Automatic Initialization for 3D Ultrasound CT Registration During Liver Tumor Ablations

This thesis is submitted in partial fulfillment
of the requirements for the degree of

Master of Science Double Degree

in

Computer Science and Electrical Engineering

by

Dirk Schut
born in Utrecht, the Netherlands

Computer Graphics and Visualization Group
Department of Intelligent Systems
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Biomedical Imaging Group Rotterdam
Erasmus Medical Center
Dr. Molewaterplein 50/60
Rotterdam, the Netherlands
www.bigr.nl

Automatic Initialization for 3D Ultrasound CT Registration During Liver Tumor Ablations

Author: Dirk Schut

Abstract

Ablation is a medical procedure to treat liver cancer where a needle-like catheter has to be inserted into a tumor, which will then be heated or frozen to destroy the tumor tissue. To guide the catheter, Ultrasound(US) imaging is used which shows the catheter position in real time. However, some tumors are not visible on US images. To make these tumors visible, image fusion can be used between the inter-operative US image and a pre-operative contrast enhanced CT(CECT) scan, on which the tumors are visible. Several methods exist for tracking the motions of the US transducer relative to the CECT scan, but they all require a manual initialization or external tracking hardware to align the coordinate systems of both scans. In this thesis we present a technique for finding an initialization using only the image data. To achieve this, deep learning is used to segment liver vessels and the boundary of the liver in 3D US images. To find the rigid transformation parameters, the SaDE evolutionary algorithm was used to optimize the alignment between the blood vessels and the liver boundary between both scans.

Thesis Committee:

Chair / Computer Science supervisor:	Dr. Anna Vilanova, Faculty EEMCS, TU Delft
Electrical Engineering supervisor:	Dr. Ir. Rob Remis, Faculty EEMCS, TU Delft
Project supervisor:	Dr. Ir. Theo van Walsum, BIGR, Erasmus MC
External committee member:	Dr. Ir. Marius Staring, LUMC

Contents

List of Figures

Chapter 1

Introduction

Liver cancer is the fifth most prevalent cancer in men and the ninth most prevalent cancer in women, and it is the second most frequent cause of cancer-related death [1]. One of the treatments for liver cancer is ablation. In an ablation procedure a needle-like catheter is inserted through the skin of the patient into a tumor and the tip of the catheter is then either heated(microwave ablation, radiofrequency ablation) or frozen(cryoablation), to destroy the tumor tissue. To guide the ablation catheter, ultrasound(US) imaging is used (Figure 1.1), but some tumors can not be detected on US images. Contrast enhanced computed tomography(CECT) or agnetic resonance(MR) images can be used to visualize these tumors [2], but they cannot be acquired in real time and are unpractical to acquire during an intervention.

The current practice for liver tumor ablations at Erasmus MC is to acquire a CECT

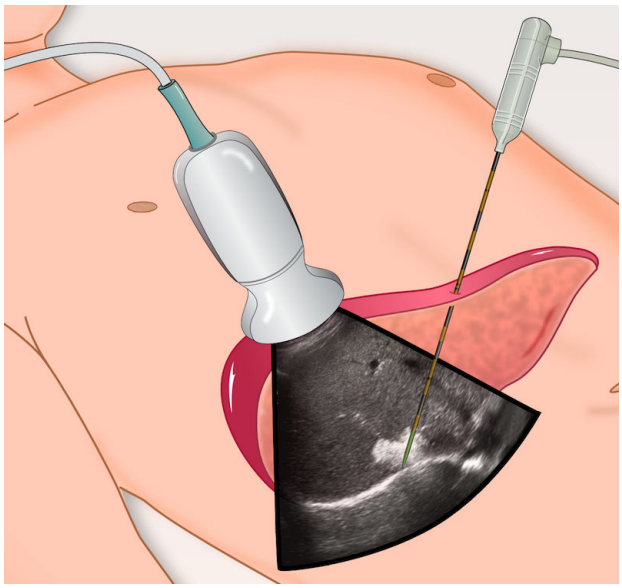

Figure 1.1: Illustration on how US guidance is used in liver tumor ablations. Image source: Brace [3]

scan of the patient before the intervention to localize and classify the tumors. During the intervention, real time ultrasound imaging is used to guide the ablation catheter. When the catheter is in approximately the right position, a non-contrast CT scan is acquired to more accurately see the position of the catheter relative to the tumors that are visible in the pre-operative CECT scan. When the catheter is not yet in the right position it will be adjusted and one or more extra CT scans may be made. When the cathether is in the right position the ablation is performed, and at the end of the intervention a CECT scan is acquired to determine whether the tumor was fully ablated. Only one CECT scan can be acquired during an intervention, because injecting more contrast agent within a short amount of time can cause kidney failure [4, 5]. To be able to perform CT scans during the intervention, the CT scanner is occupied for the entire duration of the intervention. Moreover, during CT acquisition all medical staff will have to leave the room to avoid radiation exposure and all medical instruments have to be clamped to the patient to avoid displacements when the CT platform is moving.

To make the information from the pre-operative CECT scan available during the intervention, image fusion between US and CECT can be used. The traditional method to achieve this is to use external sensors to track the movements of the US transducer. Two types of tracking systems are most used: 1.) Electromagnetic(EM) tracking systems use a field generator to generate a varying magnetic field. Sensors on the transducer derive its position and orientation by measuring the EM-field. 2.) Optical tracking systems use multiple cameras to derive the position and orientation of the transducer in 3D. Markers are often attached to the US transducer to get better visibility. Tracking systems track the orientation relative to either the field generator or the cameras. To find how this orientation corresponds to the position in the CT scan an initial registration is needed. This can be done manually which takes approximately 5–20 min [6]. In a recent study [6], an EM-tracker was used for image fusion between US and CT or MR images. 295 Tumors that were not visible on contrast enhanced US were included and 95.6% of these tumors were correctly targeted and 90.2% were completely ablated.

Nevertheless tracking systems have several downsides: EM trackers are sensitive to in-

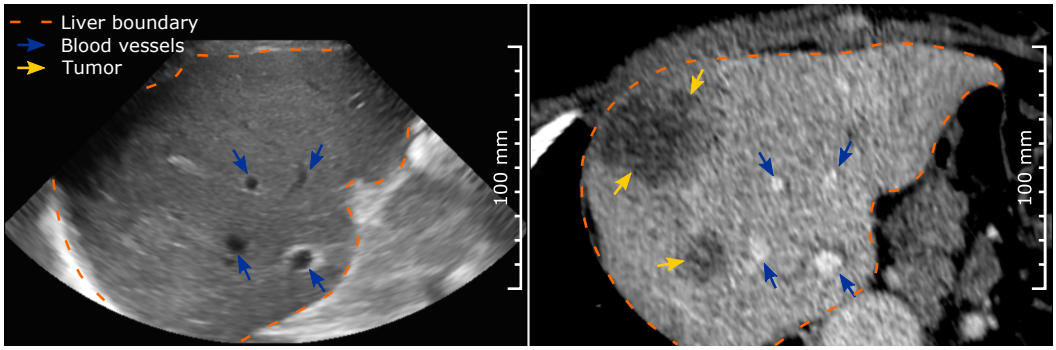

Figure 1.2: Side by side view of US(left) and registered CECT(right). The tumors are visible in the CECT scan, but not on the US scan. Some features, such as the liver boundary and the blood vessels are visible in both modalities.

terference, which can be caused by ferromagnetic and electrical devices within the EM-field [7]. Optical trackers require that the cameras have an unobstructed view of the transducer. Moreover, both types of trackers can only track rigid transformations, so non-rigid deformations inside of the body will cause errors. Lastly, there is still a manual registration required to get the initial alignment between the coordinate systems.

1.1 Research goal

At the Biomedical Imaging Group Rotterdam(BIGR) of the Erasmus MC, research has been ongoing to develop a fully image based tracking system [8, 9]. Such a system would use image registration between the inter-operative US and pre-operative CECT images to provide real time image fusion between these two imaging modalities. The goal is to design a system that does not rely on external sensors, that can track non-rigid deformations and that does not need a manual initialization. To make more information available for registration in the US domain 3D ultrasound scans are used.

Banerjee et al. [8] already developed a technique for fast multimodal 3D US CECT registration, which could be used for tracking the movements of the US transducer and the patient. Because this system has a relatively small capture range it still needs an initialization, similarly to the EM and optical trackers. In this thesis we will investigate how to automatically obtain an initial registration between 3D US and CECT scans.

1.2 Requirements

Initialization can be seen as a registration problem with specific requirements: 1.) The technique has to be able to find an initial registration from any orientation of the US transducer where the liver is in view. 2.) The error of the initialization has to be low enough for the tracking algorithm to converge. The allowed error for the technique of Banerjee et al. [10] is approximately 10 mm translations or 15° rotations. 3.) To be useful, the automatic initialization would have to be faster than a manual initialization, which takes 5-20 minutes [6]. When the initialization fails, the doctor can slightly adjust the orientation of the US transducer and restart the initialization. Moreover, if the tracking algorithm is not always able to maintain tracking, multiple initializations may be needed during a procedure. Because of this, the aim is to get the run time to be less than ten seconds. However, in this thesis we focus on meeting the first two requirements, leaving the task of speeding up the implementation to future work.

1.3 Contributions

This thesis presents the following contributions:

1. A novel method for initializing registrations between 3D US and CECT scans. On a clinical dataset of 38 scan pairs the results were manually scored as follows: 23 as good, 7 as fair, 5 as poor and 3 as bad. On the same dataset, rigid registrations based

on manual annotations by radiologists were scored as follows: 27 as good, 8 as fair, 1 as poor and 2 as bad.

2. A novel method for segmenting blood vessels and the liver boundary in 3D US scans of the liver. To the authors' knowledge this is the first time a neural network has been applied to segment these features from 3D US scans.

3. Annotations and extra medical information about the used dataset that can be used for further research at BIGR.

1.4 Thesis structure

Chapter 2 provides background information about the medical context and the imaging techniques relevant to this thesis. Chapter 3 gives an overview of the prior work in multimodal US registration and registration initialization. Chapter 4 describes how the proposed system works. Chapter 5 describes the experiments that were performed to evaluate the system and gives the results. Chapter 6 concludes the thesis, discusses its contents and proposes future work. Appendix A explains the working of the neural network techniques that were used in this thesis and appendix B lists additional figures and results of the experiments that were performed.

This thesis is best viewed in color. All figures created for this thesis have been tested to be colorblind friendly by using the Color Oracle colorblindness simulator [1].

[1] https://colororacle.org/

Chapter 2

Background

Because this thesis spans multiple topics some readers may not be familiar with all of them. The goal of this chapter therefore is to provide the necessary background information. A more in depth look into registration of the liver will be provided Chapter 3 and neural networks are explained in Appendix A.

2.1 Medical terms of location

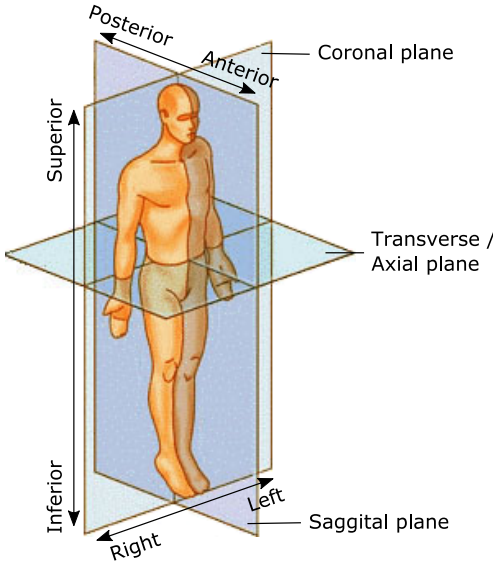

Figure 2.1: Medical terms of location relative to a human body.

To describe the space in and around the patient several medical terms are used, which are illustrated in figure 2.1. This three dimensional space is defined by three orthogonal axes. The **superior-inferior axis** is along the length of the body. The **right-left axis** is along the width of the body, with right corresponding to the right hand side of the patient,

so it does not depend on the viewer. The **posterior-anterior axis** distinguishes between the back and the front of the body.

In medical imaging you will often look at 2D slices of the body. To describe these slices it is useful to also name the planes that can be spanned in the medical space. **Transverse planes** (also called axial planes) are spanned by the right-left axis and the posterior anterior axis. CT image slices are acquired as transverse planes. **Sagittal planes** are spanned by the inferior-superior axis and the posterior-anterior axis. **Coronal planes** are spanned by the inferior-superior axis and the the right-left axis.

2.2 Anatomy of the liver

The liver is located in the front of your body mostly behind your ribs, underneath the lungs and the diaphragm (Figure 2.2). It consists of two lobes, separated by the falciform ligament, which secures the position of the liver. The right lobe is bigger than the left lobe, so the liver occupies more room in the right side of your body than the left side.

The blood supply of the liver is different than that of other organs, because it is connected to two veins instead of one. The portal vein transports blood from the intestines to the liver, so that it can be filtered before entering the rest of the body. The portal vein is the main supply of blood to the liver, but the hepatic artery also supplies the liver with fresh blood from the heart. The hepatic vein is connected to the inferior vena cava, which brings the blood back to the heart. This is illustrated in Figure 2.3.

One function of the liver is that it produces bile. The bile is transported to the gallbladder via bile ducts. The gallbladder is located underneath the liver and it is partially surrounded by the liver. Bile can be stored in the gallbladder until it has to be discharged into the start of the small intestine, where it helps with breaking down fats.

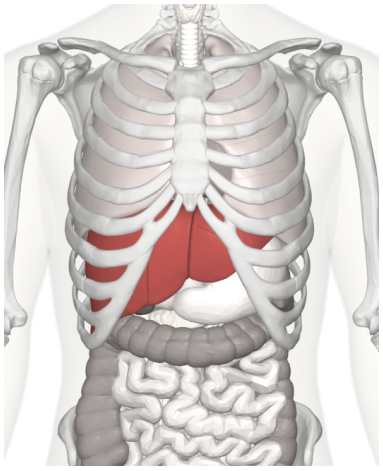

Figure 2.2: Position of the liver in the body. Source: Database Center for Life Science

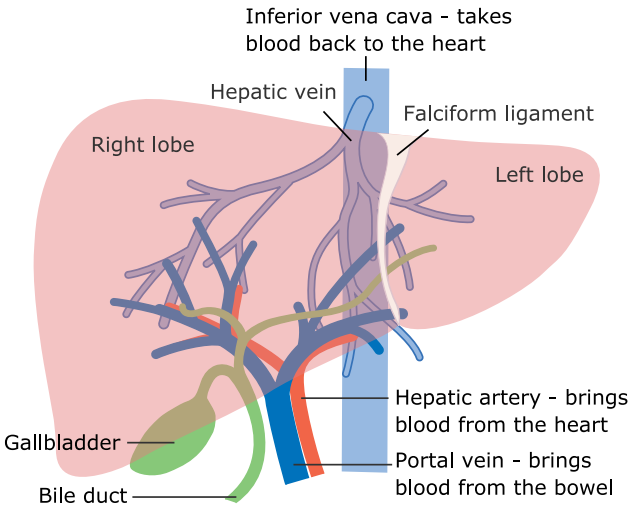

Figure 2.3: Blood supply of the liver. Modified from: Cancer Research UK / Wikimedia Commons

2.3 Contrast enhanced CT

CT imaging creates 3D images of a patient based on the x-ray attenuation of different tissues in the body. CT images can be acquired in seconds and can show the entire abdomen in high resolution (<1 mm voxel size within transverse slices and typically 2-3 mm between slices). Because the x-ray attenuation of blood and soft tissues is similar, the liver shows up as a uniform area on CT.

To make blood vessels, tumors and lesions within the liver visible contrast agent is used. The contrast agent is injected into the arm or leg of the patient, and it has a high x-ray attenuation, so the parts of the body that contain contrast agent will show up brightly on the scan.

2.3.1 Contrast phases

After the moment of injection the contrast agent will travel trough the body. Because of this the appearance of a scan will change depending on how long you wait after the injection. The passage of contrast through the body is divided into multiple phases, which are displayed in Figure 2.4 and explained more thoroughly below [11, 12, 13]. For liver tumor detection and characterization, scans at multiple phases are acquired. There is some variation in the protocols [12, 13], but a late arterial scan and a portal venous scan are included in both.

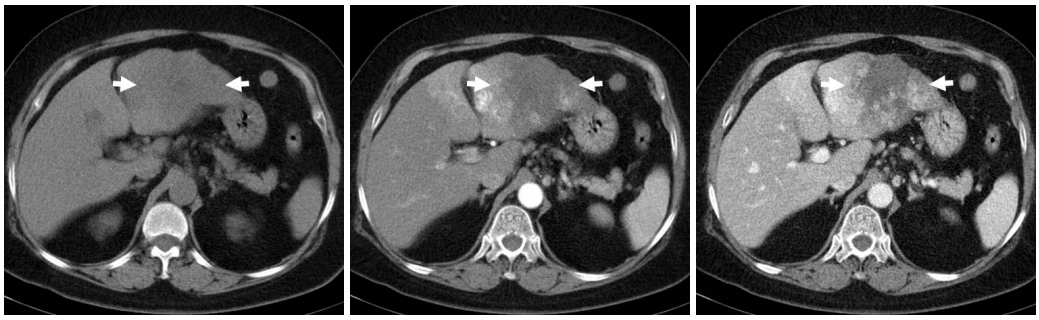

Figure 2.4: Multi-phase CT study of a 64-year-old woman with a tumor (sclerosing hemagioma). From left to right: pre-contrast image, arterial phase, portal-venous phase. Both the blood vessels (bright spots) and the tumor (indicated by arrows) are only recognizable after contrast injection. Source: Song et al. [14]

The first phase is the **arterial phase**. In this phase the contrast agent has been pumped back through the heart, and the contrast agent enters the liver through the liver artery. In the early arterial phase, around 20 seconds after contrast injection, the liver artery will be brightly visible. In the late arterial phase, around 30-35 seconds after contrast injection, hypervascular tumors will also show up brightly, because they will retrieve more blood from the artery than the surrounding tissue.

The second phase is the **portal-venous phase**, which occurs around 60-70s after contrast injection. During the arterial phase the contrast agent will also reach the intestines.

From the intestines the contrast agent will flow to the liver through the portal vein. At the same time contrast rich blood will flow away from the liver through the hepatic vein. Because of this, the portal vein, the hepatic vein, and their branches will be visible in the portal venous phase. This phase highlights most of the vessels within the liver. Hypovascular tumors take up less blood and contrast agent than the surrounding tissue so they can be best detected in this phase.

The last phase is the **equilibrium phase**, which occurs around 2-10 minutes after contrast injection. In this phase most of the contrast will have flowed away from the liver. Fibrotic lesions(scar like damaged tissue) will hold contrast longer, making them best visible in this phase.

The speed at which contrast agent propagates is dependent on the patient's physique and other factors [15], so the start time and duration of each phase can vary. Therefore, automatic bolus triggering may be used. This tracks whether the contrast agent has reached the region of interest by making repeated low dose scans of a small region of interest. When a certain contrast threshold is reached the high quality scan of the entire abdomen is acquired.

2.3.2 Downsides

Contrast enhanced CT scans have several downsides: (1) Contrast agent is toxic to the kidneys, which has a risk of causing kidney failure especially when the patient has renal impairment [4, 5] (reduced kidney function). This also limits the amount of contrast enhanced scans that can be acquired during an intervention. (2) CT scans expose the patient to radiation, which increases the risk of cancer. According to Smith-Bindman et al. [16] multiphase CT studies of the abdomen and pelvis have the highest median dose of all commonly performed CT studies at a median of 31mSv. The expected development of cancer caused by one multiphase abdomen and pelvis CT study varies with age and sex between 1 in 250 studies for a 20 year old female and 1 in 700 for a 60 year old female. It has to be noted that the radiation dose varies strongly between CT studies within this category, and that intra-operative CT scans are usually taken at low quality and within a small area, which reduces the radiation dose. (3) The scans can not be acquired in real time and the doctor and assistants have to leave the room during image acquisition, which makes CT scans unpractical for imaging during a procedure.

2.4 3D ultrasound

An ultrasound scanner creates an image inside of the body using high frequency sound waves. Ultrasound imaging has many advantages: Ultrasound scanning is harmless to the patient, the machines are portable, scans can be acquired in real time in 2D and almost real time ($\pm$5 Hz) in 3D, and the machines are cheaper than CT and MR scanners. However, US images can only look at a small area of the body at the same time and they are susceptible to artifacts. In this section we will briefly review US image reconstruction and what artifacts this can cause.

2.4.1 Physics and reconstruction

The transducer sends high frequency (1-18MHz) pulses into the body. As a pulse traverses through the body it will react to the tissues in roughly four possible ways: The pulse may traverse through the tissue, it may be reflected back, it may be converted into heat, or it may be scattered in all directions. Between sending two pulses the transducer measures the sound that is reflected back and these measurements are used to create an image.

Reflections occur on boundaries between tissues. The amount of sound that is reflected back depends on the acoustic impedances Z_1, Z_2 of the tissues and the angle between the pulse and the boundary θ. The impedance Z depends on the speed of sound c and density ρ of a tissue:

$$Z = \rho c \tag{2.1}$$

The reflection ratio R, describes the ratio between the reflected and transmitted pressure p_r, p_t. For perpendicular incidence it is defined as follows [17]:

$$R = \frac{p_r}{p_t} = \frac{Z_2 - Z_1}{Z_2 + Z_1} \tag{2.2}$$

By measuring the time between sending out the pulse and measuring the reflection, the distance of the reflecting edge can be measured. This assumes the speed of sound is constant within the body at 1540 m/s, which is approximately correct in soft tissue, but not in bone or air (Table 2.4.1).

Tissue	**Speed** (m/s)	**Acoustic Impedance** $(kg/m^2/s)$
air	330	0.0004×10^6
fat	1460	1.34×10^6
water	1480	1.48×10^6
liver	1555	1.65×10^6
blood	1560	1.65×10^6
muscle	1600	1.71×10^6
skull bone	4080	7.80×10^6

Table 2.1: Ultrasound characteristics of different tissues. Source: Konofagou [17]

A phased array 2D ultrasound transducer has an array of piezoelectric elements that can create a pulse at any angle within the plane parallel to the transducer by using beamforming (Figure 2.5). After sending a pulse at a certain angle the transducer will measure the reflected sound to determine the position of the reflectors along that angle. By sending and measuring for a range of angles a 2D image can be generated, where every measurement contributes to one line of the image.

3D images can be created by using a 2D matrix of transducer elements to steer the pulses with two angles. Alternatively, a 2D transducer can be moved or rotated over the body either manually or automatically and the resulting 2D images can be stitched together to obtain a 3D volume. Huang and Zeng [18] provide a review of how 3D ultrasound volumes can be obtained and rendered, and how they are used in practice.

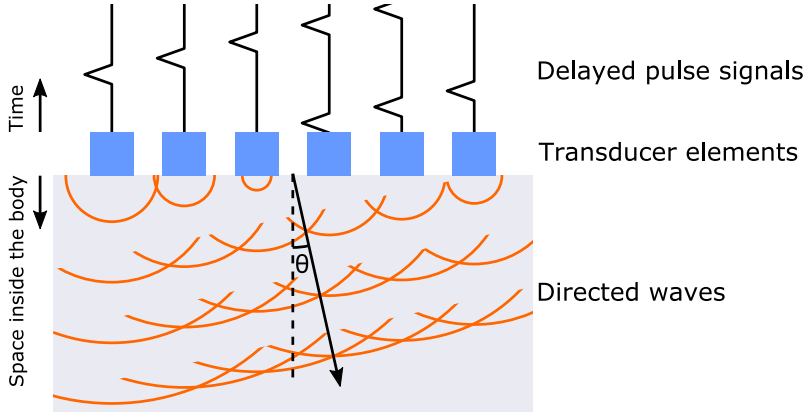

Figure 2.5: Illustration of beamforming: By sending the same pulse signal to an array of transducer elements with a different delay for each element, it is possible to create waves that propagate at any angle $\theta \in (-\frac{1}{2}\pi, \frac{1}{2}\pi)$.

2.4.2 Artifacts

While ultrasound imaging has many positive qualities, it is also susceptible to multiple types of artifacts. These artifacts tend to reduce image quality more than the artifacts that occur in CT or MR.

Shadows

Some interfaces between different kinds of tissue reflect almost all sound. This most often occurs within the body on interfaces including bone or air, and it can also occur when the transducer does not make good contact with the body. In those cases, the interface itself shows up brightly, but behind the interface nothing is visible. This is illustrated in Figure 2.6.

Speckle

Speckle is an artifact that has a noise like grainy appearance and it appears all over the image. It is caused by interference of scattered waves [19] (Figure 2.7). Bright spots are caused by constructive interference and dark spots by destructive interference. It is a deterministic result of the pulse signal, but because it is not modelled during image reconstruction its position does not directly correspond to any structure inside the body. Nevertheless, the speckle pattern does stay the same over time and over small movements, so it has been used to track images over time [20, 21].

Orientation dependence

The intensity of a reflection is not only dependent on the tissue properties, but also on the angle of the interface relative to the propagation direction of a wave. Edges perpendicular

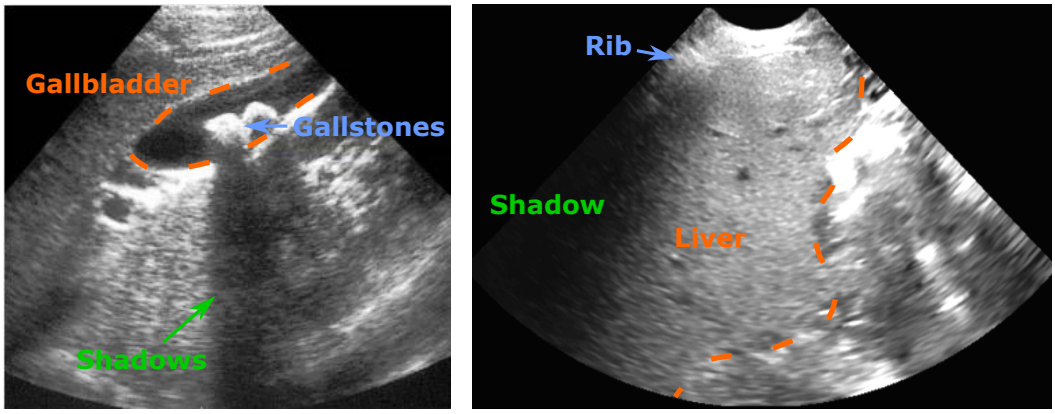

Figure 2.6: Left: Example of a shadow in 2D ultrasound, caused by gallstones in the gallbladder, source: Støylen [22]. Right: Example of a shadow in our dataset.

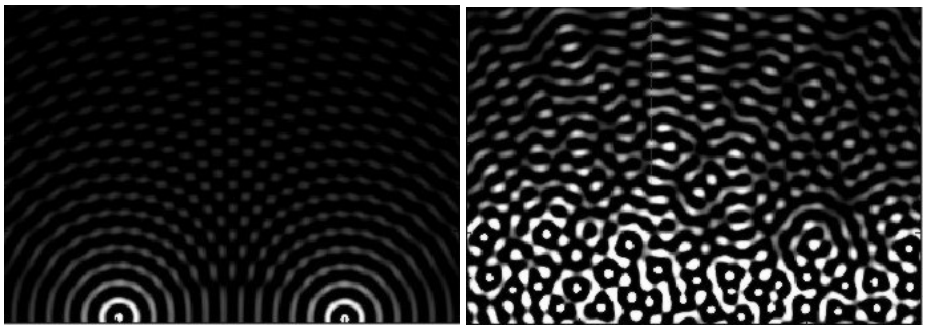

Figure 2.7: Left: Wave pattern of two sources. Right: Wave pattern of several randomly placed sources. Similar patterns can be seen in US images. Source: Støylen [22]

to the scan direction show up brightest, and the more close to parallel an edge is to the scan direction the less clearly visible it will be. Because of this, the same structures will look different when scanned from different directions.

Time gain compensation

Reflections that occur further inside the body in general reflect less power to the transducer because there is more tissue in between. To compensate for this, the US machine amplifies measured reflections more when they are further away. How this amplification is performed exactly varies between machines, and it often contains user tunable parameters. Therefore, this can be a source of variation between different scans even when they are acquired with the same scanner.

Log compression

The typical distribution of the measured ultrasound intensities contains mostly low values with a few values that are much higher, so a typical image would look very dark, with a few bright spots. To reveal more information, a logarithm-like function is applied to each pixel. Log compression also reduces the visual effect of the orientation dependence of the scanner. Again, the exact function is machine dependent and may contain user tunable parameters.

2.5 Coordinate transformation models

The result of a registration is a set of transformation parameters that maps the coordinate system of one scan onto the other. How the coordinates of one scan can be mapped from one scan to the other is defined in the coordinate transformation model. Selecting the right transformation model is a trade off between accuracy and problem complexity. If you use too few parameters you will not be able to register the images accurately. However, if you use too many parameters your registration result may contain deformations that are not physically possible, and your optimizer might get stuck in a local optimum. Therefore many registration approaches increase the transformation model complexity as the registration progresses [23, 24]. Some of the most common transformation models are described below:

Rigid transformations allow for a translation and rotation between the images. Angles, distances, lines and planes are preserved. It can be described as a 3D vector for the translation and another 3D vector or a unit quaternion for rotation.

Affine transformations are a more general model, allowing for translations, rotations, scaling along each axis, mirroring and shearing. Lines and planes remain lines and planes, but angles and distances can change. Affine tranformations can be described by a 3x3 matrix and a 3d vector.

Deformable transformations allow for locally different bending. Cubic B-splines [25] are often used, where the local transformation parameters are specified on a grid, and between the grid points, the transformation is interpolated. Deformable transformation models are typically unable to describe big rotations and translations and they use many more parameters than rigid or affine transformations, making them not suitable to be directly used for initialization.

In the case of 3D US CECT liver registration, the position of the US transducer can vary, which can be described as a rigid transformation. These motions can be tracked using an EM or optical tracking system. There are also several causes of non-rigid deformations: The pose of the patient can differ, the US transducer and other medical instruments may apply pressure to the body and the respiratory(breathing) motion also causes deformation. When the CT scan is taken the respiratory state can be controlled by asking the patient to hold his/her breath at inhalation or exhalation, but this is not possible during the intervention because the patient is sedated with the natural breathing intact.

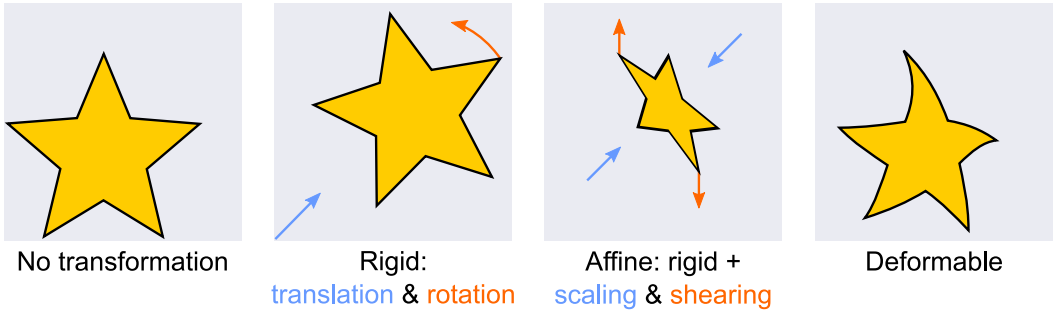

Figure 2.8: 2D illustration of different transformation models

The non-rigid deformation of the liver caused by respiratory motion was measured by Rohlfing et al. [26] on gated MRI data. They measured the distances of the liver surface and the biggest vessels between rigid and non-rigid registrations. These distances are on average around 10 mm and at most 34 mm in their measurements. According to Wein et al. [27] most of the deformations caused by respiratory motion occur in the saggital plane.

The non-rigid deformation inside of the liver between intra- and pre-operative CT scans for liver tumor ablations was measured by Luu [9]. They measured the distances of equally sampled points within the liver between rigid and non-rigid registrations. The average measured error in rigid deformations over the entire liver was 5.9 $\pm$1.6 mm. They also measured the error within the field of view of a 3D US scan acquired from either a inter-costal(between the ribs) or a sub-xiphoidal(under the sternum) position, which are also the acquisition positions in our dataset. The measured error was 4.6 $\pm$ 1.2 for inter-costal scans and 4.8 $\pm$ 1.5 for subxiphoidal scans. However, they did not take into account the extra deformations that are caused by pushing the US transducer against the body

Chapter 3

Prior Work

In this section, the prior work is reviewed. Not a lot of research has been preformed on registration initialization between 3D US and CECT scans. Therefore, a broader overview of the literature on multimodal US registration and registration initialization is given.

Most work on multimodal US registration uses an intensity based approach, but these methods are not very suitable for finding an initialization. Nevertheless they give an indication of what kind of features can be used for US registration, so they are reviewed in Section 3.1. Feature based approaches can be used for initialization, but we could only find two feature based methods for US registration: one method was only evaluated on two scans and the results of the other method were not reproducible. They are presented in Section 3.2. Another way to find an initialization is to use global search. To the authors' best knowledge this has not yet been applied for US registration initialization, but good results have been achieved in registration initialization between other modalities. These results are presented in Section 3.3.

3.1 Intensity based multi-modal US registration

Intensity based registration approaches formulate registration as an optimization problem. A cost function is defined that is based on the features that are overlapping between a fixed image, and an image that is being transformed according to a coordinate transformation model. The transformation parameters with minimum cost should describe the transformation from one image space to the other.

Images acquired with different modalities display the same tissues in different ways, so to be able to use a cost function for registration, the cost function should be mostly invariant to these differences in appearance. Commonly used cost functions are the correlation ratio and mutual information [28, 29]. The correlation ratio between two images is invariant to linear intensity changes, so when you apply a function $ax+b$ to the intensity of all voxels the optimum will not change. Mutual information(MI) is in theory invariant to remapping intensities using any bijective function f, as long as the same function is used on all voxels. In practice MI is calculated using a joint histogram, so there might be some changes due to the discretization into bins, especially when f is not a smooth function. A differ-

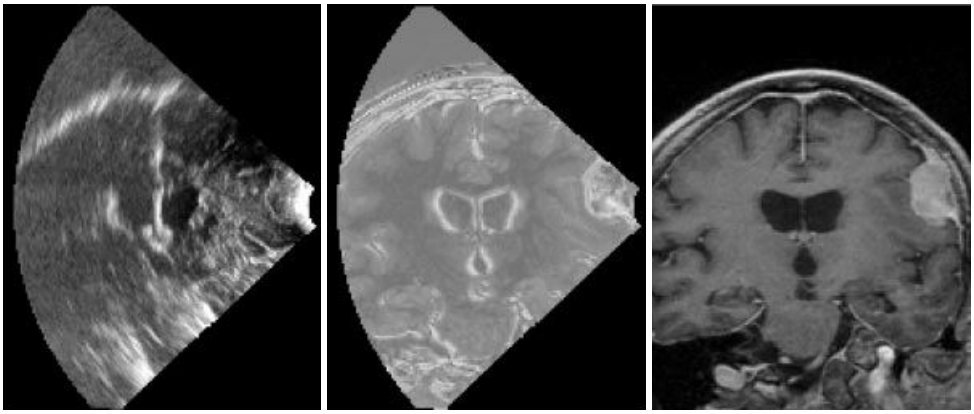

Figure 3.1: US simulation from MR using intensity and gradient magnitude as described in Roche et al. [33]. Left: US image, middle: US simulation, right: MR image slice

ence between ultrasound and other modalities such as MR or CT is that ultrasound images display a higher intensity on the boundary between different tissues. Cross correlation and mutual information are not invariant to such changes, so cost functions have been developed specifically for multi-modality registration with ultrasound.

To optimize the cost function an iterative local search strategy is used. Local search strategies start with a certain set of parameters and search in every iteration for a small change of parameters that will improve the cost function. Commonly used local optimization strategies in multimodal US registration are the Nelder-Mead method [30] and the combined methods of Powell [31] and Brent [32]. Both methods do not use the gradient of the cost function, so the cost function does not have to differentiable. Because local search algorithms only take small steps that have to improve the cost function, they tend to converge to a local optimum that is close to the initial set of parameters. Because intensity based cost functions typically have many local optima a close initialization is required.

3.1.1 MR-2D US registration

Roche et al. [33, 34] proposed a method for rigid MR to 2D US registration based on a generalization of the correlation ratio cost function. Their cost function is invariant to certain combinations of the intensity and the gradient magnitude of the MR volume. At every iteration of the optimization process, a new volume $\hat{I}$ is calculated from the MR volume intensity I and gradient G that is supposed to look more like an US image. For every voxel $\boldsymbol{x}$ the following formula is applied:

$$\hat{I}(\boldsymbol{x}) = \sum_{p=0}^{3} \sum_{q=0}^{3-p} w_{pq} I(\boldsymbol{x})^p G(\boldsymbol{x})^q \tag{3.1}$$

The weights w_{pq} are chosen in such a way that they minimize the difference between $\hat{I}$ and the US image for the current transformation. This can be formulated as a weighted least squares problem, which can be solved using singular value decomposition. One result

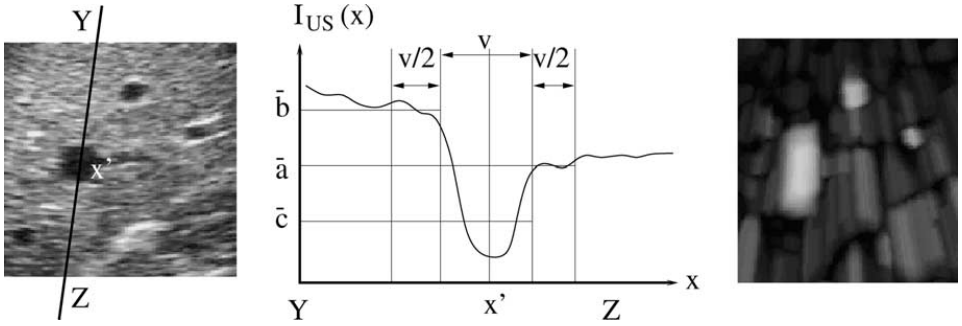

Figure 3.2: Dip image calculation from Penney et al. [35]. Left: scan direction in US image, middle: intensity profile along line YZ and the regions where the central $(\bar{c})$, before $(\bar{b})$ and after $(\bar{a})$ mean intensity values are calculated, right: resulting dip image

is shown in Figure 3.1. Afterwards, one update step of the Powell local optimization method [31] is performed on the cross correlation between $\hat{I}$ and the US image. After that a new iteration starts, and these two steps are repeated until convergence.

In another approach, by Penney et al. [35], multiple 2D images that were tracked using an optical tracker are rigidly registered to an MR volume. First the vessel probability for each pixel in the US images and for each voxel in the MR volume is approximated. A custom derivative free local optimizer is used for optimizing the transformation parameters over the cross correlation between the vessel probability images from the US and MR scans.

The US vessel probability images are obtained in three steps. First the region of shadow artifacts is estimated by tracing rays from the back of the scan towards the scanner and discarding all pixels until a threshold is reached. Secondly, local intensity dips are calculated and stored in an image: For each pixel in an US image, along the scan direction, the average intensity is compared between the center region $\bar{c}$ and the two regions before $\bar{b}$ and after $\bar{a}$ it. This is illustrated in Figure 3.2. In the third step the US intensity and dip image value are combined into the US vessel image by means of a lookup table. The lookup table was trained on manual vessel annotations. For the MR images, another lookup table was trained on manual annotations that directly mapped the MR intensity to the vessel probability.

3.1.2 CT-3D US registration

A method for registering 3D US and CECT images was developed by Wein et al. [36, 27]. A 2D US transducer with a small motor was used to create a sweep of 2D US images of the liver in a range of angles. Not all images are used: the range of angles is divided into 20-30 bins and for every bin the image with maximum entropy is selected. Two features are calculated for every pixel of the CT scans, which are combined in the cost function. This is similar to the method of Roche et al. [34], but the features of Wein et al. are more strongly based on US physics. The first feature p is the estimated tissue density, which can be obtained by applying the mapping displayed in Figure 3.3 to the CT intensities. The density of a tissue determines the intensity of a uniform region of that tissue in an US scan. The difference between two densities determines the intensity of the boundary between

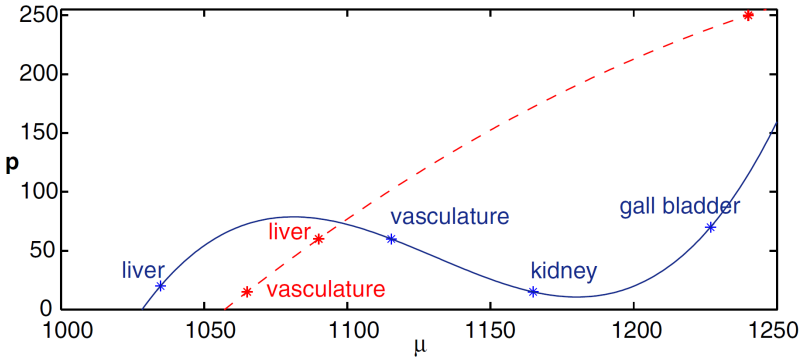

Figure 3.3: Mapping from the intensities of normal CT(red dashed line) and contrast enhanced CT(blue line) to tissue density, which is important in US images [27].

.

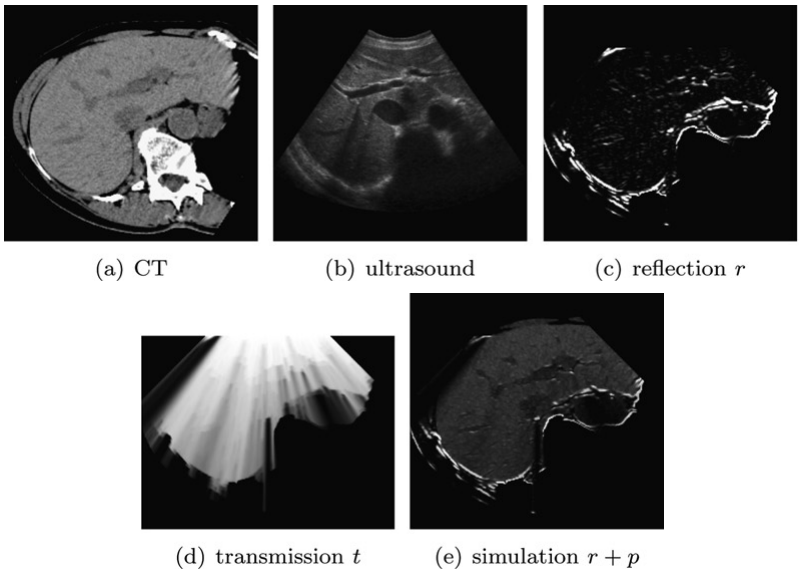

Figure 3.4: US simulation from CT [27]. (a) and (b) are the original US and CT images, (c) is the simulated reflection, (d) is the simulated transmission and (e) is the combination of the simulated density and reflection images

those two tissues in an US scan. This is simulated by tracing rays from the scanner through the image and at each pixel updating how much the ray is reflected and how much energy is left. This is Illustrated in Figure 3.4.

To combine the estimations of the density $p(\boldsymbol{x})$ and reflection $r(\boldsymbol{x})$ at pixel $\boldsymbol{x}$ a linear model is used: $f(\boldsymbol{x}) = \alpha p(\boldsymbol{x}) + \beta r(\boldsymbol{x}) + \gamma$. The weights α, β and γ are updated for every pixel separately at every update step of the optimizer. They are calculated by applying linear regression in a 11x11 pixel neighborhood around the pixel of interest. Because the US simulation depends on the current registration estimate, the simulation quality will get better

when the registration progresses. The final cost function is the correlation ratio between the real US image and the simulated US image. For optimization the Nelder-Mead method [30] is used to fit a rigid model. After that, optionally a semi-affine model can be fitted, which includes one shearing and two scaling parameters in the sagittal plane.

3.1.3 Block Matching

A different method for CECT-3D US registration was developed by Banerjee et al. [8, 10]. They used block matching [37] as the optimization strategy. First random points were selected in the US scan. A 15mm^3 block of the US scan centered around each selected point is matched to the CECT scan with the multiple correlation coefficient between US intensity and CECT intensity and gradient as the cost function. To find the optimal corresponding block, exhaustive (grid) search is applied in a 40 mm^3 region around the corresponding center point in the CECT scan according to the initial registration estimate. Some of the found correspondences might be erroneous, so an outlier rejection step is performed. This is based on two criteria: The geometric criterion specifies that two pairs of matched points should have similar distance in both scans. The smoothness criterion specifies that the displacement vector from US to CECT should be similar for points that are close together. On the remaining points an affine model is fit minimizing the squared distance between the two scans [38].

The original block matching paper by Ourselin et al. [37] used a simpler form of outlier rejection. It assumes that outliers have a bigger distance between the scans than inliers. Therefore only the 50% of the point pairs with the lowest distance between them are included when calculating the transformation. This approach is called least trimmed squares (LTS) and it originated from robust (linear) regression [39]. Modat et al. [40] made block matching symmetrical by sampling points in both scans and then matching in both directions. This improved the performance over the asymmetrical (original) approach on all their registration experiments between images acquired using different MR sequences(T1, T2& PD), CT and PET.

3.1.4 Initializing intensity based methods

All the US registration methods in this section rely on an initialization for the registration to converge to the global optimum. In the method of Roche et al. [33, 34] the initialization was entered manually, but they "always made sure that a slight misalignment was still visible", so that the images were not already registered before running the algorithm. The approach by Penney et al. [35] used a camera tracking system for initializing the rotation and they manually aligned the translation by selecting the center of the IVC in both images. Moreover, they only registered images at maximum exhalation so there were no deformations caused by breathing. The method by Wein et al. [36, 27] either initializes the rotation parameters using an electromagnetic tracker or by entering an estimate manually. Afterwards an exhaustive search is performed over all translations where the US transducer is on the skin. All translations "suggesting an optimum" are further optimized using a local

optimization approach. Finally Banerjee et al. [8] used a manual initialization, based on selecting multiple corresponding points in the US and CECT scans.

3.2 Feature based multi-modal US registration

Feature based approaches try to detect corresponding features in both scans, and derive the transformation parameters so that each feature is matched as closely as possible. The detection of features is not dependent on the orientation of the scans, making these approaches suitable for initialization.

3.2.1 Registration based on IVC and liver boundary segmentation

A feature based method for registering MR to 3D US volumes was proposed by Weon et al. [41]. They assume the edge of the liver and the inferior vena cava(IVC) can be segmented in both modalities. Furthermore they assume that close to the liver the IVC can be approximated by a straight line. For both modalities the direction of this line is calculated by taking the largest principal component of the segmentation. When the lines that are found in both modalities are aligned only two degrees of freedom remain: the translation along the line, and the rotation around the line. Both the translation and rotation are sampled at fixed intervals, and for each combination a ray is cast away from the IVC line to calculate the distance to the closest point on the liver surface in that direction. These results are stored in a geometric distance map. These two distance maps are aligned by minimizing their average absolute distance, and from the alignment result the translation and rotation parameters can be retrieved. The method was evaluated on only two scans resulting in a mean error between annotations of $4.10 \pm (0.54)$ and $4.39 \pm (0.62)$.

The above method is extended by Hwang et al. [42] by specifying how to segment the IVC and the liver edge in both scan modalities. In the MR scan the IVC is segmented using a level set approach and a single manually placed seed point. In the US scan they use adaptive thresholding for segmenting both the surface and the IVC. The threshold is based the mean and standard deviation within a local window, but they do not specify the window size. As a next step for the IVC segmentation one connected component is selected that has a shape similar to a cylinder and has the biggest diameter. Again this method was only evaluated on (possibly the same) two scans, resulting in an average annotation error of $11.31 \pm (3.37)$.

3.2.2 Registration based on vessel segmentation

Another method for automatic US-CECT registration was developed by Nam et al. [43]. They use a segmentation of the liver vessels and liver boundary as features. Because the US image has a limited field of view and less image quality, they only try to find some vessels and a part of the liver surface in the US image, while they assume the whole segmentation is available for the CECT scan. For segmentation in the US scan they first filter the image to remove noise and speckles. After that they use Hessian based filtering to detect plane like structures, which should be parts of the boundary of the liver. After applying a threshold they only use the largest connected component, which should be the interface between the

liver and the lung. For segmenting the vessels they first fit a quadratic surface through the liver segmentation and only use the part above the surface as a rough estimate of the liver volume. Within this volume, local mean and variance based thresholding is used to find blood vessel candidates. For each connected component the similarity in shape to a vessel is calculated using a shape measure and if this is above a threshold the component is included within the vessel segmentation.

In the first step of the registration only the blood vessels are registered. To do this, the centerlines of the vessels in the US and CT scans are extracted and sampled at a constant interval. Each edge between two samples in the US segmentation is matched to an edge in the CT segmentation in such a way that the relative distances and angles are preserved as good as possible. To find this matching, the Viterby algorithm is used. From the corresponding edge centerpoints the optimal rigid transformation is calculated using a singular value decomposition [44]. In a second step this result is refined by using the iterative closest point(ICP) method on the vessel and liver boundary segmentations to find the optimal affine transformation with the result from the previous step as the initialization.

This method was evaluated on a simulated dataset and on 20 scans obtained from a mechanical 3D US scanner, which were obtained so that the liver boundary and vessels were conspicuous. 6-8 corresponding points were annotated by a radiologist in the clinical dataset, and the mean RMS error after the first step was $4.30(\pm 1.07)$ and after the second step $3.0(\pm 0.95)$. Manual rigid registration had an error of $3.59(\pm 1.0)$.

This method appears to solve the same problem that we are aiming to solve. However, their results have proven difficult to reproduce: They don't report the values that were used as thresholds and both their code and dataset are not made public. We tried implementing the liver boundary segmentation method ourselves. This did not work well on our dataset. The Hessian based filter did output high values on the edge of the liver, but it also outputted high values on other parts of the scan and these parts were often connected. Therefore we were unable to find a threshold for which the largest connected component contained only a substantial part of the liver boundary and nothing else. We also emailed the authors but they did not reply.

3.3 Global optimization in registration

Feature based methods require a certain the structure of the detected features to be able find the transformation parameters, and intensity based methods require an initialization. An alternative is to use a global optimization algorithm. These algorithms can be applied to any cost function to find their global optimum, which allows for more freedom in designing a solution. However as proven by the No Free Lunch Theorem for Optimization [45] "All algorithms that search for an extremum of a cost function perform exactly the same when averaged over all possible cost functions. So, for any search/optimization algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class." Therefore, if you don't make any assumptions about your cost function, you cannot make any assumptions about the performance of you optimization either. Nevertheless, global optimization approaches have been successfully used in some areas of registra-

tion problems, so they give an indication of what kind of cost functions and optimizers may work well in registration problems.

3.3.1 Evolutionary algorithms and other metaheuristics

Evolutionary algorithms are optimization algorithms inspired by the biological process of evolution. They maintain a population of possible solutions to the optimization problem that you want to solve. From the first population new generations are generated by applying operations on the previous generation. Typical operations are *mutations*, which cause a small modification to one or more parameters and *crossover*, which combines parameters from two members of the population. These operations are applied multiple times until ideally all members of the population converge to the same optimal solution. There are many different evolutionary algorithms, using different ways of encoding solutions and different operations to generate new solutions. Metaheuristics are a broader class of algorithms which includes evolutionary algorithms. They are used to solve the same kind of problems, but they are not necessarily inspired by evolution. Many of these algorithms use other parallels to nature such as ant colony optimization [46] or simulated annealing [47], which is inspired by a chemical process. In a recent survey by Boussaïd et al. [48] many metheheuristics, including evolutionary algorithms, are reviewed.

Evolutionary algorithms and other metaheuristics have been used to solve registration problems over a large search space. In the overview of Damas et al. [49], 8 evolutionary algorithms, 5 other metaheuristics and 2 robust local optimization methods were compared for use in registration. A brain and a wrist dataset were used, from which ridge line features were extracted. A transformation had to be found over a space of all rotations, $\pm 40mm$ translation over all axes and $(0.5, 2)$ times uniform scaling. Scatter search [50] performed best, closely followed by differential evolution [51, 52].

Another evolutionary algorithm was used by Otake et al. [53] to solve the 2D X-ray to 3D CT registration problem. They applied the covariance matrix adaptation evolution strategy(CMA-ES) evolutionary algorithm to find an automatic registration with a 99.993% success rate over a search space of $\pm 200mm$ translation along each axis and $\pm 10^{\circ}$ rotation around each axis.

3.3.2 library based search

When performing a registration during an intervention you want the run time to be as short as possible. Sometimes it is possible to use a computationally expensive method, by doing most calculations before the intervention, when a long run time is less of a problem. This approach has been used multiple times in X-ray to CT registration initialization [54, 55, 56, 57]. Before the intervention, a library is generated by simulating X-Ray images from the pre-operative CT scan at many orientations and performing further pre-processing. When the real X-ray image is acquired during the intervention, the cost function can be evaluated quickly for all orientations available in the library.

Chapter 4

Method

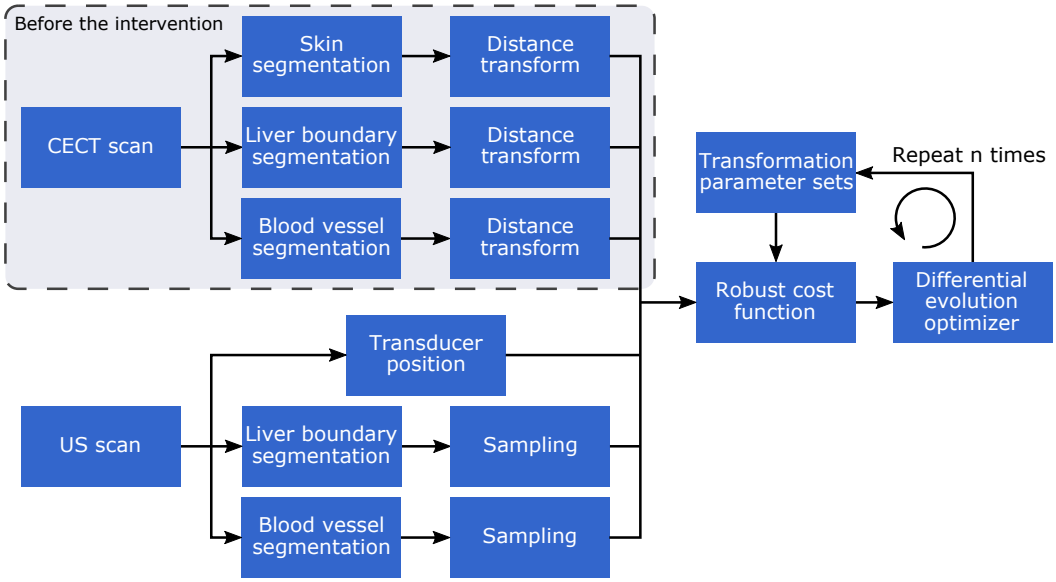

Figure 4.1: Overview of the registration pipeline

To be able to find an initial registration from any position of the US transducer that shows the liver, we present a global optimization based method, based on the work by Santamaría et al. [50]. In the cost function we use the distance of the liver boundary and the blood vessels between both scans. Both features have been used for multimodal US registration before [35, 43, 41, 42]. Neural network based segmentation is used to extract these features in the US scan, and more traditional approaches and already available manual labels are used for segmenting the CT scan. Moreover, we also include the distance from the US transducer to the skin in the cost function, similar to the initialization method used in Wein et al. [27]. The method is illustrated in Figure 4.1. To reduce the computation time during the intervention, computations are performed before the intervention whenever possible.

4.1 Registration cost function

A cost function is defined to quantify the goodness of fit of a coordinate transformation T between the two scans. The cost function we use is a weighted combination of three cost functions:

$$C_{registration}(T) = w_{liv}C_{liv}(T) + w_{ves}C_{ves}(T) + C_{trans}(T) \tag{4.1}$$

$C_{liv}(T)$ describes the fit of the segmented liver boundaries, $C_{ves}(T)$ describes the fit of the segmented blood vessels and $C_{trans}(T)$ describes whether the US transducer is in a feasible position. w_{liv} & w_{ves} are weights that can be used to take each part more or less into account. A separate weight for $C_{trans}(T)$ is not needed, because only the ratio between the weights has influence on the optimum of the cost function. We combine the different cost functions in this way to eliminate false optima. Each cost function component may have a low cost for several values of T, but we only expect them all to have a low cost when the scans are correctly aligned. $C_{liv}(T)$ and $C_{ves}(T)$ both describe the distance between two segmentations, so the same distance function can be used for both functions. For $C_{trans}(T)$ a different approach is used, because it consists of only one point.

4.1.1 Quick distance lookups using sampling and the distance transform

To quantify the distance between two segmentation masks we use the distance from sampled points in the US segmentation to the the closest point that is part of the CT segmentation. Only the distance from US to CT was used instead of from CT to US or both directions, because the US scan area does not cover the entire liver, and the segmentation of the CT scan is more reliable. In other words, we assume that whenever a structure is segmented in the US it will also be available in the CT segmentation, but not the other way around.

To sample points in the US segmentation mask, first the coordinates of all voxels in the segmented area are stored in a list by iterating over the volume line by line. Afterwards, an equally spaced subset of 1000 samples is selected from that list. This results in a more or less evenly spaced set of samples in 3D as well.

To be able to quickly look up the distances of these points to the closest point in the CT segmentation, the euclidean distance transform is calculated for each CT segmentation. In a distance map every voxel corresponds to the distance to the closest voxel within the segmentation mask. It can be calculated in linear time [58] before the intervention, because the CT scans are acquired before the intervention. This step is illustrated in Figure 4.2.

4.1.2 Robustness to segmentation errors

The available segmentations will not be perfect, so we present three cost function variants that provide some robustness to segmentation errors. All three can be used for $C_{liv}(T)$ and $C_{ves}(T)$, and we will evaluate how well each one performs in chapter 5.

Least trimmed squares of closest distances(LTS-CD)

The LTS-CD cost function takes the average over the fraction f lowest distances between closest points, with f being a tunable parameter. When n points are sampled in the US

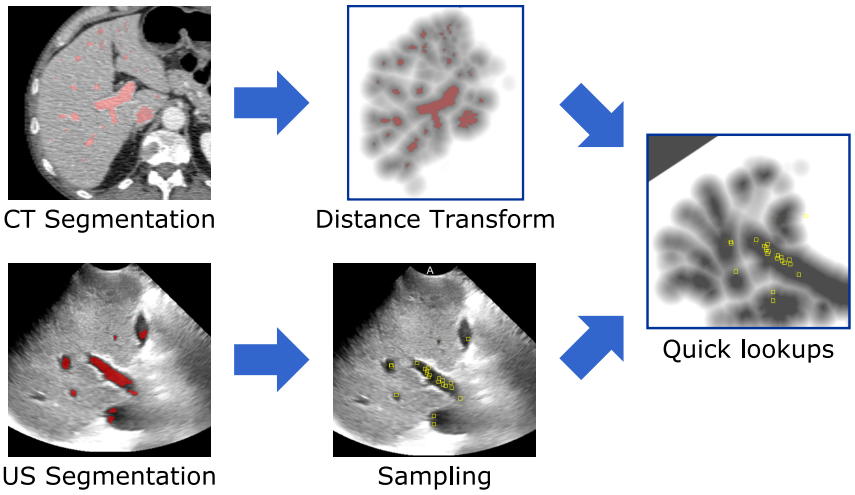

Figure 4.2: The distance to the closest point in a segmentation can be quickly looked up for multiple points by calculating the euclidean distance transform.

segmentation, $D_{sorted}(i,T)$ returns the sorted distances to the closest points in the CT segmentation for a given transformation T. The distances are sorted in ascending order and i is the index in the sorted list, so $D_{sorted}(0,T)$ would return the distance of the sample closest to the CT segmentation. Using this, LTS-CD is defined as:

$$C_{LTS-CD}(T,f) = \frac{1}{\lfloor f \cdot n \rfloor} \sum_{i=0}^{\lfloor f \cdot n \rfloor} D_{sorted}(i,T)^2 \tag{4.2}$$

The assumption that is made in this cost function is that distances corresponding to outliers are usually higher. Therefore only the smallest $f \cdot n$(rounded down) distances are included and the others are discarded. We slightly modified the original LTS [39] formulation by adding the weight $\frac{1}{\lfloor f \cdot n \rfloor}$. This does not affect the optimum of C_{LTS-CD} by itself, but it does affect the optimum when the different cost function parts are combined into $C_{registration}$.

Trimmed average closest distance(TACD)

The TACD cost function is the same as the LTS-CD cost function, but without squaring the distances. The LTS-CD is similar to the L2-norm of the closest distances, while TACD is similar to the L1-norm, which is generally less sensitive to outliers.

$$C_{TACD}(T,f) = \frac{1}{\lfloor f \cdot n \rfloor} \sum_{i=0}^{\lfloor f \cdot n \rfloor} D_{sorted}(i,T) \tag{4.3}$$

Clipped average closest distance(CACD)

The CACD cost function takes the average of all distances to closest points, but when the distance to the closest point is far away the distance is clipped to some maximum value

d_{max}, which is selected by the user. Just like LTS-CD and TACD, it also assumes false correspondences between segmentations will result in bigger distances, but it uses a different method for handling these distances. For this cost function the distances to the closest point $D(i,T)$ do not have to be sorted.

$$C_{CACD}(T,f) = \frac{1}{n}\sum_{i=0}^{n} \min\left(D(i,T), d_{max}\right) \tag{4.4}$$

4.1.3 Transducer distance function

For the distance between the US transducer and the skin a different distance function is used because the skin is flexible and can deform by the pressure applied to the transducer. To account for this, the transducer distance is only taken into account if it is bigger than a user defined threshold d_{min}. Here D_{trans} represents the distance from the transducer position to the closest point on the skin in the CECT scan.

$$C_{trans}(T, d_{min}) = \max\left(D_{trans}(T) - d_{min}, 0\right) \tag{4.5}$$

4.2 Registration optimization

The Self-adaptive Differential Evolution(SaDE) [59] is used to optimize the transformation parameters over the cost function. This algorithm is an extension of the Differential Evolution algorithm, which performed well in the comparison by Damas et al. [49].

4.2.1 Transformation parameterization

Because the cost function is based on looking up US samples in CT distance transforms, the optimizer will not try to find the transformation from the CT space to the US space directly. Instead the optimizer will try to find a transformation from US to CT, which can be inverted to get the transformation from CT to US.

First a constant matrix is applied to the US samples that transforms the US voxel coordinates to the distance in millimeters from the center of the scan. Secondly a rotation followed by a translation is applied, which are optimized by SaDE. Lastly the positions in millimeters are transformed to CT voxel coordinates using another constant matrix.

The translation is parameterized as a 3D vector and the rotation is parameterized as a unit quaternion, resulting in a total of 7 variables to be optimized. A unit quaternion is used to describe the rotation because it does not exhibit gimbal lock like Euler angles.

4.2.2 Differential Evolution

The Differential Evolution optimizer [52] tries to find the global optimum of the cost function C by calculating the cost function for each transformation of a population P for G generations resulting in a total of $P \times G$ evaluations. In the first generation each transformation is randomly generated in such a way that the center of the US scan is within the liver and a random rotation around the center is applied.

After initialization, each following transformation $T_{p,g}$ is generated from the transformation in the previous generation $T_{p,g-1}$ and three randomly selected transformations $T_{r_1,g-1}, T_{r_2,g-1}, T_{r_3,g-1}$ where $r_1 \neq r_2 \neq r_3 \neq p$. First a mutation vector is generated using the following equation, which relies on a user specified mutation factor F:

$$T_{mutation,p,g} = T_{r_1,g-1} + F\left(T_{r_2,g-1} - T_{r_3,g-1}\right) \tag{4.6}$$

Secondly a crossover operation is applied between each parameter $t_0 \ldots t_6$ of the mutated vector and the vector of the previous generation. The crossover operation is based on a user specified crossover ratio CR, and the result is called a trial vector.

$$t_{trial,p,g,i} = \begin{cases} t_{mutation,p,g,i} & \text{if } rUni(i) < CR \text{ or } rInd(p) = i \\ t_{p,g-1,i} & \text{else} \end{cases} \tag{4.7}$$

$rUni(i)$ is a random uniform number $[0,1]$ unique for each transformation parameter and $rInd(p)$ is a random index $[0,6]$ to make sure that at least one transformation parameter is crossed over.

Finally a trial vector is only accepted into the new generation when its cost is lower than the cost of the previous vector in that position:

$$T_{p,g} = \begin{cases} T_{cross,p,g} & \text{if } C_{registration}\left(T_{trial,p,g}\right) < C_{registration}\left(T_{p,g-1}\right) \\ T_{p,g-1} & \text{else} \end{cases} \tag{4.8}$$

Because of the random nature of the algorithm it is not guaranteed to converge to the global minimum of the cost function. To reduce the chance of getting stuck in a local optimum, the algorithm is started multiple (R) times. Because of the random initialization and random mutations the restarted algorithm may converge to a different optimum. In the end the result with the lowest cost function value is selected as the final result.

4.2.3 Self-adaptive Differential Evolution (SaDE)

To optimize $C_{registration}$ we use the SaDE algorithm Qin et al. [59]. This algorithm extends normal differential evolution by adding multiple combination strategies for generating new transformations. The idea is that each strategy has different strengths that may be more or less useful depending on the problem, the problem instance and the current state of the optimization process. Moreover, SaDE adaptively tunes the occurrence of each strategy and their parameters, such as CR and F, so they can change over time, and they no longer have to be tuned by the user.

SaDE combination strategies

In total four different combination strategies are used in SaDE, which are listed below.

DE/rand/1/bin:

$$T_{mutation,p,g} = T_{r_1,g-1} + F\left(T_{r_2,g-1} - T_{r_3,g-1}\right) \tag{4.9}$$

$$t_{trial,p,g,i} = \begin{cases} t_{mutation,p,g,i} & \text{if } rUni(i) < CR \text{ or } rInd(p) = i \\ t_{p,g-1,i} & \text{else} \end{cases}$$

DE/rand/2/bin:

$$T_{mutation,p,g} = T_{r_1,g-1} + F\left(T_{r_2,g-1} - T_{r_3,g-1}\right) + F\left(T_{r_4,g-1} - T_{r_5,g-1}\right) \tag{4.10}$$
$$t_{trial,p,g,i} = \begin{cases} t_{mutation,p,g,i} & \text{if } rUni(i) < CR \text{ or } rInd(p) = i \\ t_{p,g-1,i} & \text{else} \end{cases}$$

DE/rand-to-best/2/bin:

$$\begin{aligned} T_{mutation,p,g} &= T_{p,g-1} + F\left(T_{best,g-1} - T_{p,g-1}\right) \\ &\quad + F\left(T_{r_1,g-1} - T_{r_2,g-1}\right) + F\left(T_{r_3,g-1} - T_{r_4,g-1}\right) \end{aligned} \tag{4.11}$$
$$t_{trial,p,g,i} = \begin{cases} t_{mutation,p,g,i} & \text{if } rUni(i) < CR \text{ or } rInd(p) = i \\ t_{p,g-1,i} & \text{else} \end{cases}$$

DE/current-to-rand/1:

$$T_{trial,p,g} = T_{p,g-1} + K\left(T_{r_1,g-1} - T_{p,g-1}\right) + F\left(T_{r_2,g-1} - T_{r_3,g-1}\right) \tag{4.12}$$

SaDE adaptive strategy selection

The strategy for generating each new trial vector is randomly selected with the probability of each strategy based on the results in a learning period of a number of generations *LP*. When the algorithm starts, the probability of selecting each strategy is equal, and they will remain equal for the first *LP* generations. During that period a success memory and a failure memory are built up, storing for each generation g and each strategy k how many transformation vectors were successfully selected to be used in the next generation $(ns_{k,g})$ and how many transformation vectors failed to improve the cost function $(nf_{k,g})$. Using this information, the probability p_{k,g_c} of each strategy for the current generation g_c is calculated as follows:

$$p_{k,g_c} = \frac{S_{k,g_c}}{\sum_{k=1}^{4} S_{k,g_c}}, \quad \text{where} \quad S_{k,g_c} = \frac{\sum_{g=(g_c-LP)}^{g_c-1} ns_{k,g}}{\sum_{g=(g_c-LP)}^{g_c-1} \left(ns_{k,g} + nf_{k,g}\right)} + \varepsilon \tag{4.13}$$

Each iteration the success and failure memories are updated and the probabilities are recalculated. $\varepsilon = 0.01$ is included to avoid that any strategy gets a zero probability.

SaDE parameter adaptation

The behavior of each strategy is dependent on two parameters: K and F for DE/current-to-rand/bin and CR and F for the other strategies. "CR is usually more sensitive to problems with different characteristics, e.g., the unimodality and multimodality, while F is closely related to the convergence speed" Qin et al. [59]. Therefore to maintain both exploration and exploitation, F is randomly sampled from a normal distribution with a mean of 0.5 and a standard deviation of 0.3, while CR is adaptively tuned to the current problem. Every time

when crossover is applied in strategy k, a new CR is generated from a normal distribution with mean CRm_k and standard deviation 0.1. If this CR value produces a successful trial vector, it is stored in the CR-memory. This memory contains a list of successful CR values for each strategy for every generation. At the start of every generation the median over the last LP generations is used to calculate CRm_k for each strategy. In the first LP generations CRm_k is fixed at 0.5. Parameter K was randomly sampled $[0, 1]$.

4.2.4 Quaternion re-normalization

The quaternion part of a transformation that is generated by Differential Evolution is not necessarily a unit quaternion, but only unit quaternions describe rotations. Therefore the quaternion is normalized before the trial vector is used to evaluate the cost function. Moreover, between mutation and crossover the quaternion part is negated ($q_{new} = -q$) if the dot product between the quaternion of the trial vector and the quaternion from the previous generation is negative. This reduces the difference in their representations, while the negated quaternion still represents the same rotation. These modifications were inspired by the paper by Kavan and Žára [60], where they blended multiple quaternions by applying a weighted average followed by a normalization.

4.3 US segmentation using a 3D/2D U-net

For segmenting the blood vessels and the liver boundary a neural network was used with the same network architecture but a different training set. The architecture was based on the U-net [61], which was first used with 3D convolutions by Çiçek et al. [62] and further refined by Milletari et al. [63], which they call the V-net. The U-net was originally developed for segmenting medical images when only limited data is available, which is similar to the problem we are trying to solve here. Moreover, we had available the work of Muhammad Arif, who had good results in segmenting needles in 3D US scans by using a 3D U-net.

The US transducer position was not segmented using deep learning. US scans are always acquired relative to the transducer position, so the transducer position has a constant position on all US scans. Therefore it does not need to be segmented.

4.3.1 3D/2D U-net architecture

Just like the 2D U-net and the V-net our architecture has four downsampling and upsampling steps resulting in five different resolution levels. The main difference and the source of the name of the 3D/2D U-net is that it performs 2D convolutions and 2D downsampling instead of 3D convolution and downsampling on every other layer. The reason for doing this is to compensate for the fact that the resolution along the z-axis is lower than the other 2 axes.

The network also uses batch normalization to speed up training. It is only applied every few layers and not at the highest resolution to reduce the amount of memory needed. To fit within the available memory, the amount of feature maps is also low.

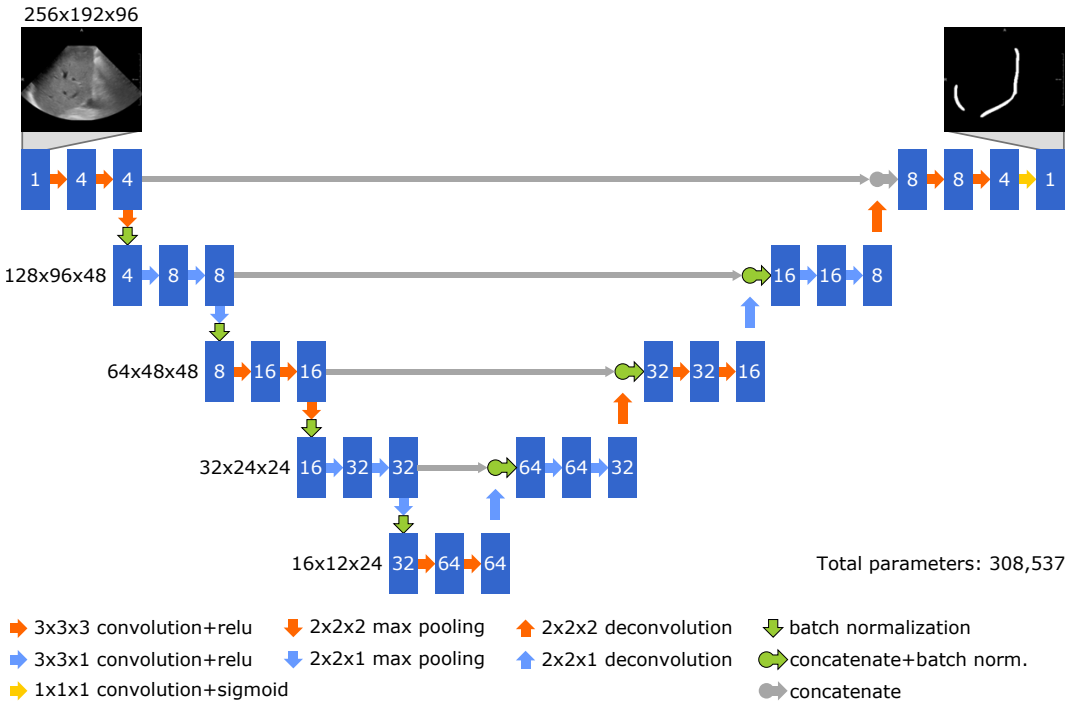

Figure 4.3: Illustration of the 3D/2D U-net architecture

4.3.2 Dice cost function

For quantifying the segmentation quality a cost function is needed. A simple choice would be to take the accuracy (fraction of wrongly classified voxels) as the cost function. However, if you do this you will run into the problem of class imbalance. Because about only one percent of the volume of the US cone consists of blood vessels and about two percent is covered by the liver boundary mask, the network is much more likely to classify a voxel as background than as foreground. To overcome this problem we use a cost function based on the Dice measure [64]. When A is the set of foreground voxels outputted by the network, and B is the set of foreground voxels in the ground truth, the Dice measure DSC is defined as:

$$DSC = \frac{2|A \cap B|}{|A| + |B|}$$

The Dice measure is always between zero and one, with one meaning perfectly overlapping classes ($A = B$). One thing to note is that this function only looks at foreground voxels. This makes the function not dependent on the size of the background and therefore less sensitive to class imbalance. Nevertheless both types of errors are penalized, but in different ways: false positives increase the denominator while false negatives decrease the numerator. Another thing to note is that the intersection operator $\cap$ is not differentiable, so

it cannot be optimized using gradient descent. Therefore we use a slightly modified version of the Dice measure to construct the cost function [63]:

$$C_{Dice} = 1 - \frac{2\boldsymbol{a} \cdot \boldsymbol{b}}{\boldsymbol{a}^2 + \boldsymbol{b}^2 + 1}$$

Here $\boldsymbol{a}$ and $\boldsymbol{b}$ are the output of the network and the ground truth in vector form. If $\boldsymbol{a}$ and $\boldsymbol{b}$ would have been binary vectors, containing only zeros and ones, $\boldsymbol{a} \cdot \boldsymbol{b}$ would be the same as $|A \cap B|$ and $\boldsymbol{a}^2$ would be the same as $|A|$. The $+1$ in the nominator and denominator, are added to avert division by zero. By using a sigmoid activation function in the last layer, all values of these vectors will be in the range $[0,1]$. However, their values can still be in between zero and one, causing a difference between this version and one minus the original Dice score. Moreover, if you want a binary mask as result you have to threshold the output of the network.

To optimize this cost function we used the ADAM optimizer [65], because it was also used by Muhammad Arif and because it was recommended by Ruder [66] as a good overall choice for deep learning optimization. In total 350 epochs are used with a learning rate of 0.001, which was divided by two every 75 epochs. The decaying learning rate was used to precisely converge to a (local) optimum and to have little variation in result near the end of the training period.

4.3.3 Data annotation

Both the blood vessels and the liver boundary were annotated by the author to be used as training data. To reduce the amount of manual work, annotations were done every few slices and interpolated on the slices in between.

The liver surface was annotated every fourth slice resulting in a 2.52mm annotation spacing. Within each slice, connected points were manually selected and a smooth curve was interpolated between these points using spline interpolation. These splines were rasterized to get a mask image for each slice and these curves were dilated using a 15×15 voxel (6.3×5.8mm) kernel. Shape based interpolation [67] was used to interpolate the masks to slices in the US volume without annotations. All slices were finally eroded again using a 9×9 voxel (3.78×3.48mm) kernel. The erosion and dilation steps were performed around the shape based interpolation to avoid getting holes between neighboring slices. To aid the annotation, a view of the (roughly) corresponding CT slice was provided side by side of the US slice view. The US CT registration needed to provide this view was based on 3-9 manually annotated corresponding points that were available with the dataset.

The blood vessels were annotated every third slice resulting in a 1.89mm annotation spacing. Within each slice each vessel was annotated by manually providing the corners of a polygon. These polygons were rasterized to get a mask image for each slice. Two interpolation methods were applied to fill in the slices that were not annotated. 1.) Shape based interpolation was used to interpolate to the slices without annotations. 2.) The slices with annotations were eroded with a 3×3 voxel (1.26×1.16 mm) kernel and then copied to the neighboring slices to get masks for the entire volume. The results from both interpolation methods were combined by taking the voxelwise maximum. The second interpolation was

performed to provide more thickness to vessels that move along the slice, which looked too thin by using just shape based interpolation. Again a CT view was provided alongside the US view to aid the annotation.

4.3.4 Data augmentation

To prevent overfitting, data augmentation is applied with small affine transformations to the training examples. New augmented examples are generated on the fly for each iteration, so new images are used every iteration. Moreover, only the images of the current mini-batch are loaded into memory, so the computer will not run out of memory even on big datasets.

To augment the image an affine matrix is composed. First the voxel positions are centered around zero and scaled to a unit voxel size, so the transformation is performed around the center of the volume. At the end, these transformations are inverted to go back to the original position and voxel size. In between, the image is randomly sheared, rotated around the z, x and y axes, and flipped along the x and z axes in that order. Because US images are orientation dependent, only flips along the x and z axes are performed. The range of rotations along the x, y and z axes is respectively [-20, 20], [-30, 30] and [-20, 20]. The range of rotations around the x and z axes are smaller because these directions affect the orientation dependence of US images. The shearing is parameterized according to the matrix below with x, y and z each within the range [-0.1, 0.1]. All random values are uniformly distributed.

$$M_{shear} = \begin{bmatrix} 1 & x & x \\ y & 1 & y \\ z & z & 1 \end{bmatrix}$$

The combined affine matrix is used to resample the image. Third order spline interpolation is used in both the US scan and the segmentation mask, but the mask image is thresholded at 0.5 to get a binary image again. Parts outside the original image are assumed to be 0, which is the same value as the area outside US cone.

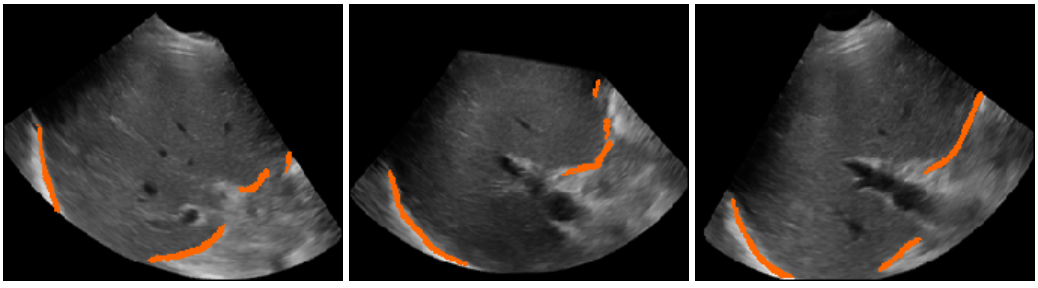

Figure 4.4: The center slice of the same volume with different data augmentation applied three times. The liver boundary mask is displayed in orange.

4.4 CT segmentation

CT segmentation is not a major focus of this thesis. Partially because manual annotations were already available, and partially because CT segmentations can be performed before the intervention, allowing for more manual interaction if needed. The parameters in this section have been chosen to get visually good results on our dataset, but have not been evaluated for performance or applicability to new data.

4.4.1 Skin segmentation

First the scan is thresholded at >400 to extract the body. Then a 5 mm (rounded to the nearest integer voxel size) morphological opening is applied to smooth the mask and to disconnect parts of clothing or medical equipment. Then the biggest connected component is selected as the body. At this point the lungs and some parts of the intestines are still not segmented. To solve this, all holes in transverse slices are filled in, by filling in all areas that can not be reached from region growing over the background in all transverse slices starting at the edges of the slice.

4.4.2 Liver boundary segmentation

From earlier research by Luu [9] manual annotations of the liver were already available for all CT scans. These annotations were performed on transverse slices, and every fifth slice was annotated. On slices without annotations we copied the mask of the closest slice. When two annotated slices were equally close the union of both masks was used. The annotator was not always consistent in whether to include the IVC, the portal vein and the gallbladder in the liver mask or not, which sometimes resulted in sudden changes in the mask.

To be used in clinical practice, this would have to replaced with a system involving less manual work. Several automatic and semi-automatic approaches have been presented for CT liver volume segmentation as part of the SLIVER07 grand challenge [68].

4.4.3 Blood vessel segmentation

For vessel segmentation, Frangi vesselness filtering [69] is used at two scales ($\sigma = 2$ and $\sigma = 4$), with $\alpha = 0.9$, $\beta = 0.5$ and $\gamma = 30$. This is thresholded at 0.01 to obtain a vessel mask. To avoid detecting vessels outside of the liver and false detections on the edge of the liver, the liver mask is shrunk by 1 cm and then applied to the vessel mask. Finally a 2 mm (rounded to the nearest integer voxel size) opening is applied to the vessel mask.

The portal vein and the IVC are not always included in the shrunk liver mask. Moreover, vesselness filtering at a big scale would be needed to detect the IVC, which would introduce extra false detections on the edge of the liver. Therefore parts of these vessels were often not segmented automatically, so they were added manually. This was done by selecting points within the vessels and region growing within a 15 mm radius over all intensities >1140.

Chapter 5

Results and Evaluations

In this chapter, the performance of the system is described. Furthermore, the system contains several parameters for which we have studied the behavior. Given the complexity of the system we evaluate the different parts incrementally. In the first experiment the neural network for US segmentation is evaluated. In the second experiment the parameters of the registration optimizer are optimized. These values are used in the third experiment to optimize the cost function parameters. In the fourth experiment a manual evaluation is performed and in the fifth experiment the results are placed in a medical context. Lastly, in the sixth experiment we measured the computation time needed during the intervention.

5.1 Data

The dataset consists of 20 CECT scans and for each CECT scan 2 3D US scans from different views, resulting in 40 scan pairs over 20 patients. It was originally acquired for the research by Banerjee et al. [8], Luu [9]. For each patient an US scan is acquired from sub-xiphoidal position (below the sternum) in the saggital plane and another from the inter-costal position(from the side between the ribs) views in the coronal plane. These are the standard scanning locations for liver tumor ablations. All scans are named with a number for the patient, and a letter for the US acquisition position: C for inter-costal and S for sub-xiphoidal.

All ultrasound scans were acquired using a Philips IU22 ultrasound system with a x6-1 3D matrix transducer, at a resolution of $512 \times 378 \times 222$ and a voxel size of $0.42 \times 0.387 \times 0.629$mm. During acquisition the patients were asked to breathe freely. Before segmentation the scans were downsampled by a factor 2 using the Lancsoz3 filter [70].

The CT scans were acquired from three different scanners(Somatom Flash, Somaton Force and Somatom AS+, all by Siemens) with slice spacing of 2 or 3 mm and a pixel size between 0.388mm and 0.625mm. For each patient multiple CT scans were acquired at different contrast phases, but only the scan closest to portal venous phase was used.

The dataset also contains manual annotations that can be used for evaluating registration results. For each scan pair, 3 to 6 corresponding points were selected by a radiologist. 16 scans were annotated twice by different radiologists resulting in up to 9 annotations per

scan. For two US scans the radiologists were unable to find corresponding points due to bad scan quality, so these scans were excluded in all experiments, resulting in a total of 38 scan pairs over 20 patients.

These annotations were originally acquired to provide an initialization for the registration method of Banerjee et al. [8], but we use them here for measuring the error of a registration. We observed that the annotation error is not a very precise measurement of registration quality, which is why we performed a manual scoring of all results in Section 5.5. This is probably caused by the fact that the scans were annotated from different 2D views, making it hard to pinpoint corresponding structures exactly. Moreover, some scans appear to contain big non-rigid deformations and only between 3 and 9 annotations are available for each scan pair, so the annotations do not capture all the deformations that are present.

5.2 Experiment 1: Neural network based US segmentation

In this section we will look at the performance of the 3D/2D U-net for segmenting blood vessels and the liver boundary. The results of these segmentations are used for registration, so the errors of the network have been investigated in this context

5.2.1 Experiment setup

During manual annotation of the blood vessels one scan was excluded due to the presence of a very big($\pm$ 200 mm diameter) tumor covering most of the visible liver area and another was excluded because the blood vessel edges were unclear. For segmenting the liver boundary the scan with the big tumor was also excluded, and another scan was excluded where the liver boundary was not clearly visible. These scans are not included in the evaluations in this section, because no reference standard is available.

The remaining 36 scans were divided randomly into four splits of nine scans for four fold cross-validation. The scans were divided in such a way that when two US scans of the same patient were available they were put into the same split. This also made sure that the different views(sub-xiphoidal and inter-costal) were evenly divided over the splits.

Neural networks were trained for the blood vessels and the liver boundary on the combined data of 3 splits and evaluated on the remaining split. This resulted in 8 networks, each using 27 scans for training and 9 for testing. After 350 epochs the network parameters were saved as the final result. No early stopping was used, because we did not observe increasing validation loss during earlier experiments. There may be a bias because the hyperparameters of the network have been tuned on the test set. This was accepted as a trade off because of the small amount of data available and we expect the bias to be small given the generic nature of the network architecture. Because of the cross-validation no network parameters are trained on the test data.

5.2.2 Training

The training curves are shown in Figure 5.1. For both segmentation tasks the training curves of the cross-validation splits with the highest and the lowest test set Dice cost are plotted. There is some difference between the worst and the best results, especially on the liver boundary segmentation task. This might be explained by the fact that there is limited data available, resulting in a difference in distribution among the splits. With more data each split would contain more samples causing the overall distribution of the split to be more similar to the entire dataset. The network is always nicely converging to an optimum, but the decrease in ripple is also caused by the fact that the learning rate is halved every 75 epochs.

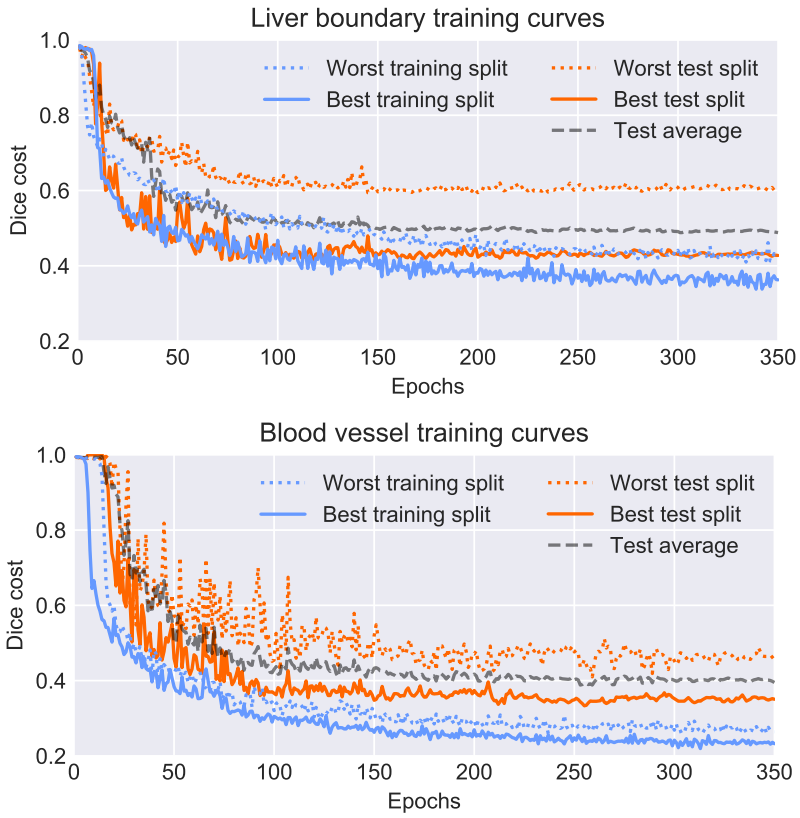

Figure 5.1: Training curves of the liver boundary and blood vessel segmentation networks. For both segmentation tasks the worst and the best results during cross-validation are plotted.

5.2.3 Result statistics

Several statistics commonly used to evaluate segmentations were calculated by comparing the output of the network to the reference standard mask. Each statistic was calculated

for each scan separately and the average and standard deviation over all scans is displayed in Table 5.1. The sensitivity, precision and accuracy are based on the ratios between true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN):

$$\text{precision} = \frac{TP}{TP+FP},\ \text{sensitivity} = \frac{TP}{TP+FN},\ \text{accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (5.1)$$

The accuracy is very high, but this is mostly caused by the class imbalance, causing there to be many more true negatives than all other categories. The same would hold for the specificity, which is therefore not included here.

	Liver boundary	Blood vessels
Dice score	0.52 (± 0.16)	0.57 (± 0.19)
precision	0.53 (± 0.12)	0.68 (± 0.17)
sensitivity	0.54 (± 0.19)	0.56 (± 0.23)
accuracy	0.99 (± 0.00)	1.00 (± 0.00)

Table 5.1: Average and standard deviation of the results of neural network based US segmentation.

5.2.4 Investigating the errors

The statistics in Table 5.1 provide an overview of the performance of the neural networks, but they also hide many details. The precision for liver boundary segmentation is on average 0.53 which would mean that almost half of the voxels detected as the liver boundary are false positives. However, by looking at the results individually we observed that big parts of the boundary of the liver were segmented almost correctly, but they did not overlap exactly. An example of this is given in Figure 5.2. Small misalignments such as these are mostly harmless during registration, and they could also be attributed to variance in the annotations, but because the liver boundary is a thin structure they have a relatively big influence on the measured performance. To see how much of the wrongly segmented voxels are still close to correct we calculated the distance from each false positive voxel to the closest positive voxel in the reference standard, and the distance from each false negative voxel to the closest voxel in the neural network output. Histograms of these properties over all scans are displayed in Figure 5.3. As you can see more than halve of the false positives in the liver boundary segmentation are within 2mm of the reference standard. Given that the registration error of the liver boundary and big blood vessels caused by using a rigid transformation model is on average around 10mm [26], this is very small by comparison.

For tuning the f_{liv} and f_{ves} parameters of the LTS-CD and TACD cost functions it is also interesting to know how the false positives are distributed over all scans. These cost functions try to remove outliers (false positives) by removing a constant fraction of samples. This works best when the relative amount of false positives is the same in all scans. Because we already concluded that many of the false positives are relatively harmless we introduce the soft precision to visualize the false detection distribution over all scans (Figure 5.4). The soft precision is the precision measure, where all false detections with a distance below a

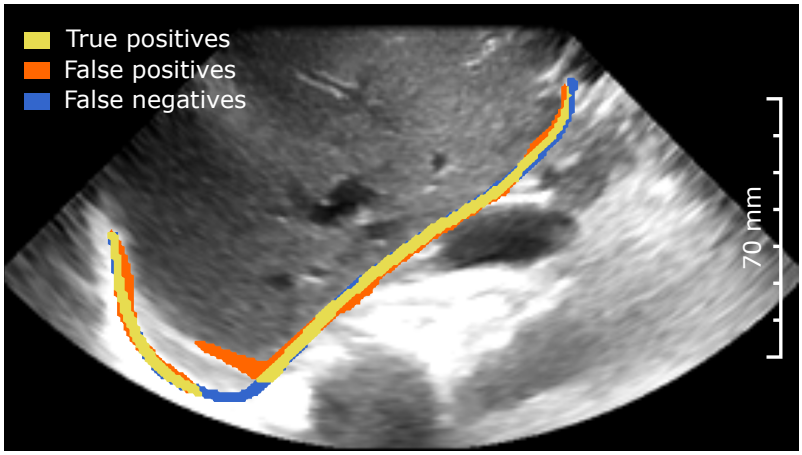

Figure 5.2: 2D slice of a liver boundary segmentation result with slight misalignments.

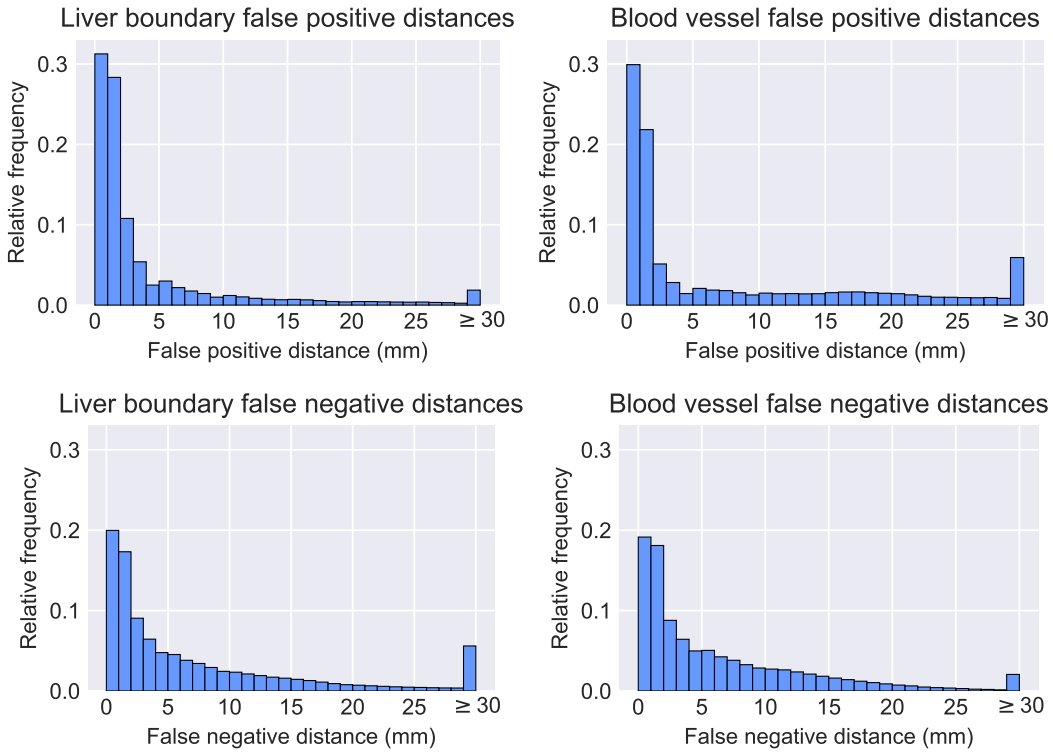

Figure 5.3: The top histograms display the distance from all false positive voxels to the closest positive voxel in the reference standard. The bottom histograms display the distance from all false negative voxels to the closest positive voxel in the neural network output. The rightmost bin contains all voxels at a distance of 30 mm or more.

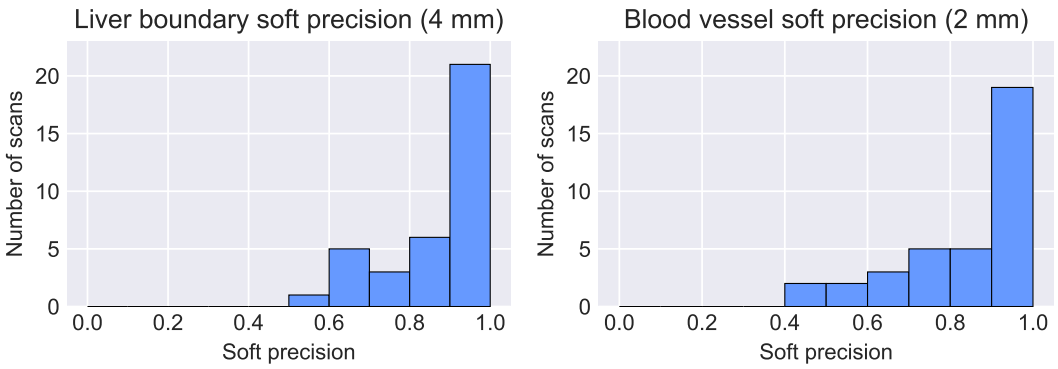

Figure 5.4: Distribution of the precision over all scans, disregarding false positives that are close to the reference standard. A 4 mm threshold was used for the liver boundary and a 2 mm threshold was used for the blood vessels.

threshold are discarded. We used a threshold of 4 mm for the liver boundary and of 2 mm for the blood vessels. The vessels should have a lower threshold because they are closer together and have better defined edges. Most scans have a soft precision of 0.9 or higher for both the blood vessels and the liver boundary. It is also possible to generate a soft sensitivity histogram using the same principle to visualize how the false negatives are distributed over all scans. These are included in Appendix B.

In the blood vessel segmentations another effect was observed: The bigger blood vessels, especially the portal vein, make up most of the volume, which has the result that the network mostly focuses on those vessels. Because of this, smaller vessels are sometimes not segmented. An example of this is displayed in Figure 5.5.

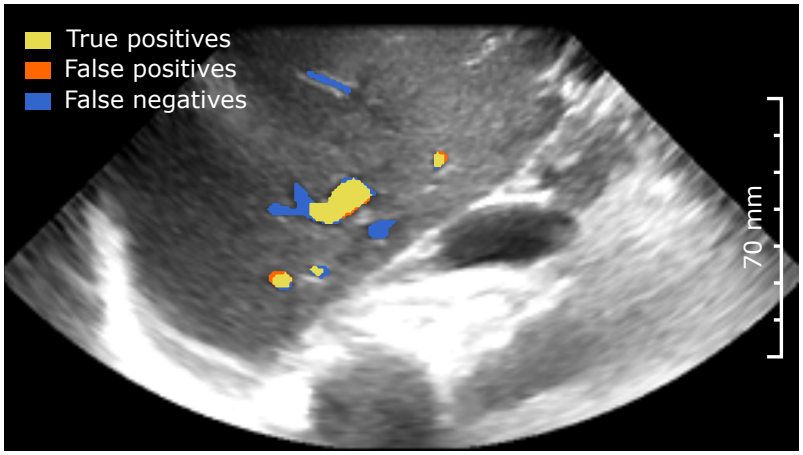

Figure 5.5: 2D slice of a blood vessel segmentation result where not all vessels were segmented.

5.2.5 Effect of the output threshold

The output of the neural network for each voxel is a continuous variable between zero and one. This value is rounded to either zero or one by a threshold. With a lower threshold you expect more positive detections, but also more false positives.

To test this influence we calculated the Dice score for each scan, for all thresholds that have a unique outcome. For the blood vessel segmentation networks, the position of the threshold has almost no influence on the Dice score because the network already pushes its outputs close to either zero or one, with only a few values in between. The average Dice score for each threshold is shown in Figure 5.6. For the liver boundary segmentation two cross-validation splits push all outputs to almost zero or one, and the two other splits push their outputs to either zero or approximately 0.74 (0.7377 in one split and 0.7425 in the other). No voxels have a value above approximately 0.74 in the output of those networks. We have no explanation why this happened. For all values between 0 and 0.74 the average Dice score is almost the same, so the threshold very robust. In the end a threshold of 0.5 was selected, so the output is rounded to the closest integer.

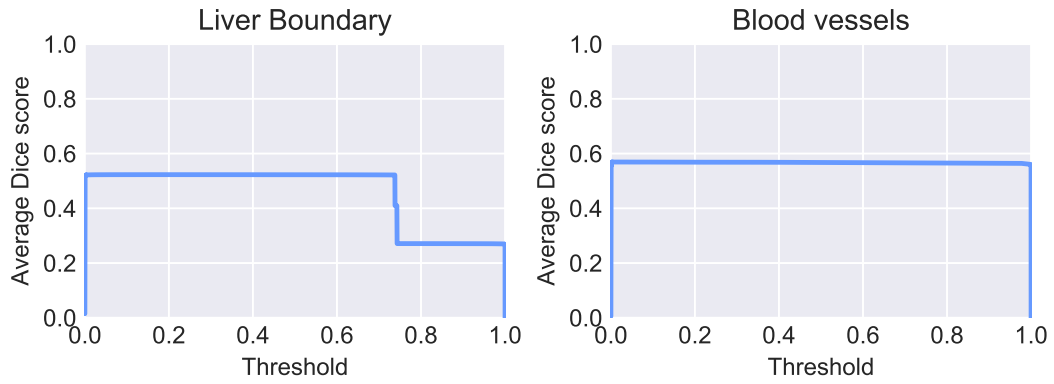

Figure 5.6: Influence of the threshold of the neural network output on the Dice score.

Note

While investigating the results of this experiment we found out that there was a small bug in the segmentation code. Instead of thresholding the output of the network at > 0.5, which was the intended behavior, the output was directly cast to an integer, which has the same behavior as thresholding at ≥ 1. Because the sigmoid activation function in the last layer of the network already pushes most of the outputs of the network to either zero or one, the differences between the bugged and the intended segmentations were small for most scans. Typically a slightly wider area is included around the same structures (Figure 5.7). However, for the two cross-validation splits of the liver boundary segmentation, where no values above ± 0.74 were outputted, nothing was segmented. This was noticed as soon as the network was executed, but when re-evaluating the network with a slightly different version of the code that did include the right threshold the network returned normal results. Because

we were not aware of the differences between the two versions we did not look into it earlier. The results in this section use the right (> 0.5) threshold, but repeating experiments 2 and 3 would take about four weeks of computation time on two computers so they still use the bugged (≥ 1) threshold, with the exception of the two liver boundary splits. In experiment 4 both thresholds were investigated and no big difference in results was found.

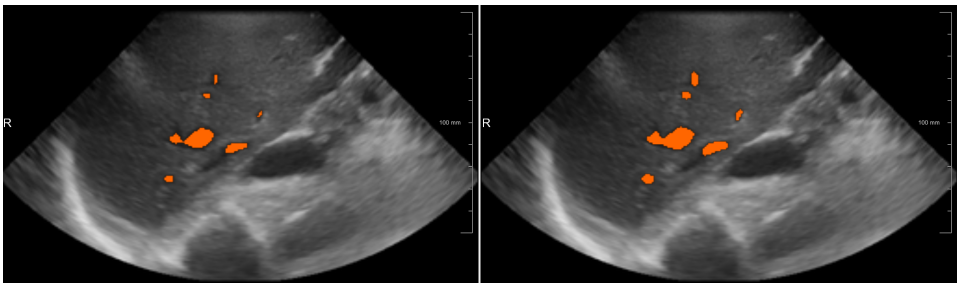

Figure 5.7: Example of the difference between vessel segmentations using the bugged threshold(left) and the right threshold(right).

5.3 Experiment 2: Registration optimizer parameters

To improve results in later experiments the parameters of the optimizer were tuned. Four parameters were taken into account: The population size P, the number of generations G, the number of repetitions R and the learning period LP. Storn and Price [52] suggested selecting setting P between five and ten times the amount of parameters that have to be optimized. Because we have seven learnable parameters we started out with $P = 50$. This converged in most cases with $G = 500$ and we observed that when P is increased, the algorithm needs longer to converge, so G should be higher too. Originally we used ten repetitions ($R = 10$). In this experiment we tested eight other parameter settings with the same amount of cost function evaluations $P \times G \times R$. In this part the learning period was kept constant at 20.

To test each combination of settings, the registration algorithm was run on all US scans for which neural network segmentations of the blood vessels and liver boundary were available. As the cost function TACD was used with $f_{ves} = 0.65$, $f_{liv} = 0.75$, $d_{max} = 10$ mm, $w_{ves} = 1.5$, $w_{liv} = 1.5$ and $w_{scan} = 1$. As the transformation model an affine model was used that added 3 scaling and 3 shearing parameters to the rigid model. In the end we did not end up using that model in other experiments due to low quality results. To measure the quality of the optimizer parameters the average of the final result of the cost function was used. This was used instead of the annotation error because it is more directly indicative of the ability of the optimizer to find an optimum, and less dependent on other factors such as the cost function parameters. The results are displayed in Table 5.2.

The best 5 parameter settings of the first part were used in the second part of this experiment. The test was the same, but now the learning period was varied over the values $20, 30, 40, 50, 60$ for each combination of other parameter settings. These values for LP were used because they were also used in the analysis of Qin et al. [59]. The results are

displayed in Table 5.3, and $P = 80$, $G = 800$, $R = 4$ & $LP = 20$ was selected as the final settings to be used in further experiments.

P	G	R	Average cost
30	420	20	1.588
50	500	10	1.437
60	700	6	1.266
70	700	5	1.304
80	800	4	1.330
90	950	3	1.398
100	1250	2	1.352
150	1700	1	1.364
200	1250	1	1.366

Table 5.2: Result of first part of the optimizer parameters experiment. The table lists $C_{registration}$ after registration for several optimizer settings. The results are color coded from worst(red) to best(green).

			Average cost					
P	G	R	LP = 20	LP = 30	LP = 40	LP = 50	LP = 60	Average
60	700	6	1.362	1.379	1.411	1.402	1.374	1.386
70	700	5	1.366	1.328	1.363	1.400	1.349	1.361
80	800	4	1.283	1.350	1.317	1.301	1.339	1.318
100	1250	2	1.347	1.291	1.411	1.499	1.377	1.385
150	1700	1	1.461	1.419	1.410	1.437	1.427	1.431
Average			1.364	1.353	1.383	1.408	1.373	

Table 5.3: Result of second part of the optimizer parameters experiment. The table lists $C_{registration}$ after registration for several optimizer settings. The results are color coded from worst(red) to best(green).

5.4 Experiment 3: Registration cost function parameters

In this experiment the influence of the cost function $C_{registration}$ and its parameters will be investigated. Regardless of what cost function variant for comparing segmentations (LTS, TACD or CACD) is used, $C_{registration}$ will have five parameters in total: w_{ves} & w_{liv}, $d_{min-trans}$ and either f_{ves} & f_{liv} or $d_{max-ves}$ & $d_{max-liv}$. We are interested in both the optimal parameter settings and the sensitivity of the algorithm to these parameters.

5.4.1 Random sampling for parameter optimizations

Performing grid search to find the optimal parameter settings within a given range would quickly become prohibitively computationally expensive because the amount of duration of

the experiment increases to the fifth power with the amount of choices you use for each parameter. For example, if four settings were evaluated for each parameter, already $4^5 = 1024$ evaluations would be needed. Given that in the current implementation running the registration algorithm on the entire dataset takes about 4 hours, this experiment would take about 170 days on one computer.

In many problems some parameters are more sensitive than others, so you might want to sample those parameters more densely. However, you often do not know which parameters are sensitive beforehand. Bergstra and Bengio [71] argue that in those cases random search on average performs better than grid search, because in random search each iteration a new value for each parameter is selected, while in grid search the same value is reused many times in combination with different values for other parameters. This is illustrated in Figure 5.8. The original application of this paper is hyperparameter tuning for deep learning, but they also performed more general experiments where for some parameters a bigger range was acceptable than others.

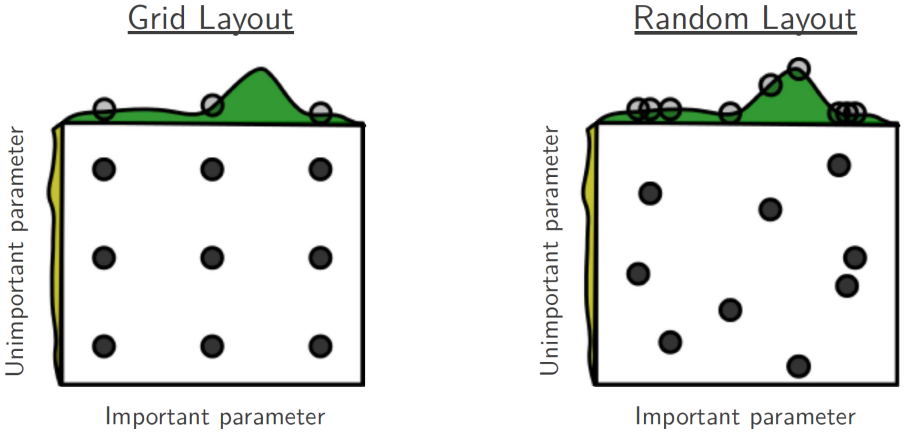

Figure 5.8: When using random sampling each parameter is more densely sampled than in grid sampling. This results in a better result when some parameters are less important than others. Source: Bergstra and Bengio [71]

5.4.2 Experiment setup

Random search was performed with 100 iterations for each cost function variant (LTS-CD, TACD & CACD, Section 4.1.2), always using the same variant for both C_{liv} and C_{ves}. For the LTS-CD and TACD cost functions the following ranges were used for each parameter: $w_{liv} \in [0.5,3]$, $w_{ves} \in [0.5,3]$, $d_{min-trans} \in [0,25]$, $f_{liv} \in [0.5,1]$, $f_{ves} \in [0.4,0.9]$. For the CACD cost function the following ranges were used: $w_{liv} \in [0.5,3]$, $w_{ves} \in [0.5,3]$, $d_{min-trans} \in [0,25]$, $d_{max-liv} \in [3,30]$, $d_{max-ves} \in [3,30]$. Uniform distributions were used to generate the random values of all parameters.

For evaluating the quality of the results the annotations of corresponding points that were provided by radiologists were used. For each scan the average distance between cor-

responding points after registration was calculated. This value was clipped at 30 mm because we observed that when the average annotation error exceeded 30 mm the scans were completely misaligned and a higher value no longer indicated a worse result. Over all scans the average over the clipped average distances was used to determine the best settings.

The best performing settings from the random search will be biased towards the dataset. To get an unbiased estimate of the performance cross-validation is applied: The same cost measure is used as in the above paragraph, but instead of selecting the same settings for all scans, for each scan the settings that perform best over all other scans are selected. This procedure can not be used to get one set of settings, but it does give an indication on how well the best settings of a cost function would perform on unseen scans.

Because we minimized over the annotation error we only wanted to include scans were the annotations were reliable. To measure this, the US and CT scans were rigidly registered by minimizing the sum of squared errors between corresponding points [44] and then the registration quality was manually judged. Four scan pairs were excluded because the annotation based registration was not successful. The three scans that were excluded for either the blood vessel segmentation or the liver boundary segmentation are also excluded here. In total 31 scans are included in this experiment.

5.4.3 Results

The best performing settings for each cost function are listed in Table 5.4. Interesting results are that the LTS-CD cost function includes almost all vessel samples, while the TACD cost function includes almost all liver samples. Moreover, the clipping distance for vessels in the CACD cost function is very low at 3.817 mm. The settings for the second and third best options look similar to the best settings in each cost function, suggesting that the found results are stable. The ten best and five worst settings for each cost function are available in Appendix B. The TACD cost function has the highest error on its optimal settings, but the the cross-validation error is lowest. Therefore it is probably overfitting less than the other cost functions.

Nevertheless the registration error appears much more dependent on the scan than on the parameter settings. Most scans get consistently good or bad results for all settings. This can be seen for the TACD cost function in Figure 5.9, and the same can be observed for the other cost functions, for which the heatmap is available in Appendix B.

Cost function	Best settings	Best cost	CV cost
LTS-CD	$w_{liv} = 1.14$, $w_{ves} = 1.71$, $d_{min-trans} = 17.51$, $f_{liv} = 0.566$, $f_{ves} = 0.891$	12.43	13.81
TACD	$w_{liv} = 2.35$, $w_{ves} = 1.43$, $d_{min-trans} = 18.57$, $f_{liv} = 0.905$, $f_{ves} = 0.685$	12.50	13.68
CACD	$w_{liv} = 1.66$, $w_{ves} = 2.19$, $d_{min-trans} = 23.93$, $d_{min-liv} = 10.85$, $d_{min-ves} = 3.82$	12.01	14.42

Table 5.4: Results of experiment 3. For each cost function the optimal settings are listed. The best error column displays the average error on the dataset. The cross-validation(CV) error is the expected error on unseen data.

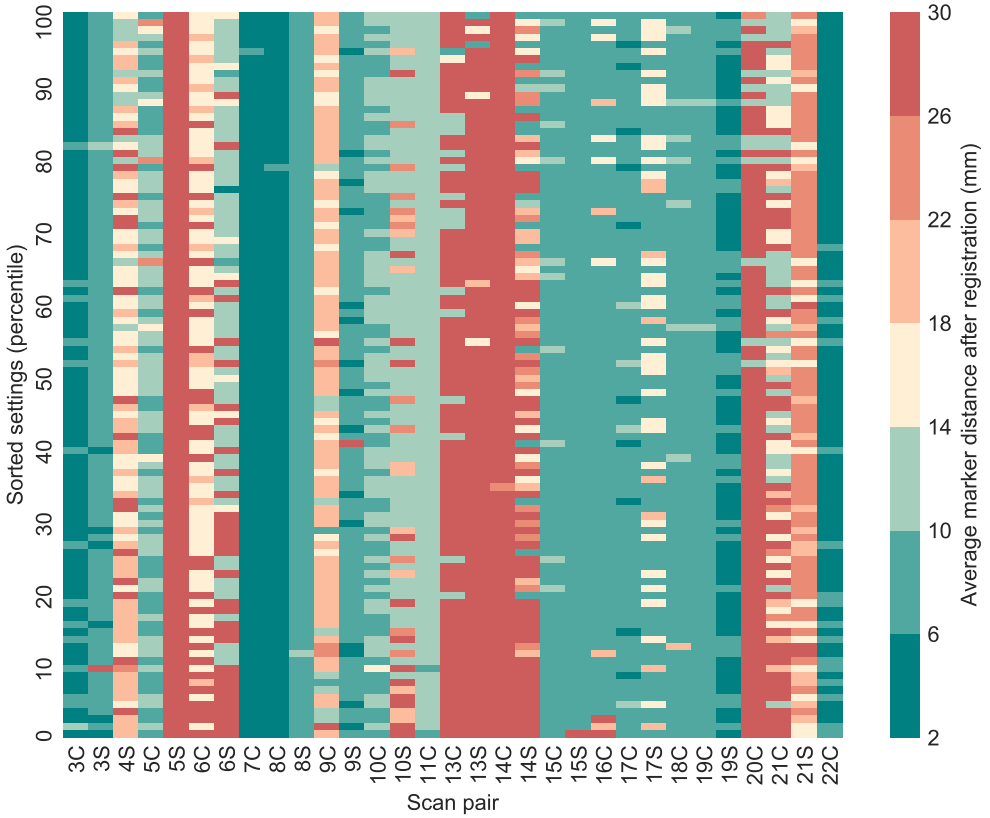

Figure 5.9: Heatmap of the registration error of each scan for each tested setting of the TACD cost function. Each rectangle represents the average marker distance, which was clipped at 30 mm. The settings are sorted from a low average over the average marker distances (top) to a high average (bottom). Most scans have a consistently high or low error for most settings.

5.5 Experiment 4: Manual scoring

Because the annotations that were included with the dataset do not give a precise measure of the registration quality, a second evaluation was performed where the author manually scored the quality of each registration result.

5.5.1 Experiment setup

Registration was performed using 13 different settings: Each of the three cost functions (LTS-CD, TACD and CACD) was used in combination with the neural network segmentations, the neural network segmentation with the threshold bug (Subsection 5.2.5) and the manual segmentations that were used for training the neural network. On the scans that were included in experiment 3, the same settings were used for registration as in the cross-validation experiment to avoid overfitting. On the scans that were not included in experiment 3, the optimal settings according to experiment 3 were used. A cost function that did not filter outliers was also included to test whether outlier removal is necessary. This was done by using the TACD implementation, but setting f_{liv} and f_{ves} both to one, which we will call ACD(average closest distance). The other parameters of the ACD cost function were set based on what gave good results in the other cost functions: $w_{liv} = 2$, $w_{ves} = 2$ and $d_{min-trans} = 25$. To have a baseline result, a registration minimizing the sum of squared distances between manual annotations was included [44]. Finally, to test our implementation of the SaDE optimizer, the same cost function was also optimized using our SaDE implementation.

During the scoring, the names of the experiments were randomized, so we did not know which folder belonged to which experiment. However it was possible to identify which experiments used the manual segmentations, because they excluded three scans.

Each registration result was scored in one of four categories: Good, fair, poor and bad (goed, redelijk, matig, slecht in Dutch). Because the rigid transformation model can not describe all the deformations that might be present, some error was still accepted even in the *good* category, especially when these errors were outside of the liver. To view the registration results, a side by side view of the US and registered CECT scans was used, with a movable cursor that was visible on the same position in both scans (SynchroView2D module in MeVisLab). In which category a scan would be rated is not exactly defined, but to give an idea of the errors present in each category four examples of each category are included in Appendix B.

5.5.2 Results

How many scans were scored in each category is shown in Table 5.5. How the scores are divided over the scans is displayed in Figure 5.10. There is only a small difference between the results of different cost functions. The ACD performs a bit worse than the other cost functions when using the neural network based segmentations, which may indicate that removing outliers improves results, but it could also be explained by the fact that the parameters of the other cost function have been tuned more extensively. All tested cost functions perform better on the neural network segmentation without the bug (Subsection

5.2.5), despite the fact that the cost function parameters were optimized on the set with the bug. All tested cost functions also perform better on manual segmentations than on automatic segmentations, indicating that improving the segmentation results would also improve the registration results.

The registration results using the closed form solution and the SaDE optimizer for minimizing the annotation distances are almost the same, which indicates that the SaDE optimizer works well. When using the manual segmentations the CACD cost function based registration performs about as good as the manual annotation based registrations. When using automatic segmentations, the TACD cost function performs best, which was also concluded in experiment 3.

Experiment	Good	Fair	Poor	Bad	Excluded
Bug NN segmentations (CACD)	21	4	6	7	0
Bug NN segmentations (LTS-CD)	20	4	6	8	0
Bug NN segmentations (TACD)	23	5	7	3	0
NN segmentations (ACD)	20	5	6	7	0
NN segmentations (CACD)	23	5	6	4	0
NN segmentations (LTS-CD)	24	1	6	7	0
NN segmentations (TACD)	23	7	5	3	0
Manual segmentations (ACD)	26	4	4	1	3
Manual segmentations (CACD)	29	4	1	1	3
Manual segmentations (LTS-CD)	25	4	6	0	3
Manual segmentations (TACD)	26	4	4	1	3
Annotations (SaDE)	26	9	1	2	0
Annotations (closed form)	27	8	1	2	0

Table 5.5: Results of manual scoring of each cost function

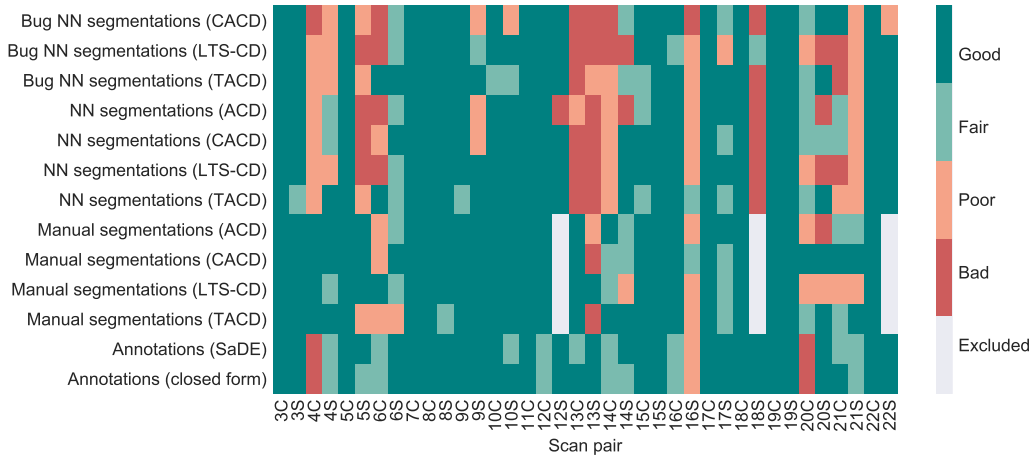

Figure 5.10: Heatmap of the manual scores of each cost function

5.5.3 Relation to the average annotation distance

The relation between the manual score and the average annotation distance is visualized in Figure 5.11. The median of the average annotation distance decreases as the score improves. However, there is a lot of overlap between the distributions of each score. This can be partially explained by errors in the annotations, but the registrations based on these annotations have mostly good results, so the errors can not be very big. A further explanation might be the fact that we use a rigid transformation model while there are non-rigid deformations present. Because of this there might be multiple solutions possible that are equally good, but have errors in different places. Because the manual annotations consist of only a few points, not all areas are penalized equally.

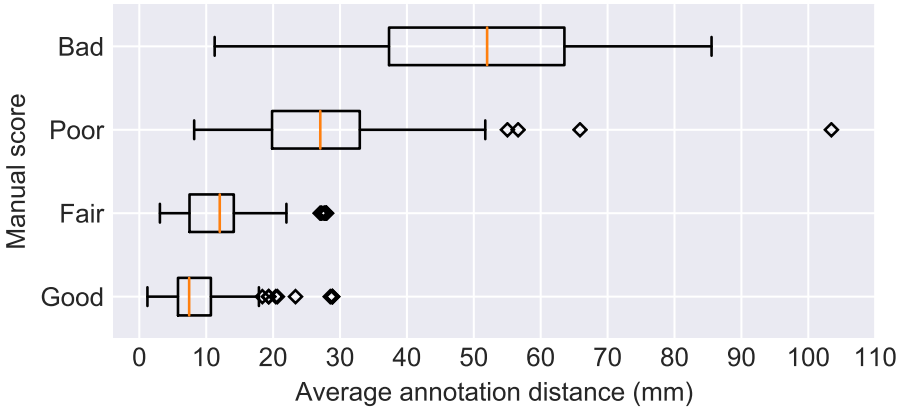

Figure 5.11: Boxplots of the distribution of the average annotation distance of scans with the same score.

5.6 Experiment 5: Scan quality and medical context

In the heatmaps displayed in experiments 3 and 4 (Figures 5.9 and 5.10) you can see vertical streaks. This indicates that the errors are not influenced a lot by changing the settings, and they mostly occur on the same scans. The goal of this experiment is to investigate if there are properties related to the medical condition of the patients that influence the registration quality. We looked at three properties: the contrast phase of the CECT scan, whether a patient has cirrhosis and the amount of tumors that a patient has. The contrast phase was included because it influences the amount of vessels that are visible. Cirrhosis was included because Hwang et al. [42] noted that less blood vessels are visible in patients with cirrhosis, making registration more difficult. The amount of tumors was included because tumors change the overall appearance of the liver which may make the segmentations more difficult. Moreover, because this system is aimed to be used during liver tumor ablations, knowing the number and size of the tumors in each patient makes it possible to predict the performance on patients that could be treated by ablation.

5.6.1 Experiment setup

All properties were determined by Adriaan Moelker, MD, PhD, interventional and cardiovascular radiologist at Erasmus MC. For detection of cirrhosis and contrast phase the same CECT scan was used as for registration. These scans were also mostly used for detecting tumors, but in some cases other CECT scans from the same series, but at different contrast phases, were used as well.

Patients with 3 or fewer tumors were assumed to be treatable by ablation [72], except for one patient, who had a tumor that was noted by Adriaan Moelker to be too big to be ablation treatable. Using this classification the patients were divided into three categories: healthy, ablation treatable cancer, non-ablation treatable cancer.

5.6.2 Results

Most CECT scans were in the portal venous phase, which was expected because the scan closest to portal venous phase was selected to be used for registration. Nevertheless, for three patients (patients 15, 20 and 21) a scan acquired exactly at portal venous phase was not available, so the scan that has been used for registration was on the boundary between late arterial phase and early portal venous phase.

Only one patient in our dataset has cirrhosis (patient 4). Because of this, only a few blood vessels are visible on the US and CECT scans, which explains why this scan is often registered incorrectly with automatic segmentations. However, with manual segmentations enough data is present to get a good registration result.

Looking at the number of tumors, only three patients are treatable by ablation, nine patients have no tumors, and in eight patients the cancer is too severe to be treatable by ablation. The manual registration score of the scans in each category is displayed in Table 5.6. The TACD cost function was used for registration in combination with the NN segmentations, because this was the registration method based on automatic segmentations with the best results according to experiment 4. For most patients two US scans were acquired, so there are six scan pairs of treatable patients, and for all of them the registration quality was rated either good or fair. However, the registration results between the healthy and untreatable categories are very similar, so the amount of tumors present does not appear to have influence on the registration quality.

Condition	Good	Fair	Poor	Bad	Nr. of scans	Nr. of patients
healthy	9	2	3	2	16	9
treatable	3	3	0	0	6	3
untreatable	11	2	2	1	16	8

Table 5.6: Results of the registration using neural network based segmentations and the TACD cost function, divided by the severity of the cancer of each patient. With treatable we mean treatable by ablation.

5.7 Experiment 6: Computation time during the intervention

In this experiment the run time of the method during the intervention was measured. The two components of the method that have to run during the intervention are the US segmentation and the SaDE based registration. The CT segmentation and the distance transform computations can be performed before the intervention so the doctor does not have to wait during the intervention for those steps to finish. We assume that the scans are already available in memory.

5.7.1 Experiment setup

The segmentation algorithm was run once for each scan in the dataset for both the liver boundary and the blood vessel segmentations. The code was run an a PC with an Intel Xeon ES-2623 v3 quad core processor running at 3GHz with 32GB RAM and a Titan XP GPU. Keras version 2.1.6 was used with a Tensorflow 1.8.0 backend, accessed from Python 3.5. The first segmentation took a little bit longer to run, due to Keras strating up. This result was discarded because Keras could be started before the intervention.

The registration algorithm was run once for each scan in the dataset for each cost function (LTS-CD, TACD and CACD), using the optimal parameters found in experiments 2 and 3. The code was run on a PC with an Intel Xeon E5520 quad core processor running at 2.26 GHz with 16GB RAM. The algorithm was implemented as a Python plugin in MeVisLab 2.7.1, using Python version 2.7.9 with Numpy version 1.6.2.

5.7.2 Results

The average run time of the segmentation was $0.187s(\pm 0.002)$ for both the vessel and the liver segmentations. It makes sense that these segmentations take equally long because they use the same network architecture and the same input data, only with different learned parameters. Because two segmentations have to be performed the total time spend on segmentation would be approximately $0.37s$.

The computation time of the SaDE algorithm is displayed in Table 5.7. The algorithm takes a bit less time with the CACD cost function. This is caused by the fact that the trimming operation applied by LTS-CD and TACD (Section 4.1.2) requires sorting the distances every time the cost function is evaluated. This is not needed for the clipping operation that is performed in the CACD cost function instead.

Cost function	Run time (s)
LTS-CD	379.5 (±24.9)
TACD	374.0 (±22.5)
CACD	343.1 (±20.5)

Table 5.7: Run time of the SaDE algortihm during registration

Chapter 6

Conclusions and Future Work

At BIGR, research has been ongoing to develop a system to provide image fusion between US and CECT scans during liver tumor ablations. In this thesis a method for finding an initial rigid registration between the US and CECT scans has been presented. This method should be used together with a follow up registration method for refining the registration result and tracking the movements of the US transducer and the patient.

In the introduction we derived three requirements for the initialization method: 1.) It should be able to find an optimum from any orientation of the US transducer where the liver is in view. 2.) The error of the initialization has to be low enough for the follow up tracking algorithm to converge. The allowed error for the technique of Banerjee et al. [10] is approximately 10 mm translations or 15° rotations. 3.) It should find a solution in ten seconds or less. In this section we will look into how much of these goals has been achieved and what can be done to improve the method further.

The SaDE evolutionary algorithm is used to find the optimal registration parameters over a large search space. In Section 5.5 the SaDE optimizer was compared to a closed form optimizer on minimizing the sum of squared distances between annotations, and they found similar solutions. Moreover, when using manual segmentations, for every scan there is at least one cost function(LTS-CD, TACD, CACD, ACD) where the result was scored at least as fair (Figure 5.10). This indicates that the SaDE optimizer is capable of finding optima in the entire search space.

The proposed method finds an initialization for most scans in the dataset, but not for all. The best performing solution based on automatic US segmentations used the TACD cost function, and its results were scored as follows: 23 as good, 7 as fair, 5 as poor and 3 as bad. As a comparison, rigid registrations based on manual annotations by radiologists were scored as follows: 27 as good, 8 as fair, 1 as poor and 2 as bad. One thing that would improve results is improving the segmentation. When providing our method with manual US segmentations, registrations were found that were about as good as when using the manual annotations for registration. To determine whether a registration is accurate enough for a follow up refinement or tracking method to converge is difficult because of the high variance in the manual annotations (Section 5.5.3). The best test would be to run that follow up method on the initialization results.

The run time of the method is currently around 6 minutes, which is on the lower side

of the 5-20 minutes needed for manual initialization [6]. Nevertheless it is much higher than the aim of at most ten seconds. Almost all of that time is spent on running the SaDE algorithm for the registration. We would like to note that there is a lot of room for decreasing the run time. This part is currently implemented in Python, and we expect that a large speedup is possible by re-implementing the code in a compiled programming language such as C or C++. Furthermore, many of the computations can be done in parallel, making a further speedup possible by running the algorithm on a GPU. SaDE has been implemented on a GPU before [73]. How much speedup was achieved in their experiments was strongly dependent on the population size, which is 80 in our method. However, we also use four restarts that could be run in parallel. Therefore we would expect the speedup achieved by Wong et al. [73] at a population size of 300 (closest to 80*4) is most indicative of the expected speedup of our method. They reported a speedup between 17.05 and 21.12 for this population size for 10 dimensional problems. By combining the speedups of using a compiled language, excecuting the code on the GPU and possibly using more modern hardware, we expect that a runtime of less than ten seconds can be achieved.

6.1 Future work

To improve the neural network architecture used for US segmentation there are many options you could consider, but based on our experiments we have a couple suggestions. The network architecture used all available memory during training on a 1070 TI with 8 GB, which limited the mini-batch size and the amount of features that could be used. With more memory you could try using more features or bigger mini-batches, which may give better results. Moreover, the training curves of the networks (Figure 5.1) show that there is some difference in performance between the training and test sets, which means the network is overfitting. Having more data would help against overfitting, but it may also be possible to reduce the overfitting and improve the test results by adding more regularization, such as data augmentation. Currently the data augmentation performs affine transformations, but this could be extended to include non-rigid deformations and intensity remapping [63]. Using dropout for regularization will probably not work well, because the network already uses batch normalization and these two techniques do not work well together [74]. Currently two separate networks are trained for blood vessel and liver boundary segmentation, but it would also be possible to train one network that outputs both segmentation masks. The advantage would be that the network could use the information about the liver boundary for better segmenting the blood vessels and vice versa. However, such a network might need more features to be able to encode the properties of both the blood vessels and the liver boundary, which would require more memory.

A different approach for getting more information from each US scan without modifying the network architecture would be to segment more classes of features in the liver. Possible extra features could be the gallbladder and the inferior vena cava(IVC), which are visible on most US scans and have been used in registration before [75, 41, 42]. Moreover, the portal vein could be segmented separately from the other blood vessels. It might be possible to segment those features using the same network architecture that was pre-

sented in this thesis. However, you would need to create manual segmentations for training these networks and a CT segmentation method for matching the features during registration. Having more features is expected to reduce the amount of local optima in the combined cost function, because there is probably not a lot of overlap between the non-global optima of each feature. However, some of these features do not show up on all US scans, which may become a problem because the current cost function has a constant weight for each class, which would be problematic when false detections occur in a non-visible class. Still, this could be solved by modifying the cost function, which can be freely done because the SaDE optimizer makes no assumptions about the form of the cost function.

The different strategies for making the cost function robust to outliers appear to improve the registration performance a little, but not much (Section 5.5.2). One explanation is that there are not a lot of outliers present and most of the outliers are so close to the reference standard that they will not harm registration much (Section 5.2.4). Originally we used a different segmentation technique which had a lot more false positives, which is why there is still a strong focus on making the cost function robust to outliers in this thesis. It may be possible to make better use of this robustness by allowing more false positives in the segmentation. In segmentation it is often possible to make a trade-off between having false negatives or false positives. Since the cost function is designed to, up to a certain degree, be insensitive to false positives, you might get better registration results if you accept more false positives in the segmentation step so you have less false negatives. The trade-off between false positives and false negatives is usually influenced by setting the threshold of the output of the network. However, in Section 5.2.5 we showed that this is not possible in our case. You could still influence the trade-off by changing the cost function of the neural network: Salehi et al. [76] propose a variant of the Dice cost function with user selectable weights for penalizing false positives and false negatives separately.

6.2 Conclusion

All in all, the SaDE optimizer is capable of finding results over the entire search space, and if it would be implemented on a GPU we expect it could run quickly enough for clinical use. The neural network based US liver boundary and blood vessel segmentation method provides enough information for an initialization to be found on most scans. Whether this initialization is close enough for a follow up registration method to converge needs further investigation. We expect that better segmentation results are needed before clinical use of our method, and we provided several suggestions for this future work.

Acknowledgements

I would like to thank Theo van Walsum, for allowing me to do this project and for his insight and encouragement during our weekly meetings, and I would like to thank Anna Vilanova for her useful feedback, especially on writing the thesis. Special thanks go to Adriaan Moelker for helping me with my medically related questions and for providing the information about the dataset in experiment 5. I would like thank the staff at the Erasmus MC department of radiology for allowing me to attend a tumor ablation intervention. Moreover, I would like to thank Muhammad Arif for helping me to get started with deep learning. Finally, I would like to thank everybody at the BIGR group, for making my time at BIGR such a nice experience.

Bibliography

[1] J. Ferlay, I. Soerjomataram, R. Dikshit, S. Eser, C. Mathers, M. Rebelo, D. M. Parkin, D. Forman, and F. Bray. Cancer incidence and mortality worldwide: sources, methods and major patterns in GLOBOCAN 2012. *International journal of cancer*, 136(5), 2015.

[2] T. W. Kang and H. Rhim. Recent advances in tumor ablation for hepatocellular carcinoma. *Liver Cancer*, 4(3):176–187, 2015.

[3] C. Brace. Thermal tumor ablation in clinical use. *IEEE pulse*, 2(5):28–38, 2011.

[4] F. Stacul, A. J. van der Molen, P. Reimer, J. A. Webb, H. S. Thomsen, S. K. Morcos, T. Almén, P. Aspelin, M.-F. Bellin, O. Clement, et al. Contrast induced nephropathy: updated ESUR contrast media safety committee guidelines. *European radiology*, 21 (12):2527–2541, 2011.

[5] M. R. Rudnick, S. Goldfarb, L. Wexler, P. A. Ludbrook, M. J. Murphy, E. F. Halpern, J. A. Hill, M. Winniford, M. B. Cohen, D. B. VanFossen, et al. Nephrotoxicity of ionic and nonionic contrast media in 1196 patients: a randomized trial. *Kidney international*, 47(1):254–261, 1995.

[6] G. Mauri, L. Cova, S. De Beni, T. Ierace, T. Tondolo, A. Cerri, S. N. Goldberg, and L. Solbiati. Real-time US-CT/MRI image fusion for guidance of thermal ablation of liver tumors undetectable with US: results in 295 cases. *Cardiovascular and interventional radiology*, 38(1):143–151, 2015.

[7] F. Poulin and L.-P. Amiot. Interference during the use of an electromagnetic tracking system under OR conditions. *Journal of biomechanics*, 35(6):733–737, 2002.

[8] J. Banerjee, C. Klink, R. Gahrmann, W. J. Niessen, A. Moelker, and T. van Walsum. *Fast 4D Ultrasound Registration for Image Guided Liver Interventions*. PhD thesis, Erasmus University Rotterdam, 2016. Chapter 7: Multiple-correlation similarity for fast CT and ultrasound fusion.

[9] H. M. Luu. *Image Analysis for Guidance in Minimally Invasive Liver Interventions*. PhD thesis, Erasmus University Rotterdam, 2017. Chapter 5: Quantification of non-rigid liver deformation.

[10] J. Banerjee, C. Klink, E. D. Peters, W. J. Niessen, A. Moelker, and T. van Walsum. Fast and robust 3D ultrasound registration–block and game theoretic matching. *Medical image analysis*, 20(1):173–183, 2015.

[11] R. Smithuis. CT contrast injection and protocols. http://www.radiologyassistant.nl/en/p52c04470dbd5c/ct-contrast-injection-and-protocols.html#in52c044713086d, 2014. Accessed: 4 June 2018.

[12] R. Baron. Liver - masses I - characterisation. http://www.radiologyassistant.nl/en/p446f010d8f420/liver-masses-i-characterisation.html, 2006. Accessed: 4 June 2018.

[13] R. M. Hammerstingl and T. J. Vogl. Abdominal MDCT: protocols and contrast considerations. *European Radiology Supplements*, 15(5):e78–e90, 2005.

[14] J. S. Song, Y. N. Kim, and W. S. Moon. A sclerosing hemangioma of the liver. *Clinical and molecular hepatology*, 19(4):426, 2013.

[15] D. Fleischmann. CT angiography: injection and acquisition technique. *Radiologic Clinics*, 48(2):237–247, 2010.

[16] R. Smith-Bindman, J. Lipson, R. Marcus, K.-P. Kim, M. Mahesh, R. Gould, A. B. De González, and D. L. Miglioretti. Radiation dose associated with common computed tomography examinations and the associated lifetime attributable risk of cancer. *Archives of internal medicine*, 169(22):2078–2086, 2009.

[17] E. Konofagou. Ultrasound imaging and therapeutics: Part 1 - fundamentals of ultrasound, 2010. International Symposium on Biomedical Imaging(ISBI) tutorial.

[18] Q. Huang and Z. Zeng. A review on real-time 3D ultrasound imaging technology. *BioMed research international*, 2017, 2017.

[19] P. Wells and M. Halliwell. Speckle in ultrasonic imaging. *Ultrasonics*, 19(5):225–229, 1981.

[20] M. Leitman, P. Lysyansky, S. Sidenko, V. Shir, E. Peleg, M. Binenbaum, E. Kaluski, R. Krakover, and Z. Vered. Two-dimensional strain–a novel software for real-time quantitative echocardiographic assessment of myocardial function. *Journal of the American Society of Echocardiography*, 17(10):1021–1029, 2004.

[21] V. D. Bello, I. Fabiani, N. R. Pugliese, L. Conte, and S. Veronica. Speckle-tracking imaging, principles and clinical applications: A review for clinical cardiologists. In *Echocardiography in Heart Failure and Cardiac Electrophysiology*, chapter 5. InTech, 2016. doi: 10.5772/64261. URL https://doi.org/10.5772/64261.

[22] A. Støylen. Basic ultrasound for clinicians. http://folk.ntnu.no/stoylen/strainrate/Basic_ultrasound, 2016. Accessed: 18 Aug 2018.

[23] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. Pluim. Elastix: a toolbox for intensity-based medical image registration. *IEEE transactions on medical imaging*, 29(1):196–205, 2010.

[24] C. V. Stewart, C.-L. Tsai, and B. Roysam. The dual-bootstrap iterative closest point algorithm with application to retinal image registration. *IEEE transactions on medical imaging*, 22(11):1379–1394, 2003.

[25] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, and D. J. Hawkes. Nonrigid registration using free-form deformations: application to breast MR images. *IEEE transactions on medical imaging*, 18(8):712–721, 1999.

[26] T. Rohlfing, C. R. Maurer, W. G. O'dell, and J. Zhong. Modeling liver motion and deformation during the respiratory cycle using intensity-based nonrigid registration of gated MR images. *Medical physics*, 31(3):427–432, 2004.

[27] 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*, 12(5):577–585, 2008.

[28] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International journal of computer vision*, 24(2):137–154, 1997.

[29] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging*, 16(2):187–198, 1997.

[30] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[31] M. J. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[32] R. P. Brent. *Algorithms for Minimization Without Derivatives*. Courier Corporation, 1973.

[33] A. Roche, X. Pennec, M. Rudolph, D. Auer, G. Malandain, S. Ourselin, L. M. Auer, and N. Ayache. Generalized correlation ratio for rigid registration of 3D ultrasound with MR images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 567–577. Springer, 2000.

[34] A. Roche, X. Pennec, G. Malandain, and N. Ayache. Rigid registration of 3-D ultrasound with MR images: a new approach combining intensity and gradient information. *IEEE transactions on medical imaging*, 20(10):1038–1049, 2001.

[35] G. P. Penney, J. M. Blackall, M. Hamady, T. Sabharwal, A. Adam, and D. J. Hawkes. Registration of freehand 3D ultrasound and magnetic resonance liver images. *Medical image analysis*, 8(1):81–91, 2004.

[36] W. Wein, A. Khamene, D.-A. Clevert, O. Kutter, and N. Navab. Simulation and fully automatic multimodal registration of medical ultrasound. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007*, pages 136–143, 2007.

[37] S. Ourselin, A. Roche, S. Prima, and N. Ayache. Block matching: A general framework to improve robustness of rigid registration of medical images. In *International Conference on Medical Image Computing And Computer-Assisted Intervention*, pages 557–566. Springer, 2000.

[38] H. Späth. Fitting affine and orthogonal transformations between two sets of points. *Mathematical Communications*, 9(1):27–34, 2004.

[39] P. J. Rousseeuw and K. Van Driessen. Computing LTS regression for large data sets. *Data mining and knowledge discovery*, 12(1):29–45, 2006.

[40] M. Modat, D. M. Cash, P. Daga, G. P. Winston, J. S. Duncan, and S. Ourselin. Global image registration using a symmetric block-matching approach. *Journal of Medical Imaging*, 1(2):024003–024003, 2014.

[41] C. Weon, W. H. Nam, Y. Hwang, J. Kim, W.-C. Bang, and J. B. Ra. Robust feature based pre-registration of 3D MR image to 3D B-mode ultrasound image of the liver. *Proceedings of the International Society for Magnetic Resonance in Medicine (ISMRM, Salt Lake City, UT, 2013)*, page 1842, 2013.

[42] Y.-K. Hwang, Y.-T. Oh, J.-B. Kim, and W.-C. Bang. One click 3D ultrasound to MR registration. In *Consumer Electronics (ICCE), 2014 IEEE International Conference on*, pages 270–273. IEEE, 2014.

[43] W. H. Nam, D.-G. Kang, D. Lee, J. Y. Lee, and J. B. Ra. Automatic registration between 3D intra-operative ultrasound and pre-operative CT images of the liver based on robust edge matching. *Physics in medicine and biology*, 57(1):69, 2011.

[44] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, PAMI-9(5):698–700, 1987.

[45] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[46] A. Maniezzo. Distributed optimization by ant colonies. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, page 134. Mit Press, 1992.

[47] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[48] I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[49] S. Damas, O. Cordón, and J. Santamaría. Medical image registration using evolutionary computation: An experimental survey. *IEEE Computational Intelligence Magazine*, 6(4):26–42, 2011.

[50] J. Santamaría, O. Cordón, S. Damas, J. M. García-Torres, and A. Quirin. Performance evaluation of memetic approaches in 3D reconstruction of forensic objects. *Soft Computing*, 13(8-9):883, 2009.

[51] I. De Falco, A. Della Cioppa, D. Maisto, and E. Tarantino. Differential evolution as a viable tool for satellite image registration. *Applied Soft Computing*, 8(4):1453–1462, 2008.

[52] R. Storn and K. Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[53] Y. Otake, A. S. Wang, J. W. Stayman, A. Uneri, G. Kleinszig, S. Vogt, A. J. Khanna, Z. L. Gokaslan, and J. H. Siewerdsen. Robust 3D-2D image registration: application to spine interventions and vertebral labeling in the presence of anatomical deformation. *Physics in medicine and biology*, 58(23):8535, 2013.

[54] M. V. Krishna. Initialization methods for 2D/3D registration of medical images during orthopedic surgeries. Master's thesis, Delft University of Technology, 2017.

[55] M. Freiman, O. Pele, A. Hurvitz, M. Werman, and L. Joskowicz. Spectral-based 2D/3D X-ray to CT image rigid registration. In *Medical Imaging 2011: Visualization, Image-Guided Procedures, and Modeling*, volume 7964, page 79641B. International Society for Optics and Photonics, 2011.

[56] S. Miao, R. Liao, J. Lucas, and C. Chefd'hotel. Toward accurate and robust 2-D/3-D registration of implant models to single-plane fluoroscopy. In *Augmented Reality Environments for Medical Imaging and Computer-Assisted Interventions*, pages 97–106. Springer, 2013.

[57] A. Varnavas, T. Carrell, and G. Penney. Fully automated 2D-3D registration and verification. *Medical image analysis*, 26(1):108–119, 2015.

[58] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory of computing*, 8(1):415–428, 2012.

[59] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[60] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16. ACM, 2005.

[61] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[62] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[63] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 565–571. IEEE, 2016.

[64] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Also available as a web page: `http://ruder.io/optimizing-gradient-descent/` accessed on 12 July 2018.

[67] G. T. Herman, J. Zheng, and C. A. Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, 12(3):69–79, 1992.

[68] T. Heimann, B. Van Ginneken, M. A. Styner, Y. Arzhaeva, V. Aurich, C. Bauer, A. Beck, C. Becker, R. Beichel, G. Bekes, et al. Comparison and evaluation of methods for liver segmentation from CT datasets. *IEEE transactions on medical imaging*, 28(8):1251–1265, 2009.

[69] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 130–137. Springer, 1998.

[70] O. Friman. MeVisLab ImageResample module documentation. `https://mevislabdownloads.mevis.de/docs/2.7.1/FMEwork/ReleaseMeVis/Documentation/Publish/ModuleReference/ImageResample.html`, 2015. Accessed: 6 Aug 2018.

[71] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[72] A. Forner, J. M. Llovet, and J. Bruix. Hepatocellular carcinoma. *The Lancet*, 379: 1245–1255, 2012.

[73] T. H. Wong, A. K. Qin, S. Wang, and Y. Shi. cuSaDE: A CUDA-based parallel self-adaptive differential evolution algorithm. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2*, pages 375–388. Springer, 2015.

[74] X. Li, S. Chen, X. Hu, and J. Yang. Understanding the disharmony between dropout and batch normalization by variance shift. *arXiv preprint arXiv:1801.05134*, 2018.

[75] D. Lee, W. H. Nam, J. Y. Lee, and J. B. Ra. Non-rigid registration between 3D ultrasound and CT images of the liver based on intensity and gradient information. *Physics in Medicine & Biology*, 56(1):117, 2010.

[76] S. S. M. Salehi, D. Erdogmus, and A. Gholipour. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In *International Workshop on Machine Learning in Medical Imaging*, pages 379–387. Springer, 2017.

[77] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[78] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[79] M. A. Nielsen. *Neural networks and deep learning (Chapter 4)*. Determination Press, 2015. Accessed on 11 July 2018.

[80] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[81] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[82] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[83] D. Masters and C. Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[84] G. Goh. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL `http://distill.pub/2017/momentum`. Accessed on 12 July 2018.

[85] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[87] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Appendix A

Neural Networks

Neural networks are a class of models that is used in machine learning. A neural network can learn to approximate a function f from a training set consisting of many examples of possible input vectors $\mathbf{a}_i$ paired with expected output vectors $\mathbf{b}_i = f(\mathbf{a}_i)$. The network designer has to provide a specification of how the input and the output can be connected, which is called the network architecture. A network architecture consists of multiple layers. Each layer performs an operation on a set of features to produce a new set of features. This is illustrated in Figure A.1. The input of the first layer is provided by the user. Then, the network will calculate one or more sets of hidden features[1] as intermediate results, and the output of the last layer is the result of the network.

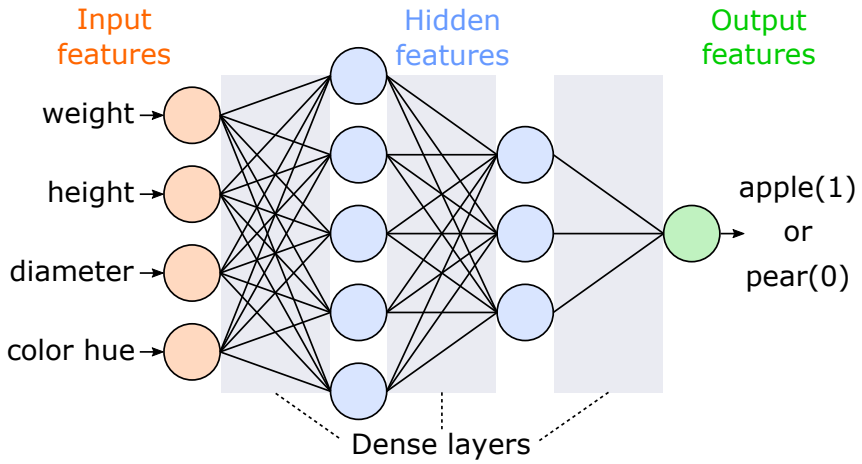

Figure A.1: Example of a small neural network for classifying between apples and pears.

Most types of layers have parameters that describe what a layer will do exactly. These parameters are learned by an optimization algorithm from a set of examples, called the training set. For the training set you already know what the output of the network should be

[1] A set of hidden features is often called a hidden layer, but I will not use that term to avoid confusion with layers that perform operations on features.

and an optimization algorithm will set the parameters in such a way that that the output of the network matches the wanted result as closely as possible. To quantify the current match of the network a cost function(also called loss function) is used. In the example of figure A.1 you might take the number of incorrectly classified pieces of fruit as the cost function. To distinguish between parameters that are learned from the data and parameters that are set by the user, parameters that are set by the user are called hyperparameters.

A.1 Traditional neural networks

A.1.1 Dense Layer

A dense layer(also called fully connected layer) describes a linear function on all input features of the layer. It will output the result: $\mathbf{y} = A\mathbf{x} + \mathbf{b}$, where $\mathbf{x}$ and $\mathbf{y}$ are the vectors of the input and output features respectively, A is a matrix of learnable parameters and $\mathbf{b}$ is a vector of learnable parameters. The width of matrix A is determined by the length n of vector $\mathbf{x}$, but the height of A and $\mathbf{b}$ can be specified by the network designer. This will determine the number of output features, which can be bigger, smaller or equal to the number of input features.

A.1.2 Activation Functions

Dense layers are linear operators, so when you apply multiple of these layers to an input vector you will still only perform a linear operation. To model more complicated functions, non-linearity is added by applying a non-linear activation function to each feature of the output of a dense layer. Because the linear operation of a dense layer is almost always followed by an activation function, the activation function is often said to be part of the dense layer.

One of the most simple activation functions is the rectified linear unit (ReLU): $ReLU(x) = max(0,x)$, which is often used within a neural network. Another option is the sigmoid function: $\sigma(x) = \frac{1}{e^{-x}+1}$. The output of a sigmoid function will always be in the range $[0,1]$ and often close to either one of these values. This can be useful when you want to output a yes/no decision.

While adding an activation function may seem like a small change it greatly increases the amount of functions a neural network is able to approximate. Cybenko [77] proved that a network consisting of two dense layers with a sigmoid activation in between can be used to approximate any continuous function on the domain of a unit hypercube $[0,1]^d$ with error smaller than some predifined constant ε using a finite amount of variables in the hidden layer. This was extended by Hornik [78] to any activation function that is continuous, bounded and non-constant and over any domain that is compact. A more visual explanation is provided by Nielsen [79]. Nevertheless, this does not mean that a network of 2 dense layers and one activation is always a good choice. It may need a lot of hidden features for a good fit, and there is no guarantee that the optimizer will able to find a useful optimum.

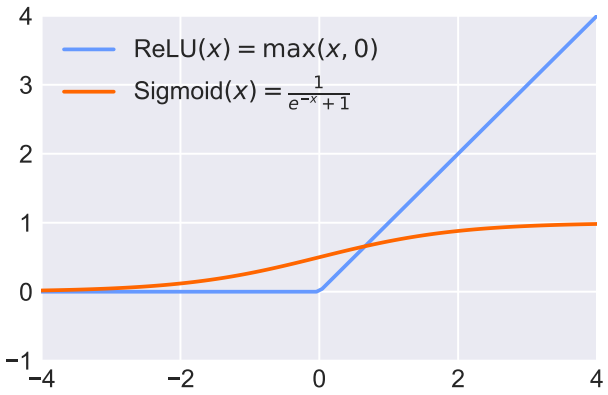

Figure A.2: Activation functions

A.2 Convolutional neural networks

In traditional neural networks no assumptions are made on the correlation of the input features. However, this is different when neural networks are applied to time series (1D), images (2D) or volumes (3D). For these types of input data, the same features are measured over different times or positions. One feature measured over the entire space is called a feature map, which is illustrated in Figure A.3. There usually is a correlation between features that were measured close together. For example in Figure A.3, when a pixel is part of the cat there is a relatively big chance the surrounding pixels are also part of the cat. Convolutional neural networks [80] make use of the local correlations to derive their results. Instead of hidden features, they derive hidden feature maps (also called channels), that have the same local structure as the input. To achieve this they use special kinds of layers. Dense layers can still be applied too by treating every element of a feature map as a separate feature.

Figure A.3: Color images are data with correlations in 2D and they have 3 feature maps: red, green and blue. To store multiple 2D feature maps you need a 3D array with shape $(x_{resolution}, y_{resolution}, n_{features})$.

A.2.1 Convolution Layer

In convolutional neural networks, convolution layers have the same role as dense layers in traditional neural networks. The main reason to replace dense layers is to reduce the amount of learnable parameters. For example: if you would put the values of all pixels of a one megapixel image into a feature vector it would have one million values. If you then apply a dense layer to this vector you would get one million and one learnable parameters for each output feature that you want to generate.

The amount of learnable parameters can be strongly reduced by performing convolutions instead of matrix multiplications, so $\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{b}$, where $\mathbf{x}$ and $\mathbf{y}$ are the input and output feature maps respectively, $*$ is the convolution operator, $\mathbf{k}$ is a convolution kernel of learnable weights and $\mathbf{b}$ adds a learnable offset to each output feature map. A convolution is also a linear operator, but it only combines values that are within the area of a convolution kernel, centered around the point of interest. Just like dense layers, they are typically followed by an activation function. The shape of the kernel has to be defined by the network designer. For images 3×3 kernels are a popular choice because they are the smallest kernel possible that has a symmetric shape around the point of interest. You can learn multiple convolution kernels to get multiple output feature maps, and when you have multiple input feature maps a separate weight is learned for each input feature map. For example, a convolution layer with 4 input feature maps, 5 output feature maps and a 3×3 kernel will have to learn, $4 \times 5 \times 3 \times 3 = 180$ parameters.

To describe the area in the input that is used to determine the value of one pixel in the output we use the term receptive field. For example, when a network consists of only one 3x3 convolution layer, the receptive field is 3x3, or when a network consists of two 5x5x5 convolution layers the receptive field is 9x9x9. If you apply a convolution to all pixels in an image, the receptive field will partially lie outside of the image for border pixels. One approach to handle this is to make the output of every convolution slightly smaller than the input, which is illustrated in Figure A.4. Another approach is to make assumptions about what is outside of the image. Often it is assumed everything outside the image is zero, which is called zero padding.

A.2.2 Pooling Layers

Pooling layers reduce the resolution of the image by combining values that are close to each other. Like in a convolution layer a kernel is moved over the image and the values are combined into one. Within this kernel the maximum value(max pooling) or the average value(average pooling) is outputted. Then the kernel is moved with a step size which determines the factor by which the image will be downsampled. A step size of 2 will downsample the image with a factor 2. When each pixel has multiple features, the average or maximum, is taken separately for each feature.

Pooling layers are often used together with convolution layers to capture large scale features with fewer parameters. Moreover, by reducing the resolution, less memory is needed per parameter during training, because the gradient with respect to fewer pixels has to be computed.

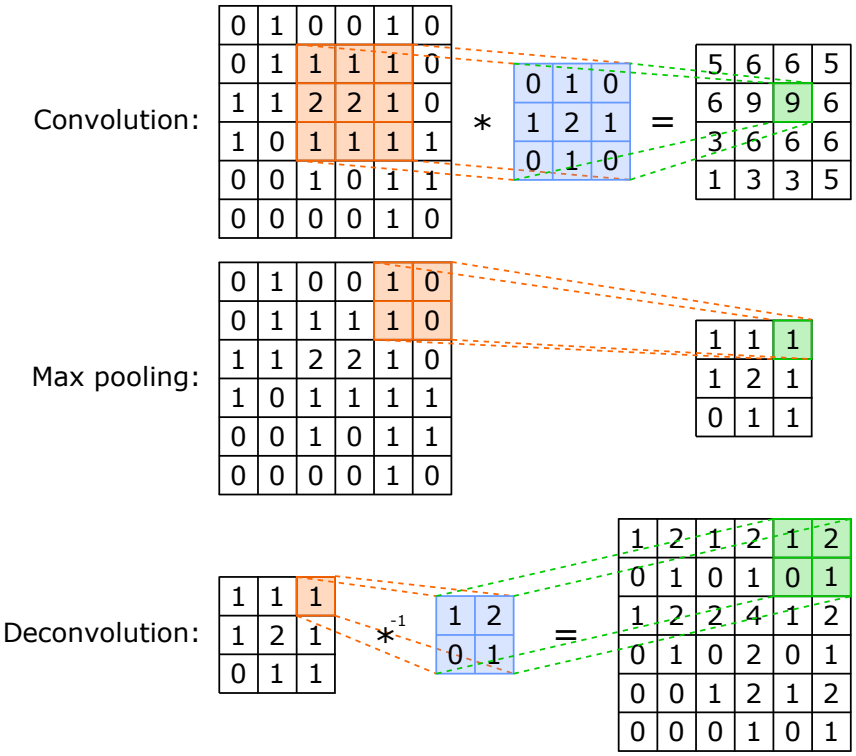

Figure A.4: Illustration of three different layers applied to a 2D image. The matrices on the left are the inputs of the layers and the matrices on the right are the outputs. The orange matrices are the (de)convolution kernels and their parameters can be learned.

An alternative to pooling is strided convolution. There you have kernels with learnable weights, just like in a convolution layer, but you also have a step size that is bigger than one, so you downsample the image, just like a pooling layer. Strided convolutions can retain more information than pooling layers, because they can output more feature maps than the original moving the information from the higher resolution to the extra features. However, pooling layers make your network less sensitive to small deformations and translations because within the convolution kernel the order of the pixels doesn't matter. In many cases this is a useful property too.

A.2.3 Upsampling Layers

An upsampling layer increases the resolution of an image with a constant factor. It is most often used when there are also pooling layers in the network to get an image at the original resolution. The standard upsampling layer replaces each pixel with an $a \times b$ region of the same value as the source pixel, increasing the resolution on the x axis with a factor a and on the y axis with a factor b. If you don't want the same value on each pixel you can also use a deconvolution layer, which will multiply each source pixel with a learnable $a \times b$ kernel.

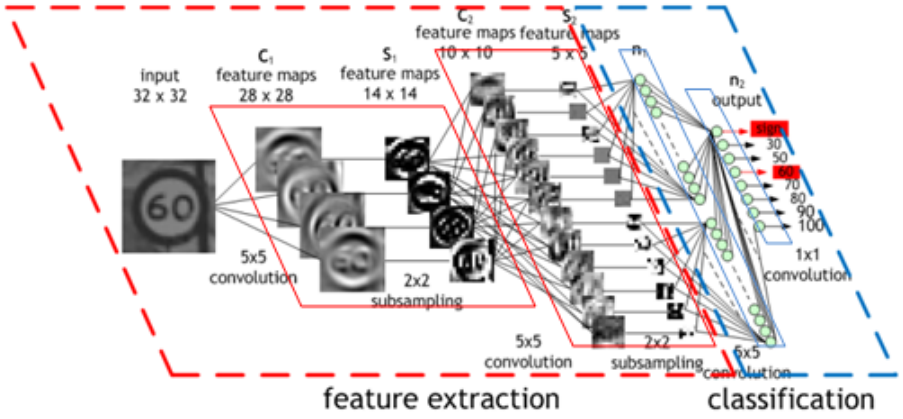

Figure A.5: Example of a convolutional neural network for classifying traffic signs. Image by Maurice Peemen

A.2.4 Concatenation

So far I assumed that a neural network always consists of layers that are performed after each other as if they were a chain. While many networks are structured like this, it does not have to be the case. Each layer can have only one input and one output, but the output of one layer can serve as the input of multiple layers and the feature maps of two equally sized outputs can be concatenated to become one image again.

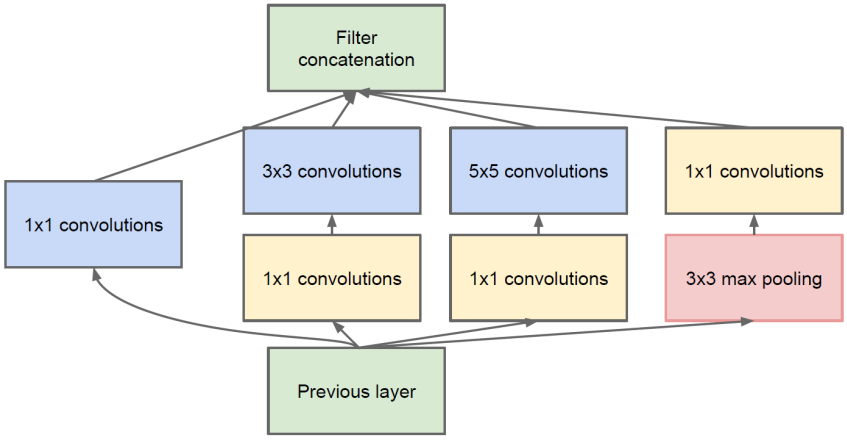

Figure A.6: Concatenation is used extensively in the GoogLeNet architecture [81]. The above block is called an Inception module, and it is repeated multiple times. The max-pooling used here has a step size of one so the image is not downsampled. Image source: Szegedy et al. [81]

A.3 Training

A neural network is a set of operations that relies on a set of learnable parameters $p_0 \dots p_N$ that can be put together into a vector $\boldsymbol{p}$. There is also a training set consisting of vectors $\boldsymbol{a}_0 \dots \boldsymbol{a}_M$. The cost function of a network describes how well the network is performing, and it is often defined describing the error of the network for one input vector $\boldsymbol{a}_m$, given all learnable parameters $\boldsymbol{p}$: $C(\boldsymbol{a}_m, \boldsymbol{p})$. To train the network, an optimization algorithm optimizes the values of the learnable parameters ($\boldsymbol{p}$) so that they minimize the average of the cost over all training vectors: $\frac{1}{M}\sum_{m=0}^{M} C(\boldsymbol{a}_m, \boldsymbol{p})$. As an optimization algorithm, gradient descent is used. To be able to use gradient descent, the cost function has to be differentiable to all parameters in $\boldsymbol{p}$. An efficient way of calculating the gradient for all parameters in a neural network is the backpropagation algorithm [82].

A.3.1 Stochastic mini-batch gradient descent

The gradient of the average of the cost functions is equal to the average of all the gradients, so for normal gradient descent you would have to calculate the gradient for all input vectors every iteration.

$$\nabla\left(\frac{1}{M}\sum_{m=0}^{M} C(\boldsymbol{a}_m, \boldsymbol{p})\right) = \frac{1}{M}\sum_{m=0}^{M} \nabla C(\boldsymbol{a}_m, \boldsymbol{p}) \tag{A.1}$$

To speed up this process, stochastic mini-batch gradient descent only averages the gradients for a subset (mini-batch) of the input vectors to calculate an estimate of the total gradient in every iteration. Typically, the mini-batches are selected in such a way that every input vector has to be used once before using the same input vector again. The number of iterations it takes to use all inputs once, is called an **epoch**. Moreover, every epoch the order of the input vectors is shuffled, so that different input vectors end up in a mini batch together. Larger mini-batch sizes result in a more reliable estimate of the gradient, but smaller mini-batches use less memory and can even help the network generalize better [83].

Several variants of stochastic mini-batch gradient descent are used in the literature, and they mostly differ in if and how they use momentum[82, 84] and adaptive learning rates. This is summarized in table A.3.1. Ruder [66] provides a more complete overview of these algorithms.

	No momentum	Momentum	Nesterov
constant learning rate	Mini batch SGD		
adaptive learning rate	Adagrad, Adadelta, RMSProp	Adam, Adamax, AMSGrad	Nadam

Table A.1: Comparison of mini batch stochastic gradient descent variants used in training neural networks.

A.3.2 Overfitting and validation

The parameters of the network are optimized to minimize the cost on the training set. However, the end goal is to use the network on data that it has not been trained on. Sometimes the network learns properties that are unique to the training set. This is called overfitting. In the worst case scenario the network would learn to recognize each example from the training set separately, without looking at any property that is shared between examples. This way the network could get very good results on the training set and very bad results on unseen data.

If you don't put in measures to prevent overfitting it almost always occurs. Some overfitting may be acceptable, but in general you want to reduce it as much as possible. To be able see how much your network is overfitting you can keep a small part of your data separate as a validation set. This set is not used by your optimizer to train your network, but the loss on the validation set is evaluated every epoch. A plot containing the loss on your training and validation sets is called the training curve plot (Figure A.7).

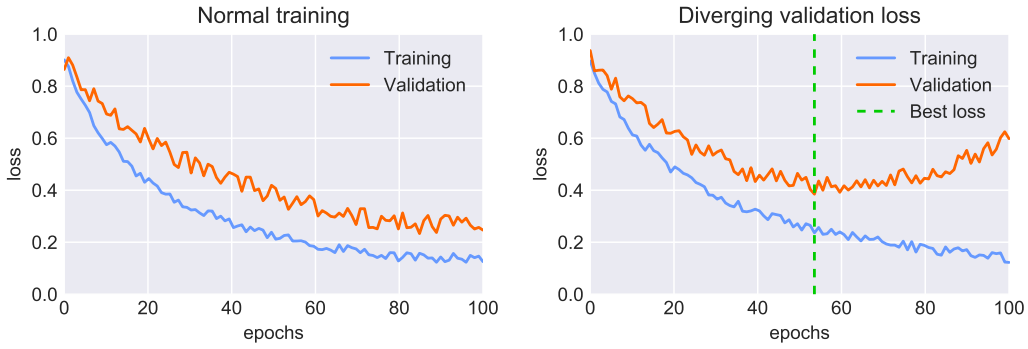

Figure A.7: Training curve examples: Both plots show overfitting, but on the right plot the overfitting gets worse over time. The green line indicates the epoch where the validation loss was lowest.

Sometimes your network will first learn properties that are generalizable, but as the training progresses it will start overfitting more and the validation will increase. One approach to counter this effect is to select the network with minimal loss on the validation instead of minimal loss on the training set. Be aware that if you do this, your result will be biased towards your validation set as well. Because of that, some part of the data is often kept separate as a test set. Experiments on the test set should only be performed during the final evaluation, when training and tuning the network is entirely finished.

A.4 Improvements

A.4.1 Data augmentation

Data augmentation is a technique for *regularization*, which is the term for reducing overfitting. There are other regularization techniques, such as dropout layers [85, 86] and L1/L2-

regularization, but data augmentation has the nice property that it allows you to make the network more robust specifically to the variations that you expect to occur in the data. It works by modifying the training data before it is send to the network. By modifying your data with the variations you expect to happen you can make your network learn these variations and make the network more robust to these variations and less strongly fitted on your training data.

For example, assume you are training a network for detecting faces in pictures. The faces can be close to the camera or further away, resulting in a smaller or bigger size. By nature convolutional neural networks are not scale invariant, but a network can learn separate features for different scales if you provide the network with examples of each scale. Even when your training set only consists of a few pictures, you can still create examples at each scale by scaling your training pictures before sending them to the network. This way it is possible to generate a new set of images each iteration, so the network will never see exactly the same image twice, which strongly reduces overfitting.

A.4.2 Batch Normalization

Batch normalization [87] aims to reduce the training time of a neural network by counteracting the problem of internal covariate shift. Internal covariate shift is that when a network is training, the distribution of the inputs of a layer will change as a result of changes in the parameters of the previous layers. Because of this the parameters of that layer may no longer derive meaningful features from its inputs. This problem becomes worse when a network has more layers because each layer amplifies the changes of the previous layers.

The proposed solution is to put batch normalization layers between the input and the output of two neighboring layers at several points in the network. In a batch normalization layer each feature x is normalized to zero mean and unit variance using the input feature mean μ and variance σ^2 (plus a small ε to avoid division by zero). During training the mean and variance over the mini batch are used, but during evaluation of the network the mean and variance over the entire training set are used instead. Also a learnable scale γ and offset β are added:

$$\mathrm{BN}_{\gamma,\beta}(x) = \gamma\left(\frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}\right)+\beta$$

The normalization makes the distribution of the features more stable during training, which increases training speed (Figure: A.8). However, having a zero mean, unit variance distribution of each feature might not be desirable. An example where a non-unit variance would be meaningful is when the batch normalization layer is followed by a sigmoid activation function. A high variance pushes the activations close to either zero or one, while with a low variance the function is almost linear. To still make it possible to use batch normalization layers everywhere in your network the linear transform relying on γ and β allows offsetting the mean and scaling the variance.

The fact that the normalization procedure is different during training and testing may reduce performance, because the network is not optimizing exactly for how it will be evaluated. This effect becomes stronger as the batch size becomes smaller. However it may also

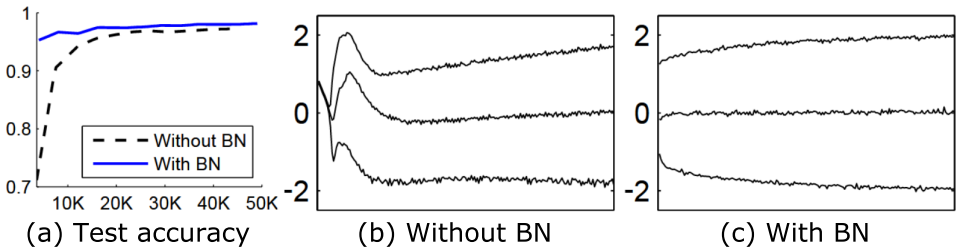

Figure A.8: (a) Test accuracy of a network on the MNIST dataset with and without batch normalization plotted against the amount of training steps. (b & c) The input distribution of one sigmoid activation function as it changes during training plotted as the {15, 50, 85}th percentile. Source: Ioffe and Szegedy [87]

have a positive regularizing effect, because training samples are in a batch together with different samples each epoch. This forces the network to learn parameters that are robust to features that are normalized slightly differently. For this to happen you should have a mini-batch size of at least two and you should shuffle the order of your training data every epoch. Both things are commonly done already because they are also beneficial to the result of mini-batch SGD without batch normalization.

Appendix B

Additional Results and Figures

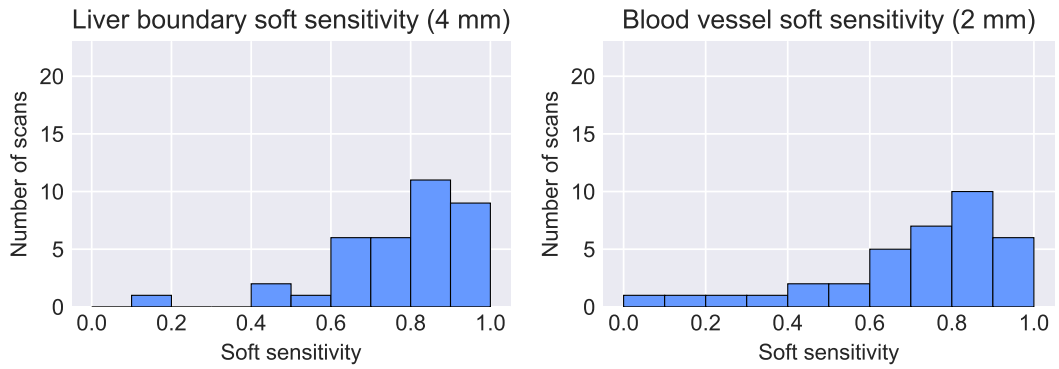

Figure B.1: Distribution of the sensitivity disregarding false negatives that are close to the neural network output. A 4 mm threshold was used for the liver boundary and a 2 mm threshold was used for the blood vessels.

	f_liv	f_ves	w_liv	w_ves	d_trans	average error
1	0.566	0.891	1.136	1.714	17.513	12.430
2	0.944	0.882	1.064	2.182	14.713	12.896
3	0.754	0.879	2.729	1.873	13.228	13.039
4	0.883	0.749	1.493	2.287	23.190	13.418
5	0.653	0.417	2.430	2.762	15.365	13.526
6	0.902	0.739	2.646	0.522	10.274	13.546
7	0.800	0.562	1.659	0.572	7.612	13.552
8	0.854	0.679	2.757	2.419	13.522	13.626
9	0.968	0.767	2.367	2.795	24.657	13.634
10	0.786	0.637	2.373	0.699	17.246	13.756
96	0.694	0.549	2.431	0.634	0.400	16.179
97	0.990	0.474	2.088	2.458	1.920	16.186
98	0.534	0.584	1.738	1.062	4.094	16.306
99	0.669	0.523	2.760	2.467	0.269	16.560
100	0.997	0.405	2.129	0.616	13.252	16.709

Figure B.2: Best ten and worst five settings of the LTS-CD cost function. Each column is color coded according to where a value lies on the range that was used to randomly generate that parameter.

	f_liv	f_ves	w_liv	w_ves	d_trans	average error
1	0.905	0.685	2.350	1.431	18.570	12.500
2	0.914	0.771	0.613	1.373	22.773	12.690
3	0.851	0.857	2.809	0.750	15.452	13.030
4	0.859	0.791	2.465	2.005	11.217	13.046
5	0.892	0.807	2.417	2.897	19.405	13.070
6	0.991	0.675	1.762	1.971	9.571	13.111
7	0.819	0.767	2.678	1.931	20.311	13.197
8	0.754	0.808	2.086	2.488	23.833	13.266
9	0.576	0.856	1.379	0.895	13.925	13.320
10	0.978	0.483	2.145	2.829	7.333	13.334
96	0.680	0.470	1.105	1.800	3.748	15.698
97	0.511	0.445	1.093	0.562	4.179	15.894
98	0.632	0.847	2.234	0.787	0.288	16.438
99	0.593	0.426	1.078	2.430	2.307	16.804
100	0.660	0.891	0.736	0.734	0.861	17.114

Figure B.3: Best ten and worst five settings of the TACD cost function

	d_liv	d_ves	w_liv	w_ves	d_trans	average error
1	10.851	3.817	1.664	2.194	23.929	12.010
2	28.549	6.206	0.819	1.011	11.950	12.319
3	25.776	3.781	0.677	1.024	10.395	12.451
4	8.309	6.027	1.296	0.864	20.446	12.462
5	28.047	5.968	0.971	0.868	22.010	12.472
6	23.803	3.628	1.261	2.366	4.145	12.476
7	9.266	15.247	1.315	1.531	24.240	12.577
8	10.358	11.125	0.633	1.669	17.840	12.651
9	22.863	7.177	2.199	2.016	12.792	12.695
10	23.761	3.980	2.430	2.789	14.866	12.796
96	13.913	12.480	0.515	2.917	23.001	16.737
97	9.937	17.790	0.515	2.447	1.291	17.256
98	4.913	13.602	0.631	2.542	5.616	17.607
99	4.326	28.701	0.570	2.159	1.469	19.245
100	3.948	13.038	1.424	1.776	0.268	19.536

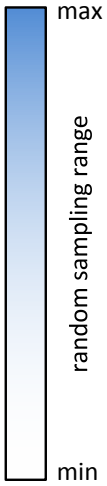

Figure B.4: Best ten and worst five settings of the CACD cost function

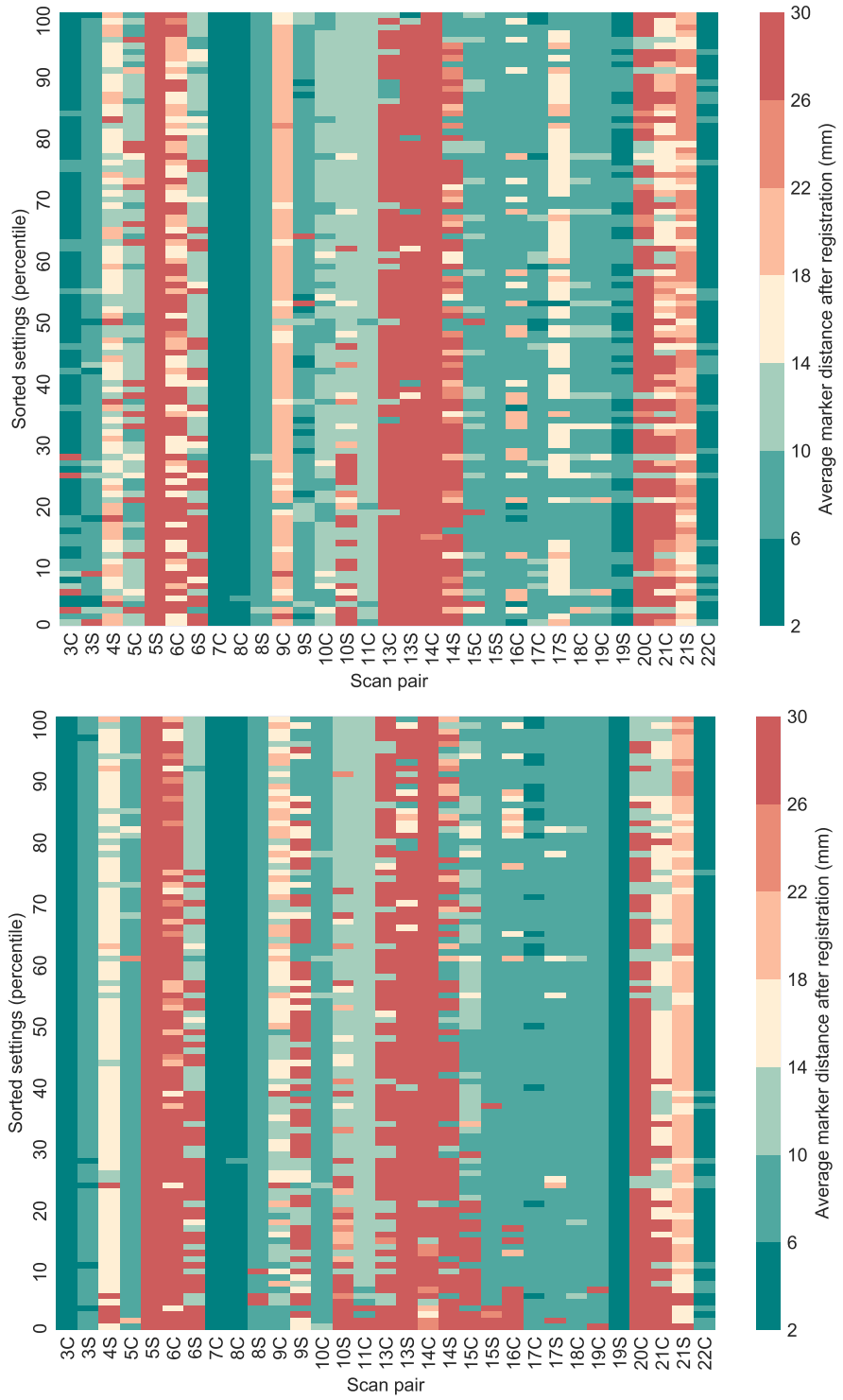

Figure B.5: Heatmaps of the registration error of each scan for each tested setting of the LTS-CD(top) and CACD(bottom) cost functions. Each rectangle represents the average marker distance, which was clipped at 30 mm. The settings are sorted from a low average over the average marker distances (top) to a high average (bottom).

Figure B.6: Examples of registration results that got the score *good*. Only one slice is displayed here, but during scoring the whole 3D volume was available.

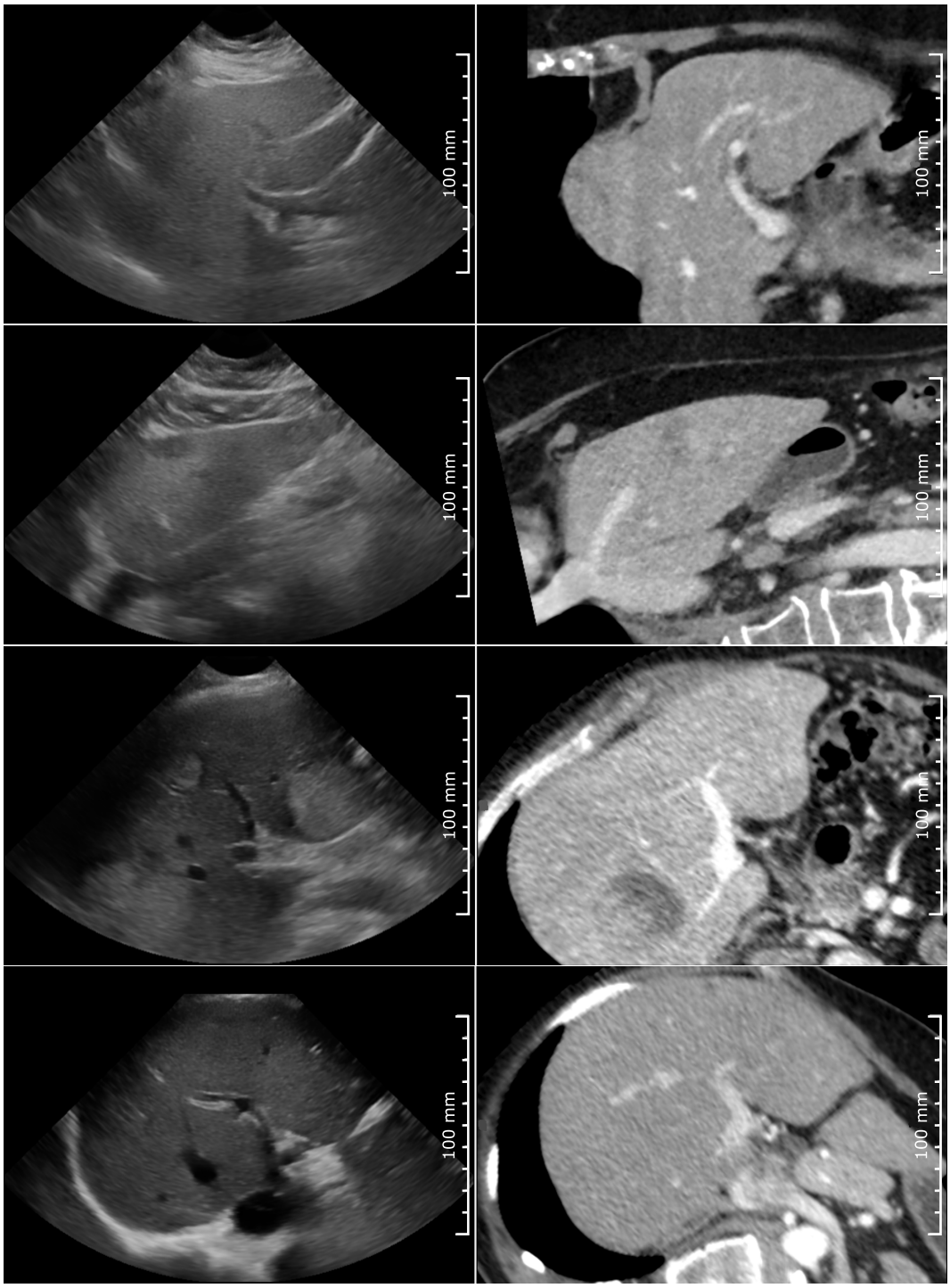

Figure B.7: Examples of registration results that got the score *fair*. Only one slice is displayed here, but during scoring the whole 3D volume was available.

Figure B.8: Examples of registration results that got the score *poor*. Only one slice is displayed here, but during scoring the whole 3D volume was available.

Figure B.9: Examples of registration results that got the score *bad*. Only one slice is displayed here, but during scoring the whole 3D volume was available.