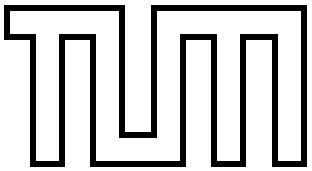


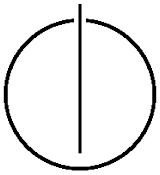
TECHNICAL UNIVERSITY OF MUNICH

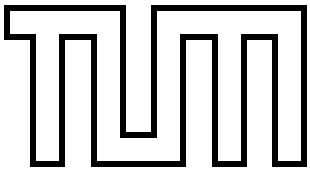
DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Modified Iterative Closest Point Algorithm Design for Needle Pose and Position Estimation using Optical Coherence Tomography Images

Ramona Schneider





TECHNICAL UNIVERSITY OF MUNICH

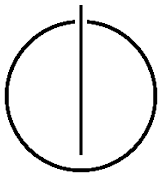
DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Modified Iterative Closest Point Algorithm Design for Needle Pose and Position Estimation using Optical Coherence Tomography Images

Modifizierter Iterative Closest Point Algorithmus für Annäherung der Nadel Rotation und Position unter Verwendung von Optical Coherence Tomography Bildern

Author:	Ramona Schneider
Supervisor:	Prof. Dr.-Ing. habil. Alois Knoll
Advisor:	Mingchuan Zhou, M.Eng.
Submission date:	Ocotber 15, 2017



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, October 13, 2017 Ramona Schneider

Acknowledgments

First and foremost, I would like to thank my supervisor Prof. Dr.-Ing. habil. Alois Knoll for giving me the opportunity to pursue my master's thesis at his chair.
Second, I wish to thank my advisor Mingchuan Zhou for his help and advice during the course of the thesis, for his continuous feedback and valuable input when I was stuck.
Finally, I would like to thank my family for their constant support throughout my studies.

Abstract

Eye surgeries, where for example drugs have to be injected into vessels in the eye, are currently performed by human surgeons. If the pose for the injected needle isn't optimal or if there are tremors in the hands of the surgeon, this can easily lead to damage on the eye because very high precision is required in this kind of surgeries. Using a robot to assist in these surgeries with automatic navigation of the needle could help to reduce the risks of a bad injection pose or damage due to tremors. But before a robot can be used, an robust algorithm has to be found with which the position and rotation of the needle can be calculated as accurate as possible in real-time. But this task can be very difficult.
The goal of this thesis is to find an approach for computing a 6DOF pose of a beveled needle used in ophthalmic surgery by matching point clouds. The input for the algorithm are images captured with optical coherence tomography where a part of the needle is visible. The needle is then reconstructed as a point cloud in 3D space and the pose is computed by matching this point cloud with a point cloud generated from a CAD model of the needle.
For computing the pose, two approaches have been implemented. The first one uses Clustered Viewpoint Feature Histograms and Camera Roll Histograms to compute a 6DOF pose that is then refined by Iterative Closest Point. This approach didn't yield any usable results in this thesis.
The second approach uses additional information about the needle, which is the needle direction and the x- and y-rotation computed with the help of the OCT images. Then only two parameters remain (the z-rotation and the shift along the needle direction) which have to be computed. This is done by defining starting intervals for them and then trying several combinations of values inside this intervals. How good the current combination is, can be determined by the number of matched points in the clouds when the current parameter combination is used for the transformation of the CAD model. The results of this algorithm were better than with the first approach. The angle could be computed for all used datasets within 10° deviation, for most datasets even within 5° deviation. The accuracy for the position estimation could be controlled within $15\mu m$.

Contents

1 Introduction

Eye surgeries, where for example drugs have to be injected into vessels in the eye, are currently performed by human surgeons. If the pose for the injected needle isn't optimal or if there are tremors in the hands of the surgeon, this can easily lead to damage on the eye because very high precision is required in this kind of surgeries. Using a robot to assist in these surgeries with automatic navigation of the needle could help to reduce the risks of a bad injection pose or damage due to tremors. But before a robot can be used, an robust algorithm has to be found with which the position and rotation of the needle can be calculated as accurate as possible in real-time. But this task can be very difficult, because for example not the whole needle is visible on the OCT images, but only the upper surface, so only few information is available to reconstruct the rotation and position of the needle. Also, the images can contain noise or deformations, so the algorithm used has to be very robust against these issues.

1.1 Objectives

The general objective of this thesis is to find a method to compute the position and rotation of a beveled needle that can be reconstructed as point cloud from images obtained through optical coherence tomography by matching the resulting point cloud with a given CAD model.

The results for the position should be as stable as possible so that the accuracy can be controlled in a range as small as possible.
The results for the angle, especially for the rotation around the angle axis should be as accurate as possible as well.
Two approaches should be implemented:

- A well developed approach for object recognition and pose estimation for rigid objects described in a paper by Aldoma et al. [5]. They use Clustered Viewpoint Feature Histograms (CVFH) and cameras roll histograms together in a recognition framework to recognize objects and compute their 6DOF pose.
- A new approach which is similar to ICP but uses the direction of the needle as information and finds the best transformation from a lot of computed possibilities which are found by using different combinations of angles and position shifts along the direction of the needle.

1.2 Related Work

A lot of approaches in retinal surgery that track the needle position use the images provided by a microscope, but since these are only 2D images, the full pose of the needle can't be reconstructed.
For example, Rieke et al. [14] use regression forests to get the positions of tool tips and the center point of forceps in real-time. The algorithm can also handle incomplete and noisy data. The tracking algorithm finds a bounding box around the tool tip by estimating a relation between the instrument motion and changes induced in the image intensities. Points of interest can then be found in this region by evaluating a learned mapping between image patches and the articulated pose.
Richa et al. [13] propose an algorithm for detecting unintentional collisions between tools and retina with the visual feedback of an ophthalmic stereo microscope. First, the instrument is tracked on both stereo images using a direct visual-tracking method based on a robust similarity metric. This method has two stages, so that a more accurate estimation of the tool tip can be obtained: First, gradient-based tracking is performed, where rotation and vertical translation are estimated with Efficient Second-Order Minimization. In the second stage, a brute force search is carried out to get the tool tip by a search along the shaft of the instrument in discrete steps. For the proximity detection, a stereo disparity map of the retinal surface is used.
Sznitman et al. [16] use a detection based scheme to track a 2D instrument tip. A gradient-based tracker is used to approximately estimate the target's new location. A detector is evaluated to predict the presence of an instrument in a reduced region of an image which is parameterized by the tracker's estimate. The detector is also used for initialization or reinitialization, so that no user input is necessary. The accurate instrument position is then found by spatial and score weighting of the detector responses. As last step, the tracker template is updated.
Li et al. [10] also use a gradient-based tracker which is capable of handling unexpected appearance changes. For instrument detection, a cascade appearance classifier is used. For initialization, the user has to click on the instrument position in the first frame. Additionally, a filtering step is applied to get a training set that augments the model. In the next step the output of the tracker and the detector are integrated into a unique target position, which yields more reliable tracking results. Finally, image patches for online updating are selected that are used for updating the appearance model of the detector.

6DOF pose reconstruction of needles in surgical interventions has also been tried with other imaging technologies than Optical Coherence Tomography, like Ultrasound or X-Rays.
For example, Chatelain et al. [6] localize and track manually inserted needles in real-time using a 3D ultrasound probe on a robotized arm. They don't need any a priori information, instead image differences are computed for initialization from which the needle can be detected. When it is visible with a certain length, it is localized with a RANSAC-based

method which fits polynomial curves to a candidate set of voxels found with a search in a volume of interest, which is predicted by a Kalman filter. Using the Kalman filter speeds up the whole algorithm and it is also used to discard unlikely configurations found during RANSAC.
Papalazarou et al. [11] estimate the 3D position and rotation of a needle during a procedure, only with the help of a simple model of the instrument and small C-arm motion. First, points that belong to the needle are detected in 2D by creating ridgeness images and choosing the maxima in scale-space as possible needle points. Next, with model fitting candidate needles are computed. For this, sparse feature points (the ridgeness maxima) and dense feature representations (label images) are used. The model consists of parameters of a polynomial curve. From the curve equations and the ridgeness maxima the candidate needles are selected and the endpoints of them are computed. Then, some constraints are applied on the endpoints which leads to a limited area where to search for them in the next frame. Candidate matches in the next frame are then selected with a cost function. As last step, the needle is reconstructed in a pointwise manner, by sampling points along the length of the curve segment given by the endpoint detection result and the needle equation. The points are then reconstructed with an iterative intersection/resection method.
Qiu et al. [12] segment needles from 3D ultrasound images by extending 2D Hough transform to 3D. The method uses 3D Randomized Hugh Transform and a coarse-fine searching strategy. 3D ultrasound images are first converted to cropped 3D ultrasound images. Next, the data is binarized and 3D Randomized Hugh Transform is used at the coarse stage to get an approximation of the needle orientation and position. At the fine stage, the approximations are used to get a limited area where the needle has to be searched.

Iterative Closest Point, as described by Zhengyou Zhang [18], is a method often used for aligning two point clouds. It needs two points clouds, as well as an initial guess, as input and outputs the rotation and translation which is required minimize the distance between the two clouds. It is explained in 2.2.5 in more detail.

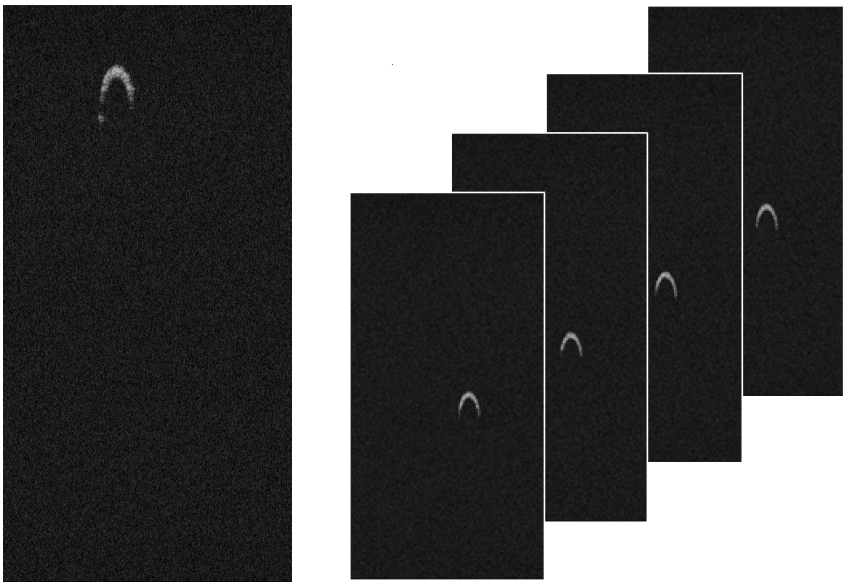
Aldoma et al. [5] propose a recognition framework for identifying and estimating 6DOF poses of CAD models in real scenes by using a new feature descriptor called 'Clustered Viewpoint Feature Histogram' and cameras roll histograms, which can handle partial occlusions and noise very well. This approach is explained in more depth in 2.2.

A method with the same goal as in this thesis has been introduced by Zhou et al. [19], who propose a framework for estimating the needle position and rotation with the help of Optical Coherence Tomography (OCT) and geometric features of the needle. This is done by reconstructing the needle from OCT images by segmenting points that belong to the needle in every frame of the OCT cube. A bounding box around these points is then projected on the x-y-plane. The leftmost and the rightmost points obtained from these projections then form two lines. Since the width of the needle increases until the body is reached, it is calculated with linear regression where the bounding box width stops grow-

ing. Then, the center line of the needle is calculated, which is the middle between the two lines formed by the outer points of the bounding boxes. Next, it is tried to fit an ellipse that best resembles the needle tip by solving a nonlinear least-squares problem. Then, the needle direction obtained through the peak points in all frames is mapped to 3D space, from which the center line of the needle can be calculated with the help of the needle diameter. The center point of the ellipse is then projected on this axis and the rotation angle of the needle is obtained from the already calculated parameters. As a last step, the whole needle transformation is computed. The results show, that the needle angle could be matched very well, and the position could be controlled within $10\mu m$ in all directions.

1.3 Optical Coherence Tomography

Optical coherence tomography is an imaging technique which is for example used to map and measure the thickness of the retina's different layers. It works similar to ultrasound or MRI, but uses light waves instead of sound or radiation to acquire images of the retina or other tissue in situ and in real time. The amount of back-scattered light is measured to image the internal micro-structure in biologic systems. Additionally, OCT can have a resolution in μm range, which is a lot higher than the resolution of MRI which is in the range of millimeters. If the images are taken during surgery, parts of the used instruments are visible. [17], [7]
The high resolution makes OCT very suitable for reconstructing the position or rotation of a needle that shows up on those images, because the instruments used in ophthalmic surgery have a very small diameter. For example, the beveled needle used in this thesis has a diameter of $0.31mm$.
The data generated by OCT are two-dimensional grayscale images that form a cube if put after one another. Each OCT cube used in this thesis consists of 128 images.
Images of the retina can for example be used to diagnose diseases like age-related macular degeneration or diabetic eye disease. [17], [7]
Examples for OCT images can be seen in Figure 1.1.



(a) Single OCT image. (b) Part of an OCT cube.

Figure 1.1: OCT images.

1.4 Outline

Chapter 2 describes the approaches implemented during the course of this thesis. First, the generation of point clouds used for matching is explained. Then, an overview over the first algorithm is given, which uses Clustered Viewpoint Feature Histograms, Camera Roll Histograms and Iterative Closest Point.
Also, the second algorithm is described more closely, as well as the experiments carried out to get results as good as possible. The experiments included for example using different models, different thresholds or different intervals for the searched parameters.

In chapter 3 the results of the second algorithm are explained more closely, by comparing different experiment results for the angle and evaluating the accuracy of the computed position.

The 4^{th} chapter summarizes the results of this thesis and chapter 5 provides an overview of some open questions that could be evaluated in the future.

2 Approach

2.1 Generation of Point Clouds

2.1.1 Generation of OCT Point Cloud

Different datasets captured with OCT were used for testing of the algorithms. Only datasets that showed only the needle and no part of the retina were used. Due to the mechanism of OCT only the upper half of the needle surface can be seen on the images. They differed in the rotation (especially the z-rotation) and the position of the needle. A microscopic view of the needle in different rotations can be seen in Figure 2.2, to make the difference in the rotation clearer. Also, some datasets contained more deformations than others which can happen due to reflections in OCT.
The needle in these datasets is a beveled needle, which can be used to inject drugs to parts of the eyes. The datasets were assembled from a needle with a bevel angle of 15°. Since the needles are manufactured very well, it can be assumed reliably that they always have a diameter of $0.31mm$. It can also be assumed that the body part of the needle (after the tip ends) is a cylinder.
To assemble an OCT cloud, the individual grayscale images are processed by applying a threshold of 0.26 to them to filter noise. Additionally, a median blur filter is applied. Afterwards, the images are labeled and for all labels a bounding box is computed, as can be seen in Figure 2.1. If the bounding box area is greater than 250 the label is considered a part of the needle.
For every label, the point with the lowest y-value in every column is added to the final point cloud so that it only consists of the outer shell. The different frames are assembled in z-direction, so that the needle is approximately aligned to the z-axis. To provide a better understanding the coordinate system is shown in Figure 2.3.
At first a scale of *(3.0, 2.0, 2.6)* has been used, but this this led to scale mismatches with the CAD model. For example, in Figure 2.4 the width in x-direction of the CAD model is smaller than the width of the OCT cloud. Using a scale of *(2.7, 2.4, 3.0)* led to better fitting results.

2.1.2 Generation of CAD Model Point Cloud

To generate a point cloud of the CAD model that can be used for matching, a model in polygon file format (.ply) is loaded with the point cloud library. Since only few points are necessary to describe the needle, more points have to be sampled on the surface as a

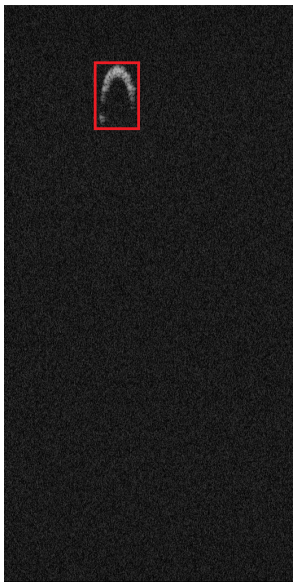
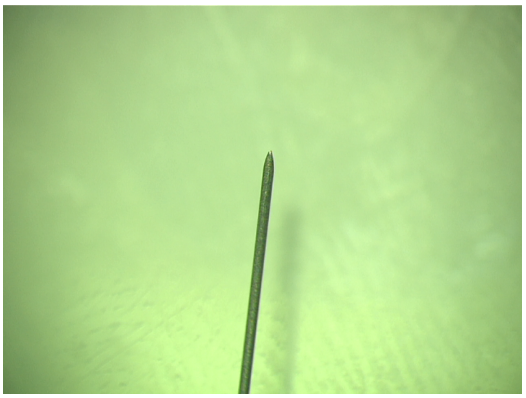
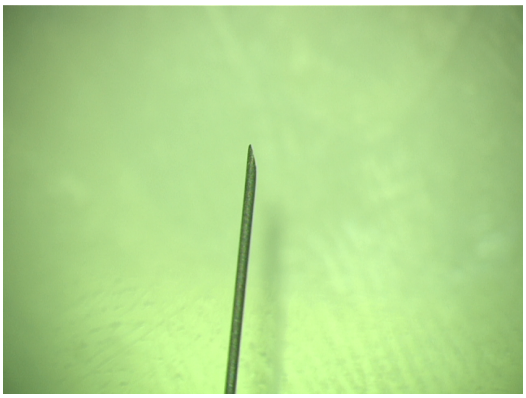


Figure 2.1: OCT image with bounding box around the visible needle part.

second step. The resolution of the model should later match the resolution of the OCT cloud, so maybe an additional filtering step is required.



(a) Microscopic view of a needle with rotation 0°.



(b) Microscopic view of a needle with rotation -45°.

Figure 2.2: Microscopic views of the needle.

2.2 Recognition Framework

For the first approach for matching the CAD model with the OCT cloud, the implementation of the recognition framework developed by Aldoma et al. [5], which can be found in point cloud library as the 3d_rec_framework[1], has been adapted so that it can be used for the tasks in this thesis.
The recognition pipeline has the following steps:

- train the CAD model
- compute Clustered Viewpoint Feature Histogram and camera roll histogram for the OCT cloud
- find the nearest neighbors of the OCT cloud in the CAD model views for every stable region
- select the n best candidates according to a metric
- compute the roll angle for those candidates
- refinement with Iterative Closest Point
- sort according to the inlier number from ICP

Some of these steps are described more closely in the following sections.

[1] https://github.com/PointCloudLibrary/pcl/tree/master/apps/3d_rec_framework

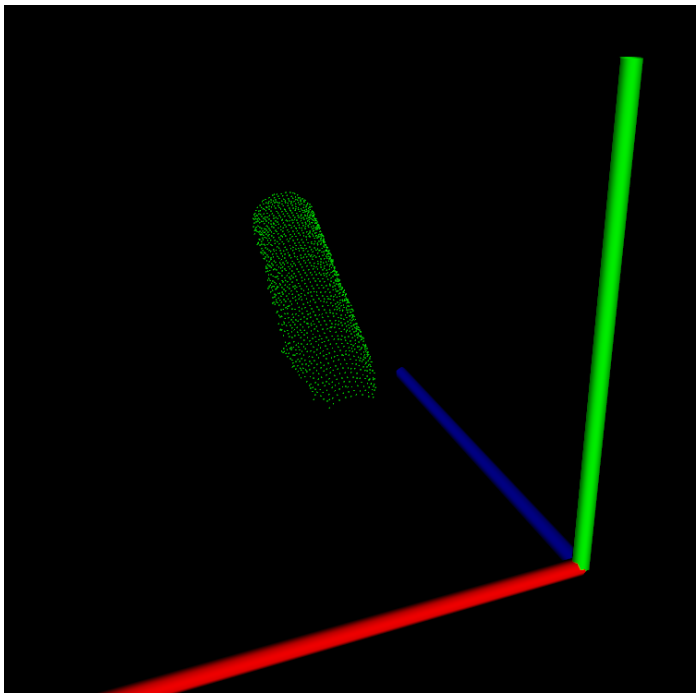


Figure 2.3: View of the OCT cloud in coordinate system (red=x-axis, green=y-axis, blue=z-axis)

2.2.1 Training of Model

As first step in the recognition pipeline, the CAD model has to be trained. This means, that different views have to be generated and the descriptors have to be computed for each of them before the main algorithm starts. In the original algorithm, a tessellated sphere is generated around the model and a virtual camera renders 80 different objects into a depth buffer, from which partial point clouds can be generated because the model is approximately aligned to the z-axis. [5]
For the needle matching, only the rotation around the z-axis is relevant, so instead of a tessellated sphere only a circle has been used to acquire 96 views that show different rotations of the CAD model around the z-axis.
For each of the views, the Clustered Viewpoint Feature Histogram and the camera roll histogram described in the next sections is computed.

2.2.2 Clustered Viewpoint Feature Histogram

For all of the views acquired in the training step, the Clustered Viewpoint Feature Histogram (CVFH) has to be computed, as well as for the OCT cloud. The CVFH is a feature descriptor developed in [5], which is closely related to the Viewpoint Feature Histogram

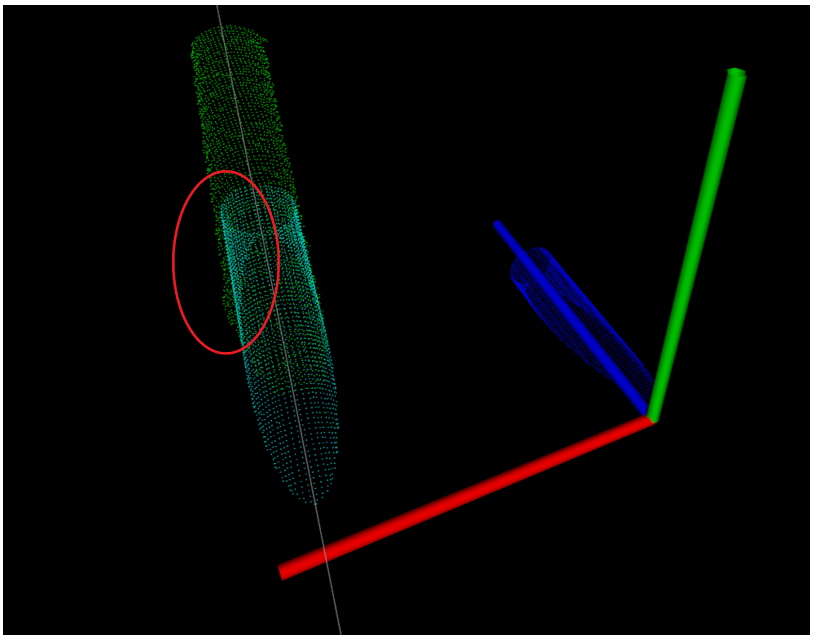


Figure 2.4: Scale mismatch of clouds, OCT cloud (green) has larger width than CAD model (turquoise).

(VFH), but is less sensitive to noise and occlusions. The normal VFH can be easily affected by this because the centroid of the object is used for the VFH computation, and when many points are missing on an object, the centroid is different to what it should be to be of use. [15], [5]
To overcome this, for CVFH not a single histogram for the complete object is computed, but one for every stable region that is found on the object. This leads to the advantage, that if one of these regions is fully visible in the scene, the object can be identified well. As an additional difference to VFH, the histogram shouldn't be normalized, so that objects of different scale aren't identified as the same object, which makes CVFH more robust to occlusion. [15],[5]
The VFH consists of four histograms for the normal angular deviations, three of them with 45 bins and one with 128 bins. For the CVFH a fifth histogram is added for the shape distribution component, which should help to differentiate surfaces with similar size and normal distribution, but different point distributions. It also consists of 45 bins, which results in a total size of 308 dimensions for the CVFH. [15], [5]
How the CVFHs for different views of the model and the OCT cloud look can be seen in Figure 2.5 and Figure 2.6.
For the needle object, only one stable region is found, which is the whole cloud.

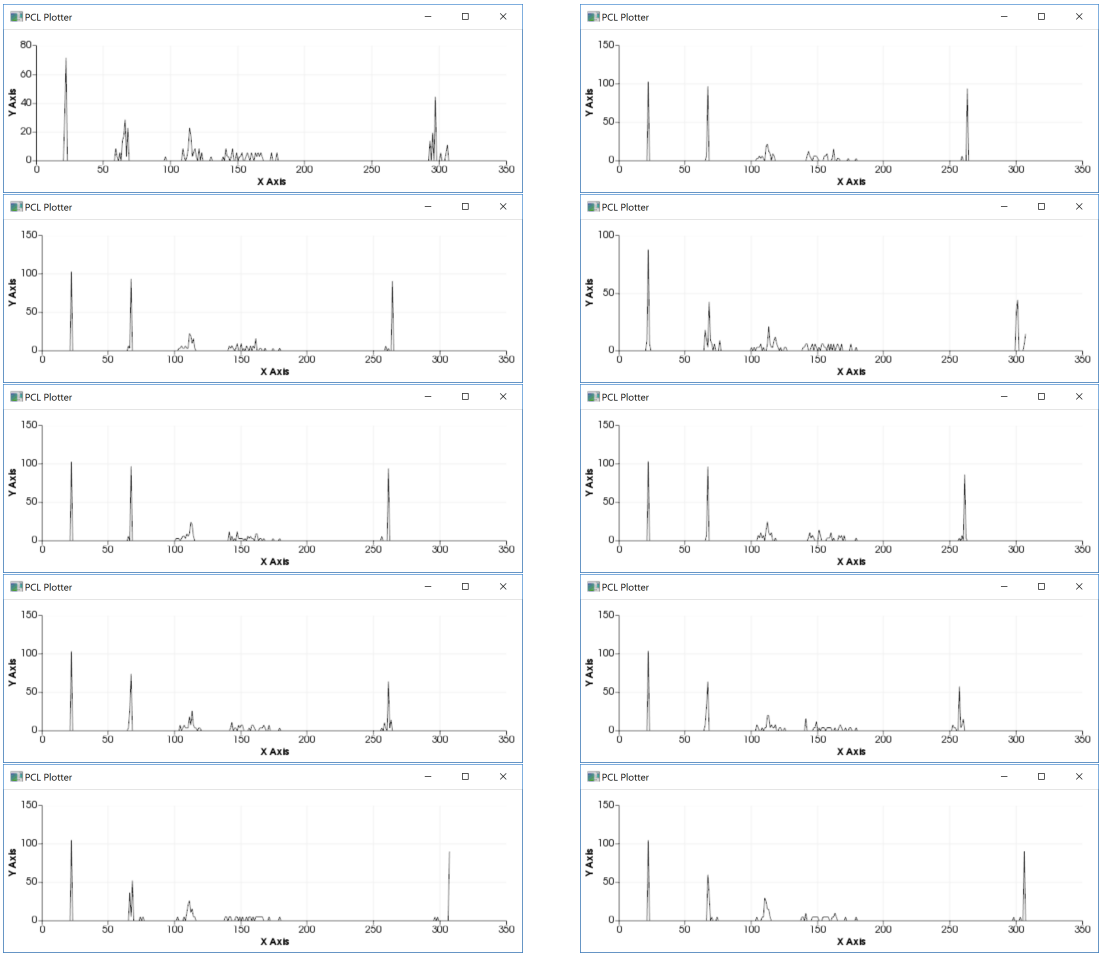


Figure 2.5: CVFHs of different views of the CAD model.

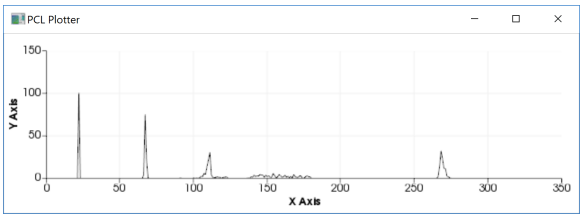


Figure 2.6: The CVFH of an OCT cloud

2.2.3 Camera Roll Histogram

Since CVFH is invariant to rotations around the the view axis of the camera, an additional step has to be performed to get a 6DOF pose in the end. For this, another descriptor called "camera's roll histogram" (CRH) is used in the algorithm in [5], which has to be computed for every stable region found with a CVFH. The histogram consists of 90 bins, one for every 4 degrees. The input for the histogram is computed by projecting each normal of the object onto a plane orthogonal to a line given by the camera and the centroid of a stable region. The angle of the projected normal to the up-vector of the camera is computed and then added to the histogram. Additionally, the projected normals are weighted by their magnitudes to reduce influence by noise. [15], [5]
An example from [5] how CRHs can look like can be seen in Figure 2.7.
The histograms for different views of the CAD model cloud can be seen in Figure 2.8, the CRH for an OCT cloud can be seen in 2.9.

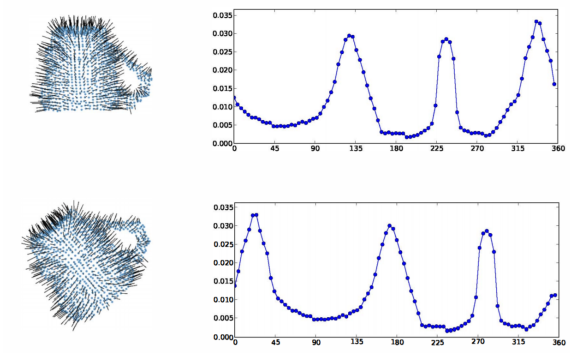


Figure 2.7: The camera roll histograms of the same object in different orientations. [5]

2.2.4 Matching

After the n best matching views in the training set are identified by a nearest neighbor search on the CVFHs, the CRHs are computed for those. To find the best roll angle, amongst other things a Discrete Fourier Transform is applied on the histograms to get

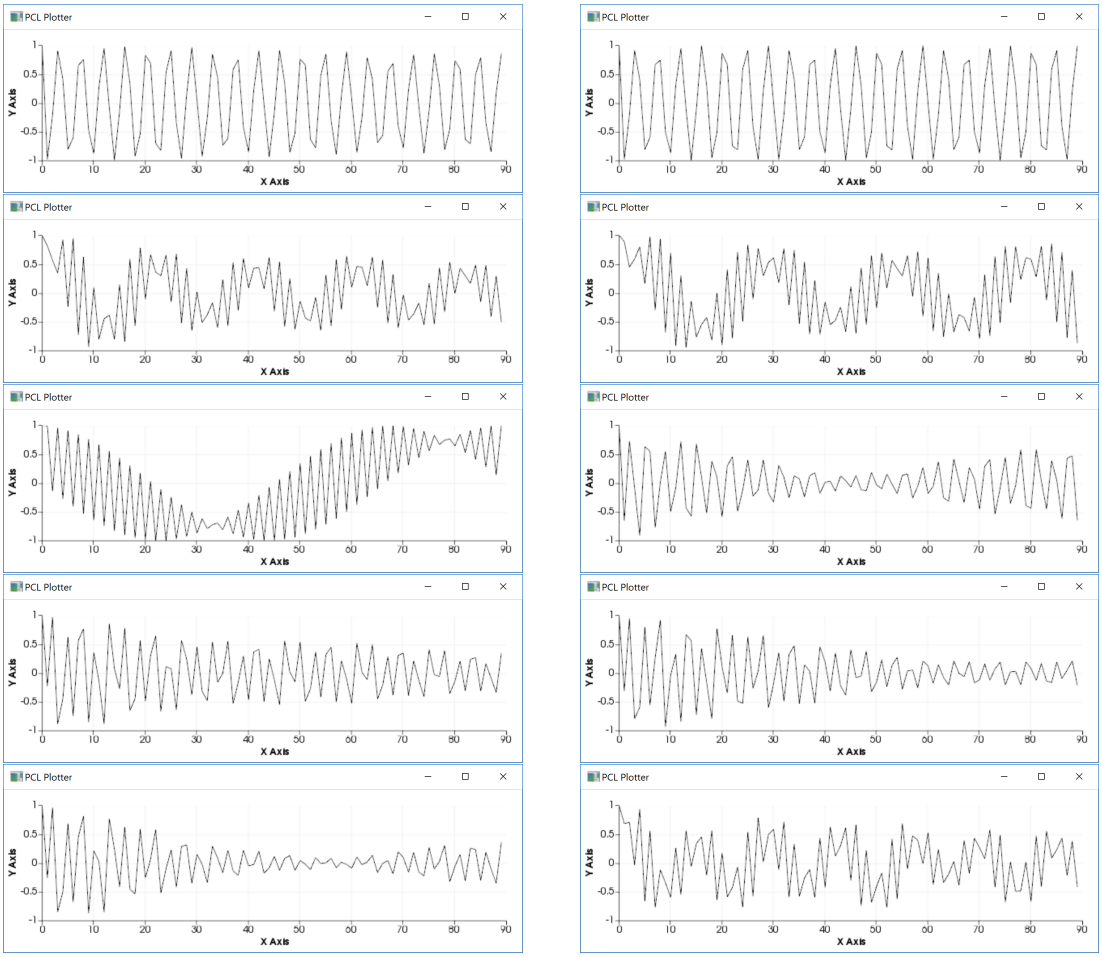


Figure 2.8: CRHs of different views of the CAD model.

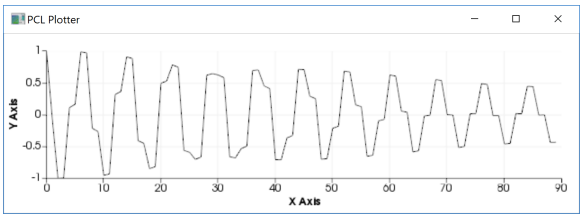


Figure 2.9: The CRH of an OCT cloud

a cross power spectrum whose peaks show the best angle candidates. The peaks are then postprocessed so that it is relied not only on the maximum peak, but very similar alignments are removed and local maxima are found. [5]
As can be seen in Figure 2.8 and Figure 2.9, the CRHs of the CAD model and the OCT cloud look very different, which makes finding a good match very hard.

2.2.5 Iterative Closest Point

Iterative Closest Point (ICP) is an algorithm which can be used to match point clouds by computing the best possible transformation which minimizes the distance between the point clouds. An initial guess for the transformation is needed, which is here given by the result of the CRH alignment.
Then, for every point in one cloud the closest point in the other cloud is determined. Next, the squared distances between those points are computed, whose sum gives the measure for how good the current transformation is. For finding a better transformation the algorithm tries to minimize the sum of squared distances by changing the transformation parameters. The new parameters are then used as input transformation for the next iteration. The algorithm iterates until a convergence criterion is hit, which can be a threshold for the squared distances sum, a maximum number of iterations or when "the transformation can't be further updated (the difference between current and previous is smaller than a threshold)" [3]. [18]
Using a high number of maximum iterations (100 or even more) and few points for the OCT cloud than would be available and also the same resolution for the CAD model improved the results so that for a few datasets the CAD model could be coarsely aligned, but this didn't work for all of the datasets.
For the problem in this thesis, it is not sufficient to just find any good match. Instead, it is important that the model is rotated in the correct direction (as didn't always work with ICP, as you can see in Figure 2.10), so that the tip can be matched well. For this, the algorithm strongly depends on a good first guess because it has to find good corresponding points whose distance is below a threshold. But a good first guess is often not given by the result of the CRH alignment and because of this only few corresponding points can be found. Maybe it is also a problem, that when only few points are matched, but matched very well, so that the distances of the matches are very small, the error is below a given

threshold and the algorithm converges.

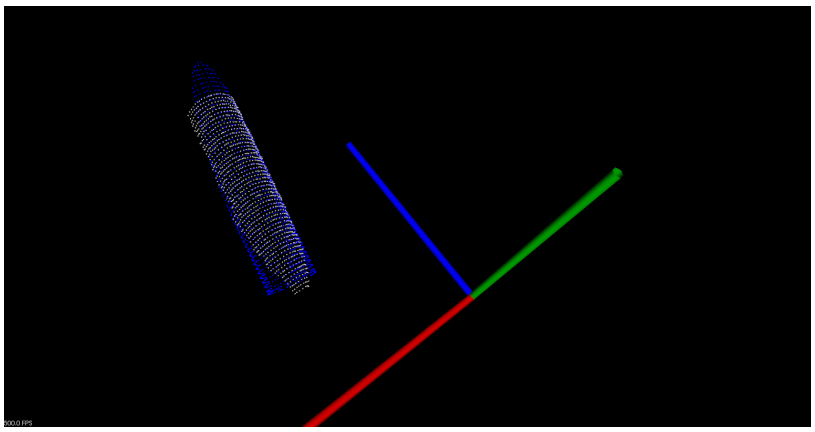
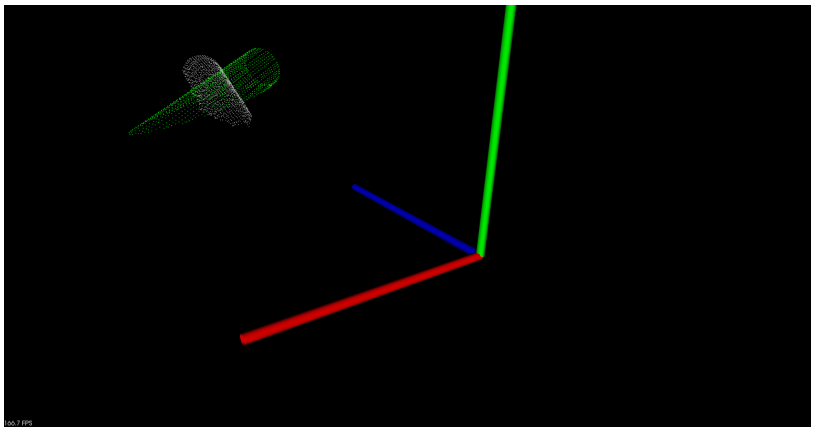


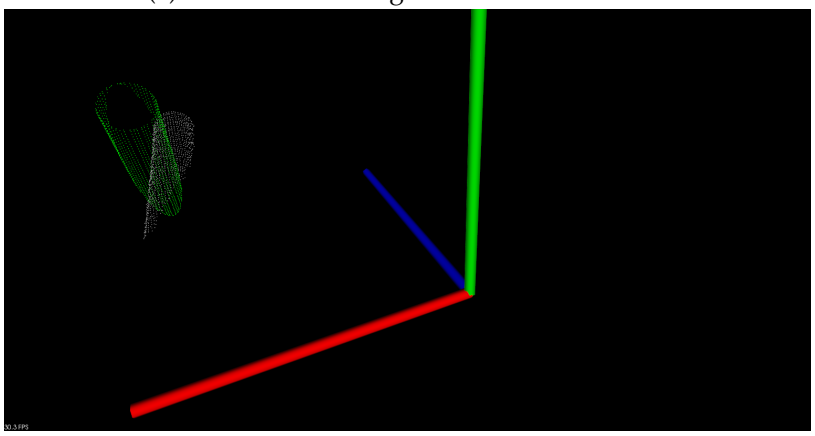
Figure 2.10: ICP result with wrong direction of CAD model.

2.2.6 Results

In general, the recognition framework didn't yield good results. The CVFH nearest neighbor search and the CRH alignment found very weird results. It has also been tried to use VFH as descriptor, but the results were equally worse. Transferring the model to the center of the OCT cloud worked, but they were in most cases not the least aligned. ICP could improve the result a great deal so that the clouds were aligned a bit after all, but it didn't work for all datasets and was in no way good enough to get results for the position or rotation of the needle. A result for a 0° dataset can be seen in Figure 2.11a, a result for a 45° dataset can be seen in Figure 2.11b. Both results were obtained with the OCT cloud cut in half. It has also been tried to use the full cloud, but then the matching was even worse because there were more points on the end of the needle where the model could be fit. Using only half of the tip of the OCT cloud also didn't work, because then there were too few points in the OCT cloud and the tip could be fit anywhere on the CAD model, as can be seen in Figure 2.12. The point where to cut for the tip half is computed as described in 2.3.1. Additionally, it seems like ICP isn't suitable for this specific task. Sometimes the results were quite good, like for an 0° dataset which can be seen in Figure 2.13a. But for other datasets, like a 45° dataset the results were really bad, as can be seen in Figure 2.13b.



(a) Result of CRH alignment for a 0° dataset.



(b) Result of CRH alignment for a 45° dataset.

Figure 2.11: The matching results after CRH alignment.

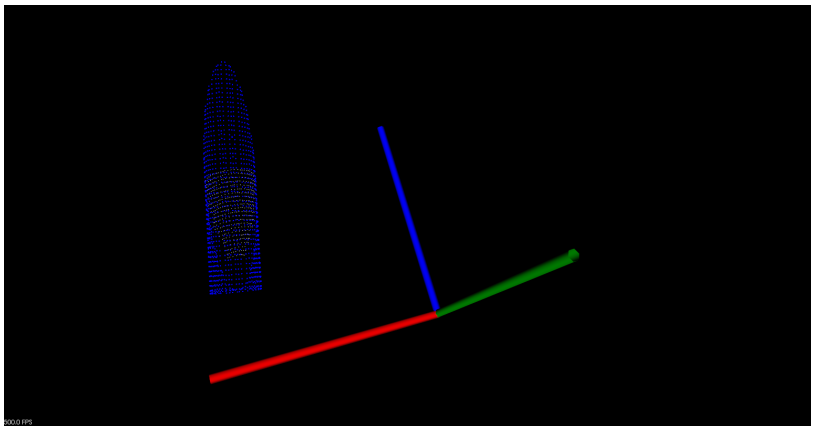


Figure 2.12: The matching result after using only the tip of the OCT cloud.

2.3 Shifting Algorithm

In this approach an algorithm similar to ICP has been implemented, but additional information about the needle, like the direction of the OCT point cloud is used. With this information, the number of parameters that have to be found is much smaller, because only the z-rotation and the shift in needle direction remain. The computation of the needle direction is described in depth in the next section. The algorithm then shifts the needle along the computed direction forward and backwards, and changes the z-rotation, to find the best combination of shift and angle. The best combination is determined by the highest number of point matches between the clouds. Then the intervals for the angle and the shift are refined by using the intervals

$$[bestAngle - 2 * angleStep; bestAngle + 2 * angleStep] \tag{2.1}$$

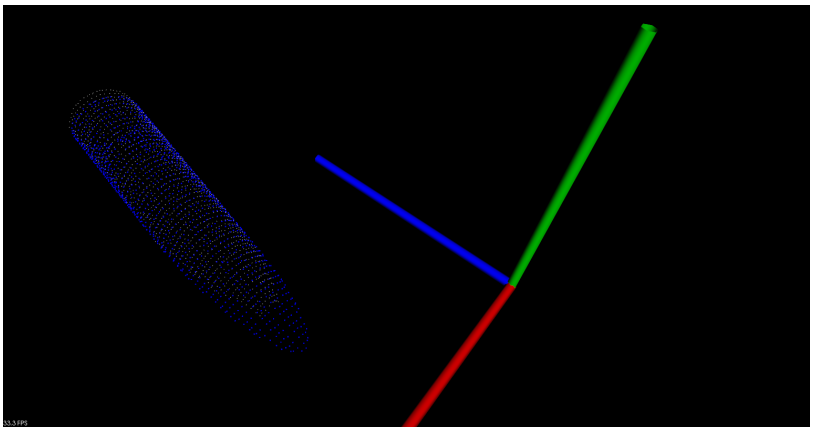
$$[bestShift - 2 * shiftStep; bestShift + 2 * shiftStep] \tag{2.2}$$

and also smaller steps, for which the step size of the round before is divided by 5. The intervals are also restricted by the interval used in the round before, for example if the start angle is -90 degree, then the next interval start can't be smaller than -90 degree.

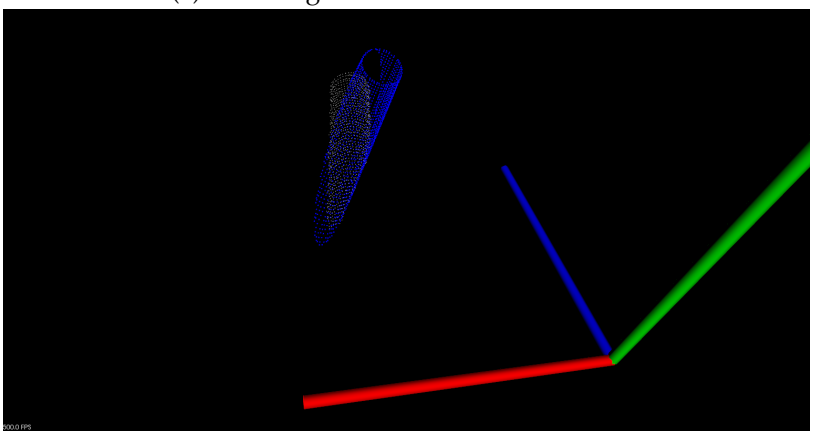
A lot of experiments were carried out to improve this basic algorithm. They are described in the following sections. Some of the results are described more closely in chapter 3.

2.3.1 Initial Guess Computation

For the initial rotation (especially the rotation around x- and y-axis) the direction of the OCT needle is computed. For this, in every frame a bounding box around the visible

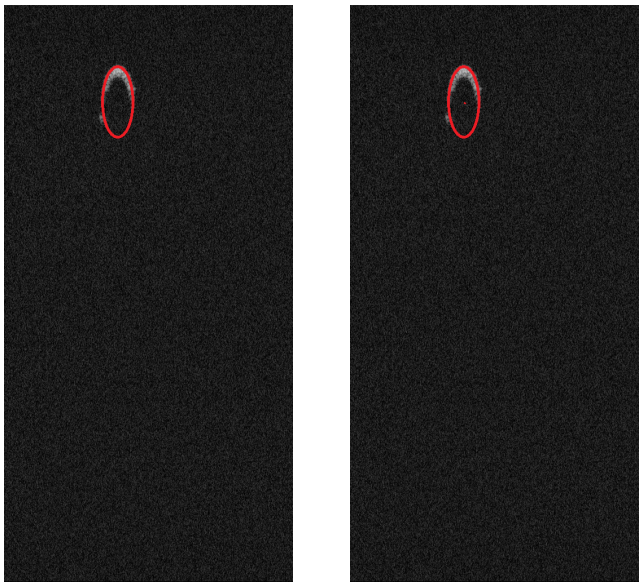


(a) Matching result of ICP for 0° dataset.



(b) Matching result of ICP for 45° dataset.

Figure 2.13: The matching results after ICP.



(a) OCT image with ellipse fit to the outer points of the visible needle part.

(b) OCT image with the center of a fitted ellipse.

Figure 2.14: Ellipse fitting.

needle part is computed, as can be seen in 2.1. The points in this frame that would be chosen for generating the OCT cloud are also used for fitting an ellipse to them, as shown in 2.14a. The center points of these ellipses (like visualized in 2.14b) are then used to fit a line which is aligned to the OCT cloud and goes through the middle of it.

For the computation of the direction, at first all points are assembled into a matrix.

$$P = \begin{bmatrix} p_x^1 & p_y^1 & p_z^1 \\ p_x^2 & p_y^2 & p_z^2 \\ \vdots & \vdots & \vdots \\ p_x^n & p_y^n & p_z^n \end{bmatrix} \tag{2.3}$$

Then the mean of every column is computed, which is later used as a point on the direction line.

$$m = \begin{bmatrix} mean_x & mean_y & mean_z \end{bmatrix} \tag{2.4}$$

The mean is then subtracted from every row of matrix P. As a next step, the conjugate transpose of P is computed and multiplied with P, which results in a symmetric matrix A.

$$A = P^H * P \tag{2.5}$$

A can then be factorized into

$$A = VDV^{-1} \tag{2.6}$$

where *D* is a diagonal matrix with the eigenvalues on the diagonal, and *V* is a matrix with the eigenvectors as its columns. The third column of *V* is then normalized and used as direction for the needle.
At first this often led to a lot of outliers because for the first few frames of the OCT cloud only few points were found, and apparently they weren't enough to find a good match for an ellipse. So a threshold of 50 points has been applied, to ensure that enough points are found in one frame to get an ellipse. Additionally, RANSAC was used for the found center points of all frames to filter other outliers that may occur. A distance threshold of 0.1 yielded the best fitting results between OCT cloud and CAD model.
It was also tried to fit a line through the top middle of the bounding boxes and subtracting the needle radius in y-direction, but this led to imprecise results for higher angles.

The initial rotation is computed by

$$R = \begin{bmatrix} rotX_x & rotX_y & rotX_z \\ rotY_x & rotY_y & rotY_z \\ direction_x & direction_y & direction_z \end{bmatrix}, \tag{2.7}$$

where

$$rotX = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} direction_x \\ direction_y \\ direction_z \end{bmatrix}, \tag{2.8}$$

$$rotY = \begin{bmatrix} direction_x \\ direction_y \\ direction_z \end{bmatrix} \times \begin{bmatrix} rotX_x \\ rotX_y \\ rotX_z \end{bmatrix} \tag{2.9}$$

The initial translation is computed by

$$T = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + direction * ((p_z - tipHalf)/direction_z), \tag{2.10}$$

where *p* is a point on the line of the needle direction, *tipHalf* the z-value where the half of the OCT cloud tip ends and *direction* the normalized direction vector.
tipHalf is computed by using the width of the bounding box which is computed around the visible needle part. For the tip of the needle, the width of the bounding boxes is constantly growing, for the needle body it is constant. Piece-wise linear fitting is used to find the point

where the width stops growing. The widths are split in several iterations so that always one part is used to fit a line with zero slope, while for the other part the same method as for the direction computation is used. For both methods the squared distances error is computed and then summed up to get the overall error. Then the split version which has the smallest overall error is chosen and the index where the widths have to be split is the value of *tipHalf*.

2.3.2 Transformations

When a new combination of angle and shift is chosen, a transformation matrix is computed and applied to all points of the CAD model cloud, so that the model is translated to the new position and rotated by the requested angles.
The rotation around the z-axis for an angle α would look like

$$R = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.11}$$

The translation for a shift δ, current translation t_c and needle direction d is computed by

$$T = \begin{bmatrix} t_c x \\ t_c y \\ t_c z \end{bmatrix} + d * (\delta / d_z). \tag{2.12}$$

The rotation and translation are then assembled into a transformation matrix which looks like the following:

$$M = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.13}$$

where *r* stands for rotation and *t* for translation.

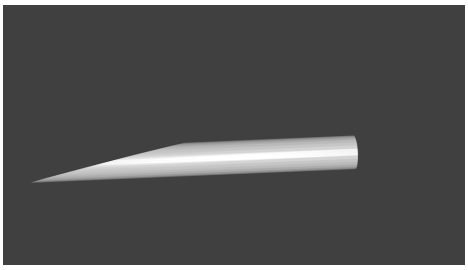
2.3.3 Summing up Point Matches

At first, the angle-shift-combination with the highest number of point matches was chosen in the algorithm. But since there isn't much difference between a lot of the values, maybe only one or two matches more or less, it could happen that an angle-shift-combination was chosen which didn't fit very well, maybe because for this case some outliers of the OCT could be matched in addition to the normal matches. Also, in choosing the best match in this way, the angle and shift depend on each other. But if for example the shift is chosen

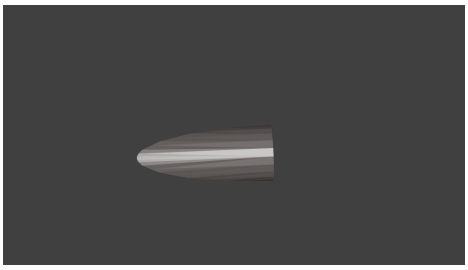
very badly due to the outlier theory just mentioned, only an interval around the shift in this combination is used for the next iteration, which would lead to bad results.
An idea to overcome this problem was to sum up the matches found for every angle, and the matches found for every shift. So for example for angle 45° the sum of the matches for shift 0.0, 0.05, 0.1 ... is computed. Then the angle with the most summed up matches is chosen for the next iteration. The same is done with the shift, for example for shift 0.0 the sum of the angles -90°, -89°, -88°, ... is computed and then the shift with the greatest sum is chosen for the next iteration. If the angle is good, a lot of matches will be found for a lot of shift steps, more than for a bad angle. The same works for the shift, if it is a good one which doesn't for example exceed the tip, then for a lot of angles a lot of matches will be found. Additionally, in this approach, the angle and the shift are chosen independently from each other, so a very good shift and a very good angle can be chosen even if they wouldn't be chosen as best combination in the basic approach.

2.3.4 Different Models

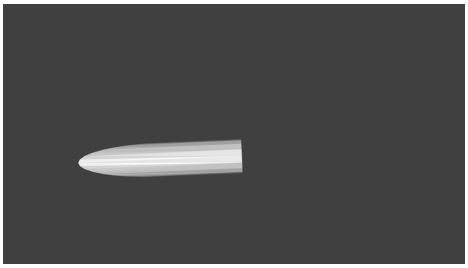
The algorithm was tested with different models of the needle to find one that fits very well. The tests included two CAD models of the whole needle (Figure 2.15a), one with a thick hull and one consisting only of the shell. The results showed that the shell model worked better because there weren't so many points that could lead to confusing matches. All other models only consisted of the shell, which include a model of a part of the tip (Figure 2.15b), and a model of the whole tip (Figure 2.15c, Figure 2.15d). When the tip model yielded the best results, also tip models with different bevel angles were tried. A model with a 12 degree angle yielded the best results.
It has also been tried to cut this model in half in y-direction, so that it resembles better the shape of the OCT cloud, but this didn't produce good results because for high angles the shape isn't similar anymore.
Another experiment was to cut off the first few points of the tip of the CAD-model, since those are always missing in the OCT cloud, but the results didn't improve with this approach.
The algorithm was also tested with the OCT cloud cut at the half of the tip, which has been computed with piece-wise linear fitting like described in 2.3.1 before. But as for the approach in 2.3.1, the cloud was too small to yield a good matching result with the CAD model.
Another test was to cut the OCT cloud in half in z-direction. This improved the results slightly, since the length of the CAD model and the OCT cloud were now more similar. The cut off part of the OCT cloud didn't seem to provide useful information anyways, since it only has the shape of a half cylinder. As a refinement of this step, since the OCT clouds from different datasets have different lengths, only so many points were cut off that the length of the OCT cloud was the same as the length of the CAD model, which can be useful when using different models, beause for example the model with a bevel angle of 15° is much shorter than a model with a bevel angle of 12°.



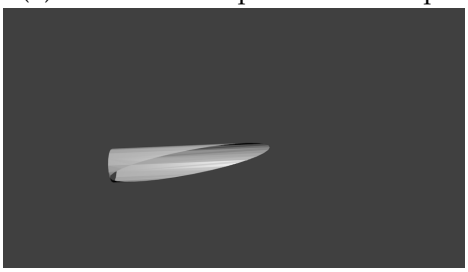
(a) CAD model of the whole needle.



(b) CAD model of part of needle tip.



(c) CAD model of the needle tip, top view.



(d) CAD model of the needle tip, open side.

Figure 2.15: The different CAD models used.

As results of experiments with the models, the tip model with a bevel angle of 12 degree and the OCT cloud with a part cut off were kept for further experiments.

2.3.5 Point Matches and Rejectors

Normal correspondence[2] estimation and reciprocal correspondence estimation was tested with a few datasets to get an impression which works better.
While normal correspondence estimation keeps all correspondences that have a distance below a defined threshold, reciprocal correspondence estimation keeps only those with a distance below a certain threshold and those where the point in the source is the best match for the point in the target, which means only those correspondences that are found when searching from source to target and vice versa [1]. The threshold in the experiments was set to the same value as the resolution used to sample the points of the clouds. The results showed that the normal correspondence estimation worked better than the method with reciprocal correspondences.

To get rid of invalid correspondences, rejectors can be applied [9].

- A distance rejector rejects all correspondences that have a greater distance than a defined threshold. Experiments were carried out with the same resolution that was

[2]Matched points are called correspondences in the point cloud library.

used to sample the model points, but at this time the threshold for normal correspondence estimation has been set to twice this threshold.
Using a distance rejector didn't improve the results that were collected without using any rejector.

- Using a oneToOne rejector also didn't improve results. This rejector looks out for correspondences for which a point in the target cloud was assigned multiple times to a point in the source cloud, and it selects only the correspondence with the smallest distance of the multiple assignments, which is then kept.
- A sample consensus rejector was also tested throughout the experiments. This rejector tries to remove other correspondences by computing the euclidean distance between points. It also didn't help to improve results.
- A median rejector computes the median of the found correspondences and rejects all correspondences with a distance greater than the median. This rejector helped to improve the results slightly, so it was kept for use in most other experiments. But after some changes in thresholds and other adjustments, it seemed to reject too many points that should be matched, as can be seen in Figure 2.16, so it wasn't used in the final algorithm.

2.3.6 Distance Metrics

An experiment, where the number of matched points wasn't considered as the indicator for the best match, was to apply distance metrics to all found matches and use the smallest result of all positions/angles as best match.
The metrics tested were:

- Manhattan distance:

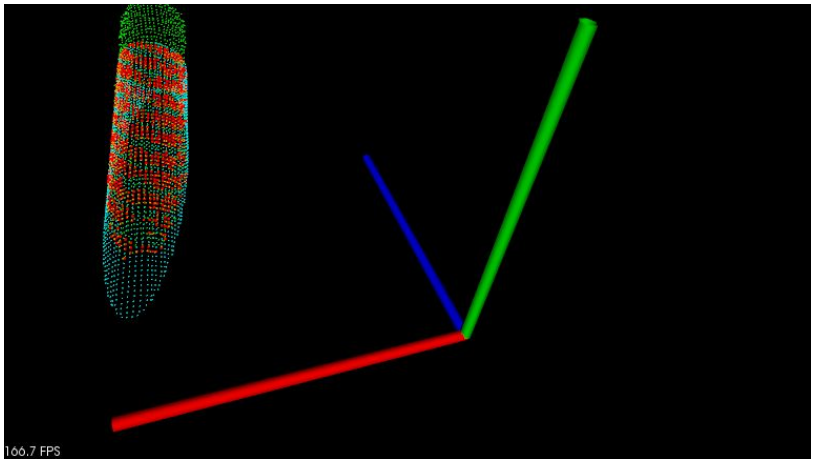
$$(x, y) \mapsto \sum_{i=1}^{n} |x_i - y_i| \tag{2.14}$$

- Euclidean distance:

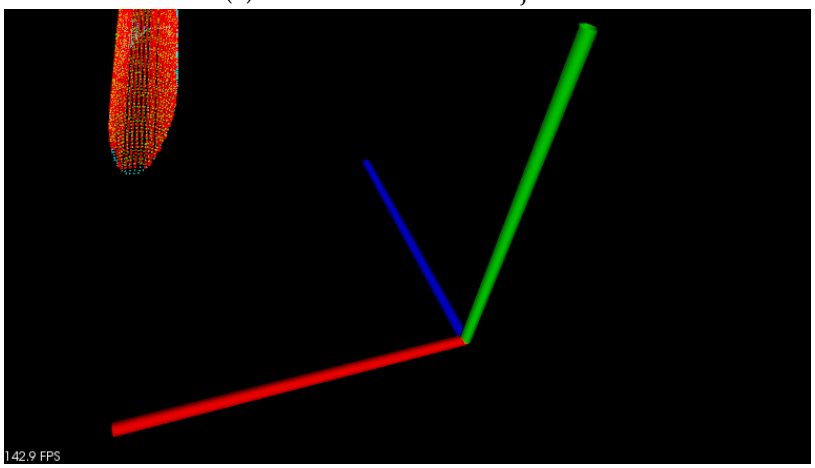
$$(x, y) \mapsto \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.15}$$

- Squared euclidean distance:

$$(x, y) \mapsto \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2.16}$$

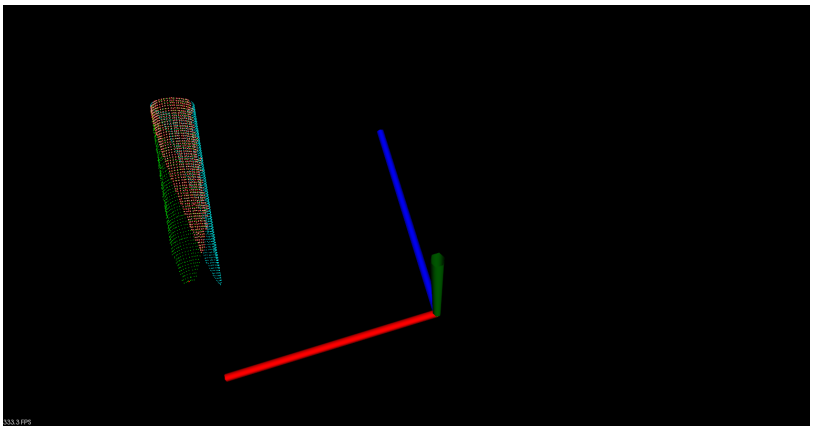


(a) Match with median rejector.

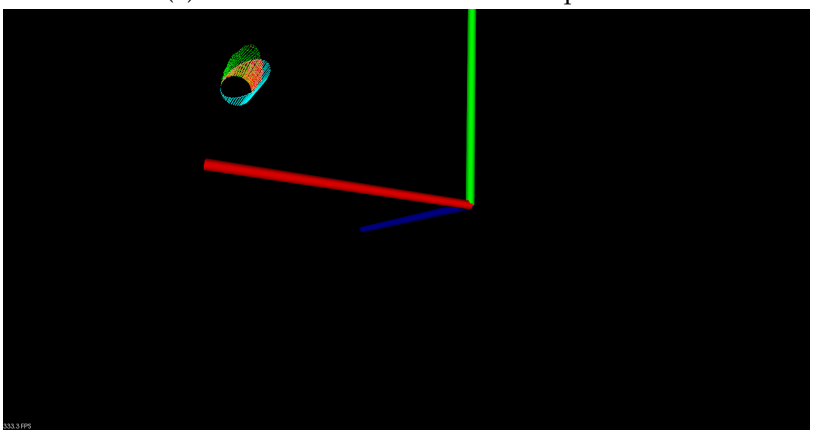


(b) Match without median rejector.

Figure 2.16: Point matches (red) with and without median rejector.



(a) Match with distance metrics top view.



(b) Match with distance metrics from below.

Figure 2.17: Match after applying distance metrics (green=OCT cloud, blue=CAD model, red=point matches).

These experiments didn't yield any usable results because always the largest angles were chosen (like -90 or 90 degree, when they were used as initial interval boundaries). This happened because for these angles only few point matches were found, but these few had a very small distance, so the computed value of the metric was very small and the large angle was therefore chosen as the best match, as can be seen in Figure 2.17.
It has also been tried to combine the normal point-match-counting-approach with these distance metrics, by counting point matches in the first iteration and applying distance metrics in the remaining iterations. But those results also were much worse than the normal approach alone.

2.3.7 Different Shift/Angle Range

Different ranges for the shift have been tried. Shifting in the negative direction was useless, because the models were always aligned in such a way that the model had to be shifted in positive direction to get a good result. So the start of the interval was set to 0. It has been tried to restrict the upper limit of the interval to a value as low as possible, and it seems that a value of 0.5 is sufficient for all models so that a good fit can be found.
The angle interval was set to range from -90 degree to 90 degree at first because the angle couldn't be restricted in any way. Instead, different steps have been tried. A step of 5 degree in the first iteration yielded worse results than a step of 1 degree, but the 1 degree step took much longer because the point matches had to be computed for many more angle/shift combinations. Nevertheless, the 1 degree step was chosen for the algorithm.

2.3.8 Tip Approximation

To get better results for the angle, it has been tried to approximate the position of the needle tip (which is mostly not present in the OCT cloud), so that the CAD model can be rotated by an angle around the needle axis for which the CAD model tip matches the OCT cloud tip.
At first, the points with the minimum values in z direction of the OCT cloud are used to compute the middle between them, which can be seen as red dot in Figure 2.18.
The same is done for the CAD model which has already been transformed so that it fits the needle. All values are projected to the x-z-plane. Then a line which goes through the computed OCT middle and which is parallel to the needle direction is crossed with a line that goes through the computed tip of the CAD model and is perpendicular to the needle direction.

$$OCT\_middle + \lambda * OCT\_direction = CAD\_middle + \mu * \begin{pmatrix} OCT\_direction.z \\ 0 \\ -OCT\_direction.x \end{pmatrix} \quad (2.17)$$

The crossing point is then the approximated needle tip, to which the tip of the CAD model should be rotated. A schematic representation of these computations can be seen in Figure 2.19.

2.3.9 Performance Improvement

Since the algorithm took very long for large initial intervals, it has been tried to improve the performance. The first approach was to stop the algorithm if for the next n angles no improvement in the number of matched points has been found. Unfortunately this didn't work very well, because it yielded different and worse results than the normal algorithm without performance improvement, except when using such large n that the performance improvement was not significant.

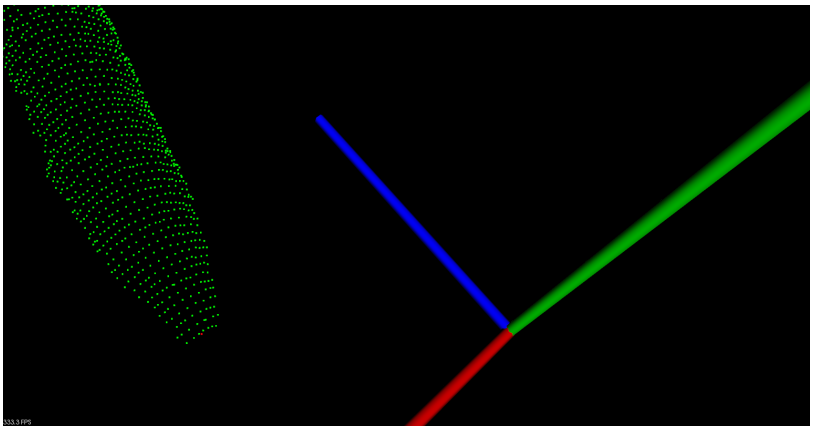


Figure 2.18: Computed middle of the OCT cloud's first frame (red dot).

Another approach was to use *Open Multi-Processing* [2] to parallelize all angles for which then all the shifts are computed. This worked very well and tests showed that the performance could be improved by approximately 48% which can be seen in Table 2.1 when comparing column 1 and 2 or column 4 and 5.
A different experiment was to use tip approximation as a first guess for the angle and then an interval of

$$[angleGuess - 5°; angleGuess + 5°] \tag{2.18}$$

as start for the angles in the first iteration which improved the performance by approximately 26%, as can be seen in Table 2.1 when comparing column 1 and 4. Using an interval of

$$[angleGuess - 10°; angleGuess + 10°] \tag{2.19}$$

has also been tried, but the results for the angle were worse, so the first one was kept.

For computing the time the algorithm takes, *ScopeTime* [4] of point cloud library has been used. The tests were carried out on a computer with an Intel(R) Core(TM) (CPU) i7-7700HQ @ 2.80GHz and a GeForce GTX 1050 (GPU).

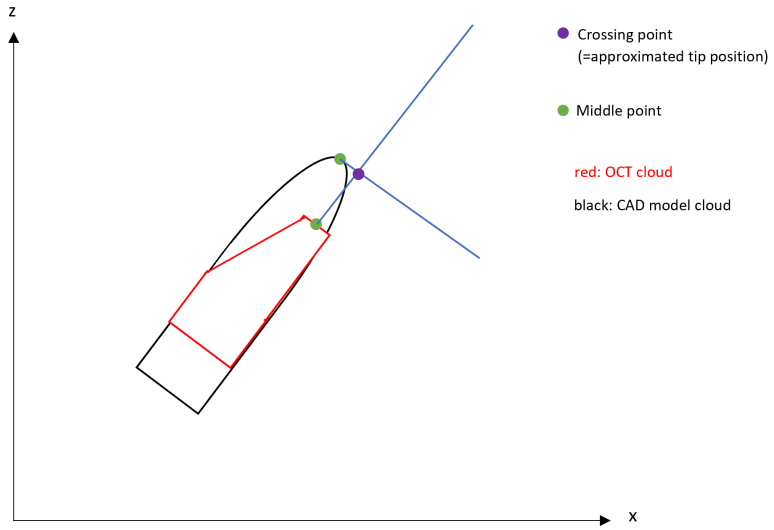


Figure 2.19: Schematic representation of the tip approximation.

Table 2.1: Results of performance tests.

parallelized with tip approximation	not parallelized with tip approximation	improve-ment	parallelized with angle range -90°to 90°	not parallelized with angle range -90°to 90°	improve-ment
16205ms	34242ms	47.32%	54614ms	110131ms	49.59%
15808ms	35297ms	44.79%	56506ms	112560ms	50.20%
13783ms	28087ms	49.07%	56552ms	117285ms	48.22%
13911ms	28762ms	48.37%	57199ms	113379ms	50.45%
13732ms	28565ms	48.07%	56306ms	113843ms	49.46%
13758ms	28295ms	48.62%	49774ms	99273ms	50.14%
13751ms	28185ms	48.79%	56674ms	112730ms	50.27%

2.3.10 Iterative Closest Point

Another approach was to use the initially computed transformation as an input for normal ICP. When the results (which can be seen in Figure 2.21a) from this experiment weren't good, it has also been tried to use the initially computed transformation as well as the z-rotation computed with tip approximation as an additional input for ICP. A result can be seen in Figure 2.21b. But as described in 2.2.5, a good initial guess is important for ICP. If many points weren't matched, which even happened with the tip approximation when the initial shift was not very good or many points of the OCT cloud tip were missing,

ICP didn't find the shift in needle direction as good as the normal shifting algorithm. An example for this can be seen in Figure 2.20. For the angle, without the tip approximation

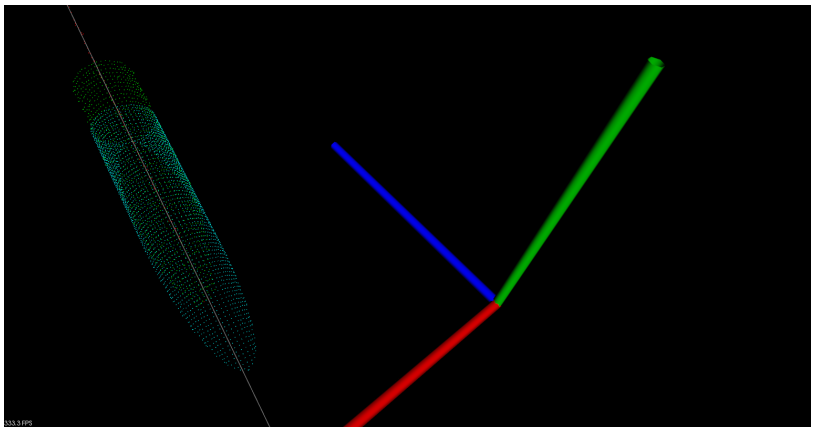


Figure 2.20: Result of ICP with bad initial guess for shift in needle direction.

always some values close to 0° seemed to be sufficient for ICP, even with tip approximation it didn't find better results than the normal shifting algorithm in this case, too.

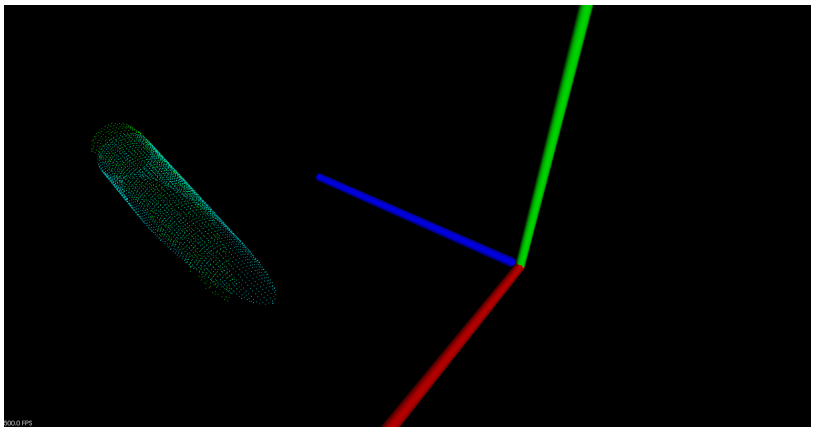
2.3.11 Thresholds and Resolutions

There were also some experiments with different thresholds and resolutions:

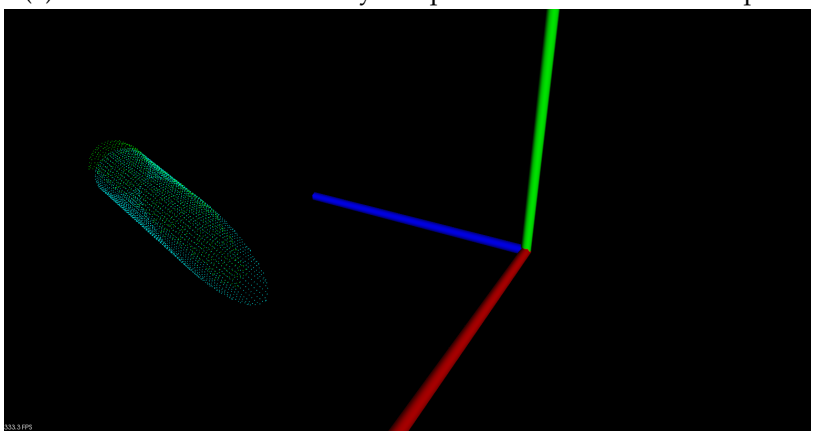
- Point Match Threshold
  Different thresholds for computing point matches between the clouds have been tried. This threshold is the maximum distance points can have to be considered a match. For greater thresholds, naturally more point matches were found, but the results were better for a very small threshold.

- Bounding Box Area
  When generating the OCT cloud, a label found in the image is considered part of the needle, when the area is greater than 250. For this, also higher values have been tried, which resulted in OCT clouds with a tip that wasn't defined very well. For lower values than 250 the OCT cloud didn't change much, but the risk of choosing a wrong labeled part of the image gets higher, so 250 was chosen.

- RANSAC Threshold for Direction Computation
  When computing the direction of the OCT cloud, RANSAC has been used to find the best direction with the certain points of the ellipses. Different thresholds like 0.01 or 0.5 have been tried, but the fitting results weren't as good as with some value in the middle between those two, so 0.1 was chosen which yielded good results.



(a) Result of ICP with initially computed transformation as input.



(b) Result of ICP with initial transformation and z-rotation.

Figure 2.21: Results of using ICP.

- Model Resolutions
  Another experiment was to use different resolutions of the OCT cloud as well as the CAD model. Using less points for both the OCT cloud and the CAD model led to worse results. But using all points available for the OCT cloud and a lot of points for the CAD model improved the results significantly, especially for the tip approximation. A matching result with more points can be seen in Figure 2.22. Unfortunately, when using more points the algorithm takes a lot longer. Compared to the tests with less points in 2.3.9, it was about 20% slower.

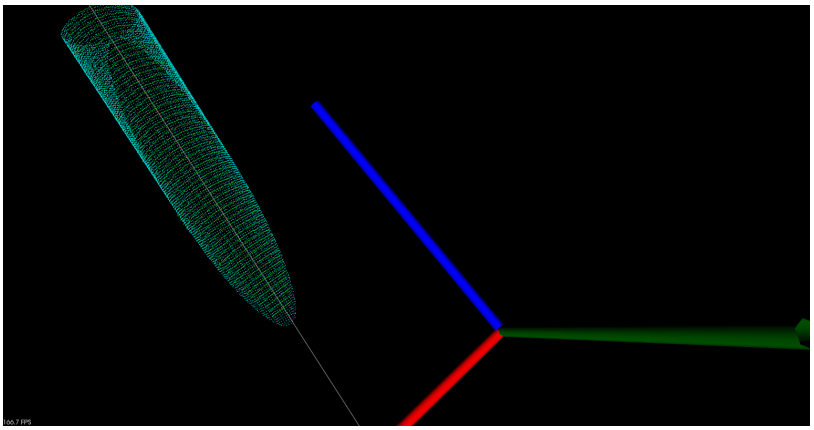


Figure 2.22: Result when using more points.

3 Results

3.1 Z-Rotation

For the evaluation of the angle of the z-rotation, 53 datasets were used. They included:

- 6 datasets with an angle of -45°
- 5 datasets with an angle of -30°
- 7 datasets with an angle of -15°
- 9 datasets with an angle of 0°
- 7 datasets with an angle of 15°
- 8 datasets with an angle of 30°
- 11 datasets with an angle of 45°

3.1.1 Summed up vs. not summed up

The first tests were performed under the following conditions:

- rejectors: median rejector
- RANSAC threshold for direction computation: 0.01
- model: tip with bevel angle 12°
- angle interval: -90° to 90°
- angle step: 1°
- shift interval: 0.0 to 0.3
- shift step: 0.05
- OCT cloud post-processing: cut in half in z-direction

Figure 3.1 shows a boxplot of the results under these conditions with summed up point matches, while Figure 3.2 shows a boxplot of the same tests without summing up point matches but with always picking the best combination of shift and angle.
The results showed that summing up point matches has a higher accuracy for more datasets, 17 datasets got a result with 5° deviation, 7 were within 10° deviation and the rest had a deviation of more than 10°. When the point matches weren't summed up, only 14 datasets had a 5° deviation, 5 within 10° deviation and the remaining 34 datasets had a deviation greater than 10°.

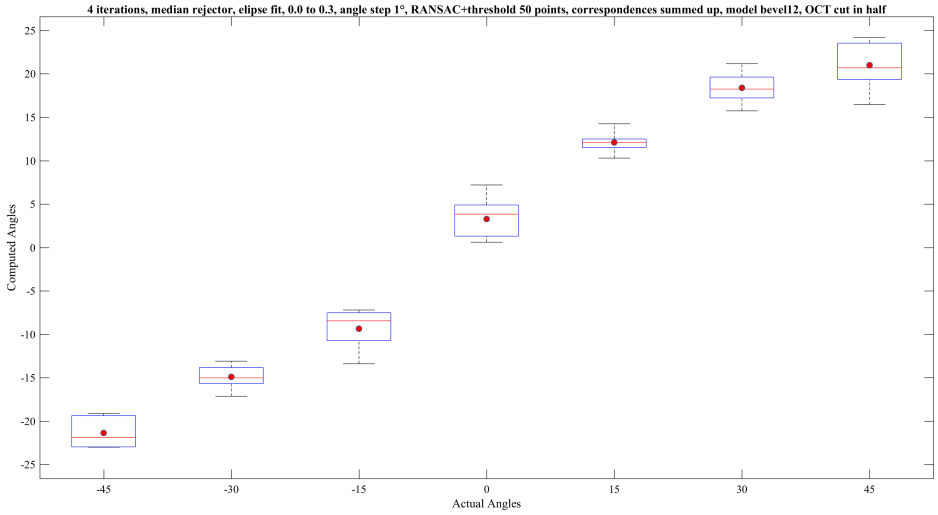


Figure 3.1: Boxplot of results with summed up point matches.

3.1.2 RANSAC Threshold

The next tests were carried out under the same conditions as in the section before, without summing up point matches, but with a different RANSAC threshold (0.1) for the direction computation. The results were not as good as with 0.01, only 11 datasets were within a 5° range, 8 within a 10° range and again 34 had a deviation of more than 10°. But nonetheless this threshold was kept because the direction results were better for this threshold which could be seen on images. Figure 3.3 shows the results for a threshold of 0.01. Also, experiments under different conditions showed that the threshold of 0.1 worked better than for example 0.5 or 0.01.

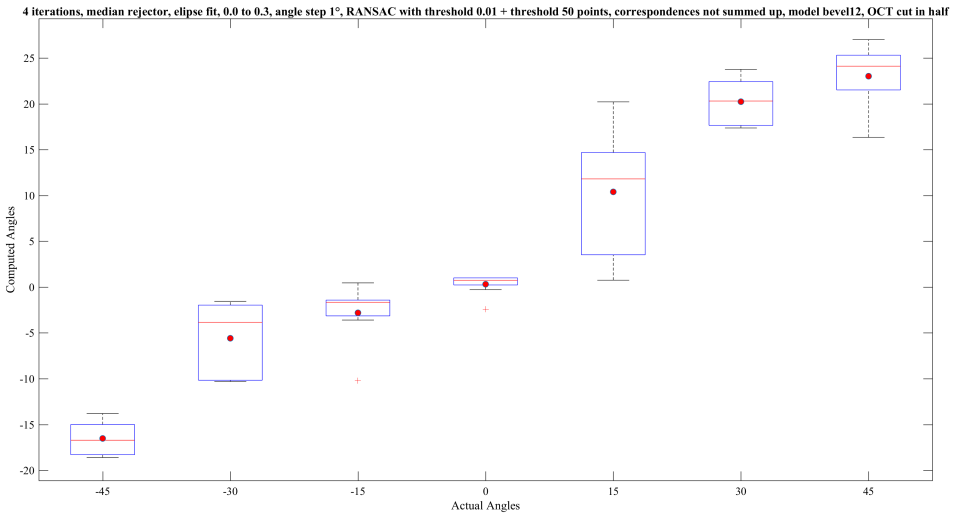


Figure 3.2: Boxplot of results without summed up point matches.

3.1.3 Tip Approximation

Before using tip approximation as help for defining the angle interval in the shifting algorithm, it was tested how good it performs alone, and if the results are usable at all. It turned out, that it yielded very good results on its own, with the same conditions as in section 3.1.2 with RANSAC threshold 0.1. The results can be seen in Figure 3.4. 25 datasets were within a 5° range, 27 within a 10° range and one was worse than 10° deviation, which is a major improvement compared to the results before.
The tip approximation even outperformed the whole shifting algorithm for the angle, except for 0° but even for this angle most results were within 5° deviation. The shifting algorithm tends to find the angle closest to zero in the defined interval instead of finding the best one, because for an angle closer to zero more point matches can be found.
When the shift was already very good or tip approximation was applied again after the shifting algorithm found the best shift, the results could even be improved a bit.

3.1.4 Limited Angle Interval

When using tip approximation to define the interval for the angle, it has been tried to use a range of

$$[angleGuess - 5°; angleGuess + 5°] \tag{3.1}$$

or

$$[angleGuess - 10°; angleGuess + 10°] \tag{3.2}$$

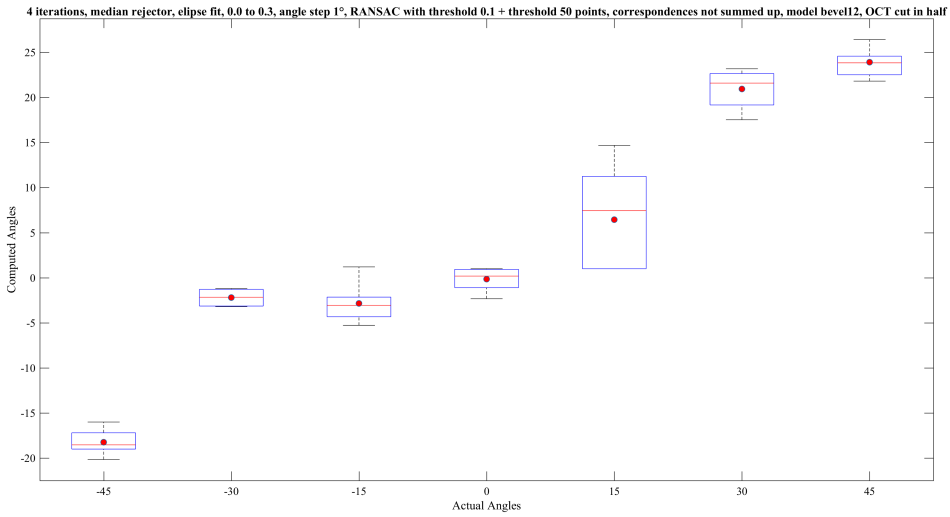


Figure 3.3: Boxplot of results with threshold 0.1.

The results can be seen in Figure 3.5 and Figure 3.6. For the 10° range, 19 datasets were within a 5° range, 22 within a 10° range and 12 had a larger deviation, which is an improvement to the normal algorithm with an interval for the angle of 180°. The results for 20° angle interval were not as good, only 12 datasets were within a 5° range, 14 within a 10° range and 27 had a larger deviation.

3.1.5 Different Models

The results for a model with a bevel angle of 15° can be seen in Figure 3.7. Compared to 3.5, the results were worse. 16 datasets were within a 5° range, 17 within a 10° range and 20 had a larger deviation.

3.1.6 More Points

When using more points for the OCT cloud and the model, the results could be improved. The shifting algorithm computed the angle for 22 datasets within 5° deviation, for 22 datasets within 10° deviation and for 9 datasets the result was worse. Compared to 3.5, this is only a small improvement, but for the tip approximation alone the results could be improved very much. 40 datasets were within 5° deviation, 13 within 10° deviation and no result was worse than that. Figure 3.9 visualizes the results of tip approximation alone, Figure 3.8 visualizes the results of the shifting algorithm.

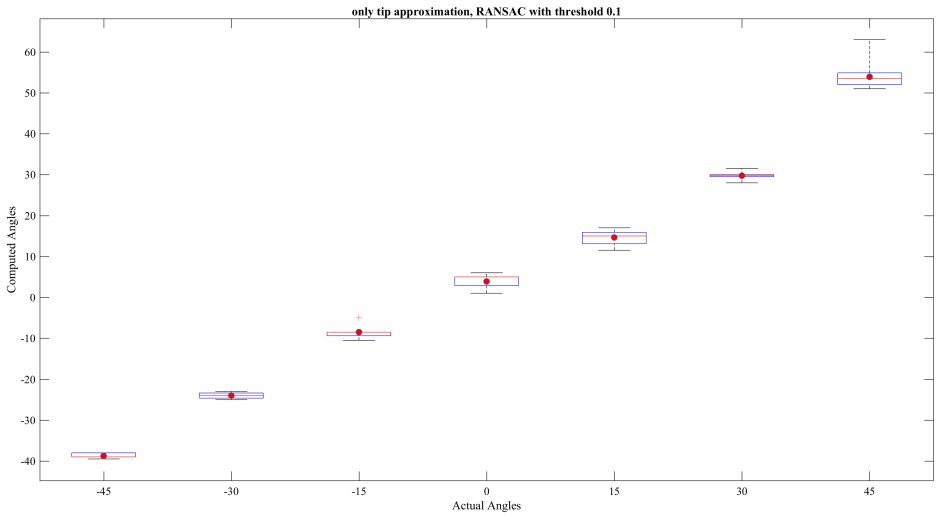


Figure 3.4: Boxplot of results of tip approximation alone.

3.1.7 Final Results

The results for the angle showed, that the shifting algorithm works best if

- the point matches are summed up
- the direction computation is done with a RANSAC threshold of 0.1
- a CAD model with a bevel angle of 12° is used
- tip approximation is used as input for the angle
- the start interval for the angle is set to 10° around the tip approximation result
- a lot of points are used for the CAD model as well as for the OCT cloud

But the results also showed, that tip approximation alone outperforms the whole ICP-like shifting algorithm and yields much better results for the angle, especially for the higher angles like 30 or 45 degree.

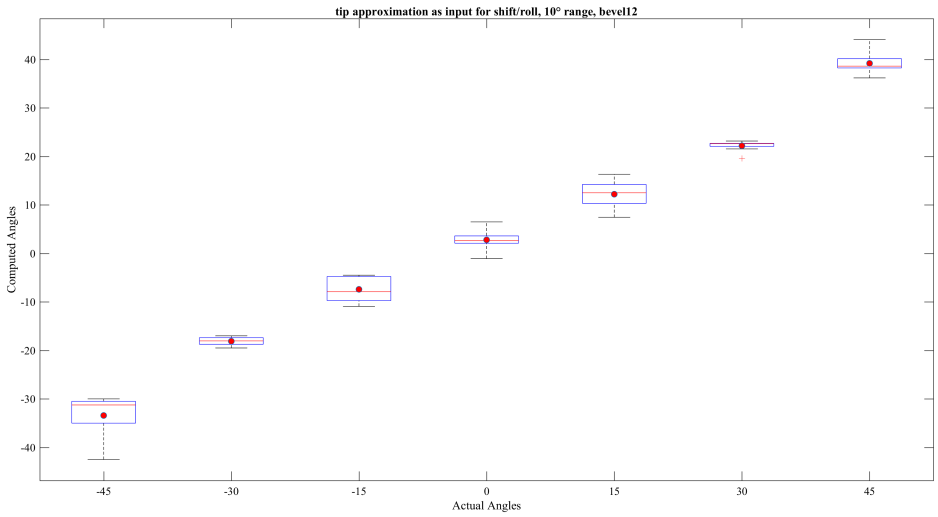


Figure 3.5: Boxplot of results with 10° angle range.

3.2 Position

Since no data for the absolute position of the needle in an OCT cube is available, the relative position has been used to evaluate how good the algorithm works for computing the needle position. The datasets used were obtained by moving the needle with a manipulator in x-, y- and z-direction without rotating it. The movements included $10\mu m$, $20\mu m$ and $40\mu m$. For every movement, 5 datasets have been used to compute the position. The positions of the centroid of the CAD model are shown in Figure 3.10. Even when only the movement in one direction is changed, there are also changes in the other directions, which could for example be a result of vibration noise when obtaining the OCT images or result from bad initial alignments of the CAD model when computing the needle direction.
Different experiments were carried out specifically with the intention of making the results more stable, like using a median rejector or not, using different lengths of the OCT cloud or a lower threshold for the number of points in a label so that it is considered part of the needle. But with none of these experiments, significant improvements could be seen. So the final result is, that with the configuration from 3.1.7, the distance error can be controlled within $15\mu m$. In Table 3.1, the individual mean difference of the 5 datasets used for each movement in x-, y- and z-direction is shown, as well as the overall distance.

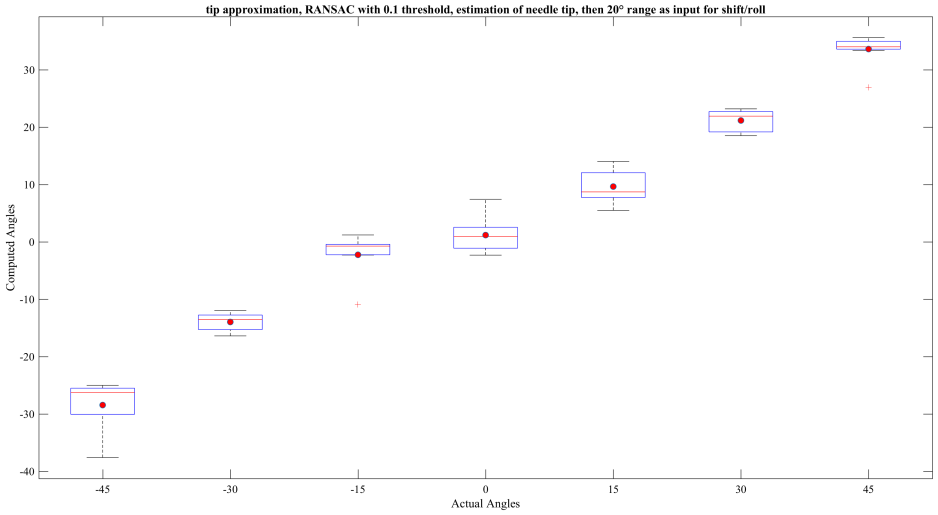


Figure 3.6: Boxplot of results with 20° angle range.

Table 3.1: Self-variance of the algorithm for datasets with different movements.

actual			computed			
Δx	Δy	Δz	Δx	Δy	Δz	distance
$10\mu m$	$0\mu m$	$0\mu m$	10,49	-11,79	7,69	17,55
$20\mu m$	$0\mu m$	$0\mu m$	20,04	2,98	4,16	20,68
$40\mu m$	$0\mu m$	$0\mu m$	33,01	6,12	9,21	34,81
$0\mu m$	$10\mu m$	$0\mu m$	3,98	8,99	5,30	11,17
$0\mu m$	$20\mu m$	$0\mu m$	2,72	-23,37	-2,97	23,72
$0\mu m$	$40\mu m$	$0\mu m$	1,84	-53,19	6,47	53,61
$0\mu m$	$0\mu m$	$10\mu m$	-1,31	0,52	7,43	7,56
$0\mu m$	$0\mu m$	$20\mu m$	1,03	-10,42	15,69	18,86
$0\mu m$	$0\mu m$	$40\mu m$	-1,07	11,33	47,02	48,38

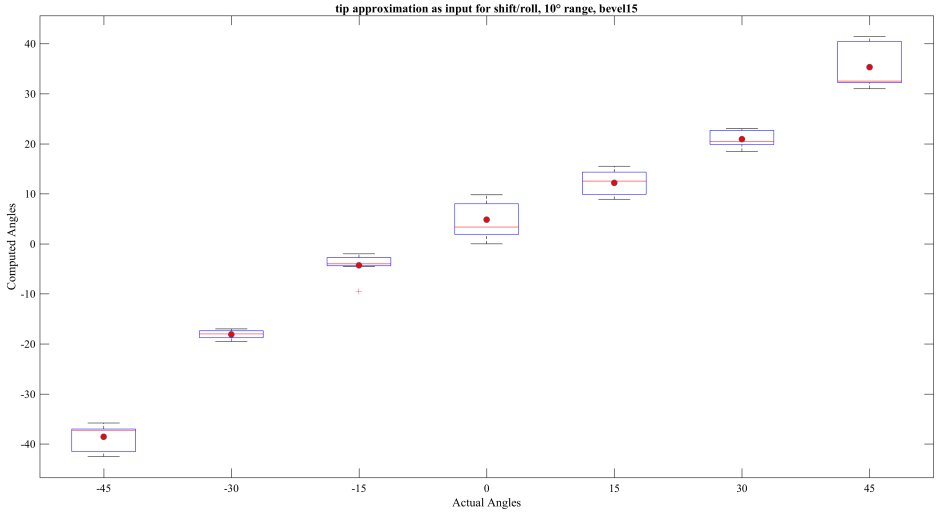


Figure 3.7: Boxplot of results with model with bevel angle 15°.

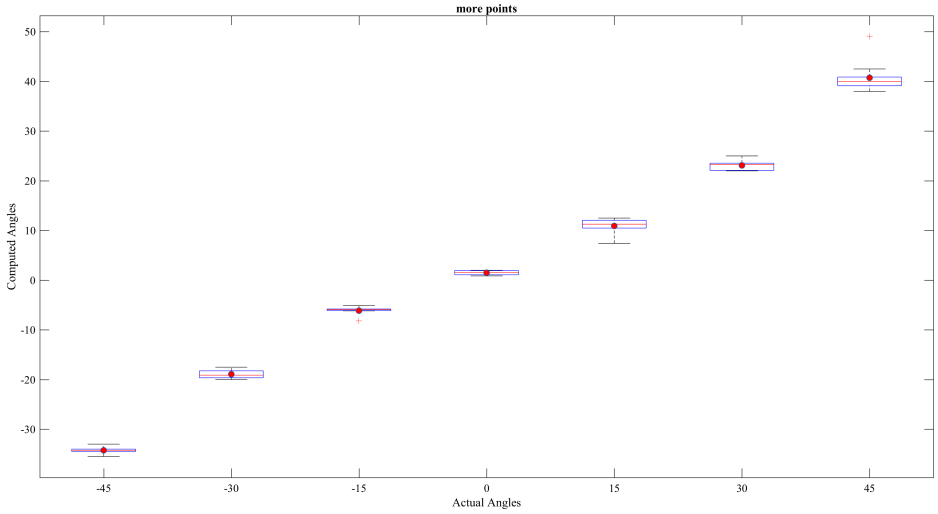


Figure 3.8: Boxplot of results with more points.

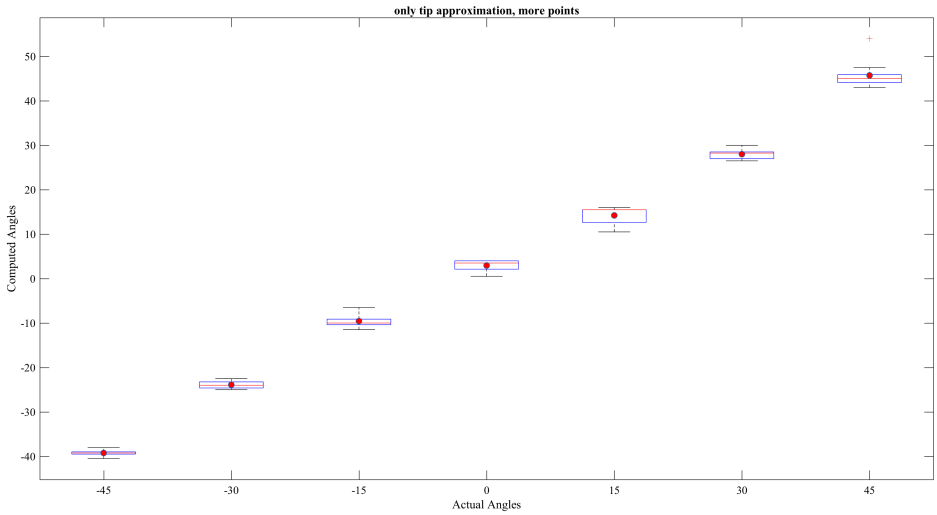


Figure 3.9: Boxplot of tip approximation with more points.

(a) Centroid position for x-movement.

(b) Centroid position for y-movement.

(c) Centroid position for z-movement.

Figure 3.10: Positions of the CAD model centroid for movement in x-, y- and z-direction, when moving 0, 10, 20 and 40μ and using 5 datasets each.

4 Conclusion

In this thesis, an algorithm has been developed, which can compute a 6DOF pose of a point cloud reconstructed from OCT images by matching it with the point cloud of a CAD model.

It turned out, that using feature descriptors like CVFH combined with CRHs and ICP for refinement doesn't perform well for the problem in this thesis. This happens mainly due to the fact, that the descriptors and CRH didn't work very well, which led to a bad initial guess for ICP. ICP then outputs a pose where in the worst case the point clouds aren't even aligned because ICP strongly depends on a good guess to get the direction of the model right. But even with given direction it didn't work as well as the algorithm developed in this thesis.

Using the needle direction as additional information was of great help, because the position and rotation then only depend on two parameters: the z-rotation and the shift along the needle direction. Counting point matches when using different combinations of these parameters yielded good results. For most datasets the angle was within 10° deviation, the position estimation accuracy could be controlled within $15\mu m$.
The best results could be achieved using the following conditions:

- A needle CAD model with 12° bevel angle
- A RANSAC threshold of 0.1 for the direction computation
- A scale for the OCT point cloud of *(2.7, 2.4, 3.0)*
- Summing up the point matches and choosing independently instead of choosing the best combination
- Tip approximation to limit the angle interval
- A shift interval of 0.0 to 0.5

5 Future Work

There are several open challenges that can be considered for future development to improve and extend the current approach. For example, there were several experiments with parameters that couldn't be carried out in this thesis due to time constraints:

- A different interval after using the tip approximation could be used.
- There could be ways to improve the computation of the needle direction, for example by using RANSAC when fitting the ellipses to remove outliers, so that the CAD model fits better for some datasets where the current approach didn't work perfectly.
- Maybe the shifting algorithm could be improved by using different steps for the shift and the angle.
- Also, another measure than counting point matches for determining how good the transformation is could be researched and applied.
- The CAD model could be made more similar to the OCT cloud by cutting it in half in y-direction after applying tip approximation.

Bibliography

[1] The PCL Registration API. http://pointclouds.org/documentation/tutorials/registration_api.php, (accessed 17-September-2017).

[2] OpenMP, 2017. http://www.openmp.org/, (accessed 14-September-2017).

[3] pcl::registration::ConvergenceCriteria Class Reference, 2017. http://docs.pointclouds.org/trunk/classpcl_1_1registration_1_1_convergence_criteria.html, (accessed 25-September-2017).

[4] Point Cloud Library Documentation, 2017. http://docs.pointclouds.org/trunk/classpcl_1_1_scope_time.html, (accessed 17-September-2017).

[5] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. CAD-Model Recognition and 6DOF Pose Estimation Using 3D Cues. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592. IEEE, 2011.

[6] Pierre Chatelain, Alexandre Krupa, and Maud Marchal. Real-time needle detection and tracking using a visually servoed 3D ultrasound probe. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1676–1681. IEEE, 2013.

[7] James G Fujimoto, Costas Pitris, Stephen A Boppart, and Mark E Brezinski. Optical Coherence Tomography: An Emerging Technology for Biomedical Imaging and Optical Biopsy. *Neoplasia*, 2(1-2):9–25, 2000.

[8] Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically Stable Sampling for the ICP Algorithm. In *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM 2003)*, pages 260–267. IEEE, 2003.

[9] Dirk Holz, Alexandru E Ichim, Federico Tombari, Radu B Rusu, and Sven Behnke. Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D. *IEEE Robotics & Automation Magazine*, 22(4):110–124, 2015.

[10] Yeqing Li, Chen Chen, Xiaolei Huang, and Junzhou Huang. Instrument Tracking via Online Learning in Retinal Microsurgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 464–471. Springer, 2014.

[11] Chrysi Papalazarou, Peter MJ Rongen, and Peter HN de With. Surgical needle reconstruction using small-angle multi-view x-ray. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 4193–4196. IEEE, 2010.

[12] Wu Qiu, Mingyue Ding, and Ming Yuchi. Needle Segmentation Using 3D Quick Randomized Hough Transform. In *Intelligent Networks and Intelligent Systems, 2008. ICINIS'08. First International Conference on*, pages 449–452. IEEE, 2008.

[13] Rogério Richa, Marcin Balicki, Raphael Sznitman, Eric Meisner, Russell Taylor, and Gregory Hager. Vision-based Proximity Detection in Retinal Surgery. *IEEE transactions on biomedical engineering*, 59(8):2291–2301, 2012.

[14] Nicola Rieke, David Joseph Tan, Mohamed Alsheakhali, Federico Tombari, Chiara Amat di San Filippo, Vasileios Belagiannis, Abouzar Eslami, and Nassir Navab. Surgical Tool Tracking and Pose Estimation in Retinal Microsurgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 266–273. Springer, 2015.

[15] Víctor Rodríguez. PCL/OpenNI tutorial, 2015. `http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors)`, (accessed 23-September-2017).

[16] Raphael Sznitman, Karim Ali, Rogério Richa, Russell H Taylor, Gregory D Hager, and Pascal Fua. Data-Driven Visual Tracking in Retinal Microsurgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 568–575. Springer, 2012.

[17] David Turbert. OPTICAL COHERENCE TOMOGRAPHY, 2017. `https://www.aao.org/eye-health/treatments/what-does-optical-coherence-tomography-diagnose`, (accessed 13-September-2017).

[18] Zhengyou Zhang. *Iterative Point Matching for Registration of Free-Form Curves*. PhD thesis, Inria, 1992.

[19] Mingchuan Zhou, Kai Huang, Abouzar Eslami, Daniel Zapp, Haotian Lin, Mathias Maier, Chris P. Lohmann, Alois Knoll, and M. Ali Nasseri. Beveled Needle Position and Pose Estimation based on Optical Coherence Tomography in Ophthalmic Microsurgery. In *2017 IEEE International Conference on Robotics and Biomimetics*, 2017.