Bachelor’s Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Circuit Theory

3D carotid artery reconstruction from 2D in-vitro ultrasound images

Markéta Kvašová

Supervisor: prof. Dr. Ing. Jan Kybic
Field of study: Medical electronics and bioinformatics
May 2022

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kvašová Markéta**	Personal ID number:	**483774**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Circuit Theory**		
Study program:	**Medical Electronics and Bioinformatics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

3D carotid artery reconstruction from 2D in-vitro ultrasound images

Bachelor’s thesis title in Czech:

Rekonstrukce 3D obrazu karotidy z 2D in-vitro ultrazvukových snímků

Guidelines:

1. Get acquainted with the data and perform a literature survey on the existing methods of 3D volume reconstruction from 2D ultrasound images without positional information.
2. Compensate for the movement of the device in the capturing plane with image registration. Register the volumes created from both capturing directions.
3. Fuse both volumes. Investigate suitable methods.
4. Iteratively improve the 3D reconstruction using 2D-3D registration of the slices with respect to the computed 3D volume.
4. Test the method on simulated and real data, visualize the results and perform quantitative and qualitative experiments.

Bibliography / sources:

[1] O. V. Solberg, F. Lindseth, H. Torp, R. E. Blake, and T. A. Nagelhus Hernes, “Freehand 3D Ultrasound Reconstruction Rlgorithms–A Review,” Ultrasound in Medicine and Biology, 2007.
[2] F. Mohamed and C. V. Siang, “A Survey on 3D Ultrasound Reconstruction Techniques,” Artificial Intelligence - Applications in Medicine and Biology, 2018.
[3] R. Rohling, A. Gee, and L. Berman, “A comparison of freehand three-dimensional ultrasound reconstruction techniques,” Medical Image Analysis, vol. 3, no. 4, pp. 339–359, 1999.

Name and workplace of bachelor’s thesis supervisor:

prof. Dr. Ing. Jan Kybic Biomedical imaging algorithms FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **02.02.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

prof. Dr. Ing. Jan Kybic	doc. Ing. Radoslav Bortel, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Acknowledgements

I would like to thank my supervisor, prof. Dr. Ing. Jan Kybic, for helping me with this thesis. I would also like to thank my family for supporting me in my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 17. May 2022

Abstract

2D–3D reconstruction is the process of creating a volume from a set of 2D images. This reconstruction process results in an estimation of the unknown intensity values of all the individual volume elements. The estimation is done by processing the original 2D images and utilizing the information that can be found in them.

The goal of this thesis is to develop a method to create a 3D model of the carotid artery from 2D in-vitro ultrasound images without their positional information. The carotid artery was scanned in two orthogonal directions, which resulted in the artery's transversal and longitudinal 2D slices. These two sets of slices are used in the reconstruction process to first create two separate volumes. These volume estimations are then improved and fused into a final 3D estimation. Registration is used in several steps of the proposed method to compensate for the lack of information about the slices' position.

The proposed method is tested on both real and artificially created data. The result of this thesis is a 3D visualization of the created volume.

Keywords: registration, ultrasound, medical imaging

Supervisor: prof. Dr. Ing. Jan Kybic

Abstrakt

2D–3D rekonstrukce je proces tvorby objemu z 2D obrázků. Výsledkem rekonstrukčního procesu je ohad neznámých intenzit jednotlivých elementů vytvářeného objemu. Tento odhad je uskutečněn zpracováním původních 2D obrázků a využitím informací, které se v nich nachází.

Cílem této práce je vypracovat metodu pro vytvoření 3D modelu karotidy z 2D in-vitro ultrazvukových obrázků bez informace o pozici. Karotida byla nasnímána ze dvou ortogonálních směrů. Výsledkem tohoto snímání jsou její transverzální a longitudinální řezy. Tyto řezy jsou v rekonstrukci použity nejprve k vytvoření dvou samostatných objemů. Tyto odhady objemů jsou poté vylepšeny a je z nich vytvořen finální 3D odhad. Registrace je použita v několika krocích navrhované metody pro kompenzaci chybějící informace o pozici jednotlivých řezů.

Použitá metoda je vyzkoušena na uměle vytvořených i reálných datech. Výsledkem práce je 3D vizualizace vytvořeného objemu.

Klíčová slova: registrace, ultrazvuk, lékařské zobrazování

Překlad názvu: Rekonstrukce 3D obrazu karotidy z 2D in-vitro ultrazvukových snímků

Contents

Tables

Chapter 1

Introduction

Carotid arteries are two arteries located on the side of the neck. Their function is to supply the head and neck area with oxygen. The arteries in the dataset used in this project may contain arterial plaque, a buildup of calcium and fat on the inner arterial walls. As this buildup grows, it causes a disease called atherosclerosis (Figure 1.1). This disease results in hardened and narrow arteries, through which blood passage is more complicated. The brain is the most crucial organ that carotid arteries supply. When the blockage caused by atherosclerosis becomes too severe, it can lead to stroke [1].

Creating a 3D model from 2D slices of this diseased area can fully capture the state of the illness and better visualize the affected tissue. One 3D model also provides an efficient visualization method in comparison with a set of several 2D slices that were used to create the model. This visualization could better show the artery blockage location created by atherosclerosis. With better visualization and more information about the diseased area, this location and the severity of the illness could be estimated more accurately.

1.1 Task definition

The goal of this project is to create a 3D volume from 2D ultrasound images of the carotid artery. The studied object was scanned in-vitro in two orthogonal directions, which resulted in its transversal and longitudinal slices. The slices' positions were not acquired during the scanning process, so they have to be estimated in the reconstruction process. The 3D volume should be computed using pixel values of the individual slices and the slices' estimated position in space. The proposed method should be evaluated both on real and artificially created data. The final volume representing the original scanned object should be visualized in 3D.

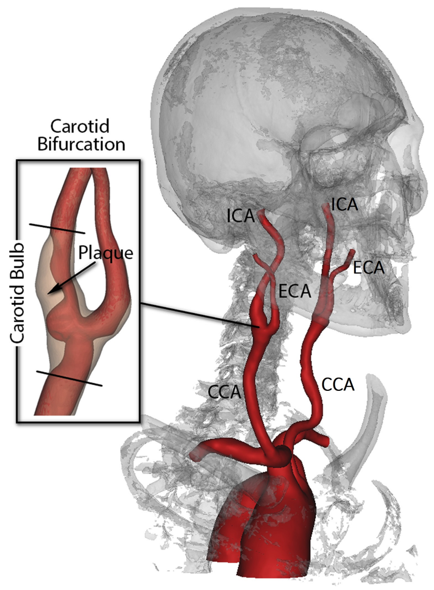


Figure 1.1: Carotid arteries affected by atherosclerosis. The image shows the left and right common (CCA) carotid arteries, which split into the internal (ICA) and external (ECA) carotid arteries. Taken from [2].

1.2 Thesis structure

- In Chapter 2, the in-vitro data is presented, and the creation of artificial data is described.
- In Chapter 3, a theoretical background about ultrasound imaging and the 2D–3D reconstruction process is given.
- In Chapter 4, the proposed solution is described.
- In Chapter 5, the results of the proposed method are summarised.

Chapter 2

Data

Data used in this project is in the form of 2D images. There is no information about the mutual position of the slices, only their order and the direction in which they cut the artery is known. The missing positional information makes volume reconstruction more difficult but not impossible. Every set of images contains data captured in two directions. This information can be used in the reconstruction process to make the resulting volume more accurate. For this project, both real and artificially created data was used. The image positions in space were estimated and used to create a 3D volume. How the images can be positioned in space is shown in Figure 2.2.

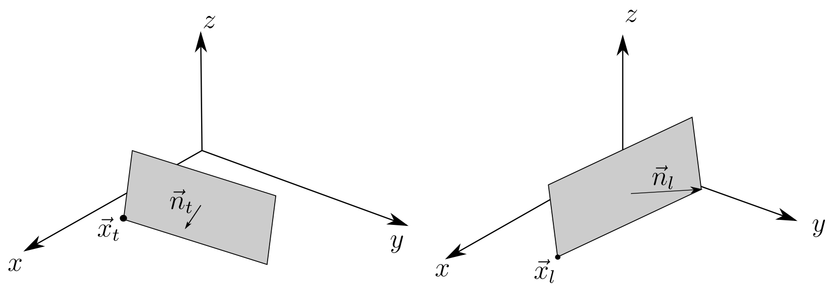


Figure 2.1: How a transversal slice (left) and a longitudinal slice (right) can be positioned in space. A slice's position can be described by the plane the slice lies in, and the slice's corner point.

2.1 Data Analysis

The in-vitro images were taken freehand by an ultrasound device with no position sensor. Dataset from each patient contains images from two orthogonal capturing directions. One direction creates transversal images, and the other creates longitudinal images. One example from each capturing direction is shown in Figure 2.2. The green triangles on the right side of the images shown in Figure 2.2 represent the captured distance of 22 mm. They are 151 pixels apart. This gives the ratio of approximately 0.15 mm/pixel.

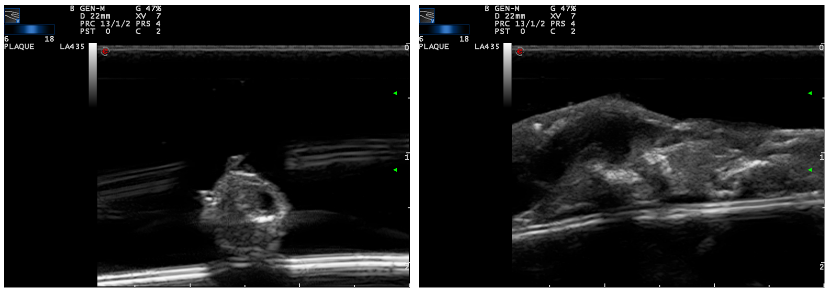


Figure 2.2: Example of a transversal image and a longitudinal image

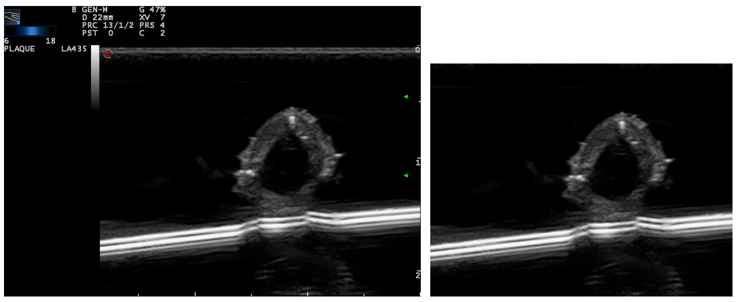


Figure 2.3: Example of an image before and after cropping

Every image is composed of 800 x 556 pixels, which corresponds to the captured area of approximately 120 mm x 83 mm. After cropping the images, to acquire only the important part of the image, they are 580 x 446 pixels large. An example of a cropped image is shown in Figure 2.3.

There are 138 sets of images in total, each set corresponding to one patient. Every set contains around 20–30 images. The approximate scanning distance between two neighboring longitudinal images is 20 mm. The distance between two neighboring transversal images is approximately 6 mm.

Where to find the ultrasound images is described in Appendix A.

2.2 Artificial Data

3D reconstruction of the real data can only be evaluated visually because the true 3D shape of the arteries shown in the images is not known. I created artificial data to test the used method and quantify its results.

The artificial artery is represented by a tube with a changing diameter and a plate beneath it. The tube lies in a uniform grid of 300 x 300 x 600 voxels, it is shown in Figure 2.4. Its outer diameter ranges from 171 pixels to 235 pixels, and its inner diameter ranges from 83 to 149 pixels. Both these diameters monotonically decrease throughout the volume.

Images from two perpendicular directions were created from this model, slicing the artificial artery in the same way it is done in the real data. This resulted in two sets of slices, one longitudinal and the other transversal.

Examples of these slices are shown in Figure 2.5 and how the volume can be cut is shown in Figure 2.6. The resulting transversal slices are 300 x 300 pixels large, and the longitudinal slices are 300 x 600 pixels large.

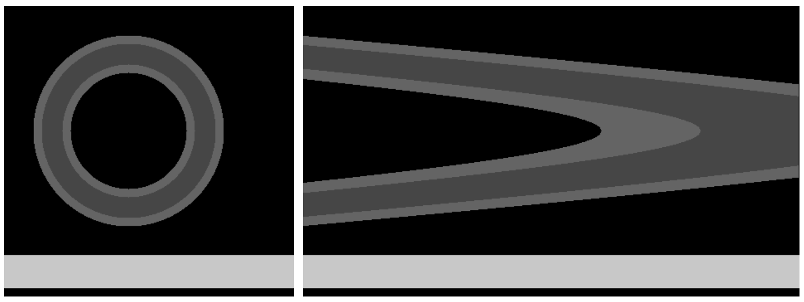


Figure 2.4: Examples of artificial slices.

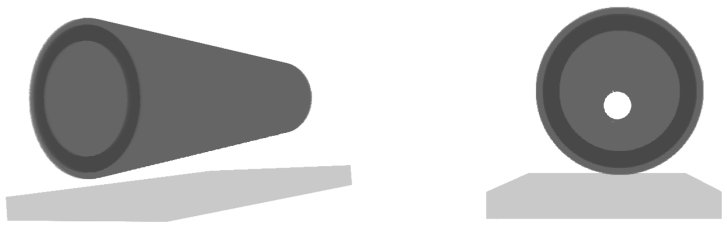


Figure 2.5: Examples of artificial data in space.

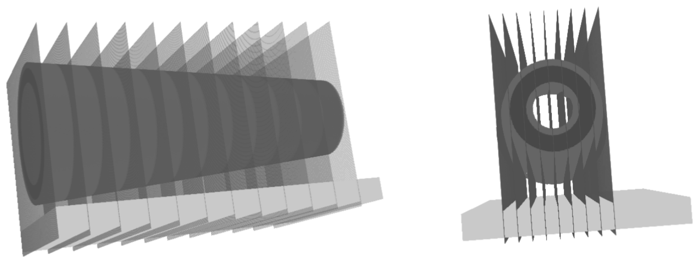


Figure 2.6: Visualisation of how the volume can be cut, creating artificial transversal (left) and longitudinal (right) slices.

Chapter 3

Background

This chapter contains background relating to ultrasound image acquisition, the process of image registration, and 2D–3D reconstruction.

3.1 Ultrasound imaging

Ultrasound images are created by an ultrasound device. This device contains piezoelectric material, which produces a sound wave when an electric field is applied to it. Ultrasound devices use sound waves with frequencies above 20 kHz. These frequencies are above the threshold of human hearing. In the scanning process, the created sound wave is reflected from the boundary of two tissues with different acoustic resistance [3]. The reflected wave is then registered and processed by the device. The ultrasonic beam is swept, and the received signal is used to construct the resulting grayscale image [3].

2D ultrasound shows only the cross-section of the examined area, but there are ultrasound devices that can generate a 3D model, usually from a series of 2D slices [4]. There are different scanning methods for 3D volume reconstruction. These methods are 2D array scanning, mechanical scanning, tracked freehand scanning, and untracked freehand scanning [4]. 2D array scanning uses a 2D array of piezo crystals to create a pyramidal volume. In mechanical scanning, the ultrasound probe is moved by a machine, and freehand scanning is done using a hand-held probe. How the images for 3D reconstruction are acquired by 2D array scanning and freehand scanning is shown in Figure 3.3.

The freehand scanning method creates a series of 2D ultrasound images, usually by moving the device in one direction. In the tracked system, there is a tracking sensor attached to the ultrasound probe. This tracking sensor gives information about its position and orientation. The sensor can be electromagnetic, acoustic, or optical [5]. Untracked freehand scanning is done without any information about the probe's position. Freehand untracked devices are the most portable 3D ultrasound systems because of their simplicity, but the resulting 3D volume quality can be affected by an inconsistent scanning rate

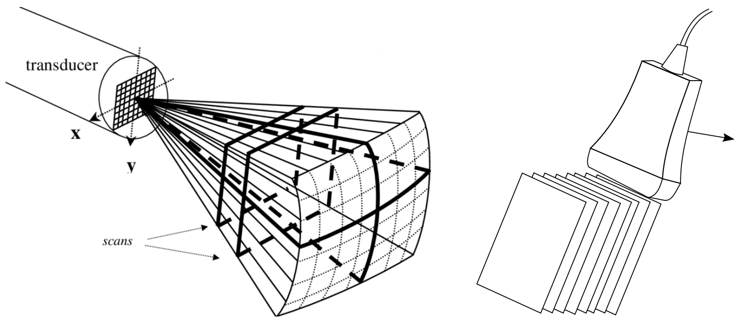


Figure 3.1: Image acquisition for 3D reconstruction. Steerable 2D array scanning (left) obtains multiple slices at once. Freehand scanning (right) results in a series of slices taken one after another. Taken and edited from [6] and [4].

or a changing angle [4].

3.2 How to reconstruct a 3D volume from ultrasound slices

A 3D reconstruction method takes 2D ultrasound data and uses these images to approximate 3D volume data. There are several methods of 2D–3D reconstruction. Some are dependent on the information about the position of the ultrasound device, which can only be acquired by a position sensor in the scanning process [4]. When the information about the slice position is not known, the data found in the images can be used to approximate their mutual position in 3D space. If the object is scanned from multiple directions, it can help with better slice localization [7]. But 3D volume reconstruction is possible even with slices acquired only from one scanning direction [8].

3.3 How the scanning process affects reconstruction quality

The quality of a reconstructed 3D volume is dependent on the capturing process. In the case when the slices are acquired by a hand-held probe, the distances between slices can vary. The angle between the ultrasound device and the captured object can also change, which results in slices that have varying angles to their neighboring slices. Sometimes the slices can even intersect, for example, when the object is scanned densely and the capturing angle changes rapidly. On the other hand, when the object is scanned too sparsely, there is not enough information to create an accurate 3D model of the original object.

If the ultrasound device also moves in any other direction but the scanning direction, it results in a change of the object's position in the image. These changes in position can be compensated by image registration (see Section 4.3).

Images acquired with ultrasound also contain a lot of noise and artifacts, for example, acoustic shadowing, speckle noise, and refraction [4]. These unwanted phenomena can make ultrasound image processing more challenging. Creating an accurate 3D reconstruction can also be made more difficult by tissue deformation, which can occur in the capturing process [9].

3.4 2D–3D Reconstruction methods

2D–3D reconstruction methods can be divided into three groups by their approaches to the problem. There are pixel-based methods, voxel-based methods, and function-based methods [4]. All these methods take a series of images and compute values in a 3D volume. This volume contains the estimation of the scanned area.

3.4.1 Pixel-based methods

Pixel-based methods consist of two steps, a bin-filling step and a hole-filling step [4]. In the bin-filling step, the algorithm goes through every pixel of every image and uses these pixels' values to fill the array of volume elements (voxels) [4]. The elements of the 3D array which were not assigned any value in the first step are computed in the hole-filling step [9].

If the position and orientation information is known from data acquisition, this information is used to determine the map function between the 2D ultrasound image coordinates and the 3D volume coordinates [10]. A transform T of image coordinates (u, v) is defined by a matrix $A \in \mathbb{R}^{3\times 2}$ and a vector $\vec{t} \in \mathbb{R}^3$, which assign 3D coordinates (x, y, z) to coordinates (u, v) as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = A \begin{pmatrix} u \\ v \end{pmatrix} + \vec{t} \tag{3.1}$$

An example of a method for the bin-filling step is the nearest-neighbor method. In this method, each pixel's value is used to fill the nearest voxel [9]. In the case where multiple pixels correspond to one voxel, their average is computed [9].

After processing every image in the first step, there can still be some empty voxels. There can be a method that assigns values to these elements of the 3D grid, but not all voxels need to be assigned a value [11].

In the hole-filling step, values of the empty voxels can be averaged based on the values of the surrounding voxels [9]. An average of a voxel can be computed from a surrounding grid of $(2l+1)^3$ voxels as

$$V'(x,y,z) = \frac{1}{n_{(V\neq 0)}} \sum_{i=-l}^{l} \sum_{j=-l}^{l} \sum_{k=-l}^{l} V(x+i, y+j, z+k), \tag{3.2}$$

where $n_{(V\neq 0)}$ is the number of surrounding non-zero voxels and V is the volume filled in the bin-filling step. This computation is done only for the voxels which have some surrounding non-empty voxels that can be averaged. If there are still some missing voxel values after this averaging step, the size of the surrounding grid can be increased, and the missing values can be averaged from the original values in this larger surrounding grid [9]. Another option for filling the empty voxels is by interpolation between the two closest non-zero voxels [11].

3.4.2 Voxel-based methods

In voxel-based methods, the whole 3D grid is traversed, and the value for each voxel is computed from the 2D images [11]. This approach results in a grid with no empty voxels, so there is no need for a bin-filling step.

Voxels' values can either be assigned directly from one pixel, as in the voxel-nearest neighbor algorithm or from multiple pixels, as in the distance-weighted method.

In the voxel-nearest neighbor method, the nearest pixel is found by calculating a normal from the computed voxel to each input image. The shortest normal gives the closest image, and the pixel's value closest to the normal is assigned to the voxel [11]. If the distance of the voxel to the closest pixel is too large, this method generates large reconstruction artifacts [12].

A normal of an image is computed from the point-normal plane equation. The normal vector is computed as

$$(n_1, n_2, n_3) = (x_{(rt)}, y_{2(rt)}, z_{(rt)}) \times (x_{(up)}, y_{(up)}, z_{(up)}), \tag{3.3}$$

where $(x_{(rt)}, y_{(rt)}, z_{(rt)})$ is a vector that defines the step to move one pixel right in the image and $(x_{(up)}, y_{(up)}, z_{(up)})$ is a vector that defines the step to move one pixel up in the image [13].

The point from the lower-left corner of the image $(x_{(c)}, y_{(c)}, z_{(c)})$ can be chosen to complete the point-normal equation:

$$n_1(x - x_{(c)}) + n_2(y - y_{(c)}) + n_3(z - z_{(c)}) = 0. \tag{3.4}$$

This equation can be rewritten as

$$n_1x + n_2y + n_3z = n_1x_{(c)} + n_2y_{(c)} + n_3z_{(c)}, \tag{3.5}$$

or

$$n_1x + n_2y + n_3z = b. \tag{3.6}$$

The distance of the plane form the calculated voxel's coordinates (x_v, y_v, z_v) is then computed as

$$d = \frac{n_1 x_v + n_2 y_v + n_3 z_v - b}{\sqrt{n_1^2 + n_2^2 + n_3^2}}. \tag{3.7}$$

The closest point (x_p, y_p, z_p) to the calculated voxel (x_v, y_v, z_v) is then computed as

$$(x_p, y_p, z_p) = (x_v, y_v, z_v) + d \cdot (u_1, u_2, u_3), \tag{3.8}$$

where (u_1, u_2, u_3) is the normalized normal vector of the image pointing away from the computed voxel [13]. Then the closest point value is bilinearly interpolated by the four enclosing pixels. Bilinear interpolation in the slice's plane at coordinates (x', y') is computed as

$$\begin{aligned} g(x', y') =& w_{u,v} g(u, v) + w_{u+1,v} g(u+1, v) + \\ &+ w_{u,v+1} g(u, v+1) + w_{u+1,v+1} g(u+1, v+1) \end{aligned} \tag{3.9}$$

where

$$w_{u,v} = (u + 1 - x')(v + 1 - y'), \tag{3.10}$$
$$w_{u+1,v} = (x' - u)(v + 1 - y'), \tag{3.11}$$
$$w_{u,v+1} = (u + 1 - x')(y' - v), \tag{3.12}$$
$$w_{u+1,v+1} = (x' - u)(y' - v), \tag{3.13}$$

are the weights of the surrounding pixels' values, considering that the pixels are one unit apart [14].

Values in a uniform grid between two known slices can also be calculated with linear interpolation [11] or by transformation interpolation, which uses the original pixel values and a transformation between the interpolated slices (see Section 4.4.2). If the slices constructing a volume V are parallel to each other, then the value of V at (x, y, z) between two known values $V(x_0, y, z)$ and $V(x_1, y, z)$ can be linearly interpolated as

$$V(x, y, z) = V(x_0, y, z) + (x - x_0)\left(\frac{V(x_1, y, z) - V(x_0, y, z)}{x_1 - x_0}\right). \tag{3.14}$$

for a case when the interpolation is done in the x-axis. Linear interpolation between two parallel slices is shown in Figure 3.2.

Another type of voxel-based method uses distance-weighing [9]. In the distance-weighted algorithm, a resulting voxel value $W(x, y, z)$ is computed from n values as

$$W(x, y, z) = \sum_{i=1}^{n} w_i \cdot W(x_i, y_i, z_i), \tag{3.15}$$

where

$$w_i = \frac{1}{d((x_i, y_i, z_i), (x, y, z))} \tag{3.16}$$

are the weights of the contributing voxels, computed from their distance from the resulting voxel. The resulting volume can be too smoothed, and some information can be lost in the reconstruction process because averaging is used in the distance-weighting method [4].

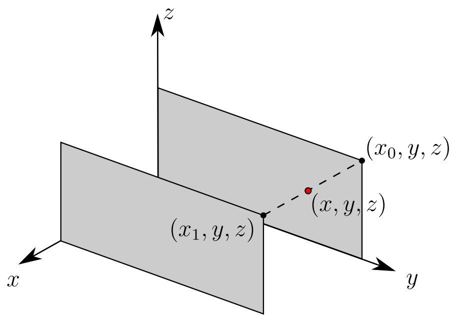


Figure 3.2: Linear interpolation in the x-axis between two known points $V(x_0, y, z)$ and $V(x_1, y, z)$ of two parallel slices.

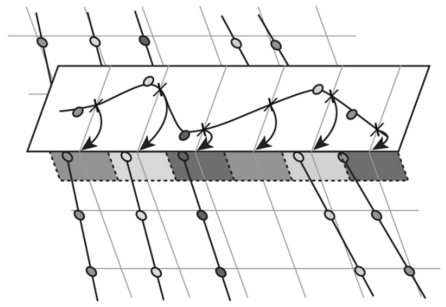


Figure 3.3: Interpolation with a function visualized along one dimension. The full lines with points represent the original images and their pixels. A function (represented by a curve) through these points is estimated. Then it is evaluated at regular intervals, and the computed values are assigned to the correct voxels. The evaluated points are represented by x-marks. Taken from [11].

3.4.3 Function-based methods

In function-based methods, the voxel values are calculated from a function, for example, a polynomial or a radial basis function [9]. In these methods, a function is chosen to pass through the pixels of the input data. Then it is evaluated at regular intervals to determine the values in the 3D grid [11]. Because of measurement errors, such as tissue motion, the chosen function can only approximate the pixel data and does not need to pass exactly through them [9].

Sometimes the data from the images can be divided into segments to reduce the required number of computations. Then the data is used to compute the approximation for that segment. These segments can overlap to create smooth connections [11]. One of the disadvantages of function-based methods is their high computational requirements. On the other hand, they can produce high-quality 3D volumes [4].

3.5 Registration

When there are two images of the same object or a scene, it can be useful to find the mapping function that relates image coordinates in the two images corresponding to the same physical point. This can also be done for two volumes representing the same object or an image and a volume. Registration can be formulated as an optimization problem that finds a transformation T that minimizes the dissimilarity measure of these two images or volumes. The result of this transformation is that for every point in one image, the corresponding point in the second image is found [14]. For the elements of a fixed image or volume $f(\vec{x})$ and the elements of a moving image or volume $g(\vec{x})$, where $\vec{x}$ are the coordinate vectors in two or three dimensions, the coordinate transformation is described by the following equation:

$$\hat{T} = \arg\min_{T} J(f(\vec{x}), g(T(\vec{x}))) \tag{3.17}$$

where J is a dissimilarity measure [15, 16].

In our case, minimizing the dissimilarity measure of two objects with registration can be used to solve the issue of non-existent positional information about the images. The problem to solve is not knowing the mutual position of the slices captured in one direction. Another problem is not knowing the transversal and longitudinal slices' mutual position.

3.5.1 Image registration

Image registration is a process in which a transformation between the coordinate systems of two images is found [16]. This transformation is used to align these images.

A transformation T of coordinates $\vec{x}$ of the moving image g creates transformed coordinates $\vec{x}'$. A similarity measure J computes how well the transformed moving image $g(T(\vec{x}))$ and the fixed image $f(\vec{x})$ match each other. An example of a similarity measure is a normalized cross-correlation function [17], defined as

$$J(f(x,y), g(T(x,y))) = \frac{\sum_x \sum_y (f(x,y) - \overline{f})(g(T(x,y)) - \overline{g_T})}{\sqrt{\sum_x \sum_y (f(x,y) - \overline{f})^2}\sqrt{\sum_x \sum_y (g(T(x,y)) - \overline{g_T})^2}}, \tag{3.18}$$

where $\overline{f}, \overline{g_T}$ are the average values of the fixed and the moving image.

An example of a dissimilarity measure is the square of the difference of image intensity, defined as

$$J(f(x,y), g(T(x,y))) = \sum_x \sum_y (f(x,y) - g(T(x,y)))^2. \tag{3.19}$$

The optimization of the similarity function is the role of the optimizer. Gradient descent [18] is an example of an optimizer, which finds a function's

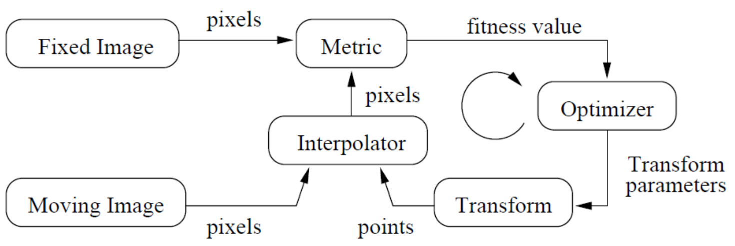


Figure 3.4: Registration process. The chosen metric compares the similarity of the moving image and the fixed image. The moving image's transform is then found by iteratively finding the local minimum of the metric. The interpolator then computes the pixel values at the desired points, so the images can be compared again. Taken from [19].

local minimum. One step of the gradient descent algorithm with parameters θ and learning rate γ is computed as

$$\theta_{n+1} = \theta_n - \gamma \nabla J(\theta), \tag{3.20}$$

where θ are parameters defining the transformation T.

After applying the transformation, the transformed image must be resampled because the values in the transformed coordinates do not need to be known. These values are computed by an interpolator. The whole image registration process is shown in Figure 3.4

3.5.2 Stopping criteria for the gradient descent optimizer

The chosen optimizer works iteratively, and its criteria determine the number of iterations that will be performed. The following parameters define the gradient descent optimizer:

- Learning rate — a multiplicative factor applied on the gradient of the measure.
- Minimum step size — the step size is the difference between the values of two consecutive iterations. The registration is stopped when this difference is smaller than the predefined minimum step size.
- Gradient magnitude tolerance — determines the minimal computed gradient magnitude. When the gradient magnitude is smaller than the defined tolerance, the iteration process is stopped.
- Number of iterations — the maximum number of iterations the registration goes through. [19]

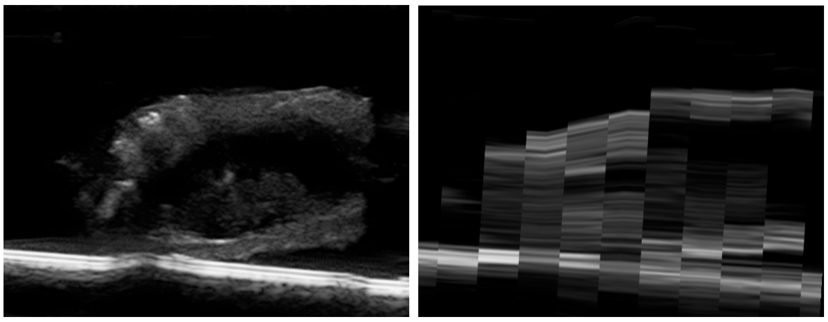


Figure 3.5: Two corresponding cross-sections of the created volumes (longitudinal on the left and transversal on the right). The transversal cross-section looks pixelated because of interpolation.

3.5.3 Mutual information

Mutual information can be used as a similarity measure for volume registration. This measure can even be used for volumes of different modalities, for example, one volume being created by magnetic resonance (MT) and another by computer tomography (CT) [20]. Our longitudinal and transversal volumes are of the same modality, both made from ultrasound images, but the corresponding volume elements can have different intensity values because the slices were captured at different times and from different angles (Figure 3.5).

Mutual information makes a similarity measure I defined as

$$I(U, V_T) = H(U) + H(V_T) - H(U, V_T), \tag{3.21}$$

where U and V are the original compared volumes and V_T was created from V by transforming the coordinates of V with T, and H is entropy [20]. Mutual information is computed from histograms of the studied volumes [20].

3.5.4 Multi-resolution registration

Multi-resolution registration can be used to improve the speed, accuracy, and robustness of the volume registration process. The principle of multi-resolution lies in creating volume pyramids that consist of scaled-down versions of the registered volumes (Figure 3.6). In multi-resolution registration, the registration is first performed with the volumes at the coarsest resolution. The transform parameters determined by the registration are then used to initialize the registration at the next finer level. This is repeated until the finest level of image resolution is reached [19].

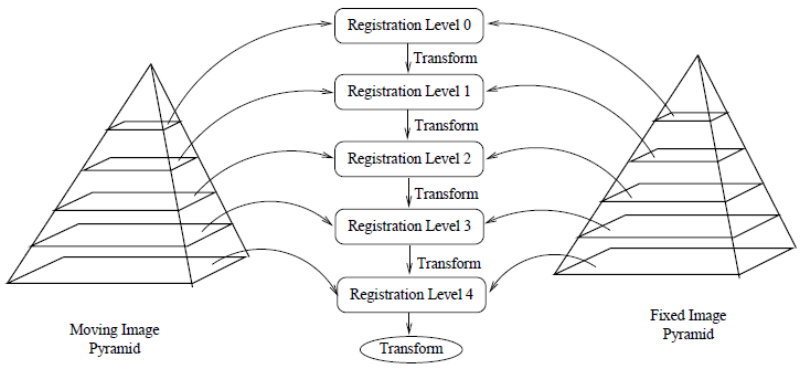


Figure 3.6: Representation of multi-resolution registration. The registered objects are subsampled at various rates, and the subsampled versions are used at different registration levels. Taken from [19].

Chapter 4

Methods

This chapter describes the full process of 2D–3D reconstruction. Technical details of the implementation can be found in Appendix A.

4.1 Proposed volume reconstruction method

The proposed method can be described by the following steps:

- Use image registration to separately align all the images in the transversal image set and the longitudinal image set. Do this to compensate for the unwanted movements of the scanning probe, which cause a change of the object's position in individual slices (Section 4.3).
- Create two volumes, one from longitudinal slices and the other from transversal slices (Section 4.4).
- Find and apply the transformation between the two created volumes. This results in the first estimation of the transversal and longitudinal slices' mutual position (Section 4.5).
- Improve the estimation of the slices' position by repeating these steps:
    - Register the longitudinal slices in the transversal volume, update their coordinates and create a new longitudinal volume from these transformed slices.
    - Register the transversal slices in the longitudinal volume, update their coordinates and create a new transversal volume from these transformed slices (Section 4.6).
- Create a final volume from both sets of slices with their updated position in space (Section 4.7).

The whole process is represented in Figure 4.1.

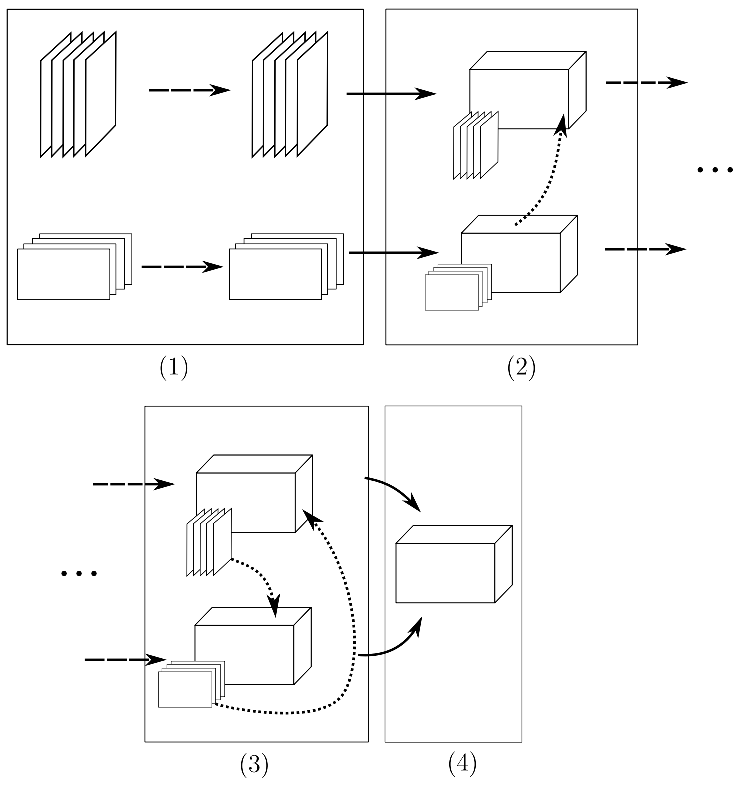


Figure 4.1: The proposed method. In part (1), images are registered. In part (2), two volumes are created from the registered images, these volumes are then registered and their mutual transformations are found. The volumes are then transformed. In part (3), slices are registered in the transformed volumes. In part (4), the final volume is created.

4.2 Preprocessing

Images corresponding to one patient had to be manually separated into two groups, longitudinal images and transversal images. In the first part of the 3D volume reconstruction, these two sets of images are handled separately to create two volumes. Where to find these separated images is written in Appendix A.

4.3 Registration of a series of images

The scanning device did not move in a perfectly straight line during the scanning process, which resulted in the change of the object's position in every image. One image can be chosen as a fixed image, and the rest of the images can be shifted and rotated to compensate for some of the scanning device's unwanted movement.

The image in the middle of the series is used as a fixed image in the

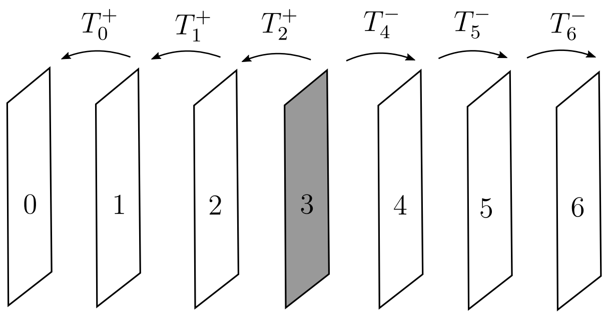


Figure 4.2: Example of the registration process of a series of images. The grey image represents the chosen fixed image which will not be transformed.

registration process to minimize the distance between the fixed image and any other moving image. The registration is done gradually by finding a rigid transformation (a transformation composed of rotation and translation) between each pair of neighboring images. To align the image that neighbors with the middle image, only the transformation between these two images needs to be applied. Every following image is transformed by composing all the transformations between the currently transformed image and the middle image.

Let us consider a series of n images $f_0, ..., f_{n-1}$, and let $k = (n-1)/2$ be the index of the chosen fixed image in the middle of the series. The rigid 2D transformations between the coordinate systems are found with registration as

$$T_i^+ = \arg\max_T J(f_{i+1}(x,y), f_i(T(x,y))) \tag{4.1}$$

for the images preceding the chosen fixed image. And

$$T_i^- = \arg\max_T J(f_{i-1}(x,y), f_i(T(x,y))) \tag{4.2}$$

for the images following the chosen fixed image, where J is the cross-correlation function. The preceding images are transformed as

$$f_i'(x,y) = f_i(T_i^+ \circ T_{i+1}^+ \circ ... \circ T_{k-1}^+(x,y)). \tag{4.3}$$

And the following images are transformed as

$$f_i'(x,y) = f_i(T_i^- \circ T_{i-1}^- \circ ... \circ T_{k+1}^-(x,y)). \tag{4.4}$$

4.3.1 Image registration results

Image registration is done for every pair of images with the SimpleITK library, using a gradient descent optimizer and the following parameters: learning rate=1, minimal gradient magitude step=0.001, number of iterations=100, gradient tolerance=0.0005. Cross-correlation was used as a metric and a

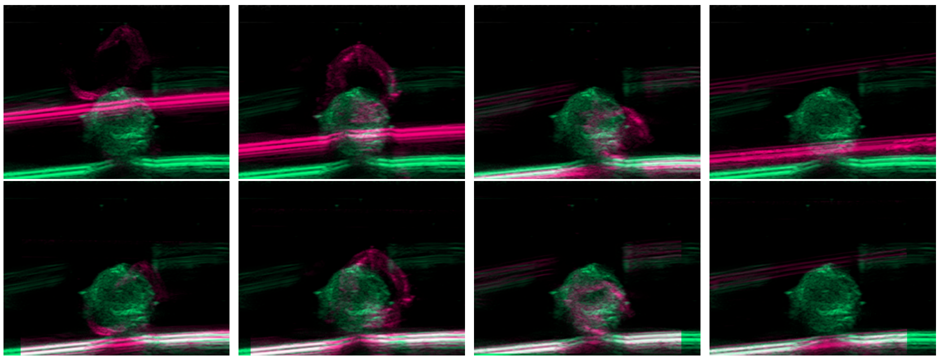


Figure 4.3: Transversal images before (top row) and after (bottom row) image registration. The green image is the chosen middle fixed image that is not being transformed. The pink image is the moving image.

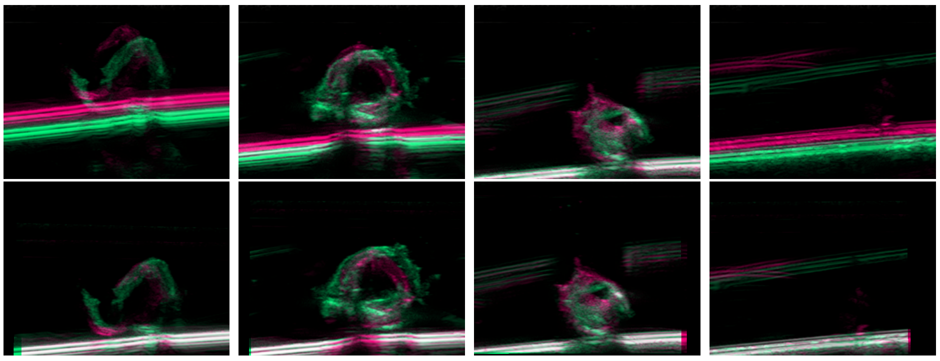


Figure 4.4: Transversal images before (top row) and after (bottom row) image registration. All images contain two neighboring images.

linear interpolator was used for resampling. The usual time for registering and transforming all the images is about one minute.

Figures 4.3 and 4.5 show how the images were aligned with the middle fixed image. Figures 4.4 and 4.6 show the alignment of several neighboring images.

Tables 4.1 and 4.2 contain the mean absolute differences for image pixel values of neighboring transversal and longitudinal images before and after the registration process. Values for every image pair decrease after registration.

image indexes	before	after
0,1	31.97	7.10
1,2	26.85	7.72
2,3	14.19	8.55
3,4	19.75	6.25
4,5	25.12	12.66
5,6	20.21	12.52
6,7	16.22	9.63
7,8	26.81	8.56
8,9	9.54	7.69
9,10	14.89	8.39
10,11	9.43	8.42
11,12	29.00	8.20
12,13	20.94	7.39
13,14	19.89	5.76
14,15	26.35	5.35
total sum	311.16	124.19

Table 4.1: Mean absolute difference between two neighboring transversal images before and after registration.

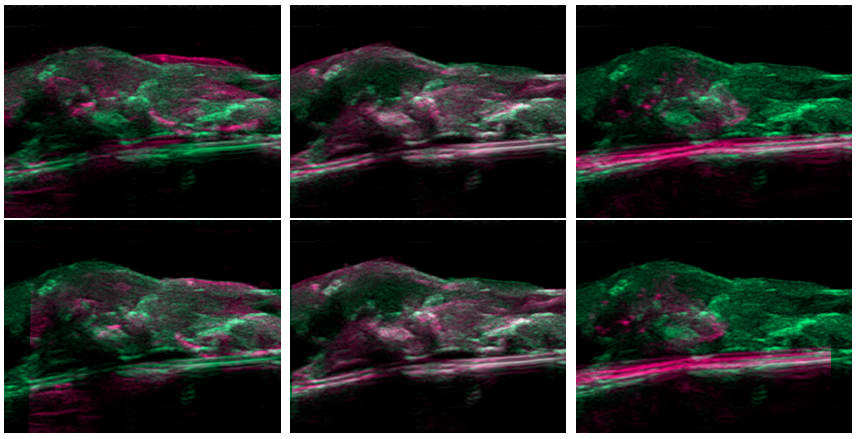


Figure 4.5: Longitudinal images before (top row) and after (bottom row) image registration. The green image is the chosen middle fixed image that is not being transformed. The pink image is the moving image.

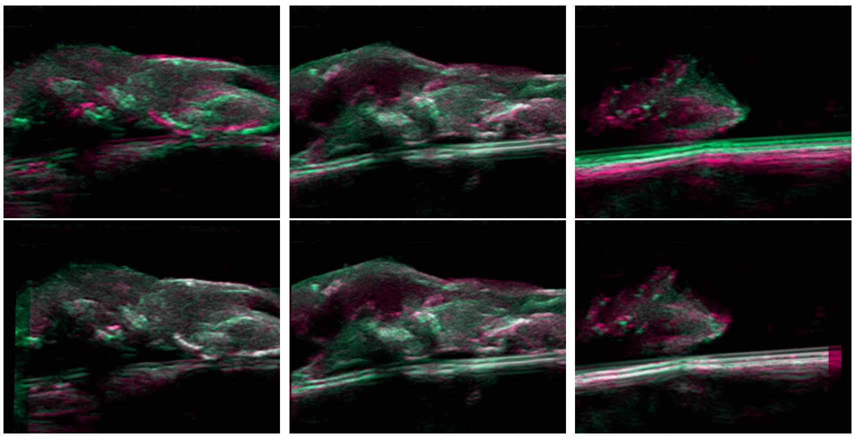


Figure 4.6: Longitudinal images before (top row) and after (bottom row) image registration. All images contain two neighboring images.

image indexes	before	after
0, 1	10.76	10.16
1, 2	18.92	16.28
2, 3	13.70	9.47
3, 4	11.55	6.94
4, 5	16.60	14.26
5, 6	12.70	12.44
6, 7	25.43	21.40
7, 8	22.99	11.18
8, 9	19.70	18.28
9, 10	19.77	9.26
total sum	172.12	129.67

Table 4.2: Mean absolute difference between two neighboring longitudinal images before and after registration.

4.4 Volume creation

One volume is created from longitudinal images and another is created from transversal images. The process of making one volume goes as follows:

- Assume that the ultrasound slices corrected by registration are spaced evenly and parallel to each other. They fill a 3D uniform grid at regular intervals. The rest of the slices creating this grid have unknown values.
- Create a 3D uniform grid of the size NxHxW, where N is the total number of known and unknown slices, H is the height of the images, and W is their width.
- Compute the positions of the known slices in the grid.
- Fill the grid with the known slices at their positions and calculate the values of the unknown slices between them.

4.4.1 The distance between two slices and the first estimation of slices' position

The distance between two known slices is measured in the number of missing slices between them. One assumption made for simplification of the first estimation is that the first slice in one set is positioned at the first voxel of the slices in the other set. Every following slice is shifted by a multiple of the distance between two slices.

The distance between two transversal slices is estimated in a way to create a volume of a depth that is close to the width of the longitudinal images. To reach this goal, the total number of missing transversal slices $N_{t_{miss}}$ is computed as

$$N_{t_{miss}} = W_l - N_t, \tag{4.5}$$

where W_l is the width of the longitudinal images and N_t is the number of known transversal slices. The number of all missing transversal slices $N_{t_{miss}}$ is also computed as

$$N_{t_{miss}} = d_t \cdot (N_t - 1), \tag{4.6}$$

where d_t is the number of missing slices between two known slices and N_t is the number of all known transversal slices.

The number of missing slices between two known slices d_t is then computed from 4.5 and 4.6 as

$$d_t = \frac{N_{t_{miss}}}{N_t - 1} = \frac{W_l - N_t}{N_t - 1}. \tag{4.7}$$

For the longitudinal volume, the depth should be close to the artery diameter because the longitudinal images slice the artery from one edge to the other. The number of all missing longitudinal slices $N_{l_{miss}}$ is computed as

$$N_{l_{miss}} = d_l \cdot (N_l - 1), \tag{4.8}$$

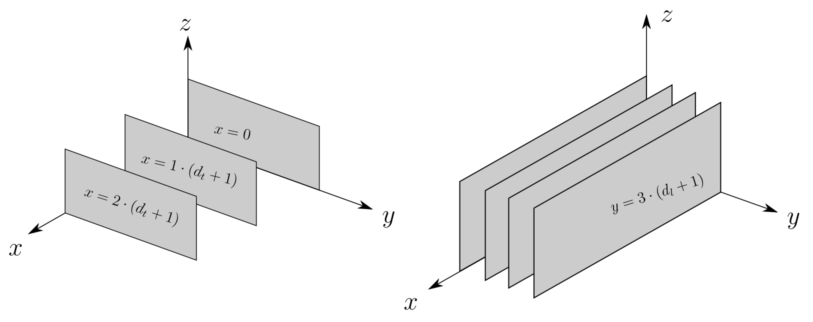


Figure 4.7: The first estimation of slices' position. The first slice is at the origin, and every following slice is shifted by the estimated distance. The image on the left shows this for transversal slices and the image on the right for longitudinal slices.

where d_l is the number of missing slices between two known slices and N_l is the number of all known longitudinal slices. The number of missing longitudinal slices $N_{l_{miss}}$ is also computed as

$$N_{l_{miss}} = d_a - N_l, \tag{4.9}$$

where d_a is the diameter of the artery in pixels and N_l is the total number of longitudinal slices. The number of missing longitudinal slices d_l is computed from 4.8 and 4.9 as

$$d_l = \frac{N_{l_{miss}}}{N_l - 1} = \frac{d_a - N_l}{N_l - 1}. \tag{4.10}$$

With this information about the number of missing slices, the slices can be put into 3D space. For the transversal volume, the first slice in the series is put at the origin. Then the next slice is shifted by $d_t + 1$ on the x-axis, to fit d_t number of slices between them. This shift is then done for every following slice. The same is done for the longitudinal slices, but the slices are put into the space rotated by 90° around the z-axis. This means that they are being shifted on the y axis by $d_l + 1$. So the i-th transversal slice u_i passes through the point

$$\vec{x}_{(i)} = (i \cdot (d_t + 1), 0, 0). \tag{4.11}$$

These points are found for each transversal slice $u_0, ..., u_{N_t - 1}$.

And the j-th longitudinal slice v_j passes through the point

$$\vec{x}_{(j)} = (0, j \cdot (d_l + 1), 0). \tag{4.12}$$

These points are found for each slice $v_0, ..., v_{N_l - 1}$.

How all slices are put in 3D space is shown in Figure 4.7.

4.4.2 Interpolation

Two methods of interpolation were implemented and tested. The comparison of these two method is made in Section 4.6.1. The first method is piecewise linear interpolation described in Section 3.4.2 and the second method is transformation interpolation.

Transformation interpolation between two slices f_a, f_b is done with the use of a transformation T_θ, which was found by registering these images, using f_a as the fixed image and f_b as the moving image. This means that T_θ is found as

$$T_\theta = \arg\min_T J(f_a(\vec{x}), f_b(T(\vec{x}))), \tag{4.13}$$

where J is a dissimilarity measure.

T_θ is defined by its parameters θ. Every new constructed slice $f_1.,.f_d$ between f_a and f_b is made by transforming one of the two original slices by a transformation T_{θ_j}, which is created from parameters θ_0 and $\Delta\theta$.

The particular transformation T_{θ_0} is an identity for some parameters θ_0. For these parameters, it holds that

$$T_{\theta_0}(\vec{x}) = \vec{x}. \tag{4.14}$$

Parameters $\Delta\theta$ are used to linearly change parameters θ_0 into parameters θ. They satisfy the equation

$$\sum_{i=1}^{d} \Delta\theta = \theta \tag{4.15}$$

A j-th slice in the first half of the interpolated volume is constructed as

$$f_j = f_a(T_{\theta_j}(\vec{x})), \tag{4.16}$$

where the parameters θ_j of T_{θ_j} are computed as

$$\theta_j = \theta_0 - \sum_{i=1}^{j} \Delta\theta. \tag{4.17}$$

Slices in the second half of the interpolated volume are constructed as

$$f_j = f_b(T_{\theta_j}(\vec{x})), \tag{4.18}$$

where θ_j is computed as

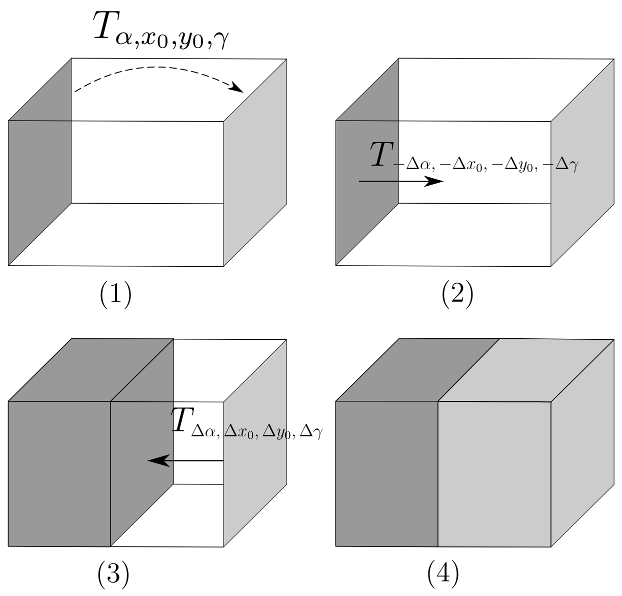
$$\theta_j = \theta_0 + \sum_{i=1}^{d-j+1} \Delta\theta. \tag{4.19}$$

If the used transform T_θ is a similarity transform, it is represented by a translation in two directions in the image plane (x_0, y_0), rotation angle around the center of the image α, and a scaling factor γ. The identity parameters θ_0 for this transform are:

$$\alpha = 0 \tag{4.20}$$

$$x_0 = 0 \tag{4.21}$$

$$y_0 = 0 \tag{4.22}$$



$T_{\alpha,x_0,y_0,\gamma}$
$T_{-\Delta\alpha,-\Delta x_0,-\Delta y_0,-\Delta\gamma}$
$T_{\Delta\alpha,\Delta x_0,\Delta y_0,\Delta\gamma}$
(1)
(2)
(3)
(4)

Figure 4.8: The process of filling the voxels between two known slices. In (1), the similarity transform between the two slices is found. In (2) and (3), the transform parameters are used to create the slices that fill the empty voxels. This results in a filled 3D array shown in (4).

$$\gamma = 1 \tag{4.23}$$

and parameters $\Delta\theta$ are:

$$\Delta\alpha = \frac{\alpha}{d} \tag{4.24}$$

$$\Delta x = \frac{x_0}{d} \tag{4.25}$$

$$\Delta y = \frac{y_0}{d} \tag{4.26}$$

$$\Delta\gamma = \frac{\gamma - 1}{d}. \tag{4.27}$$

How these parameters are used in the creation of the unknown images is represented in Figure 4.8.

4.5 Volume registration

The relative position of the two created volumes needs to be found to align the corresponding voxels. This is done with 3D Euler registration using the SimpleITK library [21]. This registration uses the Euler transform, whose parameters are rotation around a fixed center and translation in three dimensions.

The transversal volume U is used as the fixed volume and the longitudinal volume V is used as the moving volume. The center of mass of V is then used as the center of rotation to better align the two volumes before the registration is started. The vector between the two centers of mass is then passed as the initial translation of the transform.

The found transform T can then be used to align the voxels of V with the corresponding voxels of U. The inverse transform T^{-1} to the transform T can be used align the voxels of U with the corresponding voxels of V.

The elements of the transformed volumes U_T and V_T can be computed from the elements of U and V as

$$U_T(\vec{x}) = U(T(\vec{x})) \tag{4.28}$$

$$V_T(\vec{x}) = V(T^{-1}(\vec{x})). \tag{4.29}$$

These transformed volumes are then used in the process of improving the slices' position.

4.6 Improving the estimation of slices' position

The improvement of slices' positions is made only for mutual slice distances. Mutual angles between two slices from one scanning direction will not be updated. Only the angle between the longitudinal and transversal slices will change. This mutual angle is acquired in the volume registration process, and it is given by the mutual angle of the two constructed volumes. If the mutual angles of the slices from one scanning direction were also changed, it might further improve the reconstruction quality.

The position of the i-th transversal slice u_i from the first estimation is:

$$\vec{x}_{(i)} = (i \cdot (d_t + 1), 0, 0), \tag{4.30}$$

where d_t is the number of missing slices between two known transversal slices. The position of the j-th longitudinal slice v_j before slice registration is:

$$\vec{x}_{(j)} = (0, j \cdot (d_l + 1), 0), \tag{4.31}$$

where d_l is the number of missing slices between two known longitudinal slices.

In the improvement process, the y coordinate of transversal slices will be updated, and the x coordinate of longitudinal slices will be updated. New volumes U', V' will then be constructed from slices $u_0, u_2, ..., u_{N_t-1}$ and $v_0, v_2, ..., v_{N_l-1}$, as is described in 4.4.2, but the slices' updated positions will be used. These positions are:

$$\vec{x}'_{(i)} = (x_{T_i}, 0, 0) \tag{4.32}$$

for transversal slices. And

$$\vec{x}'_{(j)} = (0, y_{T_j}, 0) \tag{4.33}$$

for longitudinal slices.

The coordinates x_{T_i} and y_{T_j} are found with registration using a translation transformation. The coordinate x_{T_i} belonging to a slice u_i is found by using a thin slice from the volume U which surrounds the slice u_i. This approach is slower than registering only the images in the volumes because it involves more elements, but the precomputed volume part, which surrounds the known image, contains more information about the shape of the area and its estimated surrounding.

The slice is chosen as to surround the updated image by a quarter of the precomputed image distance d on both sides, this creates a slice of a depth $d/2$. This thin slice is then registered in the transformed volume V_T, and the coordinate at the center of the slice in the x-axis is considered the new coordinate of the image u_i. A similar thing is done for the longitudinal slices $v_0, ..., v_{N_t-1}$, which are registered in the volume U_T, and their y coordinate is then updated.

Mutual information is used as a similarity metric. Only two iterations of slice registration are done in the proposed method because of high time requirements of registering every slice. If the number of iterations was increased, it could lead to improvement in reconstruction quality.

After all the positions are updated, new volumes are created, the coordinate update is then repeated.

4.6.1 Results of slice coordinate update

Both registration of slices in volume and volume registration were done with mutual information as the similarity measure and a gradient descent optimizer with the maximum of 20 iterations in slice registration and 30 iterations in volume registration. A learning rate of 0.1 was used. Thirty histogram bins were used for the computation of mutual information, which was computed from 2 % of randomly selected elements to reduce the time needed for computation. The registration was computed at three levels of resolution. At the first level the volume was reduced by a factor of 4 and at the second level a factor of 2. 3D Euler transformation is used in volume registration and translation is used in slice registration. The usual time for two iterations of the coordinate update is 15–20 minutes.

Table 4.3 contains mutual information of the transversal and longitudinal volumes during the reconstruction process. The first value is computed before slice registration, and the second value is computed for volumes that were constructed from slices with their updated positions. The mutual information was computed for several volumes with a different number of original transversal and longitudinal slices. The computation is done with the MattesMutualInformation metric implemented in the SimpleITK library [21]. Mutual information is greater on average for volumes created with linear interpolation. Linear interpolation was chosen in the final implementation based on these results.

The quality of artificial volumes is computed from their signal to noise ratio (SNR) defined as

$$SNR = 10\log_{10}\left(\frac{\sum_x \sum_y \sum_z O(x,y,z)^2}{\sum_x \sum_y \sum_z (O(x,y,z) - W(x,y,z))^2}\right), \tag{4.34}$$

where O is the original volume and W is its reconstruction. The artificial slices in the experiments were chosen to have different mutual distances. These sets of slices were created manually by picking slices that are approximately the same distance apart.

The SNR values are computed in Tables 4.5 and 4.6 for different numbers of used slices. The proposed method struggles to improve this value in cases with a small number of original slices in the case of linear interpolation.

slice id	**number of known slices (transversal+longitudinal)**	**MI before**	**MI after**
100027	$16+11$	0.183	0.222
105520	$16+12$	0.223	0.276
128207	$13+12$	0.192	0.224
137930	$20+9$	0.027	0.154
159463	$11+9$	0.186	0.306
160177	$11+9$	0.149	0.229
252957	$12+9$	0.149	0.287
496029	$19+11$	0.074	0.151
559832	$12+10$	0.155	0.208
587463	$12+10$	0.155	0.189
646157	$17+8$	0.336	0.347
794335	$24+10$	0.157	0.229
855886	$12+10$	0.271	0.264
955148	$9+12$	0.331	0.347

Table 4.3: Mutual information (MI) of the transversal and longitudinal volumes before and after two iterations of slice registration, using MI as the registration similarity measure and Transformation interpolation between slices. Computed for several volumes (higher value of MI is better).

slice id	number of known slices (transversal+longitudinal)	MI before	MI after
100027	$16 + 11$	0.227	0.316
105520	$16 + 12$	0.231	0.350
128207	$13 + 12$	0.249	0.268
137930	$20 + 9$	0.205	0.281
159463	$11 + 9$	0.299	0.356
160177	$11 + 9$	0.197	0.311
252957	$12 + 9$	0.278	0.283
496029	$19 + 11$	0.160	0.218
559832	$12 + 10$	0.197	0.257
587463	$12 + 10$	0.251	0.261
646157	$17 + 8$	0.375	0.350
794335	$24 + 10$	0.233	0.223
855886	$12 + 10$	0.309	0.333
955148	$9 + 12$	0.381	0.398

Table 4.4: Mutual information (MI) of the transversal and longitudinal volumes before and after two iterations of slice registration, using MI as the registration similarity measure and Linear interpolation between slices. Computed for several volumes (higher value of MI is better).

number of slices (transversal+longitudinal)	SNR before	SNR after
$14 + 13$	9.45	10.10
$14 + 11$	9.87	10.08
$13 + 10$	9.97	10.69
$10 + 9$	8.08	10.11
$9 + 8$	10.01	10.01

Table 4.5: Signal to noise ratio (SNR) computed for volumes created from a different number of artificial slices. Trasformation interpolation was used for volume creation. SNR is computed before and after two iterations of slice registraion.

number of slices (transversal+longitudinal)	SNR before	SNR after
$14 + 13$	9.93	9.95
$14 + 11$	8.52	10.47
$13 + 10$	10.27	10.37
$10 + 9$	9.01	8.69
$9 + 8$	9.36	9.01

Table 4.6: Signal to noise ratio (SNR) computed for volumes created from a different number of artificial slices. Linear interpolation was used for volume creation. SNR is computed before and after two iterations of slice registraion.

4.7 Fusing two volumes into one

The resulting volume is computed from the transversal and longitudinal volumes by summing the elements at the corresponding coordinates. The weight of a resulting element $W(x, y, z)$ is determined by its inverse distance to the nearest known slice and is computed from U and V as

$$W(x, y, z) = \frac{w_U \cdot U(x, y, z) + w_V \cdot V(x, y, z)}{w_U + w_V}, \tag{4.35}$$

where $U(x, y, z)$ and $V(x, y, z)$ were acquired with interpolation, and the weights w are computed as

$$w_U = \frac{1}{d((x, y, z), (x_{Us}, y_{Us}, z_{Us}))}, \tag{4.36}$$

$$w_V = \frac{1}{d((x, y, z), (x_{Vs}, y_{Vs}, z_{Vs}))}, \tag{4.37}$$

where $d((x, y, z), (x_{Us}, y_{Us}, z_{Us}))$ and $d((x, y, z), (x_{Vs}, y_{Vs}, z_{Vs})$ are distances from the nearest known original tranverse and longitudinal slices. This distance is computed as the number of voxels separating the currently computed voxel and the nearest known slice.

The actual computation of the final volume is done by separately computing the transversal volume weights and the longitudinal weights. Because the transform between these two volumes is known, the weights can then be applied at the correct coordinates.

4.8 Results

Linear interpolation is used in the final implementation because it gives better volume reconstruction results. The result of running the script, which contains the implementation of the proposed method, is shown in Figure 4.9. It is a window containing a created 3D isosurface on the left side and three orthogonal slices of the volume on the right side. The value at which the surface is shown can be changed by a slider at the top of the window. The slices on the right side can be moved to a different position by dragging them with the mouse cursor. Both the volume and the slices can be rotated to view the result from any direction.

Some examples of the generated volumes are shown in Figures 4.10, 4.11, 4.12, 4.13, and 4.14.

Reconstructed artificial volumes are shown in Figures 4.15, 4.16, and 4.17.

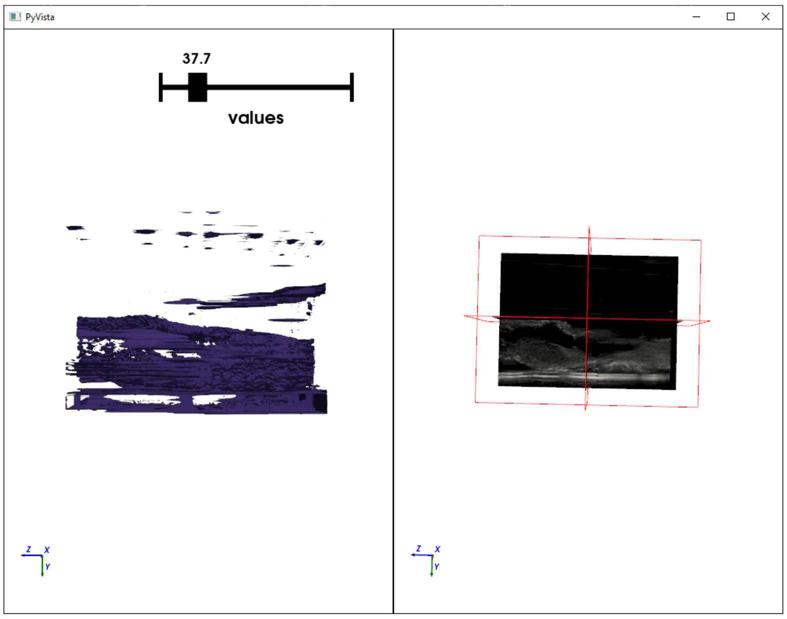


Figure 4.9: The output of the script which implements the proposed method.

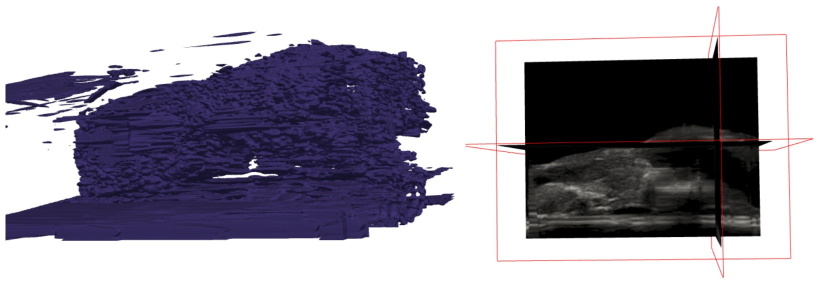


Figure 4.10: An example of a reconstructed volume and its cross section.

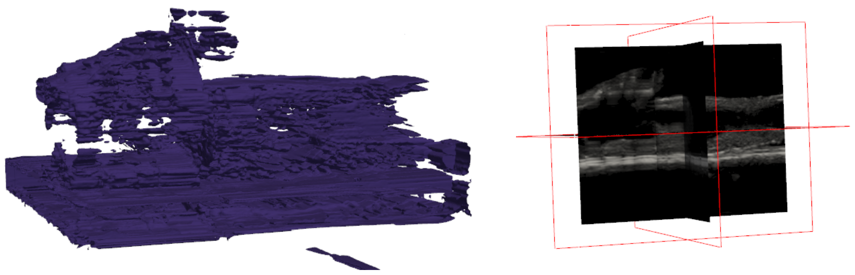


Figure 4.11: An example of a reconstructed volume and its cross section.

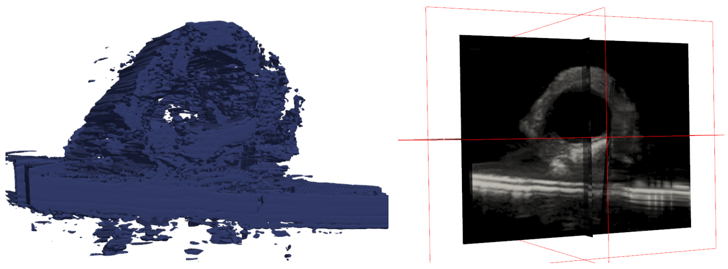


Figure 4.12: An example of a reconstructed volume.

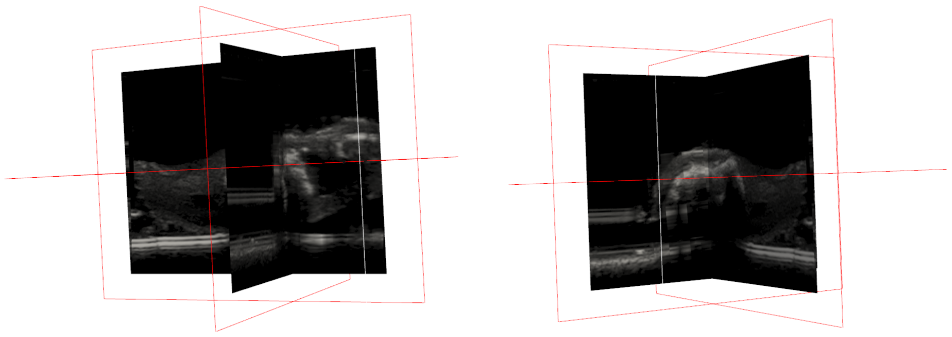


Figure 4.13: An example of slices taken from a reconstructed volume.

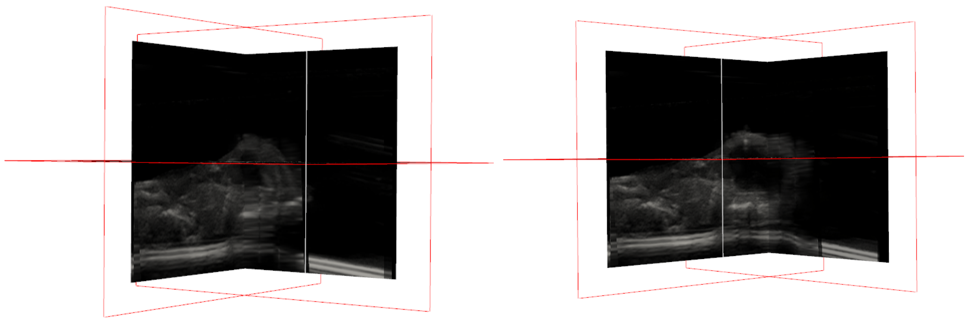


Figure 4.14: An example of slices taken from a reconstructed volume.

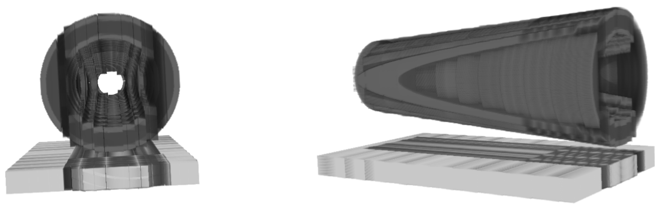


Figure 4.15: Artificial volume recreated from 12 transversal and 9 longitudinal slices(using transformation interpolation).

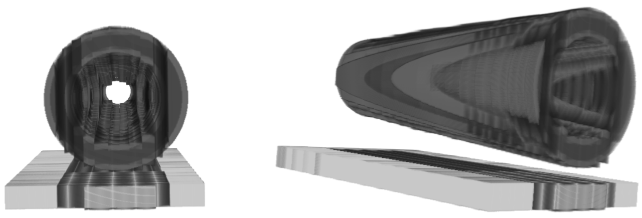


Figure 4.16: Artificial volume recreated from 13 transversal and 11 longitudinal slices(using linear interpolation).

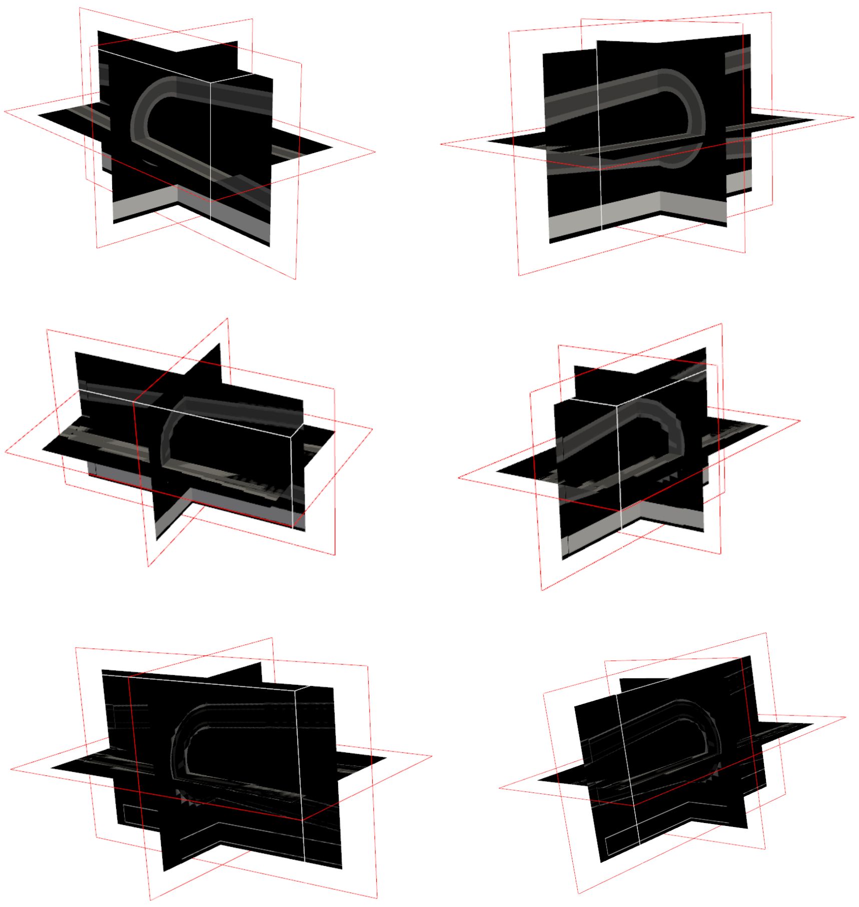


Figure 4.17: Reconstructed artificial volume slices. The first row shows the original volume, the second row the reconstruction and the third row their absolute difference. The artificial volume was created from 12 transversal and 9 longitudinal slices (using transformation interpolation).

Chapter 5
Conclusions

The aim of this thesis was to create a 3D volume from 2D ultrasound images of the carotid artery. There is no information about the position of the images, so a method of estimating the slices' position in 3D space had to be implemented before the volume reconstruction itself. The process of determining the slices' positions was made easier by the fact that the studied object, the carotid artery, was scanned in two directions. Both sets of slices were first used to create two separate volumes, and with the help of registration, a final volume was computed. The resulting visualization shows how the transversal and longitudinal slices intersect each other and gives an estimate of the original volume's shape.

Some assumptions were made during the reconstruction process because they simplified the implemented methods and reduced the computation times. The volume reconstruction could be made more accurate if these simplifications were omitted. For example, only one coordinate was updated in the computation of the slices' positions. The mutual angles of slices scanned in one direction also were not updated, only the mutual angles between the transversal and longitudinal slice sets were changed. Some improvements could be made if the unchanged parameters were changed in the reconstruction process. More iterations of slice registration should also improve the reconstruction quality.

Some of the input datasets include too few slices (below 20 in total) or contain images with many artifacts. Unfortunately, the proposed method cannot reliably reconstruct a volume from these images.

Bibliography

[1] A. Waugh and A. Grant, *Ross and Wilson: Anatomy and Physiology in Health and Illness Volume 1*. Churchill Livingstone, 9 ed., 2001.

[2] A. V. Kamenskiy, I. I. Pipinos, J. S. Carson, J. N. MacTaggart, and B. T. Baxter, “Age and disease-related geometric and structural remodeling of the carotid artery,” *Journal of Vascular Surgery*, vol. 62, no. 6, pp. 1521–1528, 2015. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0741521414019697`.

[3] A. Carovac, F. Smajlovic, and D. Junuzovic, “Application of ultrasound in medicine,” *Acta Informatica Medica*, 2011. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3564184/`.

[4] F. Mohamed and C. V. Siang, “A Survey on 3D Ultrasound Reconstruction Techniques,” *Artificial Intelligence - Applications in Medicine and Biology*, 2018. [Online]. Available: `https://www.intechopen.com/chapters/63949`.

[5] M. H. Mozaffari and W.-S. Lee, “Freehand 3-D Ultrasound Imaging: A Systematic Review,” *Ultrasound in Medicine and Biology*, vol. 43, no. 10, pp. 2099–2124, 2017. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0301562917302776`.

[6] E. Light, S. Mukundan, P. Wolf, and S. Smith, “Real-Time 3D Intracranial Ultrasound,” May 2022. [Online]. Available: `https://www.researchgate.net/publication/265182285_Real-Time_3D_Intracranial_Ultrasound`.

[7] R.-F. Chang, W.-J. Wu, D.-R. Chen, W.-M. Chen, W. Shu, J.-H. Lee, and L.-B. Jeng, “3-D US frame positioning using speckle decorrelation and image registration,” *Ultrasound in Medicine and Biology*, vol. 29, no. 6, pp. 801–812, 2003. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S030156290300036X`.

[8] R. Prevost, M. Salehi, J. Sprung, A. Ladikos, R. Bauer, and W. Wein, “Deep learning for sensorless 3D freehand ultrasound imaging,” in *Medical*

Image Computing and Computer-Assisted Intervention - MICCAI 2017 (M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne, eds.), (Cham), pp. 628–636, Springer International Publishing, 2017.

[9] R. Rohling, A. Gee, and L. Berman, "A comparison of freehand three-dimensional ultrasound reconstruction techniques," *Medical Image Analysis*, vol. 3, no. 4, pp. 339–359, 1999. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1361841599800280`.

[10] D. E. O. Dewi, M. H. F. Wilkinson, T. L. R. Mengko, I. K. E. Purnama, P. M. A. van Ooijen, A. G. Veldhuizen, N. M. Maurits, and G. J. Verkerke, "3D ultrasound reconstruction of spinal images using an improved Olympic Hole-filling method," in *International Conference on Instrumentation, Communication, Information Technology, and Biomedical Engineering 2009*, pp. 1–5, 2009. [Online]. Available: `https://www.researchgate.net/publication/228749622_3D_ultrasound_reconstruction_of_spinal_images_using_an_improved_Olympic_Hole-filling_method`.

[11] O. V. Solberg, F. Lindseth, H. Torp, R. E. Blake, and T. A. Nagelhus Hernes, "Freehand 3D ultrasound reconstruction algorithms–a review," *Ultrasound in Medicine and Biology*, 2007. [Online]. Available: `https://www.academia.edu/22595898/Freehand_3D_Ultrasound_Reconstruction_Algorithms_A_Review`.

[12] T. Wen, Q. Zhu, W. Qin, L. Li, F. Yang, Y. Xie, and J. Gu, "An accurate and effective FMM-based approach for freehand 3D ultrasound reconstruction," *Biomedical Signal Processing and Control*, vol. 8, no. 6, pp. 645–656, 2013. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1746809413000839`.

[13] J. W. Trobaugh, D. J. Trobaugh, and W. D. Richard, "Three-dimensional imaging with stereotactic ultrasonography," *Computerized Medical Imaging and Graphics*, vol. 18, no. 5, pp. 315–323, 1994. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/0895611194900027`.

[14] A. A. Goshtasby, *Image Registration: Principles, Tools and Methods.* 2012.

[15] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0262885603001379`.

[16] D. L. H. Joseph V. Hajnal, *Medical Image Registration.* CRC Press, 2019.

[17] *IMAQ Vision: IMAQ Vision Concepts Manual.* National Instruments Corporation, 2003. [Online]. Available: `https://www.ni.com/pdf/manuals/322916a.pdf`.

[18] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: `https://arxiv.org/abs/1609.04747`.

[19] L. Ibanez, W. Schroeder, L. Ng, J. Cates, and I. Consortium, "The ITK Software Guide, Second Edition, Updated for ITK version 2.4," November 2005.

[20] W. M. Wells, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis, "Multi-modal volume registration by maximization of mutual information," *Medical Image Analysis*, vol. 1, no. 1, pp. 35–51, 1996. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1361841501800049`.

[21] B. Lowekamp, D. Chen, L. Ibanez, and D. Blezek, "The Design of SimpleITK," *Frontiers in Neuroinformatics*, vol. 7, 2013. [Online]. Available: `https://www.frontiersin.org/article/10.3389/fninf.2013.00045`.

[22] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009.

[23] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. [Online]. Available: `https://rdcu.be/b08Wh`.

[24] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020. [Online]. Available: `https://doi.org/10.1038/s41586-020-2649-2`.

[25] C. B. Sullivan and A. Kaszynski, "PyVista: 3D plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK)," *Journal of Open Source Software*, vol. 4, p. 1450, May 2019. [Online]. Available: `https://doi.org/10.21105/joss.01450`.

[26] A. Clark, "Pillow (PIL fork) Documentation," 2015.

[27] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

Appendix A

Implementation details

The proposed method was implemented in Python 3.9 [22]. The implementation can be found on GitLab[1]. The GitLab repository contains the following directories:

- **data**: contains a few sets of slices, which can be used to create a 3D volume
- **image_manipulation**: contains functions that work with 2D images
- **registration**: contains registration algorithms
- **volume_manipulation**: contains functions that work with 3D arrays

Some data is included in the GitLab project, but more in-vitro images can be found on Google Drive[2], or downloaded using an account in the CMP Unix network[3].

The libraries used in the project can be found in table A.1.

Library	Usage
SimpleITK [21]	registration, transformations
scipy [23]	some computations
numpy [24]	manipulation of arrays
PyVista [25]	visualisation of volumes
Pillow [26], OpenCV [27]	image manipulation

Table A.1: Libraries used in the project.

[1] `https://gitlab.fel.cvut.cz/kvasoma1/bakalarska_prace`

[2] `https://drive.google.com/drive/folders/1Vh4dlS3hPiq-GXgxg1VDJ5Vw3fit7xhR?-usp=sharing`

[3] the images are located in the /datagrid/Medical/ArteryPlaque/2018_10_1_hist_us directory

The implemented algorithm can be run by the script **run.py**. The runtime is around 20–30 minutes, depending on the number of images. Detailed information about running the script can be found in the **README.md** file.