, FMH13b] by using FEM software and appropriate tissue properties

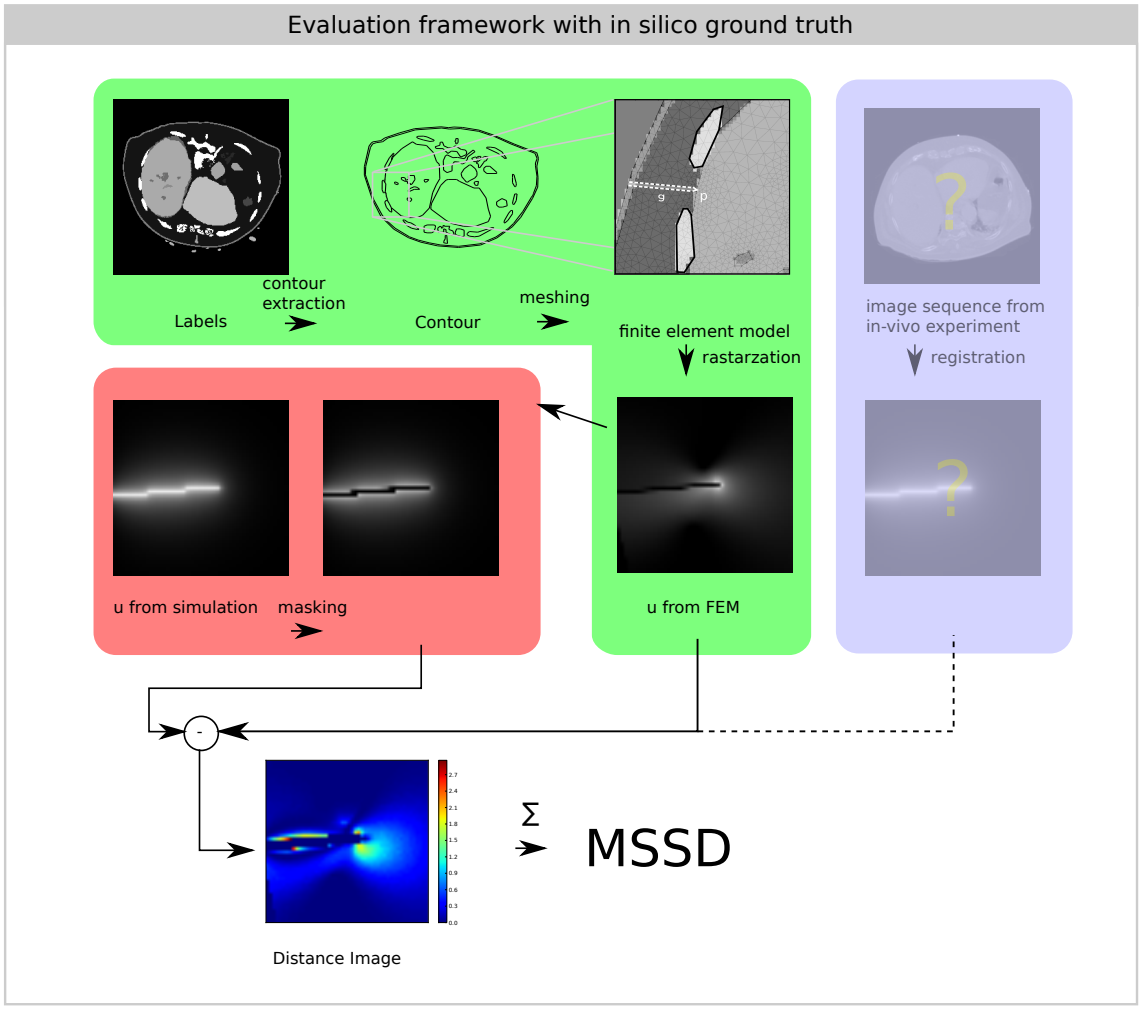

Figure 4.7.: FEM evaluation pipeline. Based on the segmentation of a single image slice, contours are extracted. From these, a FEM mesh consisting of triangular elements is created. For simulation with a sliding boundary condition and fixed offset at the needle tip, deformations are computed and the resulting displacements are rasterized and compared to displacements computed by the real-time simulation. The result is a scalar value representing the mean squared distance between both solutions. Ideally, in-vivo experiments would replace the FE model.

	layout	w	mean	stddev	N
Normal					
Diff	Column		3.67	0.042	272
Diff	Cube	4	1.05	0.032	952
Diff	Cube	6	0.65	0.032	1525
Diff	Cube	8	1.02	0.039	973
LE	Column		5.21	0.044	192
LE	Cube	4	1.41	0.024	706
LE	Cube	6	0.87	0.031	1146
LE	Cube	8	1.33	0.032	751
Multigrid					
Diff	Column		6.24	0.228	161
Diff	Cube	4	1.85	0.068	540
Diff	Cube	6	1.35	0.073	739
Diff	Cube	8	1.80	0.073	554
LE	Column		8.54	0.064	118
LE	Cube	4	2.35	0.058	424
LE	Cube	6	1.63	0.054	612
LE	Cube	8	2.22	0.070	450

Table 4.1.: Mean processing times in ms for the application of Alg. 4.1 using the diffusive (Diff) and linear elastic (LE) regularization kernels and the multigrid approach. Column and cube layouts are analyzed.

and models. This way, the performance of the algorithms can be quantified with regard to accuracy and convergence behavior.

In Fig. 4.7, the evaluation process is illustrated. The procedure is as follows: First, a single slice of the patient image data is selected. A segmentation has to be available for it. From the segmentation, the contours are extracted using 2D marching cubes and post-processed manually using a 2D CAD software to obtain a 2D vector representation of the organ boundaries. The contour of the simulated needle insertion channel is also added to the CAD image with the tip of the needle at the interface between soft tissue and liver tissue. Using the contours, a FE model consisting of triangles was created. The surface of the needle insertion channel is modeled as a sliding boundary and the tip is displaced by a fixed offset. After solving the FEM for the resulting displacements, the displacement field is rasterized to make it comparable to the results of the real-time algorithm.

4.6. Experiments & Results

Here, the different thread layouts and kernels are analyzed in terms of their computational efficiency. For an in-depth comparison of the accuracy, see [FMH13c, FMH13a]. As can be

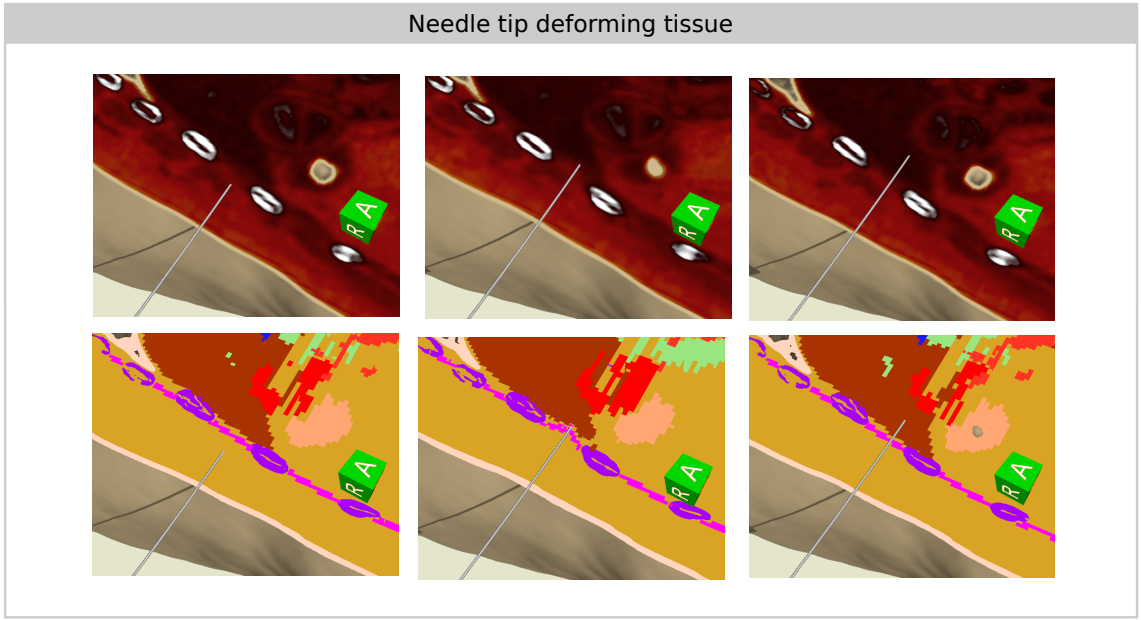

Figure 4.8.: Deformations caused by an advancing needle in the region of the intercostal fascia. In the upper row a cutaway view with transfer function rendering is used. For the lower row label tagging is used.

seen in Tab. 4.1, the choice of the block/thread layout has a big influence on the actually achieved processing times.

Additionally, Figs. 4.8, 4.9 and 4.10 shows visual results from the simulator. The first shows the advancing needle and resulting deformations at the needle tip using ray casting. For visualization of the different tissues, a 1D transfer function resp. label tagging is used. Fig. 4.9 shows deformations issued by a palpating finger. To demonstrate that the finger indents the tissue deeper at the intercostal spaces, a virtual force of 1 N is applied on the haptic device, i.e. it is simulated that the user presses against the skin with a constant force. Finally, Fig. 4.10 shows the propagation of deformations around the needle by visualization of the displacement field by color coding its orientation and magnitude.

4.7. Discussion

The presented methods for simulation of soft tissue deformation not incorporate segmentation mask or a labeling function directly but rely on a material function based on the image data. The function used for this only distinguishes between deformable soft tissue and undeformable hard tissue. Similar to the palpation algorithm from the previous chapter, this function uses a threshold to perform this distinction between soft and hard tissue. Here, this threshold has been set to a fixed conservative value and might have to be adjusted for exceptional image data. Regarding time effort of adjustment of the threshold, it should be

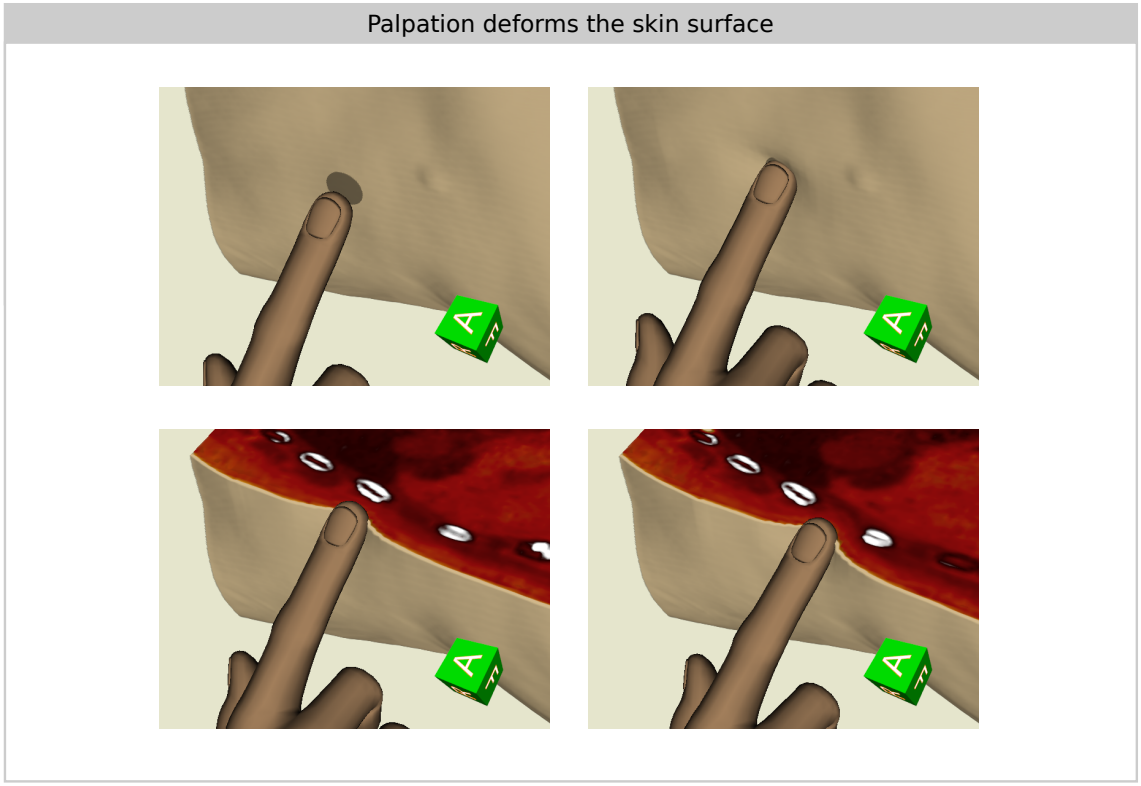

Figure 4.9.: Deformations caused by palpation with and without clipping enabled. The force exerted on the virtual finger is 1 N.

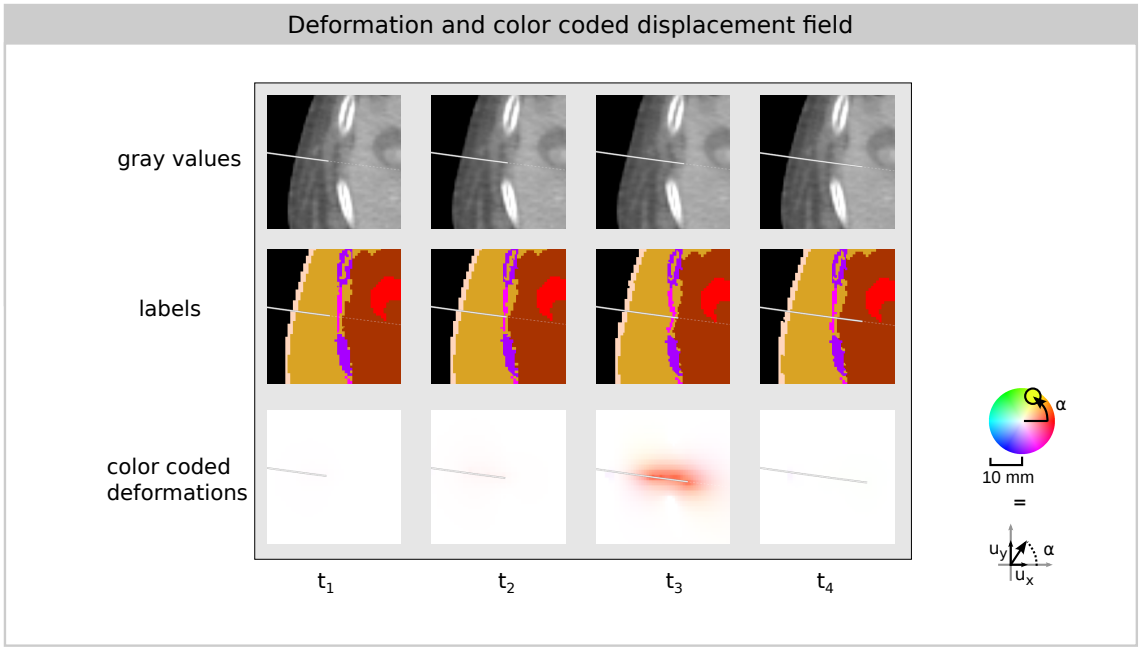

Figure 4.10.: Needle advancement as seen in multiplanar reformations with CT image values and label taging. Third row: HSV color coded displacement field whereas the hue represents the angle between the x-axis and the deformation vector projected to the image plane; the saturation indicates the magnitude of the projected vector.

sufficient to use the one also used for the distance mask. However, it will be necessary to check if such an approach is feasible.

Comparing the methods used for speeding up the simulation, the ChainMail approach was usable but it relies on parameters that determine the positional constants. These have no direct link to the soft tissues parameters and thus, this approach is considered rather inelegant. Best results have been obtained with the multi resolution strategy, which is also independent of the choice between linear elastic and diffusive approaches. Overall, the difference between the diffusive and linear elastic approach regarding simulation times do not differ widely and thus the linear elastic approach should be preferred since it is represents physical reality better. Without question, the choice of CUDA block and thread layout influences the processing times to a large extend and the optimal choice of layout ($w = 6$) is rather unexpected and seems to be highly hardware dependent. Future implementation and ports to different hardware must be performed with this in mind. The choice of the extent of the region of interest for which the deformations are computed is a trade-off between computational effort and plausible results. Ideally, the field should be of infinite size or in the range of the extent of the image data.

In general, the presented approach give good visual results. However, it is arguable how physically accurate the resulting deformations are. To make them comparable with a better suitable ground truth in-vivo or in vivo data should be acquired since the solutions of FEM models are only an approximation to reality. In practice, the differences between the diffusive and linear elastic approaches are not easily distinguishable to the untrained eye. Nevertheless, future work should include more sophisticated physical models with varying material parameters, non-linear materials and higher resolution of patient data and grid.

5. Visuo-haptic Rendering using Respiratory Motion Models

In this chapter, a novel method is presented that can be used to provide a breathing virtual patient model. This method, which has been published in [FWMH15] and [WFMH15], includes both visual and haptic volume rendering based on a model of the patient's breathing. The necessity of visuo-haptic methods that can render a breathing patient model arises from the fact that the behavior of needles that are inserted into the abdominal region are highly influenced by organ motion induced by respiratory motion. An example for this are the parts of the liver that are close to the diaphragm; the diaphragm itself moves up to 5 cm between full inspiration and full expiration [KYS13]. This displacement is not limited to the region close to the diaphragm, but also includes other visceral organs. Thus, this movement not only affects these tissues but also bends flexible needles while being inserted, leading to variations in the insertion channel of the needle. Additionally, bending of the needle inside of the patient causes tilting that can be seen or felt by the surgeon.

The requirements of convincing simulation of the phenomenon thus is twofold: On the one hand, the displacement of tissue and needle inside the patient has to be visualized and on the other hand, the tilting and torque has to be rendered haptically to the user via the haptic device. This has been addressed by introduction of motion models into the rendering methods that were presented in chapter 3. The key idea is to use a motion model that yields a time varying displacement field describing the movement for all image elements. For visual rendering, the motion model is used to deform the rays of the ray casting procedure, see section 3.2. Similarly, the motion model displaces the proxy positions used for haptic rendering as introduced previously in section 3.5 and 3.6. The overall result is a new and distinct real-time capable approach that unifies the visual and haptic rendering using a single motion model.

This chapter is organized as follows: First, related work on real-time respiratory motion simulation is given followed by a description of the time varying displacement field function. Afterwards, the motion models are specified in detail then the new methods for visuo-haptic rendering a presented. Finally, results are presented and discussed.

5.1. Related work

Previous work on respiratory in virtual reality simulations includes the work of [SID+08] and [HNR+10]. The former presents a method for real-time simulation and rendering of a deformable lung model based on 3D triangle meshes. Also, in [HNR+10] a fast to compute motion model for the abdominal organs is given and recommended for real-time tumor motion prediction using a depth image of a patient's skin surface. For simulation of angiography, the work of [WAC07] included a simple breathing model. This model is based on an affine transformation that is driven by a sinusoidal function. These methods only compute deformations for nodes of the 3D mesh of a patient data set and do not represent surgery simulations that include needle insertion or haptics.

For liver biopsy, the simulator presented in [VVA+13, VBBG11] includes simulation of needle insertion and visualization of respiratory motion of the liver and internal organs. For this, the Generalized ChainMail algorithm was used to propagate the deformations and parameters for the simulation were determined in [VVBJ12, VVL12]. However, the influence of the breathing motion on needle bending is not covered in this simulator and the breathing is driven by a sinusoidal function.

An other framework [DWJ10] covers the visualization of breathing motion in an ultrasound simulation of a liver puncture simulator. The respiratory motion is based on a sinusoidal displacement. Additionally, the resulting respiratory motion has no effect on the haptic rendering of needle insertion forces.

5.2. Respiratory Motion as a Transformation

Without modification, the methods presented in chapter 3 can only be used for static volumetric image data. Using the methods introduced in chapter 4, it is possible to deform locally restricted regions of the image data but due to limitations imposed by hardware, it is not possible to use these for global deformations of the virtual patient.

In this chapter, a time varying global displacement function $u(\mathbf{X}, t) : \Omega \to \mathbb{R}^3 (\Omega \subset \mathbb{R}^3)$ is used, which maps points $\mathbf{X} \in \Omega$ in the static reference space (see section 4.2) to their corresponding position $\mathbf{x} \in \Omega$ in the world space. This formulation uses Lagrangian coordinates, i.e., for a particle placed at $\mathbf{X}$, the displacement function $u(\mathbf{X}, t)$ describes its trajectory in dependence of the current time $t \in \mathbb{R}$. For illustration, Fig. 5.1 shows this relation for $u(\mathbf{X}, t)$ and the inverse mapping $u^{-1}(\mathbf{x}, t)$. Ideally, the relation is required to be bijective, that is, the inverse mapping $u^{-1}(\mathbf{x}, t)$ has to exist.

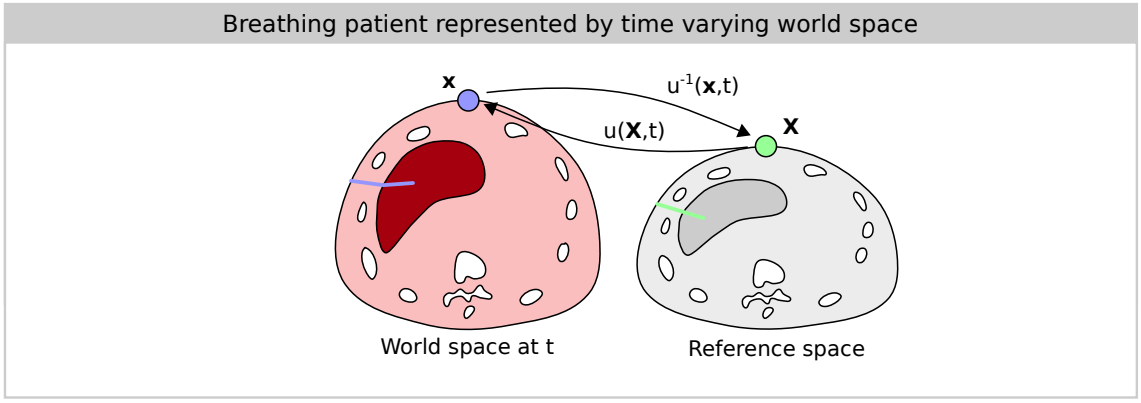

Figure 5.1.: For each time step t in the simulation, a relation between the the current state of the breathing virtual patient and the reference state of the patient exists. This relation is a transformation $u(\mathbf{X}, t)$ between the undeformed reference space and the deformed world space at time step t.

For this displacement function, different implementations are possible. A simple example would be $u(\mathbf{X}, t) = (0, a \cdot \sin(t), 0)^\top$, which models a sinusoidal displacement with amplitude a for all elements of the reference space. Such a function is obviously not a suitable choice to represent the complex motion of internal organs. Indeed, three main requirements exists for this function:

- It should model patient specific organ motion under respiratory breathing.
- It should include the naturally occurring variations of breathing.
- It has to be fast to compute to be usable in ray casting based visual rendering (high rate of sampling points, low update rate) and haptic rendering (low rate of sampling points, high update rate).

Following this specifications, the next section will present a detailed description of motion models that have been integrated into the simulation framework.

5.3. Respiratory Motion Models

To simulate respiratory motion using a the time varying displacement field, i.e., the motion model function $u(\mathbf{X}, t)$, spatio-temporal 4D CT image data [EWS+07] can be used. Using this 4D CT data, it is possible to analyze the motion of internal structures of a patient with the goal of improving radiation therapy in case of tumors present in the thorax or abdominal region. Given the 4D CT image data in form of sequence of 3D images, it is possible to perform non-linear image registration between the elements of the sequence. Precise registration techniques have been in the focus of many researchers in the recent years as well as the analysis and modeling of the respiratory motion present in the image data [EL13].

Here, the basis of the motion models are the displacement fields estimated from the 4D CT data set. A data set is a sequence of n 3D CT images $I_{j\in\{1,\dots,n\}} : \Omega \to \mathbb{R}$ that includes a respiratory cycle from maximum inspiration over expiration back to total inspiration. Following the convention, the cycle includes the states end inspiration (EI), mid inspiration (MI), end expiration (EE) and mid expiration (ME). Without loss of generality, the image I_1 is considered a reference phase. Similar to the methods in chapter 4, the image associated to the reference phase is supposed to be defined on the reference space, i.e., it is the static state of the undeformed virtual patient. Now, a non-linear transformation $\varphi_j : \Omega \to \Omega$ describes the motion between the reference phase and the phase associated to I_j. Instead of representing the transformation by φ_j, it can be represented by a displacement field $u_j : \Omega \to \mathbb{R}^3$ using $\varphi_j = id + u_j$. This displacement field u is similar to the one defined in chapter 4 for the local deformations.

Based on the fact that the 4D CT data set is a time series of 3D CT images representing consecutive states in a breathing cycle it would be straight forward to directly use the image sequence and display it in the visuo-haptic simulation. This could be done by replacing the currently rendered image data in the rendering algorithms with the next image in the series as done in stop motion video animation. Of course, this is impractical and has the following problems: First, in each frame, all used image data including segmentation mask and distance images would have to be replaced by the respective versions. Second, the number of images in the sequence is low, i.e., a sequence of 7–14 phases is reconstructed to form a 4D CT data set, which in relation to the length of the breathing cycle results in a very low frame rate. This frame rate is not high enough to create the illusion of fluid visual animation in a human observer. For haptic rendering, it is not clear how the transition between two images should be handled. At least, the spatial relation of two adjacent images should be used for the transition, making a registration that yields a displacement field necessary.

Generally, non-linear image registration techniques are used to compute the transformations between the phases to estimate the respiratory motion. An active research area is the case of lung motion estimation [SVR13, WSRHE14, MVR+11], which mainly focuses on the lungs and lesions in the lung. However, for convincing simulation of breathing motion in a virtual reality setting, it is necessary to model the movement of all structures of interest. This is a challenging problem to solve, since the the motion fields are not generally smooth, i.e., at sliding interfaces around organ borders such as the interface of lung and liver and the surrounding tissue [SRWHE12]. To overcome this limitiation, the registration approach presented by [SRWHE12] is used. It is able to handle sliding interfaces by modifying the

smoothness constraints at organ boundaries. After computation of the transformations φ_j, these are filtered by principal component analysis. This is a common approach [McC13] to mitigate inconsistencies in the motion estimation that can occur because of independent registration of each phase to the reference image. Noise and image artifacts present independently in each image can be the source of these inconsistencies.

Now, to use the estimated motion in the simulator, it is necessary to know the transformation for each point in time and space to the reference phase. For this, in each phase j, it is necessary to have available the inverse of u_j to be able to relate a point in the world space to the reference space. With a high resolution of the 3D displacement field and several phases, the memory consumption of a model easily can be very high. Thus, it is necessary to either directly implement the function $u(\mathbf{X}, t)^{-1}$ or perform an inversion of it during runtime of the simulation. Theoretically, by inverting each of the resulting displacement fields or by diffeomorphic registration, it is possible to directly create an inverse model. However, in practice, no appropriate inverse model could be created. In the framework presented here, the method of inversion of the displacement field on-the-fly at run-time is used and in the following this will be detailed. For this several approaches have been developed. The first category considers the idea of using the reference phases as key frames of animation of the patient's respiratory motion and linearly interpolate between key frames. Also to enable out-of-sample predictions, that is states that are not captured in the cycle of transformations, extrapolation takes place in a second approach. Different from this, a third approach uses a surrogate signal to drive a linear model yielding the respiratory motion vectors.

5.3.1. Key Frame Approach with a Single Respiratory Cycle

To perform animation of virtual objects, virtual reality often uses the technique of defining key frames of the pose of the virtual objects. For example, to make an object follow a path, only start and endpoint of the path have to be defined. To animate the object, these two positions then can be interpolated over time. For more complex movement, more key frames have to be introduced to the path. This idea can be employed for using the motion fields associated to the 4D CT image data. For each phase j with associated displacement field u_j and image I_j, a key frame can be defined, forming a circular sequence of key frames. Each element can be considered to be placed on a normalized time line with associated $\tau_j \in [0, 1)$. For a single point in reference space $\mathbf{X}$, this is illustrated in the left part of Fig. 5.2 during a full cycle. During this cycle, the displacement of the point in world space can be expressed by $b(\mathbf{X}, \tau) : \Omega \to \mathbb{R}^3$, whereas $\tau \in [0, 1)$ is a normalized time value corresponding to the current simulation time. It can be assumed that that for any τ two adjacent key frames with

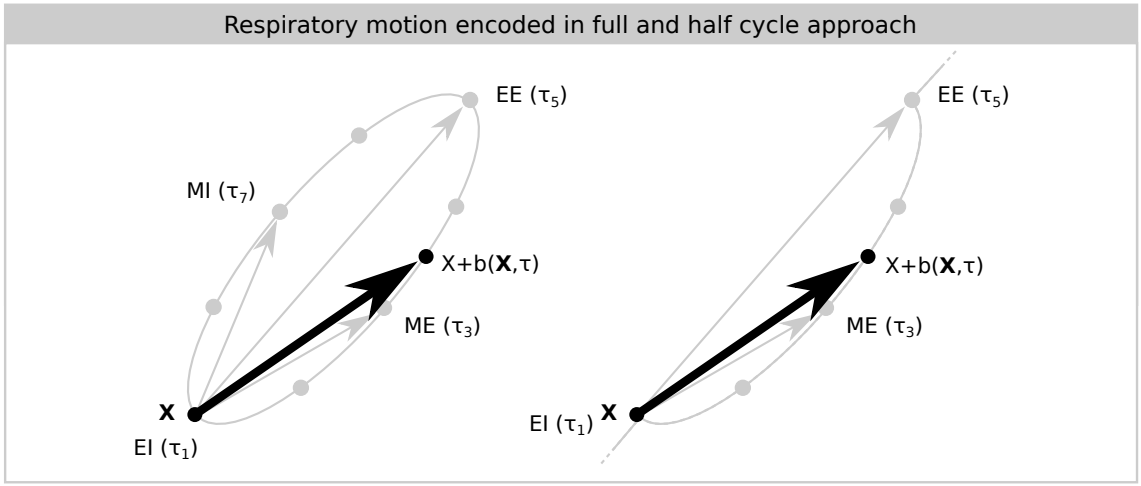

Figure 5.2.: With end of inspiration as reference phase, the respiratory motion for a single point **X** for the full cycle approach (left) and the half cycle (right) is illustrated. For the full cycle approach, $n = 8$ phases resp. key frames are used and in between key frames, linear interpolation takes place. For the half cycle approach, only $n = 5$ phases from the expiration phase are used and to predict samples that are not within the cycle, extrapolation takes place beyond end of inspiration (EI) and end of expiration (EE). (MI = mid inspiration, ME = mid expiration).

displacement functions u_i and u_{i+1} and normalized time values τ_i and τ_{i+1} are available that can be used for linear interpolation, giving a definition

$$b(\mathbf{X}, \tau) = (1 - \alpha(\tau))u_i(\mathbf{X}) + \alpha(\tau)u_{i+1}(\mathbf{X}) \tag{5.3.1}$$

with $\alpha(\tau) = \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i}$ being the weighting factor.

To relate the displacements given for the cycle on the normalized time line to the displacement during simulation, i.e., $u(\mathbf{X}, t) = b(\mathbf{X}, \tau)$, it is necessary to define a function relating t to τ, that is

$$\tau = f(t) : \mathbb{R} \rightarrow [0, 1) \tag{5.3.2}$$

In the upper part of Fig. 5.3, three function types are proposed for this purpose: Their common idea is that they run trough the cyclic sequence in a sawtooth shaped function. Starting at EI with $\tau = 0$, they run towards $\tau = 1$, passing the values corresponding to ME, EE, MI. As soon EI is reached again, the cycle starts anew. To introduce more natual variation ob breathing into the cycles, the length of each period can be adapted by randomly choosing a new cycle length or resp. choosing a new random derivative $f'(t) = \alpha$ after each cycle. Alternatively, the second derivative $f''(t)$ can be randomly estimated after each cycle, giving a smooth slope for the function $f(t)$.

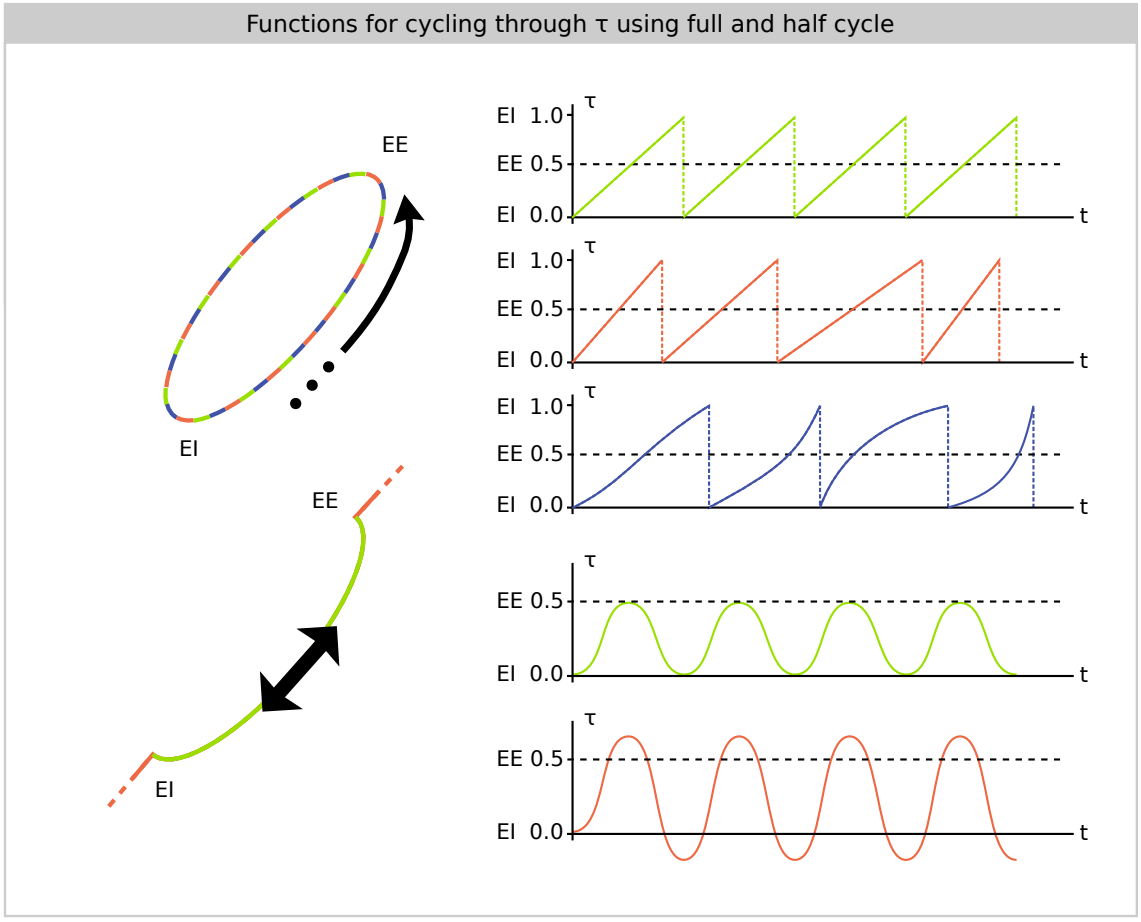

Figure 5.3.: Functions for cycling through the image sequence for full cycle (upper) and half cycle approach (lower). (EI = end of inspiration, EE = end of expiration).

This full cycle approach is straight forward and naturally follows the fact that the 4D CT image consists of several phases. To reflect the variations and hysteresis included in the breathing motion of the patient, it is necessary to at least use key frames for each the phases EI, ME, EE and MI. Of course it is desirable to use a high number of phases resp. all available phases, but the memory consumption of these can easily exceed the available resources, see also [FWMH15].

5.3.2. Key Frame Approach using a Half Cycle

The full cycle approach is only capable of looping through the sequence of phases of the 4D CT image data and variation of the respiratory motion is only possible in terms of overall cycle length resp. variation of the playback rate, i.e., time derivative of $f(t)$. Maximum inhalation and exhalation is fixed to the states captured by the passes associated to EI and EE. Of course it is desirable to also be able to display states that represent inhalation and exhalation beyond the data present in the key frames, i.e., it is necessary to perform out-of-sample prediction to be able to include more natural variation.

A possible approach to do so is to only use half of the phases and perform extrapolation for the parts beyond EI and EE, see Fig. 5.3. In-sample values are all values of τ in the range $\tau_{EI} = 0 < \tau < \tau_{EE}$, for which Eq. 5.3.1 is used. Values not within this range are considered to be out-of-sample (OOS), making it necessary to perform extrapolation. This can be done by using the displacements associated to EI and EE

$$b_{\text{oos}}(\mathbf{X}, \tau) = (1 - \alpha_{\text{oos}}(\tau))u_{\text{EI}}(\mathbf{X}) + \alpha_{\text{oos}}(\tau)u_{\text{EE}}(\mathbf{X}) \tag{5.3.3}$$

with $\alpha_{\text{oos}}(\tau) = \frac{\tau - \tau_{\text{EI}}}{\tau_{\text{EE}} - \tau_{\text{EI}}}$. For this approach, a different relation between normalized time τ and simulation time t is needed. Here, it is necessary to use a smooth function that oscillates between τ_{EI} and τ_{EE} (excluding out-of-sample prediction) resp. values below τ_{EI} and above τ_{EE} (including OOS). In contrary to the full cycle approach, this method does not include hysteresis, i.e., the displacements for both inspiration and expiration are on the same path.

5.3.3. Model using Surrogate Signals

Both the key frame approaches are limited in their possibility to include natural variation of the respiratory motion by either being limited to a single sequence or by not being able to include hysteresis. In a third approach, a motion model using a surrogate signal and its time derivative is used. A surrogate signal is a measurement of the patient's breathing that is acquired by for example a spirometry device or abdominal belt, yielding a 1D signal as shown in Fig. 5.4. Apart from a simple 1D signal, it is also possible to use higher resolution surro-

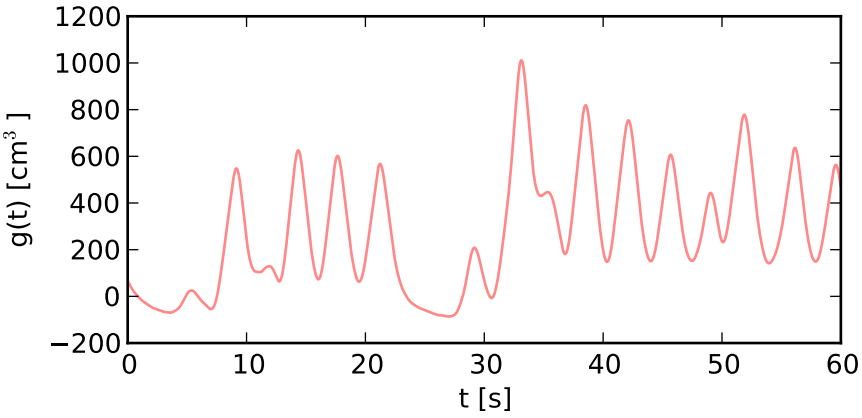

Figure 5.4.: A surrogate signal measured by a spirometry device.

gate signals as for example a depth image captured by a time of flight camera. To use the signal to build a model of the motion of the organs and tissues of the patient, it is assumed that a linear correspondence between the motion and the signal exists. For the reconstruction of a 4D CT image, it is necessary to have available a surrogate signal measurement, making it unnecessary to capture these in an additional step.

In the following, a motion model evaluated for lung motion is used [WWE+14]. The surrogate signal and its time derivative is denoted by $\hat{\mathbf{z}}(t) = (g(t), g'(t))^T : \mathbb{R} \rightarrow \mathbb{R}^{n_{\text{sur}}}$. The purpose of using the time derivative is to include an indicator for expiration resp. inspiration for modeling hysteresis. Employing multivariate regression on the transformations φ_j and associated $\mathbf{z}_j$, a linear model can be learned by using the before mentioned linear correspondence between surrogate signal and respiratory motion. To do so, the transformations are interpreted as column vectors using lexicographical ordering of the m image voxels of I_j, giving $\mathbf{b}_j \in \mathbb{R}^{3m}$. The resulting linear model yields displacements $\mathbf{m} \in \mathbb{R}^{3m}$ based on the mean motion $\bar{\mathbf{b}}$ and a system matrix $\mathbf{B} \in \mathbb{R}^{3m \times 2}$

$$\mathbf{m} = \bar{\mathbf{b}} + \mathbf{B}\hat{\mathbf{z}}(t) \tag{5.3.4}$$

To get to a form suitable for implementation, the Eq. 5.3.4 can be rewritten as

$$\mathbf{m} = \mathbf{a}_1 g(t) + \mathbf{a}_2 g'(t) + \mathbf{a}_3 \tag{5.3.5}$$

with $\mathbf{a}_{1..3} \in \mathbb{R}^{3m}$ being column vectors.

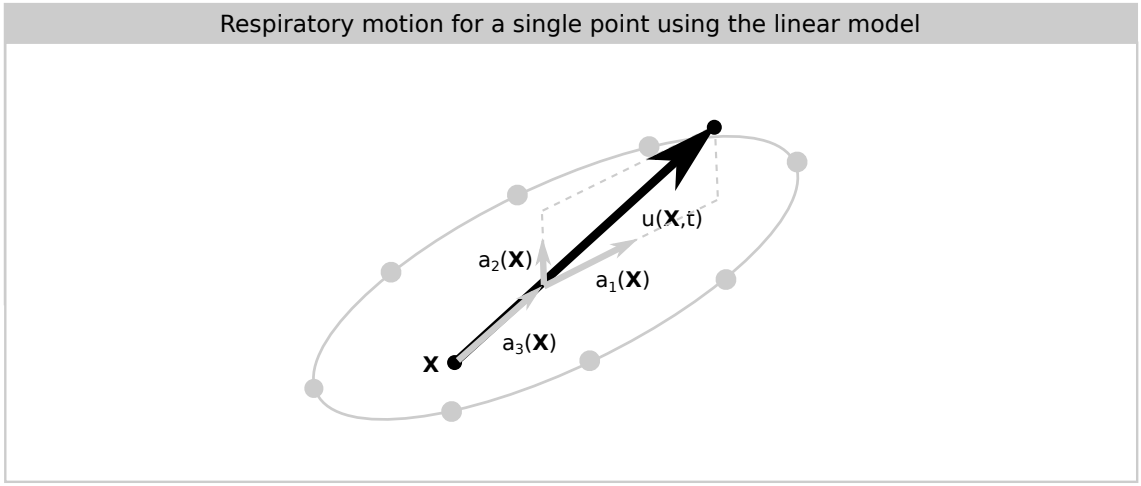

Figure 5.5.: Schematic representation of the linear motion model, which yields a displacement for each point based on a linear combination of three vectors.

Each of these then can then be again transformed back to functions $a_{1..3} : \mathbb{R}^3 \to \mathbb{R}^3$ yielding a displacement vector for a position in world space by reverting the lexicographical ordering

$$u(\mathbf{X}, t) = a_1(\mathbf{X})g(t) + a_2(\mathbf{X})g'(t) + a_3(\mathbf{X}) \tag{5.3.6}$$

This linear combination is shown in Fig. 5.5 for a single image element. Evaluation of the model for a position $\mathbf{X}$ then can be implemented as three texture lookups and summation.

5.4. Direct Visuo-haptic Rendering using Motion Fields

Having defined a displacement function, it will now be detailed how the displacement function can be used in order to render a virtual patient model visually and haptically.

5.4.1. On-the-fly Inversion of the Displacement Field

To introduce breathing motion into the visual rendering by the ray casting method, the displacement function is used. Normally, ray casting uses straight lines along which sampling takes place. The idea is that instead of using straight rays for rendering a deformed body, deformed rays are used on an undeformed body but giving the same result, see Fig. 5.6. For rendering of the breathing virtual patient model, the render integral is evaluated for rays in world space. Thus, it is necessary to find the corresponding position in reference space for a sampling position on a ray, making it necessary to use the inverse displacement function $u^{-1}(\mathbf{x}, t)$. Even if $u^{-1}(\mathbf{x}, t)$ exists for the whole image domain, it generally is not available for a given $u(\mathbf{X}, t)$ in the simulation environment. It is of course possible to perform the

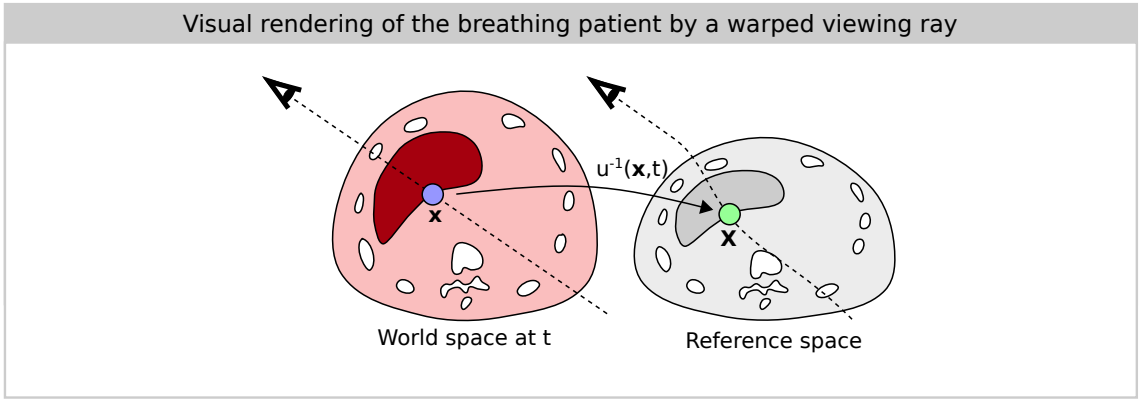

Figure 5.6.: Ray casting of the deformed state of the virtual patient can be interpreted as ray casting using a warped viewing ray in the undeformed state. The same can be applied to haptic rendering and the location of the haptic device.

required inversion, but real-time requirements prohibit to do this on the fly for each single rendering frame.

To face this problem, an approximate on-the-fly inversion of $u(\mathbf{X}, t)$ is performed along the sampling of the rays by applying an iterative scheme. Sampling along the ray is accompanied by iteratively updating the inverse, using an modified fixed point inversion scheme [CLC+08]. For rendering methods that do not include sampling along viewing rays, but instead samples on a regular grid as for example multiplanar reformations, the iterative fixed point scheme can be directly applied without modification with several iterations per point. Requiring an invertible displacement field, the method of [CLC+08] can be formulated so that iteratively evaluating the sequence $\mathbf{X}_n$=$\mathbf{x} - u(\mathbf{X}_{n-1})$ for a given position in world space gives a corresponding positions in material space. It is shown there, that this methods only needs a small number of iterations to reach a stable solution. For ray casting, iterative update of the sampling position along the viewing ray in world space and simultaneous update of the sampling position in the reference frame mimics the fixed-point approach along a viewing ray. Here, the fact that the change of displacement between sample points is generally small is employed. The iterative formulation of this along the ray can be written as

$$\mathbf{x}^+ = \mathbf{x} + \mathbf{d} \tag{5.4.1}$$

$$\mathbf{X}^+ = \mathbf{x}^+ - u(\mathbf{X} + \mathbf{d}, t) \tag{5.4.2}$$

with $\mathbf{d}$ being the direction vector of the ray and $\|\mathbf{d}\|$ the sample distance. This approach results in a quasi bended viewing ray and can be used for direct volume rendering, simulated X-ray and simulated ultrasound as presented in chapter 3.

5.4.2. Surface based Rendering

Rendering of surfaces that were reconstructed from the image data by for example the Marching Cubes algorithm is an alternative to direct rendering by ray casting. Introducing breathing motion to surface meshes that consist of vertices and polygons is straightforward by calculating the position in world space by $\mathbf{x} = \mathbf{X} + u(\mathbf{X}, t)$ of the vertices based on their position $\mathbf{X}$ in reference space.

5.4.3. Haptic Rendering

As for visual rendering, the haptic rendering methods of chapter 3 can be modified to include the breathing motion modeled by the displacement function $u(\mathbf{X}, t)$. In contrast, instead of modifying the sampling positions, the idea is to translate the position of the haptic device from world space to reference space using the displacement function. This can be illustrated by considering the haptic device position being close to the virtual patient's chest surface at maximum inhalation; in this case, the haptic device position in world space is close to the displaced surface in world space too. The same spacial relation in reference space, where the chest surface position is known but the haptic device position is not, can be enforced by setting $\mathbf{X} = \mathbf{x} + u^{-1}(\mathbf{X}, t)$.

Again, the inverse $u^{-1}(\mathbf{x}, t)$ would be needed to perform this step but can be substituted by $\mathbf{X}^{+} = \mathbf{x} - u(\mathbf{X}, t)$, i.e., a single inversion step in each iteration of the haptic update loop.

Forces computed by the haptic algorithms can then be send to the haptic device for output without having to translate back the resulting force vector to world space. This is possible since the force vector would only change under rotation inflicted by the displacement function. Theoretically, the haptic device rotation should be considered and translated to reference space as well. In practice, the rotation presented in the displacement function is very small for positions near the skin surface and thus rotational transformation can be neglected.

5.4.4. Modifications to Haptic Rendering of Needle Insertion

The previous considerations regarding the haptic rendering of ultrasound probing and for palpation simulation can be applied to the proxy positions of the needle, see Fig. 5.7. This suffices to introduce breathing motion into the needle simulation. For needle insertion, rotation induced by the displacement function at the needle tip causes different path taken by the needle and thus should also be considered.

To modify the algorithm to include the displacement at the tip, the needle tangent direction $\mathbf{d}_l$ used in Alg. 3.5, which handles the advancement of the needle, is modified by the

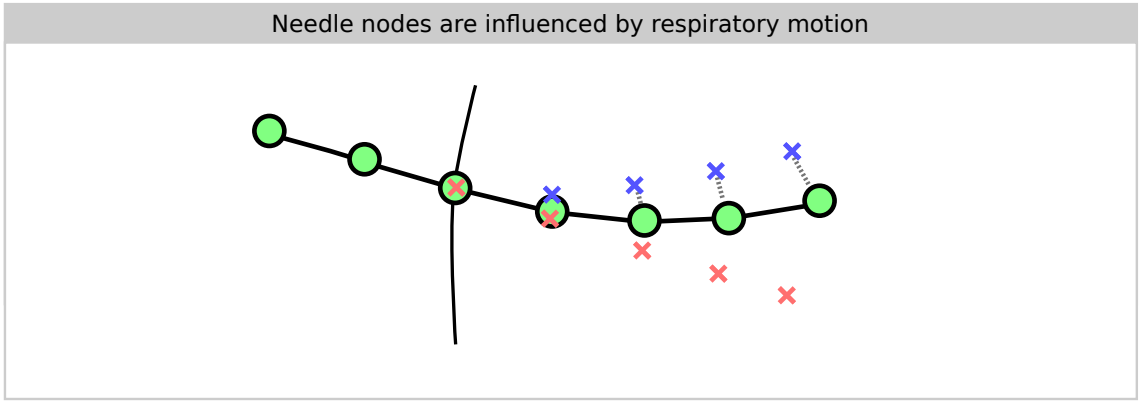

Figure 5.7.: The needle nodes representing the discretized needle are connected to proxies (red) by springs. These proxies are displaced by the breathing motion, resulting in adjusted proxy positions (blue) and thus to a bended needle and adjusted insertion direction.

following procedure. First, the deformation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ is computed based on the needle tip's proxy position $\mathbf{p}_l$:

$$\mathbf{R} = \partial u(\mathbf{p}_l, t)/\partial \mathbf{p}_l + \mathbf{I} \approx \begin{bmatrix} u(\mathbf{p}_l + (1,0,0)^\top, t)^\top - u(\mathbf{p}_l, t)^\top \\ u(\mathbf{p}_l + (0,1,0)^\top, t)^\top - u(\mathbf{p}_l, t)^\top \\ u(\mathbf{p}_l + (0,0,1)^\top, t)^\top - u(\mathbf{p}_l, t)^\top \end{bmatrix}^\top + \mathbf{I} \tag{5.4.3}$$

and then the transformed needle tangent is calculated by $\mathbf{d}_l = \mathbf{R}^{-1}\mathbf{r}_m / \left\|\mathbf{R}^{-1}\mathbf{r}_m\right\|$ using the undeformed needle tip direction vector $\mathbf{r}_m$ in world space coordinates.

5.5. Implementation

During run-time of the simulation, a rather large amount of data is accessed. Especially the ray casting visualization has to take into account nearly every element of the image data that is intersecting with viewing rays. In contrast to the the visualization, which computes 2D images at a low update rate with high parallelization, the haptic simulation has to access and evaluate the motion model with a very high rate only for a few elements. Therefore, a twofold implementation consisting of a GPU and CPU implementation had to be developed and model data has to be stored twice in both CPU and GPU memory.

5.5.1. Remarks on Visualization Implementation on the GPU

For both the key frame approaches and the linear model, each frame resp. linear component is stored as a CUDA 3D texture with four float components for each element (16 bytes). The

elements store the displacement field as x-, y- and z-component, the additional byte does not store information but is necessary to perform faster texture lookups. Before evaluation of $u(\mathbf{X}, t)$ takes place for the visualization component on the GPU, preliminary steps are performed on the CPU. For the key frame model, this consists of evaluation of the indexes for adjacent frames i and $i+1$ and the weighting factor $\alpha(\tau)$. These values are the only information that now has to be uploaded to GPU memory to be able to perform the visual rendering. Regarding the surrogate signal approach, it is only necessary to upload the surrogate signal, that is $g(t)$ and $g'(t)$.

It is worth noting that the linear model needs three texture look-ups to evaluate a single value of $u(\mathbf{X}, t)$, whereas for the key frame approaches it is sufficient to only perform two look-ups. However, for the latter, a larger number of textures (one for each phase) have to be held in memory in contrast to the three necessary ones for the linear components of Eq. 5.3.6.

Using the above, the visualization components can use $u(\mathbf{X}, t)$ and perform the necessary on-the-fly inversion and texture lookup for the actual image value or label in I and J resp. Apart from the ray casting, it is of course possible to deform triangular surface models using $u(\mathbf{X}, t)$ directly. For this, the vertices' positions of the triangles have to be stored in a pixel buffer object accessible from CUDA. After computation of their new positions in worldspace $\mathbf{x} = \mathbf{X} + u(\mathbf{X}, t)$ in parallel, these can be rendered by OpenGL.

5.5.2. Remarks on CPU Implementation for Haptics

Similarly to the GPU implementation, the CPU side relies on texture look-ups. In this component, the image data is stored as 3D VTK images. To save performance, the haptic algorithms first determine the positions for which a texture look-up have to be performed. These are the proxies for palpation and ultrasound probing resp. the needle node of the bending needle. Together, these are evaluated by VTKs image interpolation methods for the adjacent key frames resp. linear components.

5.6. Input data and Preparation of the Virtual Patient Model

For demonstration of feasibility of the approach, a single virtual patient was created from a low dose 4D CT data set. For the data set, spirometry data is available as a surrogate signal for each of the phases and also as continuous sequences, for an example see 5.4, that can be fed to the simulation. The image date consists of 14 phases and has a resolution of $512 \times 512 \times 460$ voxel elements. Using the full resolution and all phases is obviously

prohibited by the large amount of memory needed for it. Only taking into account the 14 phases at full resolution result in a memory demand of $512 \times 512 \times 460 \times 14 \times 16\,\text{bytes} \approx 25Gbyte$, which is clearly not available on consumer grade graphics hardware at time of writing.

To be able to fit the key frames resp. linear components into memory they have been reduced to sizes of 64^3, 128^3 and 256^3 voxels. The reference image was also resampled to a resolution of 256^3 elements.

For low dose 4D CT data, the stuctures inside the liver are hard to distinguish due to low contrast and noise. To overcome this limitation for a training scenario, the liver blood vessels and bile ducts have been transfered from a different patient image by warping segmentation masks of these structures onto the 4D CT reference phase. To introduce these artificial structures into the reference image, the image values of the CT data are adjusted to common values for liver blood vessels and bile ducts. Also, an artificial mock-up lesion is added to the image data by artistic means.

From the segmentations of the three structures, surface models have been created by using the Marching Cubes algorithm [LC87] followed by surface smoothing and decimation. Furthermore, threshold based segmentations of patient's skin surface and bone structures were created.

5.7. Experiments & Results

The applicability of the presented method is now first shown for visualization without tool interaction and afterwards results including haptic interaction are given.

Fig. 5.8 shows screenshots of a sagittal multiplaner reformation during respiratory motion. Both the half cycle approach and the linear model are used to visualize the phases EI, ME, EE and MI and also out-of-sample predictions. For computation of the deformed slice, the fixed-point inversion has been applied to each element of the grid. In Fig. 5.9, screenshots of the visualization methods ray casting together with triangular surface models, X-ray simulation and ultrasound simulation are shown.

In Fig. 5.10, the effect of the respiratory motion on the bendable needle is shown. Also, Fig. 5.11 shows a sequence of needle movements of a freely moving needle. For this, the user first inserted the virtual needle via the haptic device into the abdominal region of the patient model and then released the haptic device handle, showing that the force feedback loop is stable and the breathing motion induces a plausible needle movement.

For palpation, Fig. 5.12 shows how a finger indents the soft tissue of the virtual patient model, the color coded displacement field shows that both local deformations and global

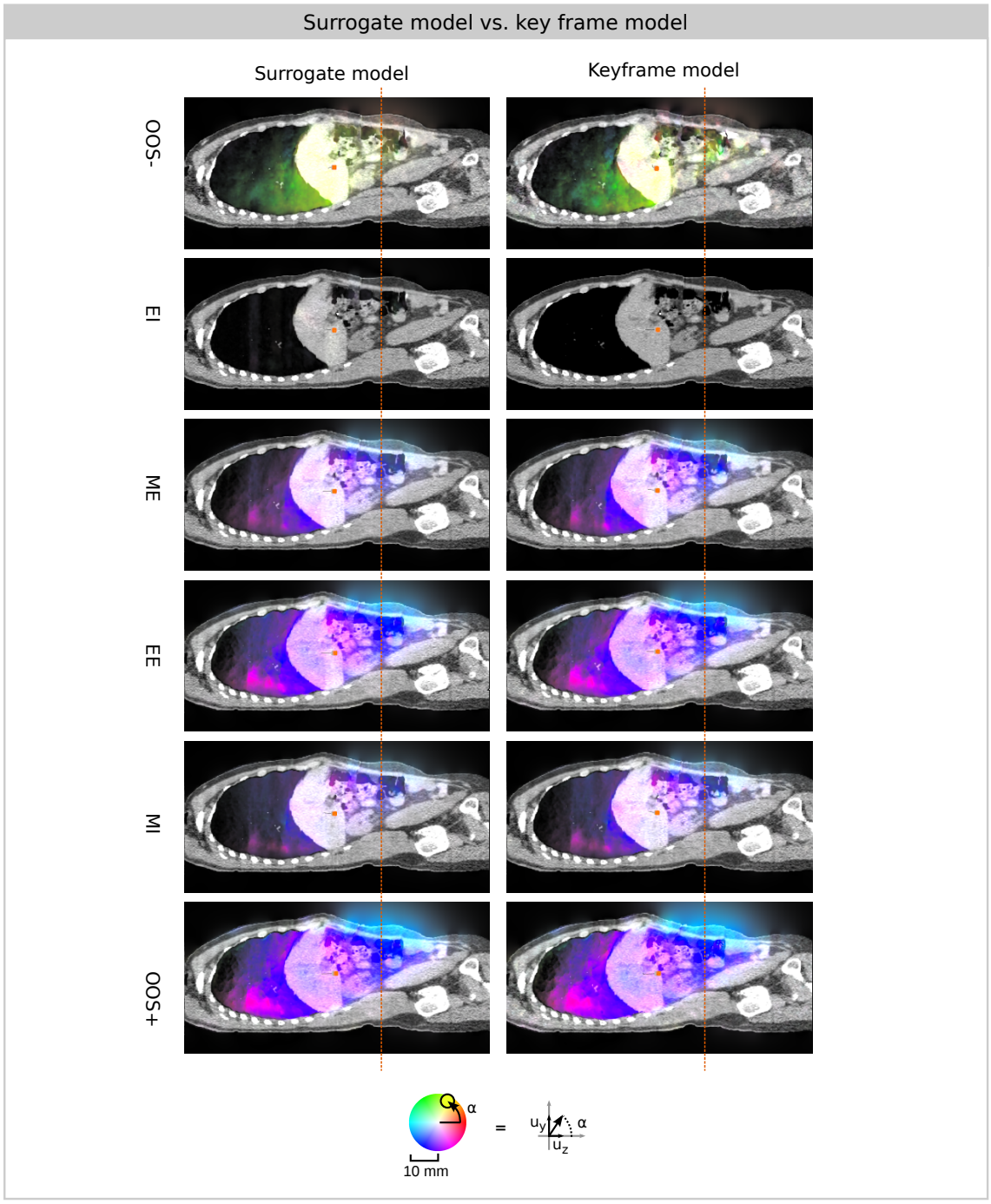

Figure 5.8.: Screenshots of sagittal slices of the virtual patient CT data under respiratory motion. Two models are shown (surrogate/linear approach vs. half cycle key frame approach). The displacements are shown using a color coding with color indicating the direction based on the HSV wheel and magnitude being indicated by opacity of the overlay. Maximal opacity of the color overlay corresponds to a displacement of 10 mm and above.

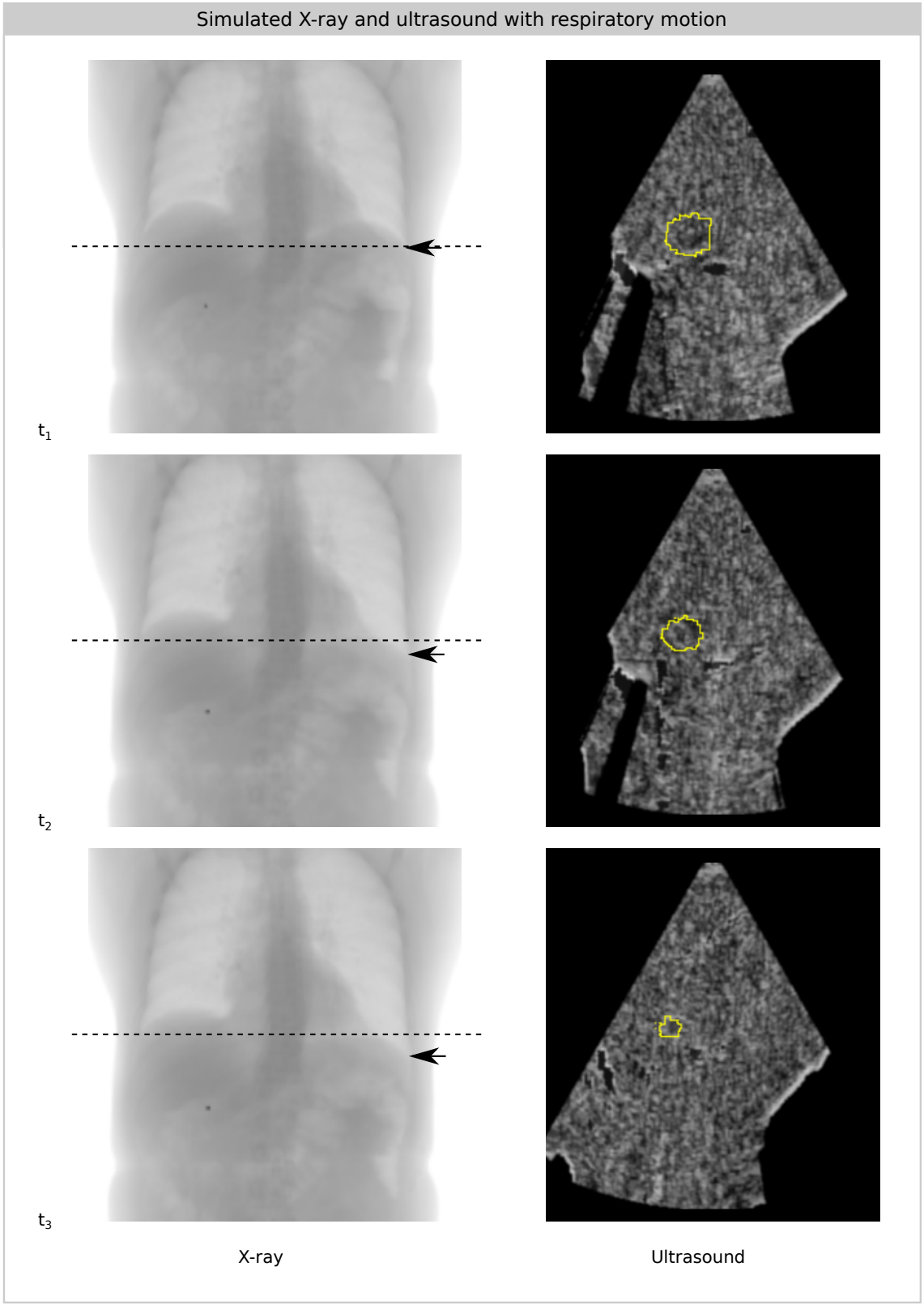

Figure 5.9.: Screenshots of simulated X-ray and ultrasound imaging influcenced by the respiratory motion for different surrogate signal values $g(t_1) = -48.1$, $g(t_2) = 491.4$ and $g(t_3) = 655.8$. Visualization of the movement is supported by arrows pointing to the base of the lung in the X-ray simulation and by delineation of the lesion segmentation in the ultrasound simulation. The ultrasound probe has been placed along an intercostal space as in Fig. 2.1.

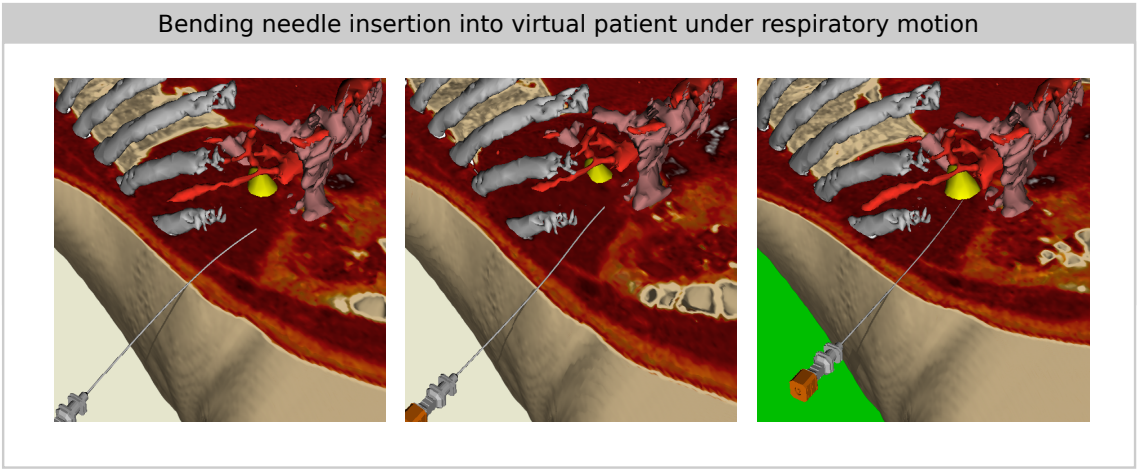

Figure 5.10.: Sequence of mixed surface and volume rendering with surface model of a lesion (yellow), bile ducts (light red), liver blood vessels (red) and bony ribs (white). In between the screenshot acquisitions, the needle was advanced deeper into the liver and finally into the lesion.

deformations from tool interaction and respiratory motion are taken into account at the same time. Discontinuities in the displacement fields as are arising at sliding tissue interfaces can lead to smearing artifacts (Fig. 5.13). These are depending on the resolution of the of the displacement fields [FWMH15] and especially visible at surface visualizations of the ribs.

5.8. Discussion

In this chapter it was shown that using 4D CT image data containing respiratory motion, it is possible to create a virtual reality simulation that includes the respiratory motion in both visualization and haptics. Its main advantage is that it is straight forward to integrate into the existing simulation components by employing the relation between world and reference space given by the displacement function.

Regarding the preparatory effort, it is necessary to segment structures with sliding boundaries for the 4D registration. Lung segmentation requires less effort in comparison to the liver in this regard due to the high contrast of Hounsfield values in this region. Liver segmentation is necessary for preparation of a new patient in any case for haptic interaction. A current limitation is the fact that 4D image sequences are necessary. Future work will investigate how to transform respiratory motion models to patient data for which only a single phase is available.

In [FWMH15], it is shown that the time needed for ray casting is affected by the ray warping, but is still within reasonable time frames. For a comparable setting, times for the ray casting without, with the key frame approach and with the linear model require ca. 16 ms,

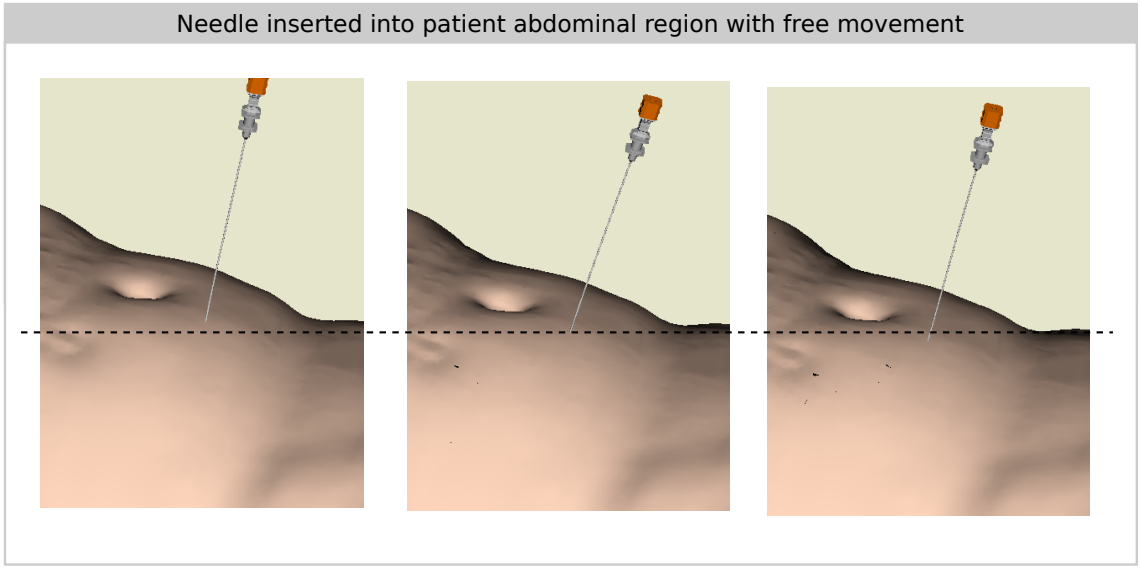

Figure 5.11.: The needle has been inserted into the abdominal region of the virtual patient by the haptic device. The user then released the haptic device handle, resulting in a freely moving needle influenced by the respiratory motion.

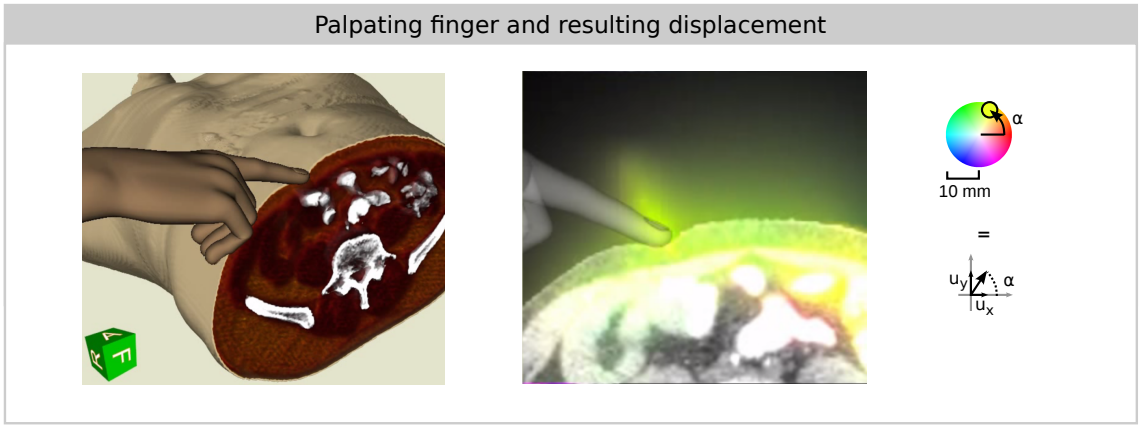

Figure 5.12.: Screenshot and MPR slice of tissue deformed by palpation together with respiratory motion.

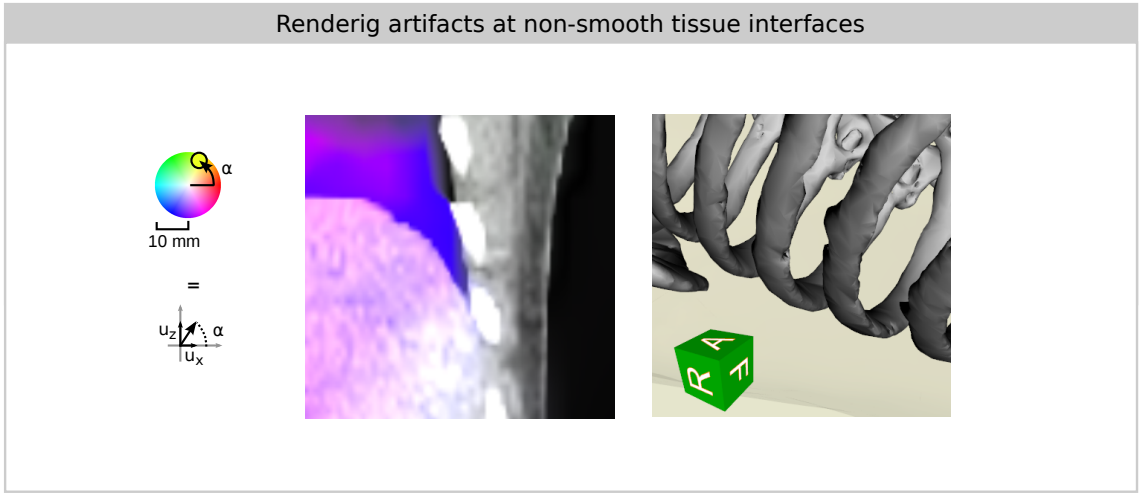

Figure 5.13.: The discontinuities at sliding interfaces can cause smearing artifacts.

37 ms and 27 ms resp. The big difference between the key frame approach and linear model can be explained by the fact, that the key frame approach has a much larger memory footprint overall.

The quality of the rendering is also highly dependent on displacement fields that are used as input. The methods developed so far for 4D image registration focus on good registration within the structures of interest. These are mainly the lungs, lesions within it or abdominal organs. Proper registration of structures such as skin or bone apart from the ribs are not subject to evaluation for these methods and so it is not guaranteed that the displacement fields computed by image registration is valid for all elements of the image. The smearing artifacts that can arise at the boundaries of sliding interfaces, i.e., large discontinuities in the displacement field are the major problem of the method. Especially, the interface between ribs and the lung are affected since the CT values are of high contrast in this interface region due to the high Hounsfield values of bone and the low values of air contained in the lungs. It is arguable that these rendering artifacts mainly occur at locations that are not in the target region of the intervention and higher resolutions of the displacement fields might reduce the problem in the future.

The effect of the respiratory motion on the haptics is well perceivable and stable. Having inserted the needle into the patient, tilting and movement of the needle can be seen and felt. Further evaluation of this effect might be necessary but challenging to perform in vivo. For virtual palpation, the effect is also perceivable and can be compared easily to user experience in the real world.

6. A Framework for Image-based Puncture Simulation

The methods from the previous chapters, namely direct visual and haptic volume rendering, rendering of local deformations caused by tool interaction and visual and haptic rendering of breathing motion are integrated in a single framework. For this, first a method to create virtual patient models using only partially segmented image data is presented and second the overall integration of hardware and software components is detailed.

The contribution of using partially segmented data was previously presented in [MFH12] for lumbar puncture and detailed in [FMSH16] for PTCD application. The second part of this chapter is also partly presented in [FMSH16], but here a more in depths description of used hardware and the developed software components is given.

6.1. Puncture Atlases with Partially Segmented Data

In chapter 3, the presented haptic and visual rendering algorithms rely on material properties of the virtual patient. These material properties are assumed to vary for different tissues and include haptic parameters for stiffness, cutting resistance force, friction force and also a RGBA-color tuple to indicate the tissue's color. Abstractly, they have been defined as functions that yield the properties for given positions in the reference space of the virtual patient. Thus, it is apparent to use the image data and segmentation mask as the basis for these functions. Apart from the image data, each virtual patient is also represented by a property tree in the framework. This tree consists of a hierarchical description of the patient's tissue classes. Based on the image data, a (partial) segmentation of tissues and the property tree, the material parameters are estimated at run-time. These three components in combination will be called a puncture atlas and the following will describe its creation and usage at run-time of the simulation.

6.1.1. Creation of Partially Segmented Data

The predecessor of the framework was depending on fully segmented patient image data for haptic rendering [Fär09]. Based on organ segmentations, the parameters for the corresponding organs were used in the rendering. Also for the visual rendering, these segmentations were needed to create triangular surface meshes by the marching cubes algorithm [LC87]. The practical limitation of this approach is the high amount of time needed for the creation of the segmentation masks. In [Dal14], these are given to be more than 60 hours for a single patient data set in a lumbar puncture scenario (see Introduction, p. 6). In an application scenario, where a patient model is needed in a short time frame, i.e. planning or pre-operative training, this is a major impediment.

To mitigate the problem, the idea is to reduce this time by only having to segment a subset of organs and, if possible, perform this segmentation with suitable supporting algorithms. For PTCD, the essential structures that have been identified to be of major importance are the liver, liver blood vessels, bile ducts and the intercostal fascia. Especially liver blood vessels and the bile ducts are risk and target structures resp. and thus have to be modeled with high precision to be able to detect harm of risk structures and successfully insertion of the needle into the bile ducts. These key structures have been segmented using Multi-atlas segmentation strategies [MFM$^+$13, BMFH14] for the liver and vesselness based techniques [BMF$^+$14]. Also, for each patient the intercostal fascia have been modeled and the resulting surface models have been converted to segmentation masks. For the fascia, this completely manual approach is necessary due to the fact that the fascia are not distinguishable from surrounding soft tissue.

Similar to the labeling function l used in the previous chapters, a partial segmentation function $p(\mathbf{x}) : \mathbb{R}^3 \rightarrow L$ can be defined based on the segmentation masks created for the key structures. These structures are listed in the Appendix A.2. Of course, other puncture scenarios than PTCD will rely on a different set of partial segmentations and partial segmentation function.

6.1.2. Label Estimation Heuristic

The partial segmentation function is fused together with a heuristic to define the labeling function l used in the PTCD framework. Apart from the essential key structures that need a segmentation mask, other structures on the needle insertion path are skin, fatty tissue and bones resp. ribs, which can be classified via threshold heuristics during run-time. Thresholds that delimit these classes, namely $t_{\text{air}}^{\text{skin}}$, $t_{\text{skin}}^{\text{fat}}$ and $t_{\text{fat}}^{\text{bone}}$, have to be estimated for this based on the image data in a first step. In A.2, labels and associates structures and Hounsfield

ranges are listed exemplary for a single patient. Generally, these thresholds have to be either re-estimated for each new image data set or new image data has to be manipulated in a way that these thresholds are valid for this new data set as well.

Given the thresholds, a heuristic that also incorporates the current needle insertion depth d can be summarized by

$$h(\mathbf{x}, d) = \begin{cases} l_{\text{risk}} & \text{if } I(\mathbf{x}) \in [-\infty, t_{\text{skin}}^{\text{fat}}) \wedge d > \delta \\ l_{\text{skin}} & \text{if } I(\mathbf{x}) \in [t_{\text{air}}^{\text{skin}}, t_{\text{skin}}^{\text{fat}}) \wedge d \leq \delta \\ l_{\text{fat}} & \text{if } I(\mathbf{x}) \in [t_{\text{skin}}^{\text{fat}}, t_{\text{fat}}^{\text{bone}}) \\ l_{\text{bone}} & \text{if } I(\mathbf{x}) \in [t_{\text{fat}}^{\text{bone}}, \infty] \\ l_{\text{air}} & \text{otherwise} \end{cases} \tag{6.1.1}$$

with δ being a threshold that is used for detection of risk structures after the needle has been inserted into the patient model. This heuristic checks if the image value $I(\mathbf{x})$ is in the the range defined by the thresholds and also checks for low image values within the virtual patient that are either cavities or other risk structures as lung or intestines, which also have low intensities in the CT image data.

A combination of the heuristic $h(\mathbf{x}, d)$ and the partial segmentations then is

$$l(\mathbf{x}, d) = \begin{cases} p(\mathbf{x}) & \text{if } p(\mathbf{x}) \neq 0 \\ h(\mathbf{x}, d) & \text{otherwise} \end{cases} \tag{6.1.2}$$

and can now be used as the labeling function for the needle insertion and other haptic algorithms. It is cheap to evaluate and thus very suitable for haptic rendering, which relies on update rates between 500 to 2000 Hz. For visual rendering, an insertion depths is not given for each rendering element, making it reasonable to either set $d = 0$ or use the insertion depths given by the needle insertion algorithm. The latter changes the visualization results during simulation in dependence of the current insertion depths, which is beneficial for inspection of the results of classification but is distracting otherwise.

6.1.3. Property Tree

All tissue classes represented in a puncture scenario and contained in the puncture atlas are part of hierarchical representation of tissues, which is called the property tree. Each leaf in the tree represents a tissue class (liver, bile ducts, bone etc.) and to each of the leafs a unique label from the set L can be assigned. Furthermore, to each inner node a group of tissues (soft tissue, vessels, hard tissue etc.) can be assigned. The purpose of this tree-like structure

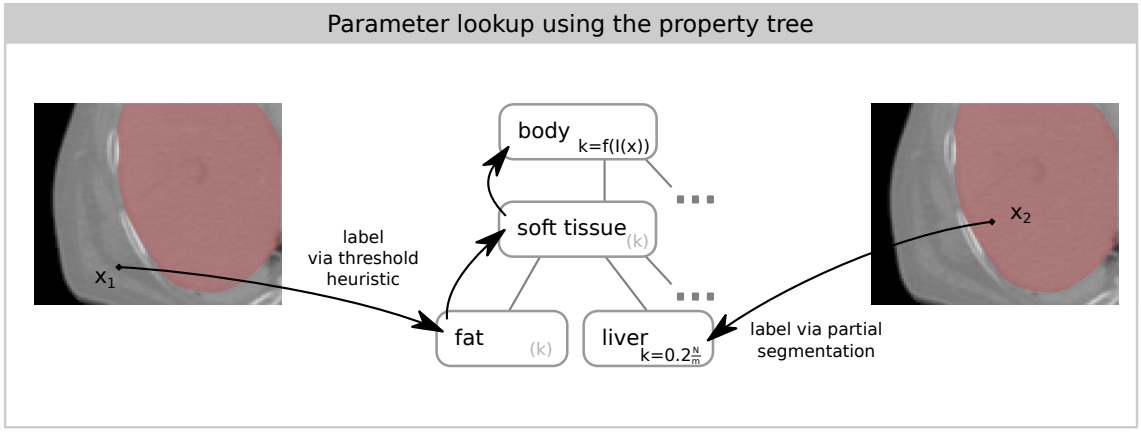

Figure 6.1.: Parameter estimation for a location in the image data first determines the appropriate tree node by applying the label function. Then, the value definition is looked up in the tree. Here, this is demonstrated for the stiffness k of the locations x_1 and x_2. For x_1, no partial segmentation is available, making it necessary to use the heuristic. Also, the property tree does not contain a definition directly for the structure, but for its parent element, which contains a transfer function definition of the value. For x_2, a segmentation is available and the definition is directly included in the tree node as a fixed value.

is to be able to define the same properties for groups of tissue and also to be able to refine properties of different members of the same structure. This is done by inheritance: For each node, properties are either defined or are inherited from the closest ancestor in the tree.

Apart from defining the tissue parameters used by the haptic algorithms as fixed values for each structure, it is also possible to define them as transfer functions that relate image intensity to the appropriate parameter value. For example, if a linear relation between image intensity and the stiffness of a tissue with label $j \in L$ exists, a linear function $f_k^j(v) = k_j \cdot v + b$ could be assigned to a tree node instead of a scalar value. To be able to approximate any function, a piecewise linear function can be specified for each tissue parameter.

6.1.4. Parameter Evaluation

For evaluation of a parameter for a given location $\mathbf{x}$, four steps are performed as illustrated in Fig. 6.1: (1) First, estimate the label $l_\mathbf{x} = l(\mathbf{x}, d)$ of $\mathbf{x}$ by using the label function from the previous section. (2) Then, find the tree node with the matching label. (3) If the parameter is present in the node as a scalar value, the evaluation terminates and returns this value. If it is a transfer function, it is evaluated for $I(\mathbf{x})$ and this value is returned. (4) In case the node does not contain a definition, the evaluation now considers the parent element of the node to contain a definition and jumps back to step (3). This way, the parameters used by the haptic algorithms are given for each location in the reference space.

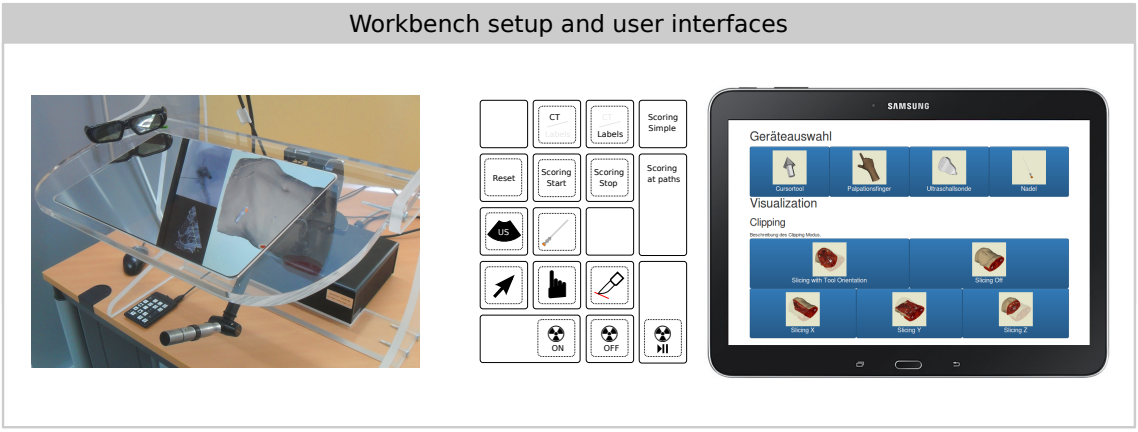

Figure 6.2.: Left: Haptic workbench running the PTCD simulation. Middle: Numpad interface. Right: Tablet PC interface to the simulator.

6.2. Haptic Workbench Hardware-Setup

The complete framework consists of both hardware and software components. For haptic interaction, the simulation frameworks supports the Geomagic Touch and Phantom Premium haptic devices (formerly Sensable Phantom Omni and Phantom Premium). Depending on the operation system on which the framework is executed, it is possible to connect up to two physical devices. This is limited by the driver support for Linux, which only supports a single Touch device. Using Windows, it is possible to either use a single Touch and a Phantom Premium or two Touch devices.

To display the rendered visual content, a haptic workbench (see Fig. 6.2) can be used. It consists of a haptic device and a LCD monitor that is capable of stereoscopic 3D and uses a semitransparent mirror to show the content of the overhead mounted monitor.

6.2.1. Workbench Constraints

This setup imposes several design constraints onto the framework. First, by mounting the monitor upside down, the resulting viewport is rotated by 180 degrees and is flipped by the mirror, resulting in an upside down configuration of the output image. In the preceding framework AcusVR [Fär09], this was dealt by inverting the scan direction of the cathode ray of the monitor by physical rewiring, which is not possible for LCD monitors. In fact, the used graphics hardware driver provided by Nvidia does support rotation of the screen, but not mirroring it, which thus makes it necessary to perform the mirroring using software. This is integrated into the framework by applying a final rendering pass in the VTK rendering pipeline that copies and mirrors the resulting output window. Performing this step is limited to the OpenGL output window and thus it is not possible to use the previously used

Qt graphical user interface (GUI) on the workbench screen. A minor issue that arises with the inverted display is that mouse interaction is also inverted. Using a tool like Autohotkey[1] on a workstation running Windows, the vertical mouse movement can be inverted in case the cursor is on the primary display.

Second, the semitransparent mirror rotates the plane of polarization of the light emitted by the screen. For stereoscopic 3D, the workbench uses liquid crystal shutter glasses. To compensate the change of polarization caused by the mirror, the manufacturer of the workbench modified the shutter glasses by rotating the lens by 90 degree. In short, this prevents the shutter glasses to be worn by a user when looking on the secondary LCD monitor of the workbench. Since it is very inconvenient to put on and remove the glasses each time the secondary monitor is needed, it is unreasonable to interface with the simulator using a graphical user interface displayed on the secondary monitor. These constraints made it necessary to come up with alternative interaction methods.

6.2.2. User Interface

To overcome the limitations imposed by workbench setup and the fact that simultaneous interaction with the haptic device and the computer mouse is inconvenient since both would be performed by using the user's dominant hand, two alternatives shown in Fig. 6.2 are provided. The first is an external number pad connect over USB: Each key can be used to issue a certain command for interaction to the simulator, which is reflected in the pictograms that have been applied to the keys. By this means, fast and robust interaction is possible. It can be used to change the current haptic device, enable and disable the X-ray simulation and change the display mode.

To provide an interface that is not limited by the amount of keys, an additional tablet PC can be used. For this purpose, the simulation framework contains a web server to which a browser can connect using the hypertext transfer protocol (HTTP). The interface provides large buttons to change the tools and visualization modes of the simulator. The motivation is that the user is able to simultaneously steer the virtual tools using the dominant hand and at the same time change the simulator settings with the other hand. In comparison, the tablet PC interaction mode can provide an interactive description of each of the steps that have to be performed for a successful virtual intervention. From a developer perspective, this mode is beneficial because it allows alterations of the interface at run-time of the simulator.

Furthermore, the framework uses the existing methods for scoring from [FHGH09]: The user can choose between free puncture and puncture along predefined insertion paths and

[1] http://www.autohotkey.com/

the results are shown on the secondary screen using the Qt framework. Future iterations of the framework should include the results in the browser-based graphical user interface. Also, it is possible for the user to select one of three different difficulty settings: beginner, advanced and expert. This selection influences the settings and availability of supporting visualization modes.

6.3. Software Architecture

The software parts of the framework are written in the languages C++ and CUDA and are subdivided into several sub-modules. Fig. 6.3 illustrates the flow of information for the major modules. These modules are decoupled as much as possible and and are layered without cyclic dependencies, see Fig. 6.4. The main dependencies of the framework are the VisualizationToolkit (VTK) for loading and processing of medical image data and OpenHaptics for interfacing the Geomagic haptic devices. Each of the components has the following responsibilities:

- HapticDevice: This is an abstraction layer for the hardware haptic devices and the only modul that interfaces the OpenHaptics library. Also, it provides a virtual dummy haptic device that can be steered by a scripted sequence or by user interaction using a GUI. The scripts are loaded from special script files and can be used for demonstration of the simulator and testing, or for path steering for evaluation of massive numbers of path generated by path planning concepts.
- VirtualPatient: This module contains a class that holds information of a virtual patient atlas, i.e. image data, segmentations, distance maps, classification heuristics, property tree and motion models. Also it consists of auxiliary classes for loading the atlas data from the hard disk using VTK and the XML library libxml2. To validate the data, it uses a Document Type Definition (DTD) ensuring syntactically correct input.
- ImageDeformation: This library reflects the methods presented in chapter 4. It includes classes that represent the image data and the deformable subimage and a class to apply the deformation models implemented in CUDA.
- HapticAlgorithm: The haptic algorithms for needle insertion, palpation and ultrasound probing presented in chapter 3 are included in this module.
- VolumeRendering: Visual volume rendering methods from chapter 3 are implemented in this library. It provides methods for rendering deformed volume data by ray casting, ultrasound and X-ray simulation and multiplanar reformations. Also, it includes rendering using the motion models presented in chapter 5. To conform to the VTK vi-

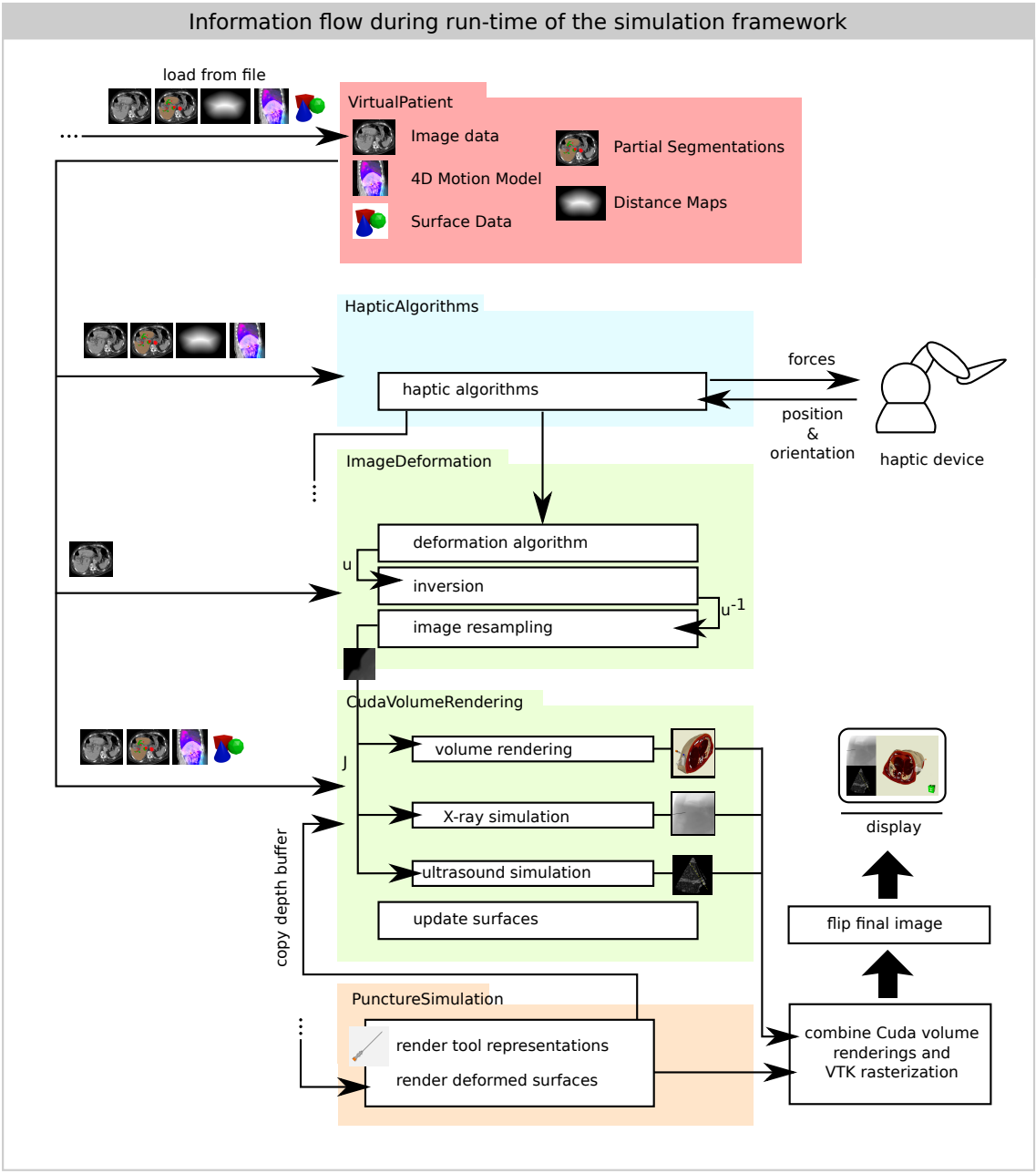

Figure 6.3.: Data flow during simulation. The user interacts with the haptic device, the resulting visualization is displayed on the screen. Haptic device and haptic algorithm form a force feedback loop and information from the haptic algorithms is given to the image deformation algorithms. Based on the the deformed image, visual rendering is performed. This includes the combination of ray-casted patient data and surface renderings.

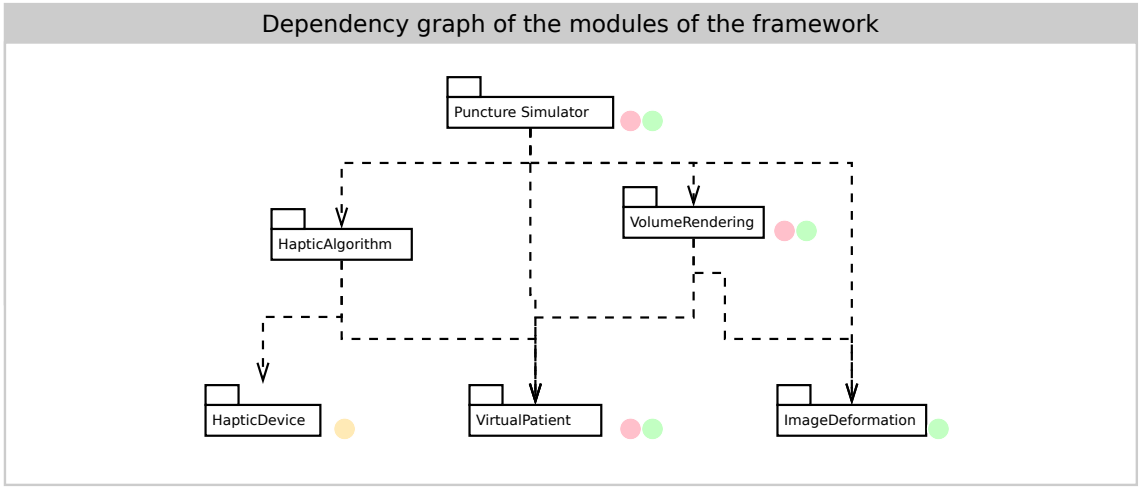

Figure 6.4.: Internal dependencies of the modules of the framework. Relevant external dependencies of each module are indicated by a green (CUDA), red (VTK) and orange (OpenHaptics) dot.

sualization pipeline, for each of the methods, a VTK actor is provided that renders the results computed by code written in CUDA using CUDA/OpenGL interoperability.

- PunctureSimulator: This module integrates all before mentioned modules and also is responsible for loading of the simulated scenario (available tools, visualization methods, predefined puncture paths). It connects the data model of the virtual patients to haptic and visual algorithms. It provides the user interface implemented using Qt, manages the display of data in various viewports (ultrasound, X-ray, main viewport, multiplanar reformations, see Fig. X). For each availabe tool, a separate visual represenation class and controller class exists to decouple model, view and controller (MVC architecture). Furthermore, it provides the web server for accessing the simulator using the tablet PC and a event pipeline.

The event pipeline in the last module is worth a detailed explanation: An event generally is a request to change the settings of a component of the simulation or to issue a command as saving of the current state of an algorithm into a file. For simplicity reasons, it is a string that contains both the name of the event and parameters. To ease development, events can be insertion into the pipeline by various interfaces. The first is a developer command console included in the Qt GUI, into which the event can be typed directly. Secondly, it is possible to insert an event by the scripted dummy haptic device. Last, the web server component can issue event processing in case a HTTP request containing an event is send by e.g. the browser of the tablet PC. As shown in Fig. 6.5, these events are passed along the components of the simulator, each having the possibility to react to the event. By this means, it is easily possible to develop new algorithms that can be configured at run-time of the simulation. It

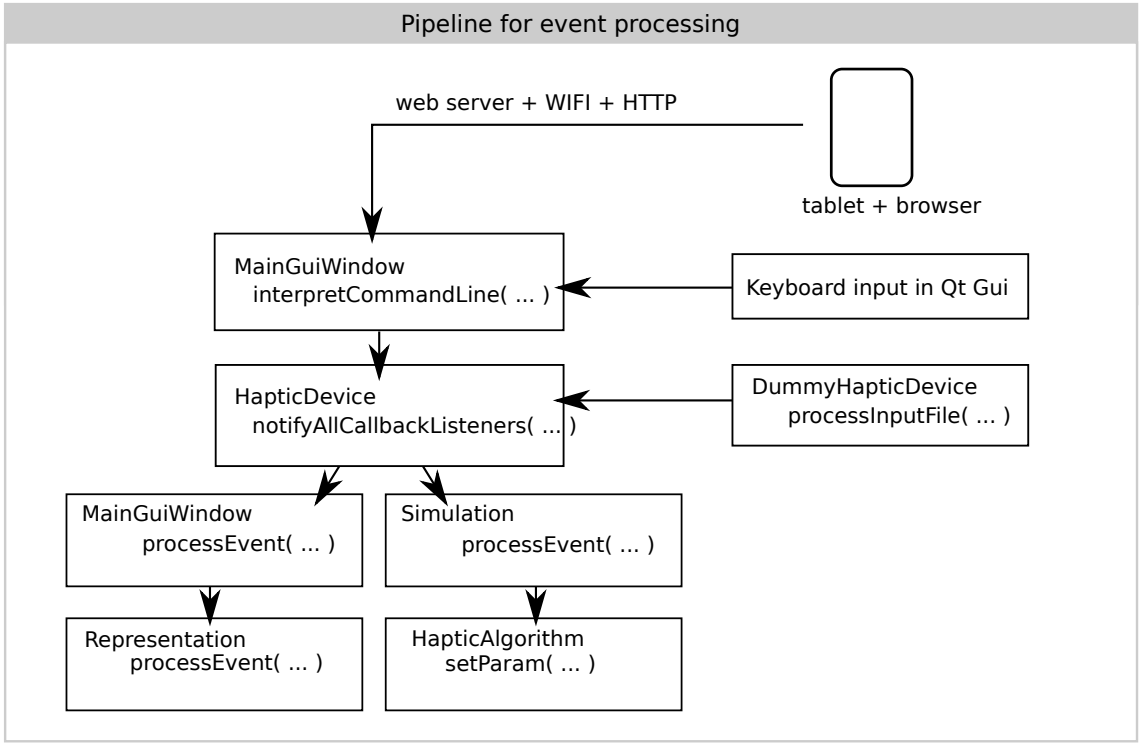

Figure 6.5.: Event pipeline for sending messages from the user interface or the dummy haptic device to the simulator components.

is possible to perform this by using the command console for a developer without having to create a GUI. Additionally, it is possible to create a suitable user interface using the web server, also at run-time.

6.4. Discussion

This chapter integrates the methods and components of the previous chapters into a single framework and details the labeling function. Using this labeling function with the proposed heuristics, it is possible limit the segmentation to key structures. For the PTCD intervention, these are liver, liver vessels (blood vessels, biliary ducts, RHD), fascia and the lung in case respiratory motion is demanded. The segmentation of the small key stuctures can be performed in several hours. With an optimized segmentation framework, the preparation of a new patient should be possible within half a working day.

Evaluation of the resulting labeling function is very fast to compute and did not noticeably affect the haptic algorithms in comparison to using a full segmentation.

An intermediate version of the simulation framework without the 4D component was evaluated by a user study in [FMSH16], which verified acceptance by the focus group of medical students. Haptic parameters were transfered from the predecessor ACUS-VR and

confirmed by two medical professionals. However, ongoing evaluation of the framework during further development is a major concern. The dummy haptic device that is used in the simulation framework uses a simple scripting language to steer a virtual haptic device and issue events into the messaging pipeline. It can be used to perform a reproducible puncture, which is not possible in case a real haptic device has to be steered by a human operator. To be able to perform such a reproducible puncture without having to execute the full simulator, the framework contains a needle path evaluation utility, which is a simplified command line utility that executes the scripted sequence faster than real-time. Scripted insertion paths can be generated by the output of a path planning algorithm [MHFH13, MHFH14], which can be converted to sequence of movement commands for the dummy haptic device. Using this tools provides a faster evaluation of the insertion sequences as would be possible by the full visuo-haptic simulator. To fully employ this feature, ground truth based on real tissue or phantoms is necessary.

The overall design of the architecture of the framework enables the replacement of components. In case new haptic algorithms are to be developed, these can be integrated into the respective libraries or new haptic devices can be integrated without having to rewrite the framework.

7. Summary & Discussion

In the following chapter, the presented methods are summarized and their limitations are discussed. Regarding the visuo-haptic rendering, it can be stated that the ray casting based volume rendering is very appropriate technique for the application in a training simulator. It has been shown that without having to create surface models of internal tissue and organ structures, it is possible to visualize the needle insertion procedure with high detail. In contrast to surface based, i.e. , indirect volume rendering, approaches, no information is lost when performing direct volume rendering. Indirect volume rendering has the disadvantage that image data is reduced to the interfaces between the tissues and everything in-between is discarded. However, the procedure used in the presented framework relies on a 1D transfer function for the relation between Hounsfield values in the CT image data and color information. Usage of 2D transfer functions that relate Hounsfield values and the local gradient magnitude of image elements to color values are available in the literature. First attempts at implementing higher dimensional transfer functions were promising.

The ray casting algorithm has been implemented in CUDA. This has been proven to be a reasonable approach concerning the successful parallelization. However, for the successful integration into the VTK or more specifically the OpenGL rendering pipeline, it was necessary to integrate an intermediate rendering to a frame buffer object to be able to copy the OpenGL depth buffer into memory available to CUDA. For a final product complexity might be reduced and rendering performance might be increased by relying on a GLSL based ray casting procedure as available in VTK. This also might be faster since CUDA does not support hardware accelerated computations of texture normals which is essential for the fast computation of lighting models.

The rendering of contrast agent in the X-ray simulation component is currently limited regarding the size of the region the diffusion process is applied to. In the case that a large region or the complete biliary tract is to be visualized this could cause unrealistic results at the borders. Normally, for the PTCD simulation where the main concern is the checking of successful puncturing of the bile ducts, this is acceptable. For other puncture scenarios the size of the region might have to be increased. Furthermore, in the implementation only a diffusive process in the bile ducts is integrated. Of course injection of the contrast agent into

a vessel creates a fluid flow that could be simulated. For blood vessels it would also be more realistic to simulate blood flow and propagation of the contrast agent. The simulation of blood flow only considers the flow velocity in the ultrasound Doppler simulation and does not incorporate the direction of blood flow. Regardless of this, the purpose of visualization of risk and target structures in the ultrasound component is achieved.

A successful virtual palpation of intercostal spaces is possible using the presented multi-proxy algorithm. Yet it is only applicable for this purpose. Palpation of the liver itself, which is an often performed diagnostic tool, is not included. For PTCD puncture this is not a part of the intervention, but nevertheless, the algorithm has the potential to include additional force terms similar to the currently included structure force by creation of distance maps for additional structures such as the liver and including them into the force term.

The simulator relies on a two step procedure in which the ultrasound probe is first used to find a proper insertion site and then the user switches to the virtual needle and performs the puncture. This procedure seems not to be intuitive for new users of the system and further development should focus on using separate devices for ultrasound probing and needle insertion each. At the same time it might be of value to also include the guide wire and drainage insertion procedure into the training simulation.

For simulation of deformations, the framework relies on the reduction of computation in a small region around needle, ultrasound probe or palpating finger. It is assumed that the main deformations for the needle occur at the tip when puncturing organ capsules or skin. For long needles, large deformations and deformations arising from the breathing motion of a patient it will be necessary to compute deformations in a larger region. Additionally, the models used for computation of deformations are of course simplifications of reality. The diffusive process used can be regarded as a plausible visualization, but closer inspection reveals that the linear-elastic model has to be applied to achieve physically founded behavior. A fully realistic simulation would also have to include anisotropy of the materials and more importantly unique material parameters for distinct tissues. In the context of this thesis the behavior of the simulated tissue has been compared to finite element simulations using non-linear materials, which is an in-silico evaluation only. It has been restricted to in-silico evaluation since it ethically unproblematic and can be performed without additional extensive experimental setups.

A central contribution of this thesis is the visuo-haptic rendering of a breathing virtual patient based on a respiratory motion sequence. As shown in chapter 5, this approach works well for a virtual patient for which both a static reference CT image and a reconstructed sequence is available. For patients for whom a PTCD is indicated such a sequence will not

normally be available and it might be unreasonable to expose the patient to the radiation dose associated with capturing a 4D image sequence. Therefore, methods for estimating a respiratory model only based on a static reference image and further easy to measure information (e.g. surrogate signals) are needed. Generally, the method shows rendering artifacts around sliding interfaces as the region between lung and rib cage, which can be attributed to the used linear interpolation. Higher resolution of the displacement fields or other interpolation methods could mitigate this problem. The resulting forces of the algorithm do not have been compared to in-vivo ground truth. It is assumed that the consideration of displacement is enough to produce realistic force feedback. This assumption neglects changes of material properties under deformation and so further experiments could verify the developed methods in this regard. For the non-invasive palpation, experiments with force sensors could be performed without concern. However, the acquisition of in-vivo needle force measurements, e.g. those published in [MMvV+12], is complex and ethically problematic.

For generation the of new virtual patients models for usage in the framework, methods have been published in [MFM+13, BMF+14, BMFH14] for segmentation of key structures. Apart from the steps presented there, a manual process is needed for the modeling of intercostal fascia, which are nearly not visible in the CT image data. Altogether, the steps needed for creation of a new patient can be performed in a reasonable time. A detailed description of these and their time demand is planned to be published in the future.

The user study with medical students [FMSH16] gave good hints for further development. This study consisted of 16 participants and supports the utility of the framework. Nonetheless, a thorough evaluation with medical experts could further support the results and could be used to identify further possible refinements.

Overall, the methods presented in this thesis are tailored for PTCD needle puncture but could be applied to other insertion scenarios as well. Regarding surgery scenarios that include cutting operations the presented methods are not be applicable without further extension. Cutting introduces changes of the topology of the virtual patient's body. By remeshing operations mesh based approaches can introduce cuts into the patients body, which is not possible when using a fixed regular grid. In the section on future work an idea to approach this challenge is proposed.

To sum up, a novel simulation framework has been presented that does not depend on surface or volumetric meshes. Specifically, methods for haptic and visual rendering have been developed for this that include up-to-date methods such as soft tissue deformation and respiratory motion. From these results further interesting research topics arise for which in the next chapter possible solutions are presented as future work.

8. Outlook & Conclusion

Based on the discussion from the previous chapter, future work for advancing the framework is suggested in the following.

8.1. Future Work

More realistic rendering A central aspect of the presented framework is the visual representation of the virtual patient by using volume rendering. From an engineering perspective, it might be interesting to integrate the latest technological developments in volume rendering, for example fast visibility encoding [KJL+11], to create a more realistic volume rendering. This approach enables self shadowing of volumetric image data by storing local visibility using a Spherical Harmonics approach. However, it is not clear if such methods are applicable for volume rendering that contains local deformations and global displacements caused by respiratory motion.

Cutting simulation To make the simulation framework applicable for additional surgery scenarios cutting should be included. In other simulations this has been performed for finite element simulation [WWD14][LZW14] or meshless approaches [JJM+14]. For a finite differences approach, cutting of tissue could be represented by a displacement field embedded in more than three dimensions. Topological changes and rupture that would introduce non-continuity of the displacement field could then be represented by keeping the field continuous while translating parts of the tissue around cuts not only in 3D but also in an extra dimension. In Fig. 8.1 this concept is illustrated for 2D. Figuratively speaking, the extra dimension acts as a "wormhole" along the cut. This approach might also benefit from the fact that it is more efficient for computation when using CUDA to store the displacement field using four components.

Better performance Scientifically more interesting will be the investigation of further developments of the finite element based deformation algorithms. Even if parallels to methods from image registration exist it is much different due to the locality of deformations, which only appear in vicinity to the interaction site of virtual tools. On the one hand, the efficiency

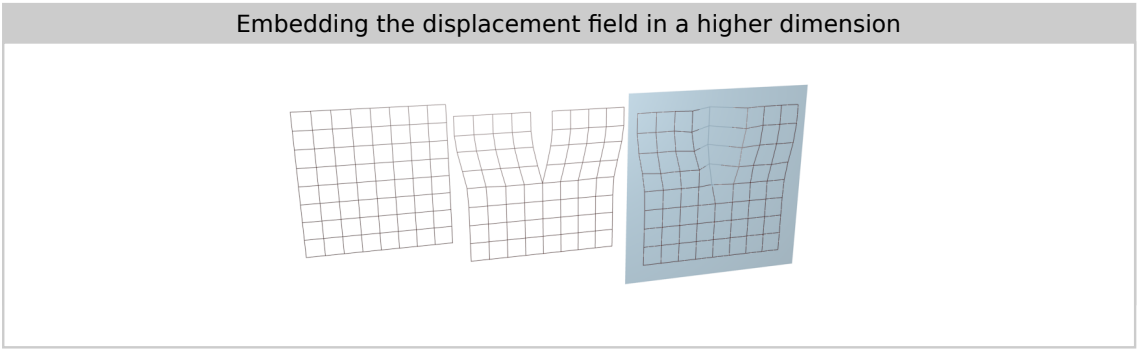

Figure 8.1.: Left: Undeformed mesh. Middle: Mesh with applied cutting, a discontinuity arises. Right:Embedding the 2D displacement field into 3D and neglecting elements with a z-value greater than zero approximates cutting.

of computation will be in the focus of further development. New generations of graphics hardware and the CUDA programming language provide new features that could be utilized. The most interesting hardware development might be dynamic parallelism, that is the launch of additional kernel calls from within the running execution on the graphics hardware [Nvi12b]. This way locally adaptive strategies could be developed that only perform computations on the grid where they are needed.

Due to the possibilities of highly parallel computation, real-time fluid simulation currently attracts lots of attention. Developments in this field might be applicable to soft tissue deformation algorithms as well. Similarly to the tall grid cell model presented in [CM11] for the simulation of fluid dynamics, the size of the grid could be dynamically adjusted to both enable fast deformation computation and overcome the limitation of a region of interest with a fixed size. In [JSP+15] a combination of fluid simulation and machine learning methods (Regression Forests) has been presented.

Improved deformation simulation Apart from improving performance, future work will include the development of deformation algorithms that include varying material parameters for each image element based on image modalities other than CT imaging. Magnetic resonance elastography (MRE) of the liver is already used for noninvasive assessment of liver diseases [VYE13]. This imaging method relies on the generation of mechanical shear waves within the soft tissue and imaging the wave by magnetic resonance imaging, giving the stiffness of the tissue that is visible in the MRI image data. Using a 2D MRE image and a registered CT image, the relation between CT Hounsfield units and the tissue stiffness could be analyzed and the resulting relation could be integrated into the linear-elastic finite difference formulation. Given a 3D MRE image, it would be possible to use the information in the haptic algorithms as well as in the computation of deformation. It will be necessary to adapt

the methods for soft-body simulation to include individual material parameters, which has been previously presented for image registration in [Kab06].

Parameters from multi-dimensional transfer functions It will be interesting to further investigate methods to derive material properties from a combination of (multi modal) image features. Using graphics hardware with a high amount of available memory, several images could be loaded and used to derive the various tissue properties, i.e., color and material properties at run-time. If available this set of images could consist of CT, MRI and MRE imaging as well as precomputed feature maps. In order to produce these, machine learning algorithms could be trained using a ground truth for the tissue properties. The result of the training could be stored in multidimensional transfer functions, or alternatively a fast implementation of the trained machine learning algorithms could be applied at run-time.

Improved Immersion The workbench setup used in the framework provides a certain immersion by stereoscopic 3D and colocating the haptic device work space and visual representation. However, problems exist with this setup. The colocation is only valid for a single and fixed viewpoint in front of the workbench. Moving of the head by the user thus leads to discrepancy in the matching of haptic device workspace and visual scene representation. This could be fixed by head-tracking as already available in consumer entertainment devices (for example the Nintendo 3DS released in 2015). Evidence presented in [SUY07] suggests that using a semitransparent mirror setup is inferior to head mounted displays (HMD). With upcoming low-cost and high-performance customer grade HMDs (Oculus Rift, Sony's Project Morpheus and HTC Vive) the workbench setup could be replaced by a combination of HMD and haptic devices. Augmented reality with consumer grade see-through HMDs (Microsoft Hololens), will probably make a noticable impact in the near future.

8.2. Conclusion

Overall, an innovative framework for image based puncture simulation has been presented that includes novel aspects regarding visual and haptic rendering as well as simulation of soft tissue under deformation and displacement by breathing motion in real-time. As intended these methods can be used without having to create surface or volumetric meshes, which distinguishes it from other existing frameworks. In comparison to its predecessor the framework circumvents the need for segmentation of the structures bones, skin and soft tissue. These alone took around 30 hours to segment previously, see section 1.2. The newly developed approach is unique and opens various possibilities for further research.

List of Symbols

I Reference image function, page 16

J Sub image function, page 16

s Image sampling function, page 16

l Labeling function, page 16

L Set of labels, page 16

L_{risk} Set of risk structure labels, page 16

L_{target} Set of target structure labels, page 16

$\dot{\Omega}$ Image grid, page 16

Ω_I Continuous image domain, page 16

$\dot{\Omega}_J$ Sub image grid, page 52

Ω_J Continuous sub image domain, page 16

$\bar{\Omega}$ Parts of the image that are hard structures , page 52

$\tilde{\Omega}$ Contact points between tissue and tool, page 52

$\mathbf{x}_i$ Position in world space, page 16

$\mathbf{x}$ Position of the haptic device, page 16

$\mathbf{q}$ Orientation of haptic device, page 16

$\mathbf{Q}$ Haptic device rotation matrix, page 16

$\mathbf{q}_z$ Direction of haptic device, page 16

R Ray used for ray casting, page 17

f_{shade} Shading function for ray casting, page 18

f_{shadow} Shadowing function used in ray casting, page 18

$\bar{l}$ Total needle length, page 20

μ Attenuation coefficient for X-ray simulation, page 23

ρ Contrast agent density, page 23

m Number of needle nodes, page 26

N Set of needle nodes, page 26

$\mathbf{n}_j$ Needle node, page 26

l Number of needle path nodes, page 26

P Set of needle path nodes, page 26

$\mathbf{p}_i$ Needle path node, page 26

$\mathbf{d}_i$ Needle path node direction vector, page 26

$\mathbf{r}_m$ Needle path node direction vector, page 26

$\mathbf{f}_{\text{cut}}$ Needle cutting force, page 31

$\mathbf{f}_{\text{fric}}$ Needle friction force, page 32

$\mathbf{f}_{\text{base}}$ Needle base force, page 32

$\mathbf{f}_{\text{skin}}$ Penalty force for skin structures , page 35

$\mathbf{f}_{\text{bone}}$ Penalty force for bone structures , page 36

$\mathbf{f}_{\text{slide}}$ Proxy-spring force sliding at skin surface, page 36

D_{skin} Euclidean distance map for skin, page 34

D_{bone} Euclidean distance map for bone, page 34

$\mathbf{X}$ Position in reference space, page 51

t Simulation time step, page 51

u Displacement field or vector, page 52

$\partial\hat{\Omega}$ Border of sub image domain., page 52

A. Appendix

A.1. Deflection Measurements for Two Needles

16 Gauge		16 Gauge*		20 Gauge	
N	mm	N	mm	N	mm
0.25	6	0.25	13	0.05	13
0.5	13	0.5	26	0.1	41
0.75	20	0.75	38	0.15	53
1	26			0.2	69
1.25	33				

Table A.1.: Forces applied to 16 resp. 20 gauge needles and resulting deflection. The (*) indicates measurements with removed stylet.

To get an impression of the bending of puncture needles, the deflection of the tip of two needles have been measured. The first needle that was analyzed is a 16 gauge (= 1.6 mm) needle, the second is a 20 gauge needle (= 0.9 mm). The shafts are ca. 165 mm resp. 200 mm long and both needles contain a removable stylet. To measure the deflection, forces were applied to the tip with a spring scale while the needle base was fixed by a bench vise. The results are given in Tab. A.1 including measurements for the 16 gauge needle with removed stylet.

A.2. List of Tissue and Organ Structures Needed for PTCD

Structure	Segmentation method	Effort	Notes
Air/Cavities	online heuristic	0 h	
Soft tissue/Fat	online heuristic	0 h	
Skin	online heuristic	0 h	
Bone	online heuristic	0 h	risk structure
Liver	Multi-Atlas segm.	< 5 h	cluster parallelization
Fascia	manual model fitting	< 1 h	
Lung	Region Growing etc.	< 1 h	risk structure
Liver Blood Vessels	Vesselness segm.	< 1 h	risk structure
Biliary Ducts	Vesselness segm.	< 1 h	
Right hepatic duct	Vesselness segm.	< 1 h	target structure

Table A.2.: Tissue and organ structures that are of major concern for PTCD with segmentation method and estimated segmentation effort.

Tab. A.2 contains the structures of major concern in PTCD simulation. These are either heuristically determined during run-time or segmented in an offline preprocessing step. The estimated effort reflects the experiences of the developers and are conservative statements. Liver segmentation was performed using Multi-Atlas segmentation methods [MFM$^+$13, BMFH14], which can be parallelized on a large cluster computer. Fascia segmentation can be performed by fitting a fascia model to the image data of a new patient model. Lung segmentation is possible by convention segmentation methods as region growing and manual post processing. Using Vesselness segmentation with manual post processing, the liver vessels can be segmented [BMF$^+$14].

Bibliography

[Ash07] J. Ashburner, "A Fast Diffeomorphic Image Registration Algorithm," *NeuroImage*, vol. 38, no. 1, pp. 95–113, 2007. 51, 64

[ASPO15] A. R. Aguilera, A. L. Salas, D. M. Perandrés, and M. A. Otaduy, "A Parallel Resampling Method for Interactive Deformation of Volumetric Models," *Computers & Graphics*, vol. 53, Part B, pp. 147–155, 2015. 49

[AW87] J. Amanatides and A. Woo, "A Fast Voxel Traversal Algorithm for Ray Tracing," *Proceedings of Eurographics*, vol. i, 1987. 18

[BBG+09] F. Bello, A. Bulpitt, D. A. Gould *et al.*, "ImaGiNe-S: Imaging Guided Interventional Needle Simulation," in *Proceedings of Eurographics 2009 Short and Areas Papers and Medical Prize Awards*. Eurographics Association, 2009, pp. 5–8. 4, 33, 50

[BG00] D. Bartz and O. Gürvit, "Haptic Navigation in Volumetric Datasets," *Proceedings of PHANToM User Research Symposium*, pp. 43–47, 2000. 33

[BHWM10] K. Burg, H. Haf, F. Wille, and A. Meister, *Partielle Differentialgleichungen und funktionalanalytische Grundlagen*. Springer, 2010. 55

[BMF+14] P. A. Behringer, A. Mastmeyer, D. Fortmeier, C. Biermann, and H. Handels, "Segmentierung intrahepatischer Gefäße mit Vesselness-Verfahren," in *Bildverarbeitung für die Medizin 2014*, ser. Informatik aktuell, T. M. Deserno, H. Handels, H.-P. Meinzer, and T. Tolxdorff, Eds., Springer, Berlin Heidelberg. Aachen: Springer, Berlin Heidelberg, 2014, pp. 150–155. 92, 105, 114

[BMFH14] J. Beuke, A. Mastmeyer, D. Fortmeier, and H. Handels, "Entwicklung und Vergleich von Selektionsstrategien zur atlasbasierten Segmentierung," in *Bildverarbeitung für die Medizin 2014*, ser. Informatik aktuell, T. M. Deserno, H. Handels, H.-P. Meinzer, and T. Tolxdorff, Eds., Springer, Berlin Heidelberg. Aachen: Springer, Berlin Heidelberg, 2014, pp. 400–402. 92, 105, 114

[BN98] M. Bro-Nielsen, "Finite Element Modeling in Surgery Simulation," *Proceedings of the IEEE*, vol. 86, no. 3, pp. 490–503, Mar. 1998. 48

[BNC96] M. Bro-Nielsen and S. Cotin, "Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation," *Computer Graphics Forum*, vol. 15, no. 3, pp. 57–66, Aug. 1996. 48

[CAR+09] N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O'Brien, "Interactive Simulation of Surgical Needle Insertion and Steering," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1–10, 2009. 26, 49

[CDA99] S. Cotin, H. Delingette, and N. Ayache, "Real-time Elastic Deformations of Soft Tissues for Surgery Simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 62–73, 1999. 48

[CJGC11] T. R. Coles, N. W. John, D. A. Gould, and D. G. Caldwell, "Integrating Haptics with Augmented Reality in a Femoral Palpation and Needle Insertion Training Simulation," *IEEE Transactions on Haptics*, vol. 4, no. 3, pp. 199–209, 2011. 3, 4, 33, 35

[CLC+08] M. Chen, W. Lu, Q. Chen, K. J. Ruchala, and G. H. Olivera, "A Simple Fixed-point Approach to Invert a Deformation Field," *Medical Physics*, vol. 35, no. 1, p. 81, 2008. 60, 81

[CM11] N. Chentanez and M. Müller, "Real-time Eulerian Water Simulation using a Restricted Tall Cell Grid," in *ACM Transactions on Graphics*, vol. 30, no. 4. ACM, 2011, p. 82. 108

[Cot77] P. B. Cotton, "ERCP," *Gut*, vol. 18, no. 4, pp. 316–341, 1977. 9

[Cou14] E. Coumans. (2014) Bullet Physics Library. Accessed on September, 10th 2014. [Online]. Available: http://bulletphysics.org/ 28

[Dal14] D. J. Dalek, "Erstellung von Fallbeispielen für einen Virtual Reality Lumbalpunktionssimulator und Evaluation der Trainingseffekte," Ph.D. dissertation, Univeristätsklinikum Hamburg-Eppendorf Institut für Computational Neuroscience Prof. Dr. Claus-Christian Hilgetag, July 2014. 6, 92

[DBI11] C. F. Dietrich, B. Branden, and A. Ignee, "Perkutane Transhepatische Cholangiodrainage," in *Interventioneller Ultraschall*, 2011, ch. 20, pp. 283–304. 9, 11, 24

[DGM+09] C. Duriez, C. Guébert, M. Marchal, S. Cotin, and L. Grisoni, "Interactive Simulation of Flexible Needle Insertions based on Constraint Models," *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 12, pp. 291–299, 2009. 49

[DGW10] C. Dick, J. Georgii, and R. Westermann, "A Real-Time Multigrid Finite Hexahedra Method for Elasticity Simulation using CUDA," *Simulation Modelling Practice and Theory*, 2010. 48, 58, 64

[DiM03] S. P. DiMaio, "Modelling, Simulation and Planning of Needle Motion in Soft Tissues," Ph.D. dissertation, University of British Columbia, 2003. 48

[DKS01] S. De, J. Kim, and M. A. Srinivasan, "A Meshless Numerical Technique for Physically based Real Time Medical Simulations," in *Studies in Health Technology and Informatics*, vol. 81, 2001, pp. 113–118. 50

[DS05] S. P. DiMaio and S. E. Salcudean, "Interactive Simulation of Needle Insertion Models," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 7, pp. 1167–79, 2005. 48

[DWJ10] N. Dong, Y. Wing, and Q. Jing, "A Virtual Reality Simulator for Ultrasound-guided Biopsy Training," *IEEE Computer Graphics and Applications*, vol. 31, no. 2, pp. 36–48, 2010. 3, 5, 72

[EHK+06] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-time Volume Graphics*. AK Peters, Ltd., 2006. 17, 18

[EL13] J. Ehrhardt and C. Lorenz, Eds., *4D Modeling and Estimation of Respiratory Motion for Radiation Therapy*. Springer Berlin Heidelberg, 2013. 73

[EWS+07] J. Ehrhardt, R. Werner, D. Säring, T. Frenzel, W. Lu, D. Low, and H. Handels, "An Optical Flow based Method for Improved Reconstruction of 4D CT Data Sets Acquired during Free Breathing," *Medical Physics*, vol. 34, no. 2, pp. 711–721, 2007. 73

[Fär09] M. Färber, "Entwicklung eines Virtual-Reality-Frameworks zur Simulation von Punktionseingriffen," Ph.D. dissertation, Universität zu Lübeck, 2009. 6, 17, 22, 92, 95

[FDBH09] M. Färber, T. Dahmke, C. Bohn, and H. Handels, "Needle Bending in a VR-Puncture Training System using a 6DOF Haptic Device," in *Proceedings of MMVR 17*, vol. 142, Jan. 2009, p. 91. 26

[FHGH09] M. Färber, F. Hummel, C. Gerloff, and H. Handels, "Virtual Reality Simulator for the Training of Lumbar Punctures," *Methods in Information and Medicine*, vol. 48, no. 5, pp. 493–501, Jan. 2009. 3, 5, 15, 96

[FMH12] D. Fortmeier, A. Mastmeyer, and H. Handels, "GPU-Based Visualization of Deformable Volumetric Soft-Tissue for Real-Time Simulation of Haptic Needle Insertion," in *Bildverarbeitung für die Medizin, BVM 2012*, ser. Informatik aktuell, T. Tolxdorff, T. M. Deserno, H. Handels, and H.-P. Meinzer, Eds. Springer, Berlin Heidelberg, 2012, pp. 117–122. 7, 47

[FMH13a] D. Fortmeier, A. Mastmeyer, and H. Handels, "Image-Based Palpation Simulation with Soft Tissue Deformations Using ChainMail on the GPU," in *Bildverarbeitung für die Medizin, BVM 2013*, ser. Informatik aktuell, H.-P. Meinzer, T. M. Deserno, H. Handels, and T. Tolxdorff, Eds. Springer Verlag, Berlin, 2013, pp. 140–145. 7, 15, 47, 65, 67

[FMH13b] D. Fortmeier, A. Mastmeyer, and H. Handels, "Image-based Soft Tissue Deformation Algorithms for Real-time Simulation of Liver Puncture," *Current Medical Imaging Reviews*, vol. 9, no. 2, pp. 154–165, 2013. 7, 33, 47, 53, 65

[FMH13c] D. Fortmeier, A. Mastmeyer, and H. Handels, "Optimized Image-based Soft Tissue Deformation Algorithms for Visualization of Haptic Needle Insertion," in *Medicine Meets Virtual Reality 20, MMVR 2013*, ser. Studies in Health Technology and Informatics, vol. 184. IOS Press, 2013, pp. 136–141. 7, 47, 64, 65, 67

[FMH14] D. Fortmeier, A. Mastmeyer, and H. Handels, "An Image-Based Multiproxy Palpation Algorithm for Patient-Specific VR-Simulation," in *Medicine Meets Virtual Reality 21, MMVR 2014*, ser. Studies in Health Technology and Informatics, vol. 196. IOS Press, 2014, pp. 107–113. 7, 15, 33, 37, 46, 47, 65

[FMSH16] D. Fortmeier, A. Mastmeyer, J. Schröder, and H. Handels, "A Virtual Reality System for PTCD Simulation using Direct Visuo-haptic Rendering of Partially Segmented Image Data," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 1, pp. 355–366, 2016. 7, 15, 46, 47, 64, 65, 91, 100, 105

[FWMH15] D. Fortmeier, M. Wilms, A. Mastmeyer, and H. Handels, "Direct Visuo-Haptic 4D Volume Rendering using Respiratory Motion Models," *IEEE Transactions on Haptics*, vol. 8, no. 4, pp. 371–383, 2015. 7, 15, 46, 71, 78, 88

[Gib97] S. F. Gibson, "3D Chainmail: A Fast Algorithm for Deforming Volumetric Objects," in *Proceedings of the 1997Symposium on Interactive 3D Graphics*. ACM, 1997, p. 149. 49

[Gok09] O. Goksel, "Meshing and Rendering of Patient-Specific Deformation Models with Application to Needle Insertion," Ph.D. dissertation, University of British Columbia, 2009. 49

[GS09] O. Goksel and S. E. Salcudean, "B-mode Ultrasound Image Simulation in Deformable 3-D Medium," *IEEE Transactions on Medical Imaging*, vol. 28, no. 11, pp. 1657–69, Nov. 2009. 4

[GSMS13] O. Goksel, K. Sapchuk, W. J. Morris, and S. E. Salcudean, "Prostate Brachytherapy Training with Simulated Ultrasound and Fluoroscopy Images," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 4, pp. 1002–12, 2013. 4, 23

[GSS11] O. Goksel, K. Sapchuk, and S. E. S. Salcudean, "Haptic Simulator for Prostate Brachytherapy with Simulated Needle and Probe Interaction," *IEEE Transactions on Haptics*, vol. 4, no. 3, pp. 188–198, 2011. 3, 4

[GW05] J. Georgii and R. Westermann, "Mass-spring Systems on the GPU," *Simulation Modelling Practice and Theory*, vol. 13, no. 8, pp. 693–702, 2005. 50

[GW06] J. Georgii and R. Westermann, "A Generic and Scalable Pipeline for GPU Tetrahedral Grid Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1345–1352, 2006. 49

[GW10] S. Grewenig and J. Weickert, "From Box Filtering to Fast Explicit Diffusion," *Pattern Recognition*, vol. 6376, pp. 533–542, 2010. 65

[HA00] V. Hayward and B. Armstrong, "A New Computational Model of Friction Applied to Haptic Rendering," *Experimental Robotics VI*, pp. 404–412, 2000. 31, 36

[HM06] E. Haber and J. Modersitzki, "A Multilevel Method for Image Registration," *SIAM Journal on Imaging Sciences*, vol. 27, no. 5, pp. 1594–1607, 2006. 51, 64

[HNR+10] A. Hostettler, S. A. Nicolau, Y. Rémond, J. Marescaux, and L. Soler, "A Real-time Predictive Simulation of Abdominal Viscera Positions during Quiet Free Breathing," *Progress in Biophysics and Molecular Biology*, vol. 103, no. 2-3, pp. 169–84, 2010. 72

[HS04] J. H. Hubbell and S. M. Seltzer. (2004) Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients (Version 1.4). Accessed on April, 21th 2015. [Online]. Available: http://www.nist.gov/pml/data/xraycoef/index.cfm 23

[IDY14] T. Iwashita, S. Doi, and I. Yasuda, "Endoscopic Ultrasound-guided Biliary Drainage: A Review," *Clinical Journal of Gastroenterology*, vol. 7, no. 2, pp. 94–102, 2014. 9

[JJM+14] X. Jin, G. R. Joldes, K. Miller, K. H. Yang, and A. Wittek, "Meshless Algorithm for Soft Tissue Cutting in Surgical Simulation," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 17, no. 7, pp. 800–811, 2014. 50, 107

[JSP+15] S. Jeong, B. Solenthaler, M. Pollefeys, M. Gross *et al.*, "Data-driven Fluid Simulations Using Regression Forests," *ACM Transactions on Graphics*, vol. 34, no. 6, p. 199, 2015. 108

[Kab06] S. Kabus, "Multiple-Material Variational Image Registration," Ph.D. dissertation, Universität zu Lübeck - Institut für Mathematik, 2006. 57, 109

[KJL+11] J. Kronander, D. Jonsson, J. Low, P. Ljung, A. Ynnerman, and J. Unger, "Efficient Visibility Encoding for Dynamic Illumination in Direct Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 3, pp. 447–462, Feb. 2011. 107

[KK86] T. L. Kay and J. T. Kajiya, "Ray Tracing Complex Scenes," *SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 269–278, Aug. 1986. 18

[KKWN09] O. Kutter, A. Karamalis, W. Wein, and N. Navab, "A GPU-based Framework for Simulation of Medical Ultrasound," in *SPIE Medical Imaging 2009*, vol. 7261, 2009, pp. 726 117–726 119. 25

[Kui02] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 2002. 16

[KYS13] P. Keall, T. Yamamoto, and Y. Suh, "Introduction to 4D Motion Modeling and 4D Radiotherapy," in *4D Modeling and Estimation of Respiratory Motion for Radiation Therapy*, ser. Biological and Medical Physics, Biomedical Engineering, J. Ehrhardt and C. Lorenz, Eds., 2013, pp. 1–21. 71

[LB03] Y. Li and K. Brodlie, "Soft Object Modelling with Generalised ChainMail - Extending the Boundaries of Web-based Graphics," *Computer Graphics Forum*, vol. 22, no. 4, pp. 717–727, Dec. 2003. 5, 50

[LC87] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, 1987, pp. 163–169. 6, 15, 85, 92

[LMH07] B. A. Lloyd, S. Member, and M. Harders, "Identification of Spring Parameters for Deformable Object Simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 1081–1094, 2007. 50

[Lui14] J. Luitjens. (2014, February) Faster Parallel Reductions on Kepler. Accessed on April, 21th 2015. [Online]. Available: http://devblogs.nvidia.com/parallelforall/faster-parallel-reductions-kepler/ 13

[LYG02] K. Lundin, A. Ynnerman, and B. Gudmundsson, "Proxy-based Haptic Feedback from Volumetric Density Data," *Eurohaptics Conference*, 2002. 15, 33, 36

[LZW14] S. Li, Q. Zhao, and S. Wang, "Interactive Deformation and Cutting Simulation Directly using Patient-specific Volumetric Images," *Computer Animation and Virtual Worlds*, vol. 25, no. 2, pp. 155–169, 2014. 48, 107

[McC13] J. McClelland, "Estimating Internal Respiratory Motion from Respiratory Surrogate Signals Using Correspondence Models," in *4D Modeling and Estimation of Respiratory Motion for Radiation Therapy*, ser. Biological and Medical Physics, Biomedical Engineering, J. Ehrhardt and C. Lorenz, Eds. Springer Berlin Heidelberg, 2013, pp. 187–213. 75

[MFH12] A. Mastmeyer, D. Fortmeier, and H. Handels, "Direct Haptic Volume Rendering in Lumbar Puncture Simulation," in *Medicine Meets Virtual Reality 19, MMVR 2012*, ser. Studies in Health Technology and Informatics, vol. 173. IOS Press, 2012, p. 280. 15, 91

[MFM+13] A. Mastmeyer, D. Fortmeier, E. Maghsoudi, M. Simon, and H. Handels, "Patch-Based Label Fusion Using Local Confidence-Measures and Weak Segmentations," in *SPIE Medical Imaging 2013, Image Processing*, Orlando, USA, 2013, pp. 86 691N–1–86 691N–11. 92, 105, 114

[MFMH15] M. Meike, D. Fortmeier, A. Mastmeyer, and H. Handels, "Real-Time Resampling of Medical Images Based on Deformed Tetrahedral Structures for Nee-

dle Insertion VR-Simulation," in *Bildverarbeitung für die Medizin 2015*, ser. Informatik aktuell, H. Handels, T. M. Deserno, H.-P. Meinzer, and T. Tolxdorff, Eds. Lübeck: Springer, Berlin Heidelberg, 2015, pp. 443–448. 49

[MG04] M. Müller and M. Gross, "Interactive Virtual Materials," in *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society, 2004, pp. 239–246. 48

[MHFH13] A. Mastmeyer, T. Hecht, D. Fortmeier, and H. Handels, "Ray-Casting-Based Evaluation Framework for Needle Insertion Force Feedback Algorithms," in *Bildverarbeitung für die Medizin 2013*. Springer, 2013, pp. 3–8. 101

[MHFH14] A. Mastmeyer, T. Hecht, D. Fortmeier, and H. Handels, "Ray-casting based Evaluation Framework for Haptic Force Feedback during Percutaneous Transhepatic Catheter Drainage Punctures," *International Journal of Computer Assisted Radiology and Surgery*, vol. 9, no. 3, pp. 421–431, 2014. 101

[MJLW07] K. Miller, G. Joldes, D. Lance, and A. Wittek, "Total Lagrangian Explicit Dynamics Finite Element Algorithm for Computing Soft Tissue Deformation," *Communications in Numerical Methods in Engineering*, no. August 2006, pp. 121–134, 2007. 48

[MMvV+12] A. Majewicz, S. P. Marra, M. G. van Vledder, M. Lin, M. A. Choti, D. Y. Song, and A. M. Okamura, "Behavior of Tip-steerable Needles in Ex Vivo and In Vivo Tissue," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2705–2715, 2012. 105

[Mod04] J. Modersitzki, *Numerical Methods for Image Registration*. Oxford University Press, 2004. 50, 56

[MOOXS08] P. Muyan-Ozcelik, J. D. Owens, J. Xia, and S. S. Samant, "Fast Deformable Registration on the GPU: A CUDA Implementation of Demons," *2008 International Conference on Computational Sciences and Its Applications*, pp. 223–233, Jun. 2008. 51

[MSN+06] W. Mollemans, F. Schutyser, N. Nadjmi, F. Maes, and P. Suetens, "Parameter Optimisation of a Linear Tetrahedral Mass Tensor Model for a Maxillofacial Soft Tissue Simulator," in *Biomedical Simulation*, ser. Lecture Notes in Computer Science, M. Harders and G. Székely, Eds. Springer Berlin Heidelberg, 2006, vol. 4072, pp. 159–168. 50

[MVR+11] K. Murphy, B. Van Ginneken, J. M. Reinhardt *et al.*, "Evaluation of Registration Methods on Thoracic CT: The EMPIRE10 Challenge," *IEEE Transactions on Medical Imaging*, vol. 30, no. 11, pp. 1901–1920, 2011. 74

[MWTT98] W. Maurel, Y. Wu, N. M. Thalmann, and D. Thalmann, *Biomechanical Models for Soft Tissue Simulation*. Springer, 1998. 48

[NMP+05] M. Nesme, M. Marchal, E. Promayon, M. Chabanas, Y. Payan, and F. Faure, "Physically Realistic Interactive Simulation for Biological Soft Tissues," *Recent Research Developments in Biomechanics*, vol. 2, no. 2, 2005. 48

[Nvi12a] Nvidia. (2012) Best Practice Guide. Accessed on April, 2th 2015. [Online]. Available: http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf 13

[Nvi12b] Nvidia. (2012) Kepler GK110. Accessed on April, 2th 2015. [Online]. Available: http://www.nvidia.com/content/PDF/kepler/NVIDIA-kepler-GK110-Architecture-Whitepaper.pdf 13, 108

[Nvi12c] Nvidia. (2012) NVIDIA GeForce GTX 680. Accessed on April, 2th 2015. [Online]. Available: http://www.geforce.com/Active/en_US/en_US/pdf/GeForce-GTX-680-Whitepaper-FINAL.pdf 12

[PCY07] K. L. Palmerius, M. Cooper, and A. Ynnerman, "Haptic Rendering of Dynamic Volumetric Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 263–76, 2007. 33

[PM90] P. Perona and J. Malik, "Scale-space and Edge Detection using Anisotropic Diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990. 57

[PND+11] I. Peterlik, M. Nouicer, C. Duriez, S. Cotin, and A. Kheddar, "Constraint-Based Haptic Rendering of Multirate Compliant Mechanisms," *IEEE Transactions on Haptics*, vol. 4, no. 3, pp. 175–187, 2011. 49

[PO09] E. G. Parker and J. F. O'Brien, "Real-Time Deformation and Fracture in a Game Environment," *Eurographics ACM SIGGRAPH Symposium on Computer Animation 2009*, vol. 1, p. 165, 2009. 48

[Pro95] X. Provot, "Deformation Constraints in a Mass-spring Model to Describe Rigid Cloth Behaviour," in *Graphics Interface*, 1995, pp. 147–154. 50

[PX11] G. Pratx and L. Xing, "GPU Computing in Medical Physics: A Review," *Medical Physics*, vol. 38, no. 5, pp. 2685–2697, 2011. 12

[RKK97] D. C. Ruspini, K. Kolarov, and O. Khatib, "The Haptic Display of Complex Graphical Environments," *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH 97*, pp. 345–352, 1997. 36

[RLAM15] A. Rodríguez, A. León, G. Arroyo, and J. Mantas, "SP-ChainMail: A GPU-based Sparse Parallel ChainMail Algorithm for Deforming Medical Volumes," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3482–3499, 2015. 50

[Rößl09] F. A. Rößler, "Bridging the Gap between Volume Visualization and Medical Applications," Ph.D. dissertation, University of Stuttgart, 2009. 65

[RPAS09] T. Reichl, J. Passenger, O. Acosta, and O. Salvado, "Ultrasound goes GPU: Real-time Simulation using CUDA," *SPIE Medical Imaging 2009*, vol. 7261, no. 1, pp. 726 116–726 116–10, 2009. 25

[RRM+05] D. B. Russakoff, T. Rohlfing, K. Mori, D. Rueckert, A. Ho, J. R. Adler, and C. R. Maurer, "Fast Generation of Digitally Reconstructed Radiographs using Attenuation Fields with Application to 2D-3D Image Registration," *IEEE Transactions on Medical Imaging*, vol. 24, no. 11, pp. 1441–54, Nov. 2005. 23

[RWE08] F. Rössler, T. Wolff, and T. Ertl, "Direct GPU-based Volume Deformation," in *Proceedings of Curac 2008*, Leipzig, 2008, pp. 65–68. 49, 50

[SASW96] V. Spitzer, M. J. Ackerman, A. L. Scherzinger, and D. Whitlock, "The Visible Human Male: A Technical Report," *Journal of the American Medical Informatics Association*, vol. 3, no. 2, pp. 118–130, 1996. 5

[SBH07] F. Schulze, K. Bühler, and M. Hadwiger, "Interactive Deformation and Visualization of Large Volume Datasets," in *Proceedings of International Conference on Computer Graphics Theory and Applications*, vol. D, 2007, pp. 39–46. 49, 50, 65

[She94] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method without the Agonizing Pain," 1994, Carnegie-Mellon University. Department of Computer Science. 59

[SID+08] A. P. Santhanam, C. Imielinska, P. Davenport, P. Kupelian, and J. P. Rolland, "Modeling Real-time 3-d Lung Deformations for Medical Visualization," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 2, pp. 257–70, 2008. 72

[SK10] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-purpose GPU Programming*. Addison-Wesley Professional, 2010. 11

[SMFH14] J. Schröder, A. Mastmeyer, D. Fortmeier, and H. Handels, "Ultraschallsimulation für das Training von Gallengangspunktionen," in *Bildverarbeitung für die Medizin 2014*, ser. Informatik aktuell, T. M. Deserno, H. Handels, H.-P. Meinzer, and T. Tolxdorff, Eds. Springer, Berlin Heidelberg, 2014, pp. 222–227. 25, 46

[SNC90] G. Sherouse, K. Novins, and E. Chaney, "Computation of Digitally Reconstructed Radiographs for Use in Radiotherapy Treatment Design," *International Journal of Radiation Oncology Biology Physics*, vol. 18, no. 3, pp. 651–658, 1990. 23

[SREWH12] A. Schmidt-Richberg, J. Ehrhardt, R. Werner, and H. Handels, "Fast Explicit Diffusion for Registration with Direction-Dependent Regularization," in *Biomedical Image Registration*, ser. Lecture Notes in Computer Science, B. Dawant, G. Christensen, J. Fitzpatrick, and D. Rueckert, Eds. Springer Berlin Heidelberg, 2012, vol. 7359, pp. 220–228. 65

[SRWHE12] A. Schmidt-Richberg, R. Werner, H. Handels, and J. Ehrhardt, "Estimation of Slipping Organ Motion by Registration with Direction-dependent Regularization," *Medical Image Analysis*, vol. 16, no. 1, pp. 150–159, 2012. 74

[SSKH10] R. Shams, P. Sadeghi, R. A. Kennedy, and R. I. Hartley, "A Survey of Medical Image Registration on Multicore and the GPU," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 50–60, 2010. 12

[SUY07] C. Sandor, S. Uchiyama, and H. Yamamoto, "Visuo-haptic Systems: Half-mirrors Considered Harmful," in *Eurohaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint*. IEEE, 2007, pp. 292–297. 109

[SVAT12] G. San-Vicente, I. Aguinaga, and J. Tomás Celigüeta, "Cubical Mass-Spring Model Design Based on a Tensile Deformation Test and Nonlinear Material Model," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 228–41, Feb. 2012. 50

[SVR13] D. Sarrut, J. Vandemeulebroucke, and S. Rit, "Intensity-Based Deformable Registration: Introduction and Overview," in *4D Modeling and Estimation of Respiratory Motion for Radiation Therapy*, ser. Biological and Medical Physics,

Biomedical Engineering, J. Ehrhardt and C. Lorenz, Eds. Springer Berlin Heidelberg, 2013, pp. 103–124. 74

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically Deformable Models," in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 205–214. 48

[UIK12] S. Ullrich, IEEE, and T. Kuhlen, "Haptic Palpation for Medical Simulation in Virtual Environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 617–620, 2012. 3, 4, 33

[VBBG11] P. F. Villard, P. Boshier, F. Bello, and D. A. Gould, "Virtual Reality Simulation of Liver Biopsy with a Respiratory Component," in *Liver Biopsy*, H. Takahashi, Ed. InTech, 2011, ch. 20. 3, 4, 72

[VHRG04] J. Vozenilek, J. S. Huff, M. Reznek, and J. A. Gordon, "See One, Do One, Teach One: Advanced Technology in Medical Education," *Academic Emergency Medicine*, vol. 11, no. 11, pp. 1149–1154, 2004. 1

[VJHG08] F. P. Vidal, N. W. John, A. E. Healey, and D. A. Gould, "Simulation of Ultrasound Guided Needle Puncture using Patient Specific Data with 3D Textures and Volume Haptics," *Computer Animation and Virtual Worlds*, vol. 19, no. 2, pp. 111–127, 2008. 4

[VVA+13] P. F. Villard, F. P. Vidal, L. Ap Cenydd, R. Holbrey, S. Pisharody, S. Johnson, A. Bulpitt, N. W. John, F. Bello, and D. Gould, "Interventional Radiology Virtual Simulator for Liver Biopsy," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–13, 2013. 4, 72

[VVBJ12] P. F. Villard, F. P. Vidal, F. Bello, and N. W. John, "A Method to Compute Respiration Parameters for Patient-based Simulators," in *Medicine Meets Virtual Reality 19, MMVR 2012*, ser. Studies in Health Technology and Informatics, vol. 173. IOS Press, 2012, pp. 529–533. 72

[VVH+09] F. P. Vidal, P. F. Villard, R. Holbrey, N. W. John, F. Bello, A. Bulpitt, and D. A. Gould, "Developing an Immersive Ultrasound Guided Needle Puncture Simulator," in *Medicine Meets Virtual Reality 17, MMVR 2009*, ser. Studies in Health Technology and Informatics, vol. 142. IOS Press, Jan. 2009, pp. 398–400. 4, 33

[VVL12] F. P. Vidal, P. F. Villard, and E. Lutton, "Tuning of Patient-specific Deformable Models using an Adaptive Evolutionary Optimization Strategy," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2942–9, Oct. 2012. 4, 72

[VYE13] S. K. Venkatesh, M. Yin, and R. L. Ehman, "Magnetic Resonance Elastography of Liver: Clinical Applications," *Journal of Computer Assisted Tomography*, vol. 37, no. 6, p. 887, 2013. 108

[WAC07] X. Wu, J. Allard, and S. Cotin, "Real-time Modeling of Vascular Flow for Angiography Simulation," *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 10 Pt 1, pp. 557–565, Jan. 2007. 23, 72

[WAVH+12] W. I. Willaert, R. Aggarwal, I. Van Herzeele, N. J. Cheshire, and F. E. Vermassen, "Recent Advancements in Medical Simulation: Patient-specific Virtual Reality Simulation," *World Journal of Surgery*, vol. 36, no. 7, pp. 1703–1712, 2012. 6

[WBK+08] W. Wein, S. Brunke, A. Khamene, M. R. Callstrom, and N. Navab, "Automatic CT-ultrasound Registration for Diagnostic Imaging and Image-guided Intervention," *Medical Image Analysis*, vol. 12, no. 5, pp. 577–585, 2008. 25

[WDL+12] M. Wagner, K. Dresing, W. Ludwig, C. A. Ahrens, and O. J. Bott, "SIScaR-GPU: Fast Simulation and Visualization of Intraoperative Scattered Radiation to Support Radiation Protection Training," in *Studies in Health Technology and Informatics*, vol. 180, 2012, pp. 968–972. 23

[WF04] X. Wang and A. Fenster, "A Virtual Reality Based 3D Real-time Interactive Brachytherapy Simulation of Needle Insertion and Seed Implantation," *2004 2nd IEEE International Symposium on Biomedical Imaging: Macro to Nano*, pp. 280–283, 2004. 49

[WFMH15] M. Wilms, D. Fortmeier, A. Mastmeyer, and H. Handels, "Modellbasierte Simulation der Atembewegung für das Virtual-Reality-Training von Punktionseingriffen," in *Bildverarbeitung für die Medizin 2015*, ser. Informatik aktuell, H. Handels, T. M. Deserno, H.-P. Meinzer, and T. Tolxdorff, Eds. Springer, Berlin Heidelberg, 2015, pp. 317–322. 7, 71

[Wil78] L. Williams, "Casting Curved Shadows on Curved Surfaces," *ACM SIGGRAPH Computer Graphics*, vol. 12, no. 3, pp. 270–274, 1978. 20

[WSRHE14] R. Werner, A. Schmidt-Richberg, H. Handels, and J. Ehrhardt, "Estimation of Lung Motion Fields in 4D CT data by Variational Non-linear Intensity-based Registration: A Comparison and Evaluation Study," *Physics in Medicine and Biology*, vol. 59, no. 15, pp. 4247–60, 2014. 74

[WWD14] J. Wu, R. Westermann, and C. Dick, "Real-Time Haptic Cutting of High-Resolution Soft Tissues," in *Medicine Meets Virtual Reality 21, MMVR 2014*, ser. Studies in Health Technology and Informatics, vol. 196. IOS Press, 2014, pp. 469–475. 48, 107

[WWE+14] M. Wilms, R. Werner, J. Ehrhardt, A. Schmidt-Richberg, H.-P. Schlemmer, and H. Handels, "Multivariate Regression Approaches for Surrogate-based Diffeomorphic Estimation of Respiratory Motion in Radiation Therapy," *Physics in Medicine and Biology*, vol. 59, no. 5, pp. 1147–64, 2014. 79

[YS13] S. Yasmin and A. Sourin, "Image-Based Virtual Palpation," in *Transactions on Computational Science XVIII*, ser. Lecture Notes in Computer Science, M. Gavrilova, C. Tan, and A. Kuijper, Eds. Springer Berlin Heidelberg, 2013, vol. 7848, pp. 61–80. 33

List of Publications by the Author

Journal articles as first author

- D. Fortmeier, A. Mastmeyer, J. Schröder, and H. Handels, "A Virtual Reality System for PTCD Simulation using Direct Visuo-haptic Rendering of Partially Segmented Image Data," IEEE Journal of Biomedical and Health Informatics, vol. 20, no. 1, pp. 355–366, 2016.
- D. Fortmeier, M. Wilms, A. Mastmeyer, and H. Handels, "Direct Visuo-Haptic 4D Volume Rendering using Respiratory Motion Models," IEEE Transactions on Haptics, vol. 8, no. 4, pp. 371–383, 2015.
- D. Fortmeier, A. Mastmeyer, and H. Handels, "Image-based Soft Tissue Deformation Algorithms for Real-time Simulation of Liver Puncture," Current Medical Imaging Reviews, vol. 9, no. 2, pp. 154–165, 2013.

Conference proceedings and presentation as first author

- D. Fortmeier, A. Mastmeyer, H. Handels, "An Image-Based Multiproxy Palpation Algorithm for Patient-Specific VR-Simulation," in Medicine Meets Virtual Reality 21, MMVR 2014, ser. Studies in Health Technology and Informatics, vol. 196. IOS Press, 2014, pp. 107–113.
- D. Fortmeier, A. Mastmeyer, H. Handels, "Optimized Image-based Soft Tissue Deformation Algorithms for Visualization of Haptic Needle Insertion," in Medicine Meets Virtual Reality 20, MMVR 2013, ser. Studies in Health Technology and Informatics, vol. 184. IOS Press, 2013, pp. 136–141.
- D. Fortmeier, A. Mastmeyer, H. Handels, "Image-Based Palpation Simulation with Soft Tissue Deformations Using ChainMail on the GPU," in Bildverarbeitung für die Medizin, BVM 2013, ser. Informatik aktuell, Springer Verlag, Berlin, 2013, pp. 140–145.
- D. Fortmeier, A. Mastmeyer, H. Handels, "GPU-Based Visualization of Deformable Volumetric Soft-Tissue for Real-Time Simulation of Haptic Needle Insertion," in Bildverarbeitung für die Medizin, BVM 2012, ser. Informatik aktuell, Springer, Berlin Heidelberg, 2012, pp. 117–122.

Other journal contribution

- J. Schwartze, B. Haarbrandt, D. Fortmeier, R. Haux, C. Seidel: "Authentication Systems for Securing Clinical Documentation Workflows," Methods in Information and Medicine, vol. 53, no. 1, pp. 3–13, 2014.
- A. Mastmeyer, T. Hecht, D. Fortmeier, H. Handels: "Ray-casting based Evaluation Framework for Haptic Force Feedback During Percutaneous Transhepatic Catheter Drainage Punctures," International Journal of Computer Assisted Radiology and Surgery, vol. 9, no. 3, pp. 421–431, 2014.

Other proceedings contributions

- M. Wilms, D. Fortmeier, A. Mastmeyer, H. Handels, "Modellbasierte Simulation der Atembewegung für das Virtual-Reality-Training von Punktionseingriffen," in Bildverarbeitung für die Medizin, BVM 2015, Informatik aktuell, 317–322, (2015)
- M. Meike, D. Fortmeier, A. Mastmeyer, H. Handels, "Real-Time Resampling of Medical Images Based on Deformed Tetrahedral Structures for Needle Insertion VR-Simulation," in Bildverarbeitung für die Medizin, BVM 2015, Informatik aktuell, 443–448, (2015)
- P. Behringer A, A. Mastmeyer, D. Fortmeier, C. Biermann, H. Handels "Segmentierung intrahepatischer Gefäße mit Vesselness-Verfahren," in Bildverarbeitung für die Medizin 2014, Informatik aktuell, 150–155, (2014)
- J. Beuke, A. Mastmeyer, D. Fortmeier, H. Handels "Entwicklung und Vergleich von Selektionsstrategien zur atlasbasierten Segmentierung," in Bildverarbeitung für die Medizin 2014, Informatik aktuell, 400–402, (2014)
- Schröder, J., A. Mastmeyer, D. Fortmeier, H. Handels "Ultraschallsimulation für das Training von Gallengangspunktionen," in Bildverarbeitung für die Medizin 2014, Informatik aktuell, 222–227, (2014)
- A. Mastmeyer, D. Fortmeier, E. Maghsoudi, M. Simon, H. Handels "Patch-Based Label Fusion Using Local Confidence-Measures and Weak Segmentations," in SPIE Medical Imaging 2013, Image Processing, 86691N-1– 86691N-11, (2013)
- T. Hecht, A. Mastmeyer, D. Fortmeier, H. Handels "4D-Planung von Leberpunktionen unter Berücksichtigung der Atembewegung," in 58. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS), Abstractband GMDS 2013, ID 193: 336–337, (2013)
- A. Mastmeyer, T. Hecht, D. Fortmeier, H. Handels "Ray-Casting-Based Evaluation Framework for Needle Insertion Force Feedback Algorithms," in Bildverarbeitung für die Medizin 2013, 3–8, (2013)

- A. Mastmeyer, D. Fortmeier, H. Handels "Anisotropic Diffusion for Direct Haptic Volume Rendering in Lumbar Puncture Simulation," in Bildverarbeitung in der Medizin 2012, Informatik aktuell, 286–291, (2012)
- C. Schröder, A. Mastmeyer, D. Fortmeier, C. Bohn, A. Nabavi, H. Handels "Optimierung von Schädelöffnungen mittels genetischer Algorithmen für die Behandlung subduraler Hämatome," in 11. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie, CURAC 2012, (2012)
- A. Mastmeyer, D. Fortmeier, and H. Handels, “Direct Haptic Volume Ren- dering in Lumbar Puncture Simulation,” in Medicine Meets Virtual Reality 19, MMVR 2012, ser. Studies in Health Technology and Informatics, vol. 173. IOS Press, pp. 280. (2012)