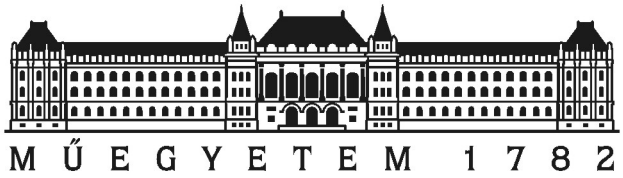


Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

Curvature adaptive 3D scanning method and its 2D simulation

Scientific Students' Associations report

Budapest, 2017

Author:
Adrián MEZEI

Supervisor:
Tibor KOVÁCS, Ph.D.

Contents

Abstract

An ideal motor movement configuration is being developed for a special 3D laser scanner which uses active triangulation. The result of a scanning process is a point cloud, which may have missing parts. These can appear for example where a concave part of the scanned object is hidden by some other part of the object itself. These missing parts can be detected, and the scanner can be reconfigured for the following scan.

In order to achieve a better scan, the transformations of the scanner's motors will be adjusted in such a way that these missing parts will be seen from a better angle. These transformations can be calculated from the results of the previous scans. The point cloud is analyzed, for example, normal vectors are calculated for the points, and the transformations for the following scan are calculated. These transformations are created so that it is the most likely to scan every part of the object based on the previously scanned point cloud.

The calculations also contain a list of points that must be visible from each scanned point (for example the camera and the laser) and it also contains an ideal point that is used for optimization purposes.

This paper contains the details of the transformation calculations, a simulator that is created for testing the calculations in 2D, and numerical and visual representation of the results with this simulator.

This method will be used for automated scans, where previous scans of the same object will make it possible for the scanner to create a better and better scanning configuration.

Hungarian abstract

Görbületkövető 3D szkennelési módszer és ennek 2D szimulációja

Egy ideális motormozgást konfiguráló modul került kifejlesztésre egy speciális 3D lézer szkennerhez, mely aktív triangulációval működik. Egy szkennelési folyamat eredménye egy pontfelhő, melynek lehetnek hiányos részei. Ezek például a szkennelt objektum olyan konkáv részeinél fodrulhatnak elő, ahol a tárgy egyik része kitakarja egy másik részét. Ezeket a hiányzó részeket detektálni lehet, majd a szkennert újra lehet konfigurálni egy következő szkenneléshez.

Egy jobb szkennelés eléréséhez a szkenner motorjainak mozgását előállító transzformációk úgy vannak tervezve, hogy a hiányzó részek jobb szögből látszódjanak. Az új transzformációkat a korábbi szkennelések eredményeiből lehet kiszámolni. A pontfelhő részletesen elemzésre kerül, például normálvektorokat kell számítani, és ezek segítségével a következő transzformáció kiszámíthtó. Ezek a transzformációk úgy vannak kialakítva, hogy a korábbi szkennelések alapján a legvalószínűbb legyen, hogy a szkennelt tárgy minden része látszódni fog.

A számítások tartalmaznak továbbá olyan pontokat, melyeknek mindenképpen látszódni kell az éppen szkennelt pontból (ilyen például a kamera és a lézer), valamint használ egy ideális pontot, ami az optimalizáció alapját képzi.

Részletesen bemutatásra kerülnek a transzformációk kiszámításának lépései, egy 2D szimulátor, mely a számítások ellenőrzéséhez, megjelenítéséhez és teszteléshez készült, valamint az eredmények numerikus és vizuális reprezentálása.

A módszer automatikus szkennelésekhez lesz használva, ahol az adott objektum korábbi szkennelései alapján egyre jobb és jobb szkennelést fog tudni készíteni a szkenner.

1 Introduction

1.1 Context

A 3D scanner [7] is being developed for which several software modules are needed. Different parts are created by different members of the team, which are integrated into a common project. The scanner creates a point cloud from each scan – as a result of several steps of the scanning pipeline. After the analyzation of this point cloud, further scans may be needed (because for example missing parts). Usually, it can be said that a general transformation setup (for example a 360° rotation) will provide an approximation of the geometry of the object, but there may be much better ways to scan it. In this paper, the steps of this optimization process are described in detail.

1.2 Problem statement

This scanner now has only manual transformation setup interface, where a fixed rotation and translation can be provided. The scanner will perform these transformations and create images with a given frequency. These fixed transformations and this frequency cannot adapt to the surface of the scanned object. This will usually result in such point clouds where the points are unevenly distributed along the scanned surface.

The goal is to create a scanning method that can adapt to the geometry of the scanned object and that can be used by the scanner to create a better scan.

It will also be a task to implement some changes to the scanner software so that it can use this method but this is not included in this paper.

1.3 Case study

A simple 360° pure rotational transformation is used as a general scanning approach. Usually, a few hundred images are created during a scan like this. This was the case when the scan of Figure 1 was created. In this case, 360 images were created, so there is an image from every whole angle of the object.

The object was placed on the scanner so that its vertical centre was approximately at the centre of rotation. A triangulation and normal vector calculation was performed on the point cloud and as a result, a shaded model can be seen in Figure 1. The areas and perimeters of the triangles are also calculated from this

Figure 1: Triangle mesh of a 360° pure rotational scan

triangulation for further analyzation. The triangles with too big perimeter are inadequate since there is not enough information from that region. These parts are highlighted in Figure 1 and Figure 2. The triangle mesh in Figure 1 contains about 120,000 triangles and their circumferences are represented by Figure 2.

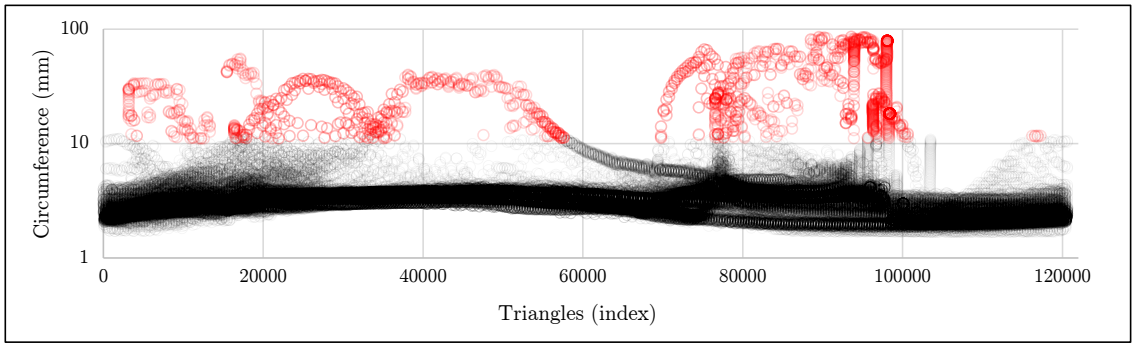


Figure 2: Distribution of the triangle circumferences of the triangle mesh in Figure 1 on a logarithmic scale

So as it was expected, some parts could not be scanned properly with this setup. It can be seen from Figure 2 that most of the triangles are accepted based on their circumferences (98.5%), but there are a few that are too big.

2 Background

2.1 Scanner hardware

The laser scanner developed to this task has two degrees of freedom. A rotational one around a vertical axis and a translational one about a horizontal axis. Combinations of the possible transformations provide a good opportunity to scan a quite wide range of objects (of course heavily concave parts cannot be scanned). The hardware, which can be seen in Figure 3, has six very important components: the controller, the laser, the camera, the two step motors and the turntable.

Figure 3: The 3D scanner with the laser, the camera, the two motors, the controller and the turntable

2.2 Scanning workflow

First of all, the scanner is calibrated with a checkerboard at a fixed position on the turntable, and a homography matrix is generated from the image of the checkerboard. After this point, the scanner always knows the exact position of the turntable and the camera, since the camera is fixed, and the movement of the turntable is controlled by the step motors, which are controlled by the computer.

The details of a scanning process are calculated on the computer, then these are sent to the controller. The two step motors of the scanner do the transformations: rotate and translate the turntable, on which the scanned object is placed, in front of the camera and the laser. The laser emits a vertical line onto the scanned object, then the controller triggers the camera to take a picture of the object and finally the

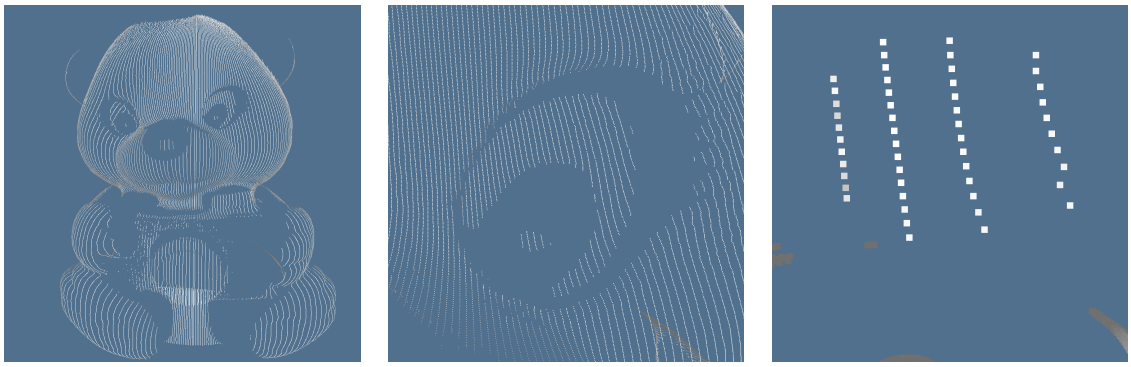
camera sends the image to the computer. The computer then makes further steps like image processing, line tracking, tessellation, mesh generation, error correction and finally the scanned point cloud is visually displayed.

2.3 Data structure handled by the scanner

As it was mentioned, the scanning pipeline consists of several steps. At a certain point, the line tracker module provides a list of consecutive points from a single image of the laser beam illuminated object and these are going to be stored.

The smallest data structure element is a point. The consecutive list of points provided by the line tracker is called a stripe. The result of a scanning transformation, which can be translational and rotational at the same time, consists of several stripes. The group of these stripes is called a batch. A scanning process consists of several transformations and the group of batches (created by the transformations) form a cloud.

This data structure is created by the line tracker and it is provided stripe by stripe when they are ready.



(a) The whole bear (b) The left eye (c) The pupil

Figure 4: A bear with a camera and its left eye zoomed

These stripes can be processed one by one with the mesh generator. It is also important that the points of these stripes are already at the correct position, which is calculated previously from the homography matrix, so the stripes do not need to be moved anywhere.

Finally, there can be more scanned objects in the scene each represented by a cloud, these are called a project.

3 Solution requirements

Given the hardware with one rotational and one translational degree of freedom, the laser, and the camera. It is also given that the hardware can only perform these two transformations and the position of the laser and the camera is basically fixed but they may be changed in the future. The setup and the results of the previous scans are stored, and these are accessible later as well, so they can be used for planning the next scan.

A scanning plan is needed for the scanner to perform its next scan. Considering the most general case, this is an ordered list in which all element contains a translational and a rotational data. An example of such transformation plan can be seen in Table 1.

Index	Rotation (degree)	Translation (mm)
1	0	0
2	36	10
3	72	20
4	108	30
5	144	40
6	180	50
7	216	40
8	252	30
9	288	20
10	324	10

Table 1: An example scanning plan

The execution of the scanning plan is the task of the scanner. First, it must translate and rotate its turntable to the required position then trigger the camera to take a picture. Then the camera sends the captured image to the computer where the image processing is done. When the line tracking is done, they form a stripe, and this is added to the current batch of the point cloud. Finally, this must be repeated for all entries of the scanning plan. The pseudocode of this process is given by Algorithm 1. It depends on the software of the controller how fast the motors are moved. The only requirement is to reach all the desired positions.

A further requirement is that the created stripes must be equally distributed around the scanned object as far as possible. It is not a good solution to have all the stripes on one side and nothing on the other side of the scanned object. This is of course not possible at the first scanning since the scanner does not know anything

about the scanned object, but later all the previous scans of the object are available. Furthermore, sometimes different parts of an object are desired to be scanned in a more detailed way, which means that such requirements can be additional parameters of the scanning plan creation process. Such detail requirements can be given as a function of the position on the perimeter.

Algorithm 1 Execution of a transformation plan

1: **Input:** $scanningPlan$ contains the transformations to execute
2: **Output:** $batch$ is the structure that contains the stripes
3: **for all** $transformation \in scanningPlan$ **do**
4: $\quad turnTable.setRotation(transformation.rotation)$
5: $\quad turnTable.setTranslation(transformation.translation)$
6: $\quad image = camera.createImage()$
7: $\quad stripe = linetracker.track(image)$
8: $\quad batch.add(stripe)$
9: **end for**

Furthermore, the first few iterations might not result in a perfect scan either, so this iterative method must be continued. It is very important to create a module for analyzing these results and evaluating the stopping criteria based on some previously provided parameters. These parameters can be for example:

- the maximum number of iterations that can be performed
- the maximum number of stripes that can be created
- the maximum distance between two stripes
- the maximum available time for a scanning process (a detailed scanning process may last longer)

4 Reducing the problem

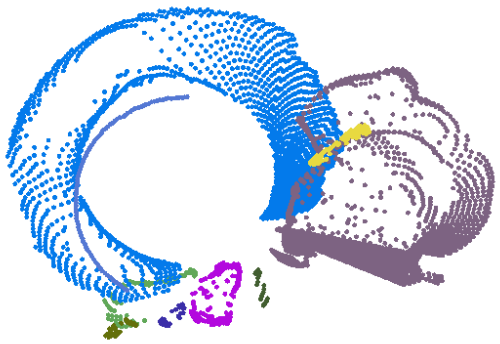
Assuming that there are some previously scanned batches of the object (this can be a simple 360° rotational scan), the parts to be repaired can be calculated. This will typically be a point cloud which can be divided into connected groups. These are called error groups. This basically means, that certain parts of the object should be repaired, and these parts are distinguished from each other.

Visually, an error group means a piece of the surface that should be repaired, and this can be described by a list of points [12]. As an example, the error groups of a scanned object can be seen in Figure 5a.

The point cloud now means the group of those points that describe one error group. This may be the whole point cloud if there aren't any distinguished error groups. To repair this point cloud, transformations are needed for the scanner, with which it can scan the object again. To calculate these transformations, the analyzation of this point cloud is needed.

4.1 Perpendicular projection to the horizontal plane

(a) The whole object



(b) The projected points of the error groups

Figure 5: A scanned object and its error groups projected to the horizontal plane

As the scanner can only translate the scanned objects along a horizontal line and rotate it around a vertical axis, the error groups are projected to the horizontal plane for further calculations. The vertical positions can be disregarded because the transformations can only be made in the horizontal plane. This means that

the vertical positions do not influence the transformations. Now the projected point cloud can be visualised as a planar set of points. This can be seen in Figure 5, where Figure 5a is the original 3D mesh and Figure 5b shows the projected point clouds. These points are to be approximated by a curve so that the scanner can schedule its next scan along this approximation.

The point cloud now means the group of those points that describe one error group. To repair this point cloud, a transformation plan is needed for the scanner, with which it can scan the object again. To calculate this, the analyzation of the point cloud is needed.

4.2 Curve approximation

There are several approaches for approximating a set of two-dimensional points with a line. For example, the least squares [3], moving least square, improved moving least squares [8] or the weighted least square methods. These are used in the iterative closest point algorithm [1]. This method always monotonically converges to the nearest local minimum of a mean-square distance metric.

This could be further improved by feature extraction [4, 16, 10], spline approximations [17, 5] or by identifying more 2D primitives. These features can be categorised and handled similarly.

Another approach is to calculate an average point for each stripe of the projected error group and this will represent that particular stripe. In this case, the projected error group will be represented by the same number of points as the number of stripes that were included in the error group. These points can now be connected, and the resulting polygon will be the approximating curve.

The two-dimensional point cloud that is to be repaired now represented by an approximating curve. This curve either represents an error group or the whole point cloud of the object if there are no error groups.

4.3 Scanning the approximating curve

Given the approximating curve that is to be rescanned, so the question is how this should be done. The turntable should somehow be transformed so that each part of the curve gets in front of the camera and the laser. Since the hardware has two degrees of freedom (the translation and the rotation), there are usually infinitely many combinations of these transformations that will be adequate for this.

4.3.1 Scanning a point of the approximating curve

A single point of the approximating curve can usually be scanned with infinitely many transformation setups. From the point of view of the laser, the best position to illuminate a point is where the normal vector of the point intersects the laser. In this case, it is the least probable to have something between the laser and the scanned point that can hide the laser beam. From the point of view of the camera, almost the same is true, since this is the least probable position to have something between the camera and the scanned point, that can hide the point from the camera.

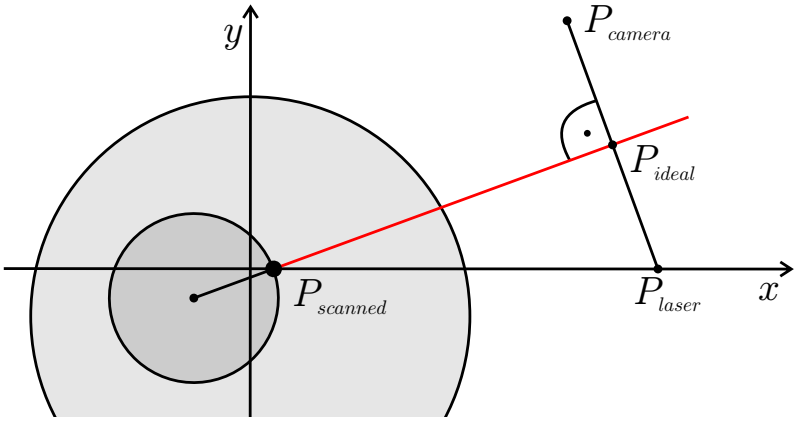


Figure 6: The ideal position for scanning $P_{scanned}$, which is a point of the approximating curve (this is now a circle)

Consequently, the best position for the scanned point would be such position from where its normal points in the direction of the camera and the laser at the same time. This is impossible, so an intermediate solution should be found. The solution is to place the actually scanned point in such a position, where

1. the line formed by the scanned point and its normal as a direction vector is the perpendicular bisector of the line segment formed by the camera and the laser and

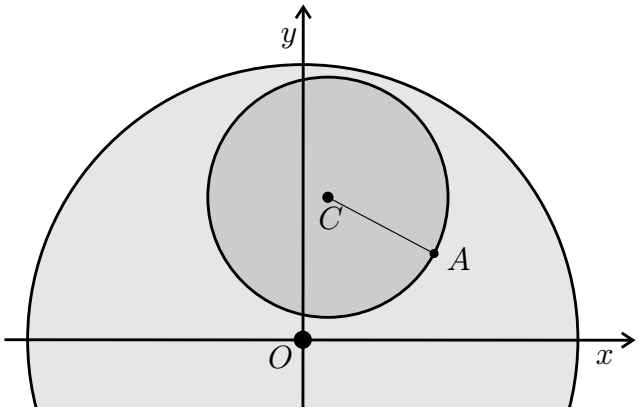
2. the normal vector of the scanned point points towards the point that is halfway between the camera and the laser (this means that the normal vector does not point in the opposite direction).

This can be seen in Figure 6 where the approximating curve is a circle, $P_{scanned}$ is the scanned point and P_{ideal} is the point that is halfway between the camera and the laser.

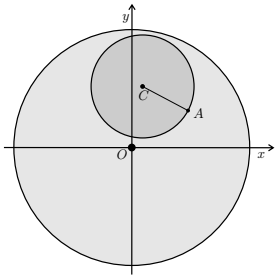
5 Ideal transformation for scanning a point

The ideal scanning position is now given, so the scanner must be transformed to this position and trigger the camera to create a picture. The question is how to achieve such position (such position can be seen in Figure 6).

A general point is given by its coordinates, which may be a calculated value from the parametric approximating curve. This can be seen in figure Figure 7. The scanner must perform a rotation and a translation to get to the ideal scanning position from this (Figure 6). If the proper rotation of this transformation is given then the translation can easily be calculated, since the rotated point only needs to be translated to the x-axis (so that the laser illuminates it).



(a) A zoomed picture



(b) The whole turntable

Figure 7: A general point which is to be scanned on the approximating curve

5.1 Translate A to the x-axis

The first step is to translate the turntable along the y-axis so that A lies on the x-axis, this translated point is denoted by A'. The length of this transformation is given by the y-coordinate of the original point A. The translated points are denoted by O', C' and A' and their corresponding y-coordinates are given by Eq. (1), Eq. (2) and Eq. (3) respectively. The result of this translation can be seen in Figure 8.

$$O'_y = -A_y \tag{1}$$

$$C'_y = C_y - A_y \tag{2}$$

$$A'_y = 0 \tag{3}$$

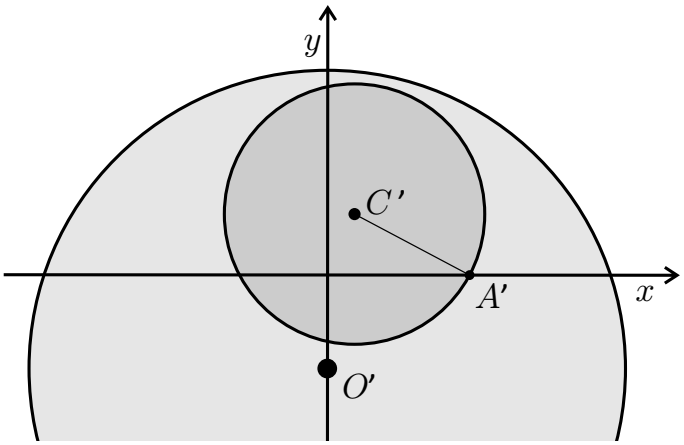


Figure 8: The turntable is translated so that A is on the x-axis (this is A')

5.2 Translate and rotate so that A' stays on the x-axis

The second step is to rotate and translate the turntable until C', A' and P_i are collinear while A' stays on the x-axis. To achieve this, the equation of the line formed by A' and P_i must be satisfied by point C'. In the following expressions, α in the subscript of a point means that the point is obtained by rotating the turntable by α and translating the turntable along the y-axis so that the image of A is on the x-axis. The equation of line A' and P_i after an α rotation about the centre of the turntable can be given by the point-slope formula Eq. (4).

$$y - y_1 = m(x - x_1) \tag{4}$$

To parametrize the formula, a few calculations should be made. The centre of the turntable O' stays on the y-axis (given by Eq. (5)) and the scanned point A' stays on the x-axis (given by Eq. (6)) after this α rotation.

$$O'_x = O'_{\alpha,x} = 0 \tag{5}$$

$$A'_{\alpha,y} = 0 \tag{6}$$

$O'_{\alpha,y}$ is calculated by rotating the $A' - O'$ point around the origin by α then the negative of the result's y-coordinate is taken. This is the distance that the turntable should be translated by along the y-axis so that A' stays on x-axis after the rotation (so $A'_{\alpha,y} = 0$). The result of the calculations is shown by Eq. (7).

$$O'_{\alpha,y} = -\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}\cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}A'_x - O'_x \\ A'_y - O'_y\end{bmatrix} =$$
$$= -\Big((A'_x - O'_x)\sin\alpha + (A'_y - O'_y)\cos\alpha\Big) =$$

$$= O'_y\cos\alpha - A'_x\sin\alpha \tag{7}$$

$A'_{\alpha,x}$ is obtained by taking the x-coordinate of the $A' - O'$ point after rotating it by α around the origin. The result of the calculations is shown by Eq. (8).

$$A'_{\alpha,x} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}A'_x - O'_x \\ A'_y - O'_y\end{bmatrix} =$$
$$= (A'_x - O'_x)\cos\alpha - (A'_y - O'_y)\sin\alpha =$$

$$= A'_x\cos\alpha + O'_y\sin\alpha \tag{8}$$

C'_α is obtained by rotating C' around O' by α then translating it by $O'_\alpha - O'$. Rotating C' around O' is obtained by rotating the $C' - O'$ point around the origin by α and translating it by O'. The result of the calculations is shown by Eq. (9).

$$C'_\alpha = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}C'_x - O'_x \\ C'_y - O'_y\end{bmatrix} + \begin{bmatrix}O'_x \\ O'_y\end{bmatrix} + \begin{bmatrix}O'_{\alpha,x} - O'_x \\ O'_{\alpha,y} - O'_y\end{bmatrix} =$$
$$= \begin{bmatrix}(C'_x - O'_x)\cos\alpha - (C'_y - O'_y)\sin\alpha + O'_x + O'_{\alpha,x} - O'_x \\ (C'_x - O'_x)\sin\alpha + (C'_y - O'_y)\cos\alpha + O'_y + O'_{\alpha,y} - O'_y\end{bmatrix} =$$
$$= \begin{bmatrix}C'_x\cos\alpha - (C'_y - O'_y)\sin\alpha \\ C'_x\sin\alpha + (C'_y - O'_y)\cos\alpha + O'_{\alpha,y}\end{bmatrix} \tag{9}$$

With the calculated values O'_α, A'_α, C'_α and the ideal point P_i, the point-slope formula can be parametrized in the following way. The slope m of the point-slope formula is expressed from points P_i and A'_α (Eq. (10)).

$$m = \frac{P_{i,y} - A'_{\alpha,y}}{P_{i,x} - A'_{\alpha,x}} = \frac{P_{i,y}}{P_{i,x} - (A'_x\cos\alpha + O'_y\sin\alpha)} \tag{10}$$

P_i is used as the point of the point-slope formula, so the x_1 and y_1 parameters are substituted by $P_{i,x}$ and $P_{i,y}$ respectively (Eq. (11), Eq. (12)).

$$x_1 = P_{i,x} \tag{11}$$

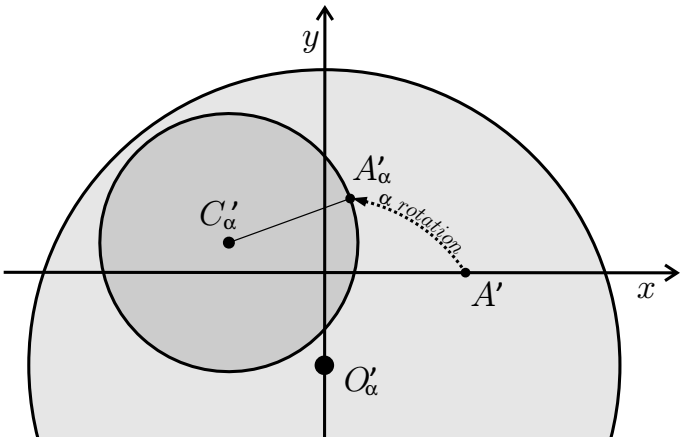


Figure 9: The turntable is rotated so that after a translation, the ideal position is achieved, this can be seen in Figure 10

$$y_1 = P_{i,y} \tag{12}$$

Thus the point-slope formula (Eq. (4)) can be parametrized in the following way shown by Eq. (13).

$$y - P_{i,y} = \frac{P_{i,y}}{P_{i,x} - (A'_x \cos\alpha + O'_y \sin\alpha)}(x - P_{i,x}) \tag{13}$$

This formula should be satisfied by point C'_α, so the x and y parameters are substituted by $C'_{\alpha,x}$ and $C'_{\alpha,y}$ respectively (Eq. (14), Eq. (15)).

$$x = C'_{\alpha,x} \tag{14}$$

$$y = C'_{\alpha,y} \tag{15}$$

Thus the final form of the point-slope formula with the parameters substituted can be expressed. The result of the calculations is shown by Eq. (16).

$$C'_{\alpha,y} - P_{i,y} = \frac{P_{i,y}}{P_{i,x} - (A'_x \cos\alpha + O'_y \sin\alpha)}(C'_{\alpha,x} - P_{i,x})$$

$$C'_{\alpha,y} - P_{i,y} = \frac{P_{i,y}(C'_{\alpha,x} - P_{i,x})}{P_{i,x} - (A'_x \cos\alpha + O'_y \sin\alpha)} \tag{16}$$

This expression contains only constants and the calculated $C'_{\alpha,x}$ and $C'_{\alpha,y}$, which can also be substituted. The result of the calculations is shown by Eq. (17).

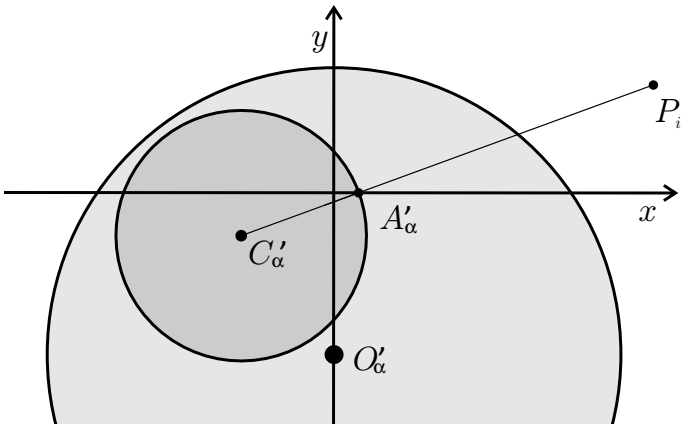


Figure 10: The turntable is rotated and translated so that C'_α, A'_α, and P_i are collinear

$$C'_x \sin\alpha + (C'_y - O'_y)\cos\alpha + O'_{\alpha,y} - P_{i,y} =$$

$$= \frac{P_{i,y}(C'_x \cos\alpha - (C'_y - O'_y)\sin\alpha - P_{i,x})}{P_{i,x} - (A'_x \cos\alpha + O'_y \sin\alpha)} \tag{17}$$

This expression contains only constants and the calculated $O'_{\alpha,y}$, which can also be substituted. The result of the substitution is shown by Eq. (18).

$$C'_x \sin\alpha + (C'_y - O'_y)\cos\alpha + O'_y \cos\alpha - A'_x \sin\alpha - P_{i,y} =$$

$$= \frac{P_{i,y}(C'_x \cos\alpha - (C'_y - O'_y)\sin\alpha - P_{i,x})}{P_{i,x} - (A'_x \cos\alpha + O'_y \sin\alpha)} \tag{18}$$

The left-hand side can be simplified since the $O'_y \cos\alpha$ terms cancel out, thus the final formula is Eq. (19).

$$C'_x \sin\alpha + C'_y \cos\alpha - A'_x \sin\alpha - P_{i,y} =$$

$$= \frac{P_{i,y}(C'_x \cos\alpha - (C'_y - O'_y)\sin\alpha - P_{i,x})}{P_{i,x} - (A'_x \cos\alpha + O'_y \sin\alpha)} \tag{19}$$

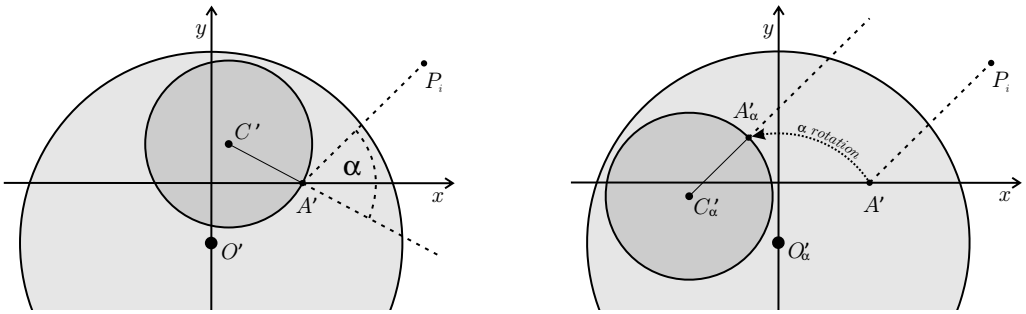
Since all the parameters are known except α, this equation can be solved. The equation is implicit and α seems to be very hard to express. The final scanner position is shown by Figure 10.

5.2.1 Rotation angle calculation

The calculated implicit equation in Section 5.2 (Eq. (19)) is going to be used frequently so its evaluation must be fast.

The following proposed solution uses an iterative method for finding the solution for Eq. (19). An angular error is calculated during the iterations that shows the required modification for the next iteration. This angular error is the counter-clockwise directed angle from line $C'A'$ to $A'P_i$ and it is always just an estimation of the required angle. This angle can be seen in Figure 11a.

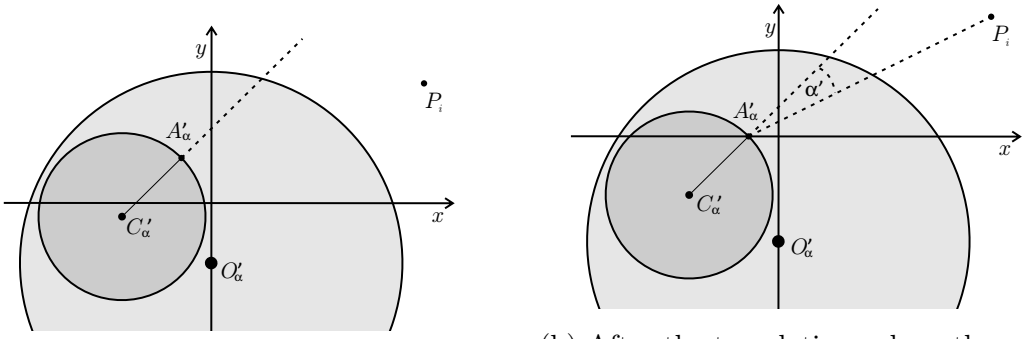
A rotation by this estimated angular error usually moves point A' from the x-axis, so in this case, the next step is to perform a translation so that A'_α gets to the x-axis again. The amount of this translation is simply the y coordinate of the rotated A'_α point. This translation can be seen in Figure 12 where the rotated position is shown by Figure 12a and the translated is shown by Figure 12b. The estimated angular error for the next iteration can now be seen in Figure 12b as well.



(a) The estimated angular error

(b) The rotation

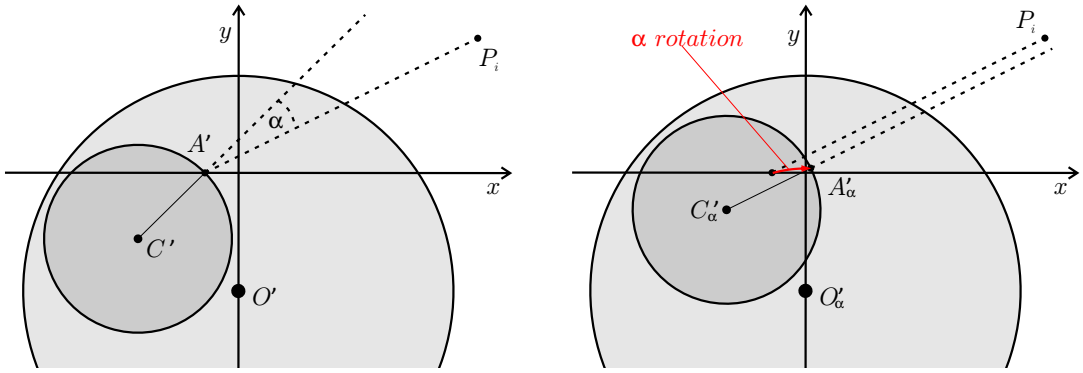
Figure 11: A rotation by the estimated angular error during the first iteration



(a) After the rotation

(b) After the translation, where the new estimated angular error α' can be seen

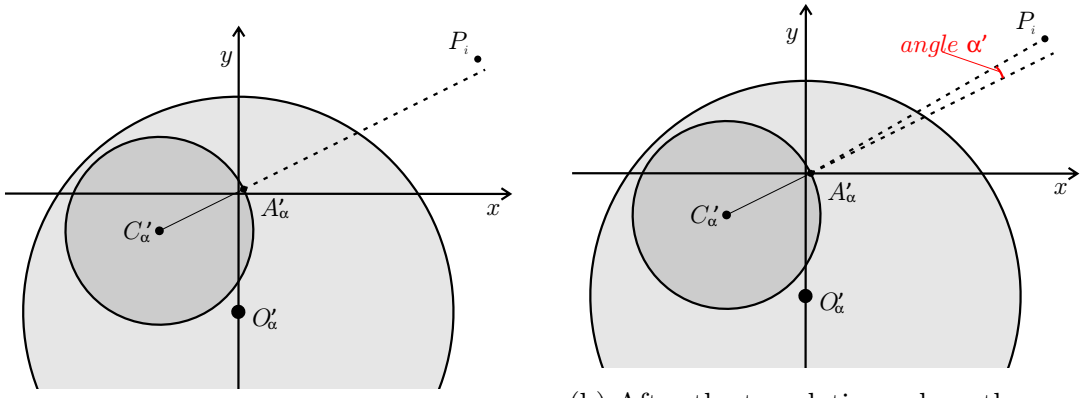
Figure 12: The translation during the first iteration, so that A'_α is on the x-axis again and the new estimated angular error can also be seen



(a) The estimated angular error

(b) The rotation

Figure 13: A rotation by the estimated angular error during the second iteration



(a) After the rotation

(b) After the translation, where the new estimated angular error α' can be seen

Figure 14: The translation during the second iteration, so that A'_α is on the x-axis again and the new estimated angular error can also be seen

In the beginning, the estimated angular error (this can be seen in Figure 11a) is 72.473°. After the first iteration, this angular error (which can be seen in Figure 12b) is only −18.233°. After the second iteration, this angle is below 4°. For a better understanding of the algorithm, the second iteration is also visualized. This can be seen in Figure 13 and Figure 14.

This process converges very fast, which means that it can be used for real-time calculations as well. After fourteen iterations the error is less than 10^{-8} degrees which is more than enough for positioning the turntable to the right position. The errors until the fourteenth iteration are shown by Table 2 and these values are also visualized on a logarithmic scale in Figure 15.

It can also be seen in Table 2 that the sign of the angular error is alternating. This means that each calculated angular error is an upper estimation of the required rotation.

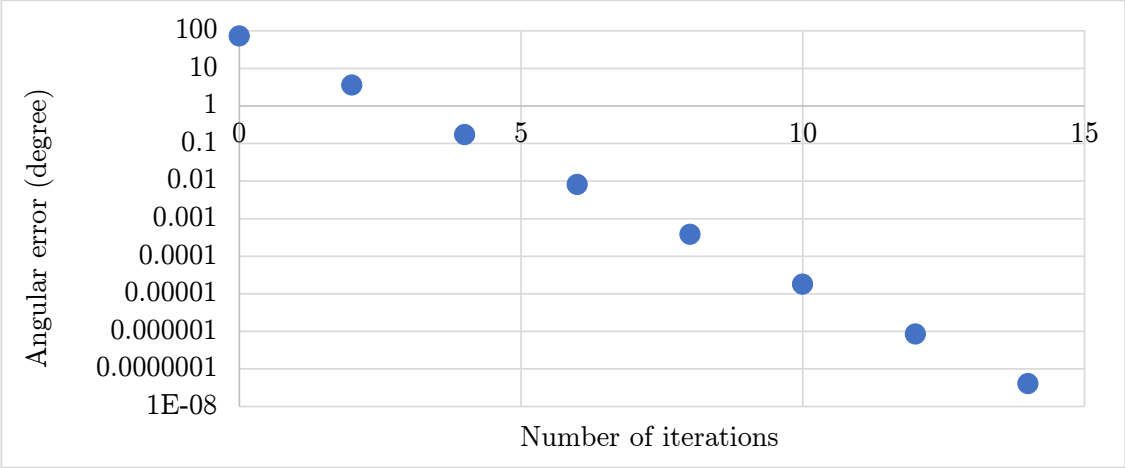


Figure 15: The absolute values of the estimated angular errors represented on logarithmic scale

Iteration number	Angular error (degree)
0	72.47264769
1	-18.23258286
2	3.623134483
3	-0.799097069
4	0.172897496
5	-0.037570399
6	0.008156432
7	-0.001771097
8	3.85E-04
9	-8.35E-05
10	1.81E-05
11	-3.94E-06
12	8.55E-07
13	-1.86E-07
14	4.03E-08

Table 2: The estimated angular errors after the iterations

5.2.2 Java implementation of the rotation angle calculation

The provided Java class (Code 1) calculates the required angle for positioning a point to the ideal scanning position. This solves the implicit equation presented in Section 5.2 with the method described in Section 5.2.1.

Code 1: The class that calculates the required angle for the ideal position

```
1  // RotationCalculator.java
2  public class RotationCalculator {
3     public static double calculateRotationDeg(double Ax, double
          Cx, double Cy, double Oy, double Pix, double Piy, double
          maxAngleErrorDeg) {
4        double Oay, Aax, Cax, Cay;
5        double errorAngle = Math.PI;
6        double maxErrorAngle = maxAngleErrorDeg / 180.0 * Math.PI;
7        double angle = 0;
8
9        while(Math.abs(errorAngle) > maxErrorAngle) {
10          // Calculated parameters
11          Oay = Oy * Math.cos(angle) - Ax * Math.sin(angle);
12          Aax = Ax * Math.cos(angle) + Oy * Math.sin(angle);
13          Cax = Cx * Math.cos(angle) - (Cy - Oy) * Math.sin(angle);
14          Cay = Cx * Math.sin(angle) + (Cy - Oy) * Math.cos(angle)
               + Oay;
15          // Calculate the error by angle PiAa and AaCa
16          double anglePiAa = Math.atan2(Piy, Pix - Aax);
17          double angleAaCa = Math.atan2(-Cay, Aax - Cax);
18          errorAngle = anglePiAa - angleAaCa;
19
20          angle += errorAngle;
21       }
22       return Math.toDegrees(angle);
23    }
24 }
```

The function in Code 1 receives the parameters required for solving the equation. The values of these parameters are based on the position where the turntable is shifted so that A' is on the x-axis. This position can be seen in Figure 8. The function also has a maximum error parameter, which is an angle in degrees that the

error of the resulting angle cannot exceed. The variables in the function are named by omitting the $'$ symbols so A' is called A for example.

The algorithm was tested with some data shown by Table 3. These are the real values of Figure 11a and the errors at the iterations are shown by Table 2.

Variable name	Value
Ax	22.986
Cx	4.406
Cy	9.873
Oy	-15.108
Pix	53.72
Piy	30.189
maxAngleErrorDeg	0.0000001

Table 3: The test parameter values

Code 2: The evaluation of the original formula Eq. (19)

```
1  if(Math.abs(errorAngle) <= maxErrorAngle){
2     // Parameters of the original formula
3     double left = Cx * Math.sin(angle) + Cy * Math.cos(angle) -
          Ax * Math.sin(angle) - Piy;
4     double right = (Piy * (Cx * Math.cos(angle) - (Cy - Oy) *
          Math.sin(angle) - Pix)) / (Pix - (Ax * Math.cos(angle) +
          Oy * Math.sin(angle)));
5
6     System.out.println("Angle: " + Math.toDegrees(angle));
7     System.out.println("Error: " + Math.toDegrees(errorAngle));
8     System.out.println("Formula left-hand side: " + left);
9     System.out.println("Formula right-hand side: " + right);
10 }
```

A debugging code snippet Code 2 is inserted to the 15th line of Code 1 for checking. This evaluates the left-hand side and right-hand side of the original formula Eq. (19) and prints the results, which are:

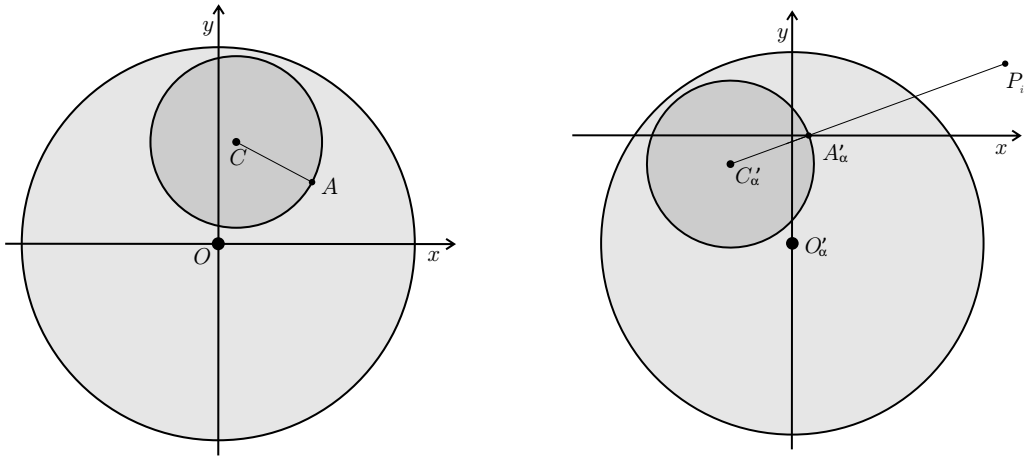
```
Angle:  57.20613059610286
Error:  4.030173709079121E-8
Formula left-hand side:  -40.460407189504586
Formula right-hand side:  -40.46040720646222
```

5.3 Translate A to the x-axis again

The required angle is already calculated in Section 5.2. After this rotation, the last required transformation is to translate the turntable so that the scanned point gets to the x-axis.

5.4 Evaluation

A method was proposed for positioning the turntable of the scanner in a position that is generally ideal both from the point of view of the camera and the laser for scanning a given point. This basically requires three steps, two of which are trivial translations and the third is the rotation that is calculated in Section 5.2.



(a) A general position

(b) The final ideal position

Figure 16: The method transforms the turntable so that the scanned point gets to the ideal scanning position

This method is ideal if a single point is to be scanned or if the approximating curve does not have too concave parts. On the other hand, there may be more difficult situations as well if the approximating curve is more complex. The shadow effect, for example, may also appear and hide the scanned point from either the laser or the camera. Some of these cases are impossible to be scanned with this hardware (described in Section 2.1) but a few of these can be handled in some way. These are described in Section 6.

6 Scanning some more concave parts

In this section, a method is introduced for scanning certain concave parts of the approximating curve that cannot be scanned with the method described in Section 5. It is obvious that certain parts of come approximating curves cannot be scanned with the hardware described in Section 2.1, an example of this is shown by Figure 17.

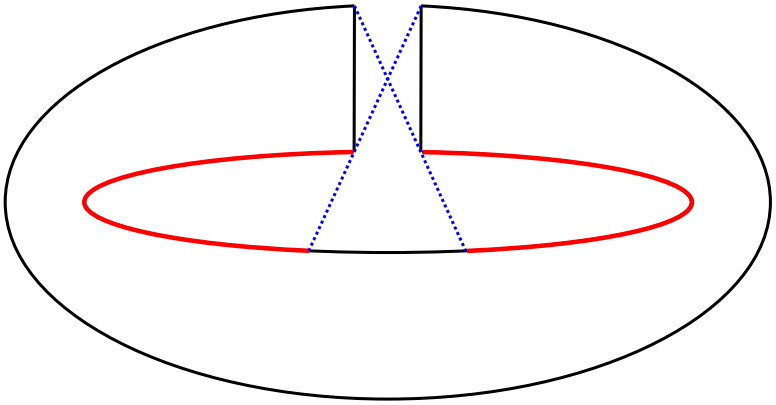
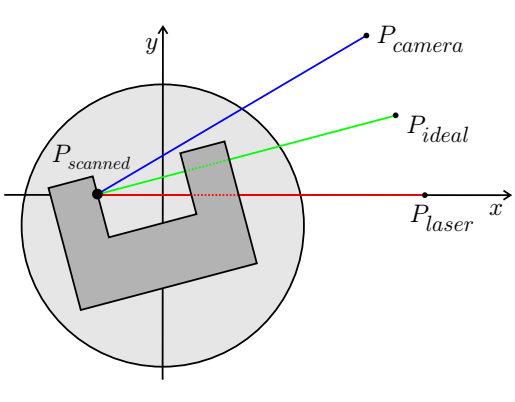
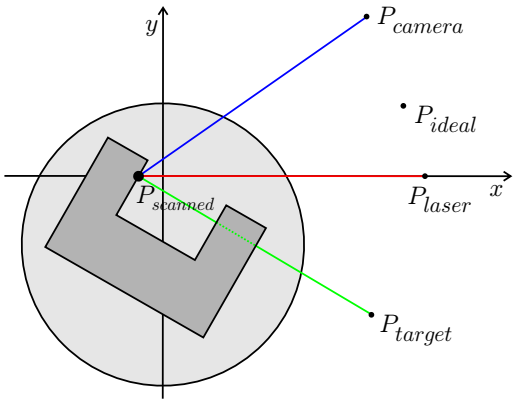


Figure 17: The highlighted parts are obviously impossible to be scanned

The interesting cases are those, that can still be scanned with the hardware, but not with the method of Section 5. An example for this position is shown by Figure 18.



(a) The ideal position based on Section 5



(b) A possible position for scanning

Figure 18: A point that cannot be scanned with the method of Section 5, but it can be scanned in some other ways

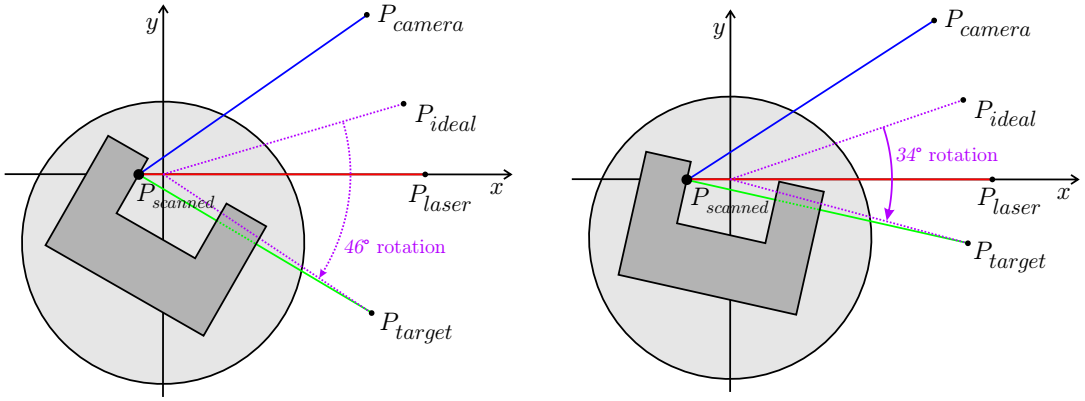
It can be seen in Figure 18a that the normal vector of point $P_{scanned}$ points in the direction of the ideal point P_{ideal} and $P_{scanned}$ is on the x-axis. This satisfies the two conditions of the ideal position stated in Section 4.3.1 but the laser cannot illuminate the scanned point. This means that this point cannot be scanned this way, while Figure 18b shows a possible position for scanning.

6.1 Closest angle to ideal that is still scannable

It can be seen in Figure 18 that there are other possible transformations as well from which $P_{scanned}$ can be scanned. All of these are other than the ideal transformation of Section 5. From these cases, the transformation closest to the ideal is the best.

All possible angles from where $P_{scanned}$ can be scanned is to be calculated first. These can be obtained by rotating the ideal point around the origin and calculating the corresponding scanning angle with the method described in Section 5.1 using Code 2. The rotated ideal point is called P_{target}, which can also be seen in Figure 18b. The best angle from these is the one for which the rotation angle of the ideal point is the smallest, thus the scanner transformation is as close to the ideal transformation as possible.

The transformation applied in Figure 18b is a good transformation since $P_{scanned}$ can be scanned, but this is not the best. The transformation that is as close to the ideal transformation as possible can be seen in Figure 19b.



(a) A possible position for scanning

(b) The best position for scanning

Figure 19: The best scanning position can be achieved by rotating the ideal point around the origin by 34° in this case

6.1.1 Determining if a point can be scanned from a position

A point can be scanned from a certain position if it can be seen from both the position of the camera and laser. Otherwise, if there is something between the camera and the scanned point then the camera cannot see the laser beam, if there is something between the laser and the scanned point then the laser cannot illuminate the scanned point.

When a point to be scanned and transformation to be applied is given then it

should be calculated whether these conditions are true or false. An approximating curve is also provided which is basically the current knowledge about the shape of the scanned object on which the scanned point lies. From these, two lines can be created. The camera line from P_{camera} to $P_{scanned}$ and the laser line from P_{laser} to $P_{scanned}$. These lines can be seen in Figure 18 and Figure 19.

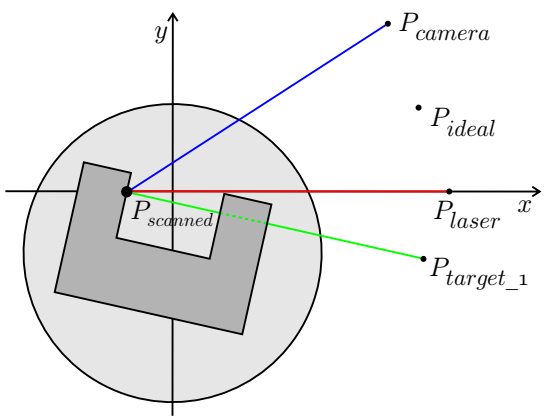
For the camera, the closest intersection to P_{camera} of the approximating curve and the camera line is to be calculated. If this intersection is not $P_{scanned}$ then the camera cannot see the scanned point, thus this point cannot be scanned from this position. The very same process should be done for the laser. If the $P_{scanned}$ is the first intersection in both cases, then the point can be scanned from this position.

6.1.2 Considering a safety angle

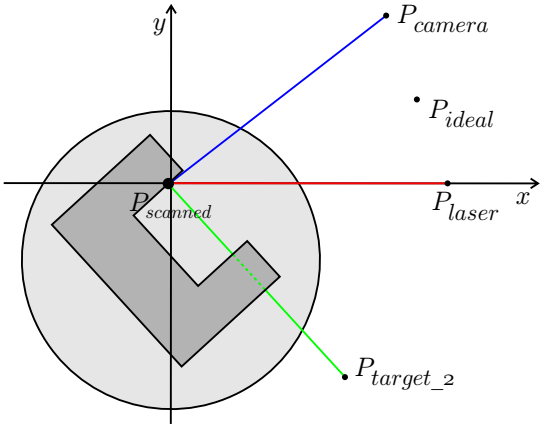
An important fact must not be forgotten, that the best transformation described in Section 6.1 is calculated based on an approximating curve. This curve may be very detailed and can approximate the object very well, but it is always the averaged cross-section of the scanned object. This means that even if the approximating curve is the best possible approximation, some parts of the scanned object may have somewhat different cross-sections.

If the best transformation is calculated, which can be seen in Figure 19b, either the laser line or the camera line may pass very close to some other part of the object. This phenomenon can be observed in Figure 19b where the laser almost intersects the corner of the object.

In order to avoid this problem, the calculated best transformation should be modified by a safety angle. The best magnitude of this safety angle may vary between different objects. A larger safety angle will provide a smaller probability to have something between the scanned point and the camera or the laser. On the other hand, a smaller safety angle also has advantages. With a smaller safety angle, more concave parts might be scanned.

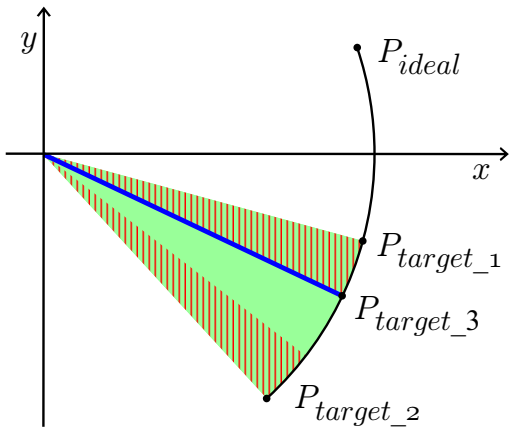


(a) The laser line is close to the object

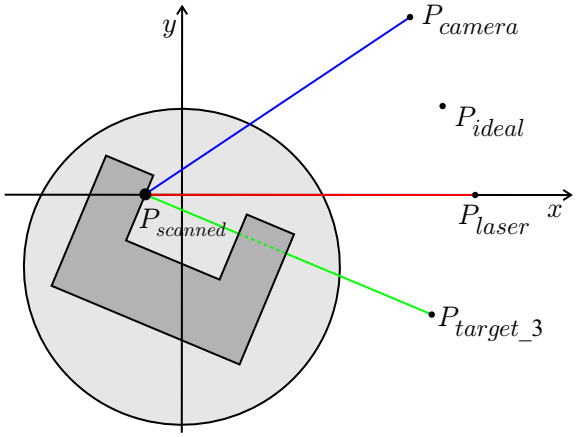


(b) The camera line is close to the object

Figure 20: The extremities from where $P_{scanned}$ can still be scanned



(a) The region from where $P_{scanned}$ can be scanned and the safety regions



(b) The best possible transformation with 10° safety angle

7 Creating a scanning plan

Based on the methods described in Section 5 and Section 6, a completely automated surface adaptive scanning process can be created. At first, there is no information about the geometry of the scanned object. It is to be scanned first somehow, then based on the results, a new scanning configuration can be created.

7.1 Iterative scanning method

The first scanning is a 360° rotational scan without any translations. The translational data of the scanning plan, therefore, consists of only zeroes. As a general approach, this scanning triggers the camera to take a photo at each whole angle, but this can be modified to be more or less frequent.

Index	Rotation (degree)	Translation (mm)
1	0	0
2	1	0
3	2	0
...	...	...
358	357	0
359	358	0
360	359	0

Table 4: The very first scanning plan for each scanning

The next scanning plan is going to be calculated based on the result of the previous scan. The result of a scanning is a batch (Section 2.3), which consists of stripes. These stripes are projected onto the $y = 0$ plane and an averaged point is calculated to each stripe that will represent the stripe. This provides a list of two-dimensional points. Then the normal vectors to the points are either calculated from the two-dimensional list of points or they can also be calculated by averaging the normal vectors of the points of the corresponding stripe. This second method is only possible if the normal vectors of the points of the scanned batch were or can be calculated. A few methods for calculating them are mentioned in Section 7.1.1. Finally, the elements of the transformation plan are calculated from these point by the methods described in Section 5 and 6.

The further scanning plans can always be calculated from the result of the previous scan. Iterating this process will generate a better and better scanning plan.

One more improvement is still needed: the polygon described by the averaged list of points should be resegmented. The details of this process can be found in Section 7.2.

7.1.1 Normal vector calculation

A proper normal vector calculation is not a trivial task for complex 3D point clouds. There are several methods for solving this, for example [9]. This method is based on a cross-curve moving mask method with which unit normal vectors can be calculated. Another approach can be found in [13], which uses fitted directional tangent vectors and a local Voronoi mesh with proposed mesh growing heuristic rules to calculate the normal of a point.

A third method should also be mentioned [14], in which first an elliptic Gabriel graph is constructed to determine neighbouring points. Given the local neighbours for each point, the normal vector of each point can be approximately calculated. A common approach is fitting quadric curves or surfaces [13, 2] and principal component analysis [15, 6, 11].

7.2 Resegmenting the approximating curve

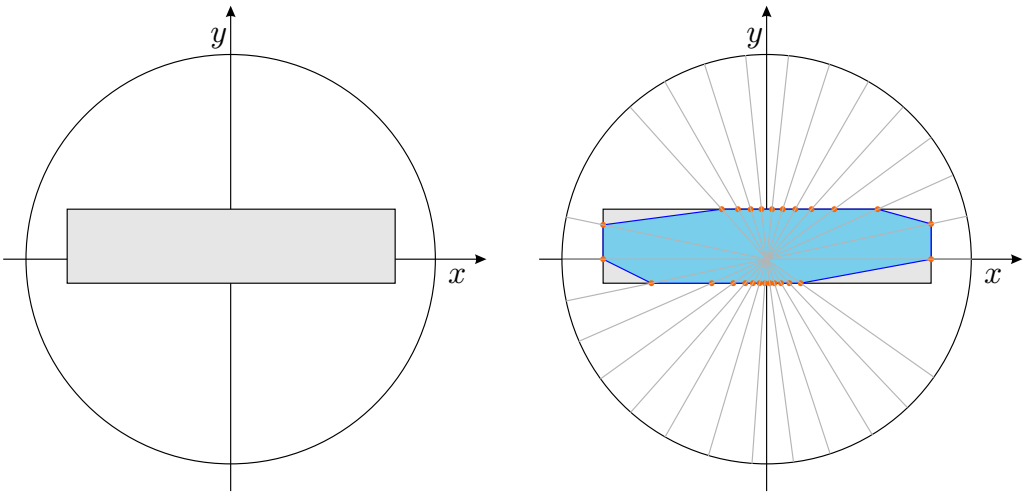
The approximating curve, in this case, is a polygon formed by the averaged points of the stripes. The sides of this polygon have different lengths which means that the scanned points are unevenly distributed along the perimeter (or on the surface in 3D). This can be improved for the next scan by evenly resegmenting this polygon. The resegmentation may be based on

1. fixed number of segments or
2. fixed length of the segments.

In both cases, the points and their normal vectors should be interpolated. This can be done by taking the weighted average of the positions and the normal vectors of the neighbouring points.

This is going to be very important because usually the scanned points of the first only rotational scan may be unevenly distributed along the scanned object. This can be demonstrated with an object that has rectangular cross-section (Figure 22).

At first, the scanner knows nothing about the scanned object, so it will perform a 360° only rotational scan with 30 triggers this time (similar to Table 4), from which 26 could successfully be scanned. The result can be seen in Figure 22b where

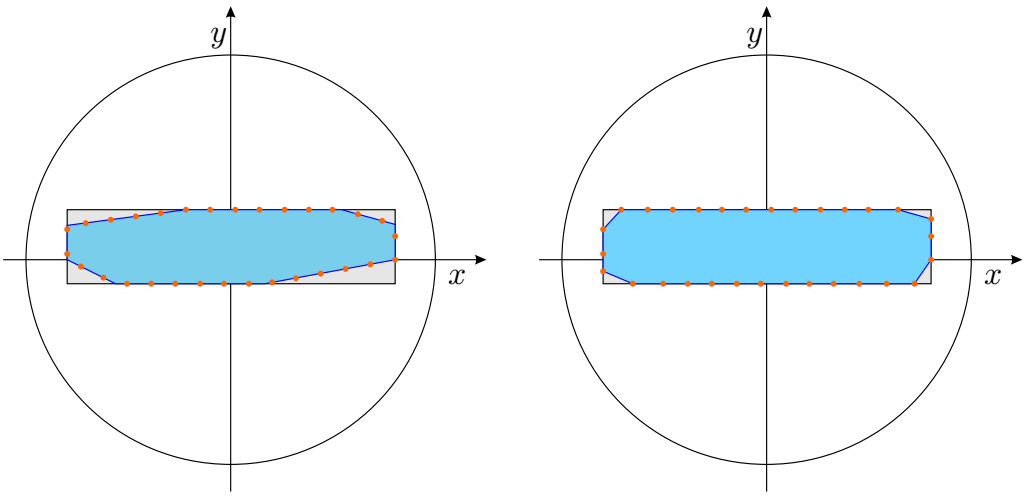


(a) An object that has rectangular cross-section is placed on the scanner

(b) The scanned cross-section of the object is the highlighted polygon

Figure 22: The result of the first scan of the object

the shadow effect can also be recognised. These parts could not be scanned because either the camera or the scanner was hidden by some other part of the scanned object.



(a) The scanned polygon is resegmented

(b) The object is rescanned

Figure 23: The result of the second scanning of the object

Based on the data of the first scan and the resegmentation, the second scan is much better. The scanned points are pretty much uniformly distributed, and all 30 triggers could be scanned. This can be seen in Figure 23.

8 Simulator

Since the hardware is not yet ready for processing the scanning plans described in Section 3, a simulator is needed for testing. What's more, a simulator is also useful for scanning some test objects designed for especially this task. The simulator contains the scanning plan calculation module as well (an extended version of Code 1) so the whole iterative scanning method described in Section 7.1 can be tested, visualised and animated.

8.1 The simulation workflow

The simulator reads a list of two-dimensional points from the provided file. These points are the consecutive points of the perimeter of the averaged cross-section of the scanned object. This perimeter is first going to be scanned with a simple rotational transformation. The scanning plans of the following scans are calculated from the previous scan results based on the iterative method described in Section 7.1. For each scan, a trigger number can be provided which means that the next scanning plan is going to have these many entries. Based on the trigger number, the perimeter is resegmented into this many equal parts. A further parameter, the safety angle, can also be set for the scans.

The simulator can play the scanning process step-by-step or it can also animate the whole process. It can as well save the image of every position and create a video from these.

8.2 Scanning results of handmade polygons

These point sequences are created for testing the algorithms and the designed methods for executing the scanning plans. Therefore, these point sequences usually contain only a few points that specify the polygon, which is going to be scanned.

8.2.1 Scanning a wide rectangle

The scan of a simple rectangle is the first example. The first and most important thing to point out is that the table is not only translated, but also rotated when the sides of the rectangle are being scanned. This can be seen in Figure 27a where the almost horizontal parts are increasing a bit. It can also be observed, that the perimeter after the third scan (Figure 26) is segmented into perfectly equal parts. The numeric details of these transformation plans can be found in Appendix A.

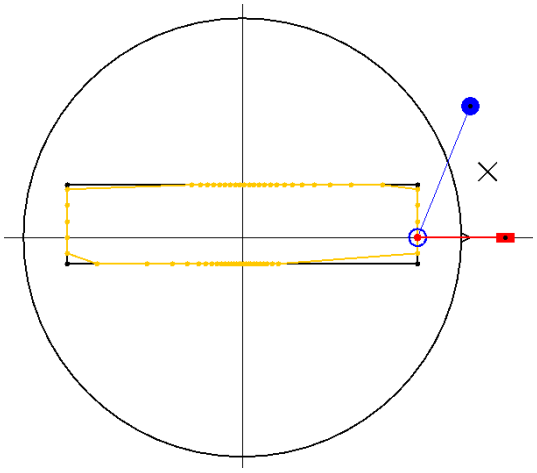


Figure 24: The first scan of the wide rectangle

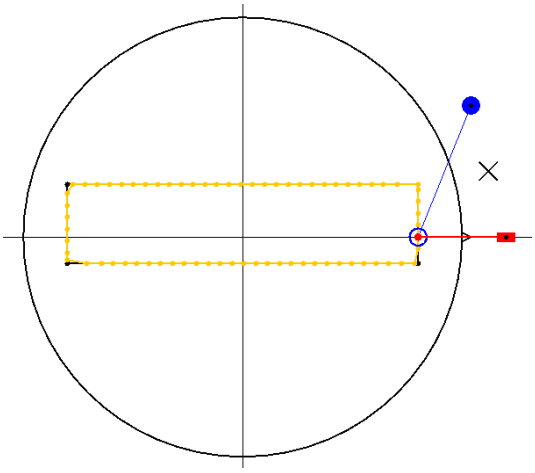


Figure 25: The second scan of the wide rectangle

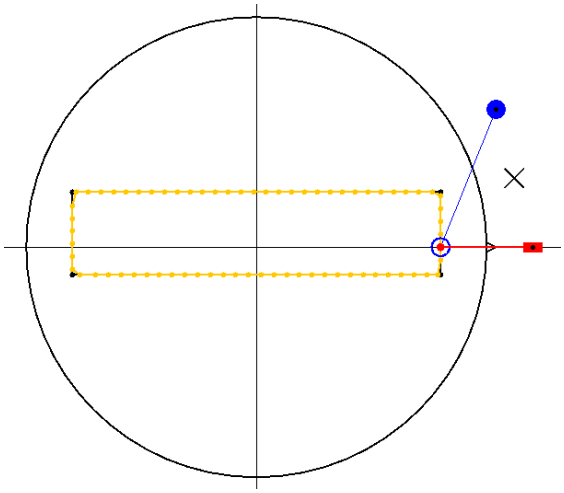
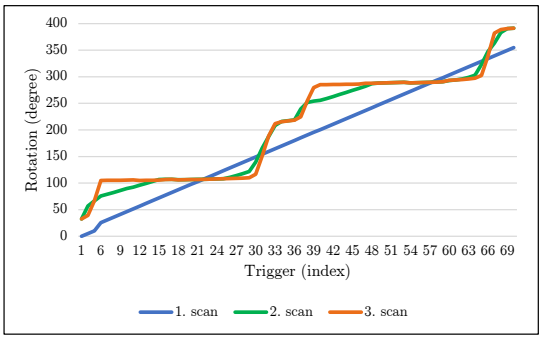
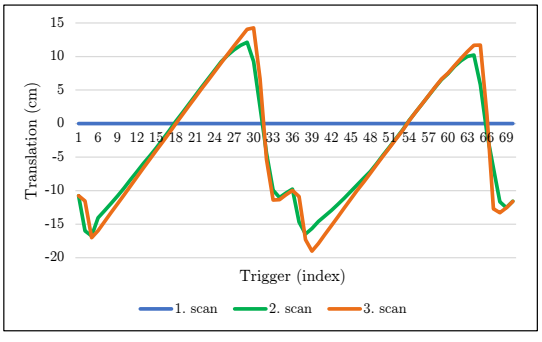


Figure 26: The third scan of the wide rectangle



(a) The rotations of the scans



(b) The translations of the scans

Figure 27: The rotation and translation functions of the first (Figure 24), second (Figure 25) and the third (Figure 26) scan of the wide rectangle

8.2.2 Scanning a concave hole

Scanning a more concave part is always an interesting thing. Previously, these objects could only be scanned after a very detailed and time-consuming manual configuration. It can be seen, that the fifth scanning plan (Figure 31) was good enough for scanning every part of the perimeter at evenly distributed points.

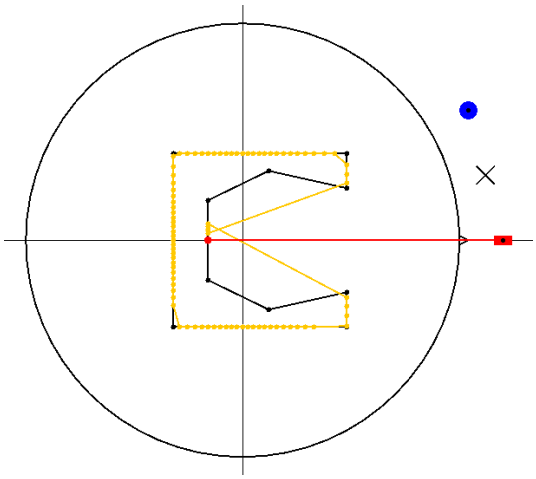


Figure 28: The first scan of the hole

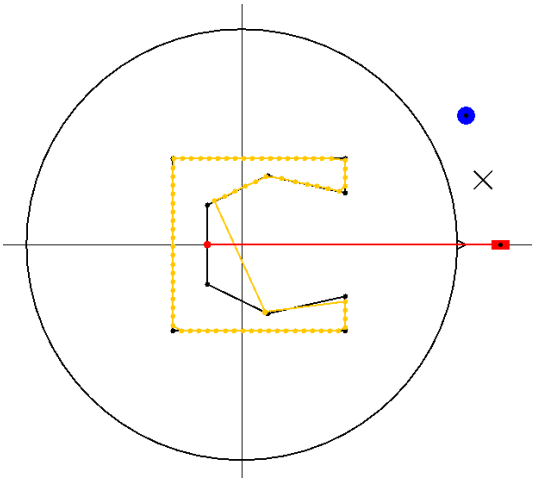


Figure 29: The second scan of the hole

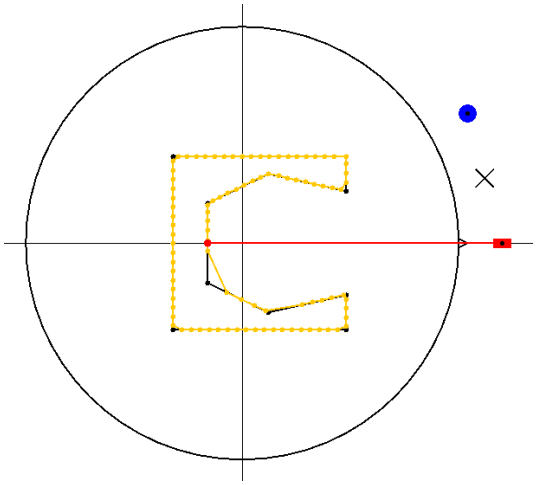


Figure 30: The third scan of the hole

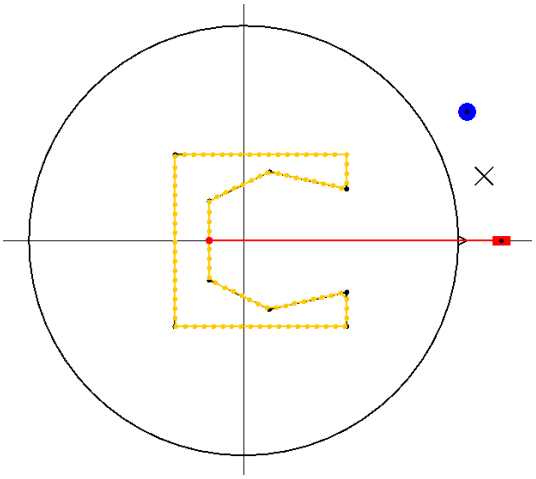


Figure 31: The fifth scan of the hole

8.2.3 Scanning a complex concave polygon

The following quite complex polygon is designed to have some difficult parts to scan. It can be seen, that the step-by-step method works nicely on this complex example as well. To scan such an object with manual configuration would almost be impossible. There were 300 triggers used for this scan, which can be further increased for a more detailed result. It can also be observed, that only the fifth scanning could create an evenly distributed point sequence (Figure 34).

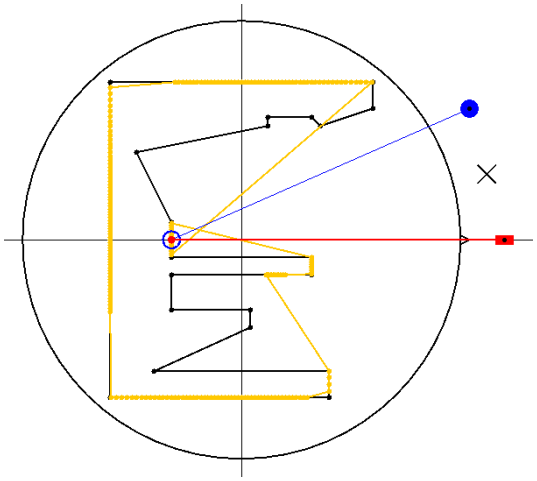


Figure 32: The first scan of the complex polygon

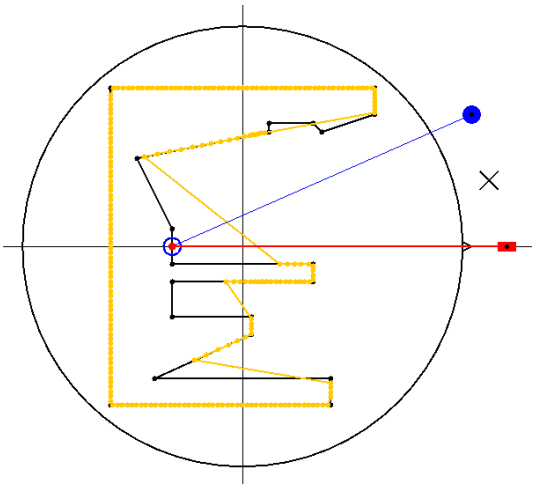


Figure 33: The second scan of the complex polygon

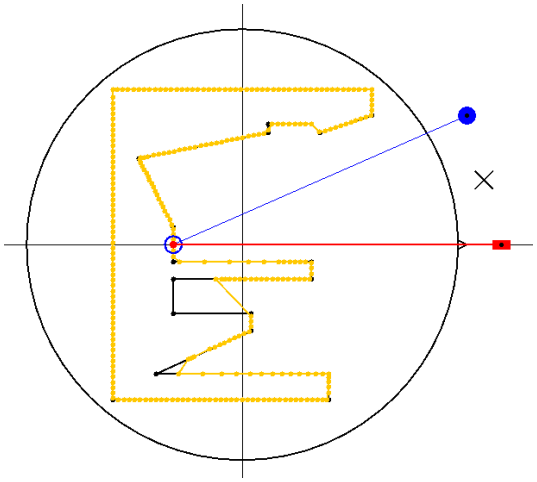


Figure 34: The third scan of the complex polygon

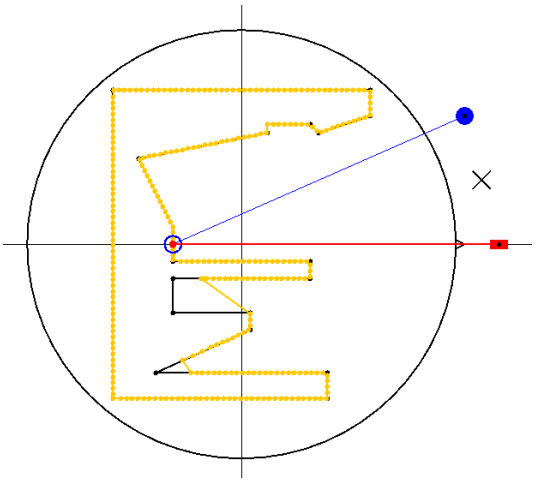


Figure 35: The fifth scan of the complex polygon

8.3 Scanning results of point clouds which were generated by the scanner hardware

A scanning plan can now be created for scanning the object mentioned in the case study (Section 1.3) again. It can be seen in the case study, that some parts of the object could not be scanned since the shadow effect. The calculated functions of the scanning plan can be seen in Figure 38 and the numeric values can be found in Appendix B.

Figure 36: The earlier scan of the object

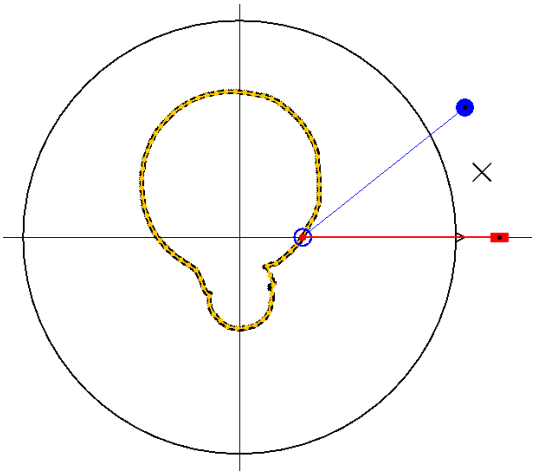
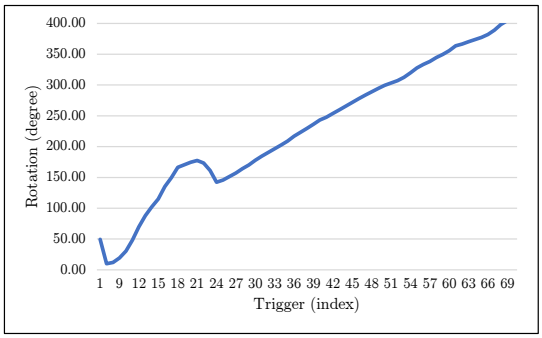
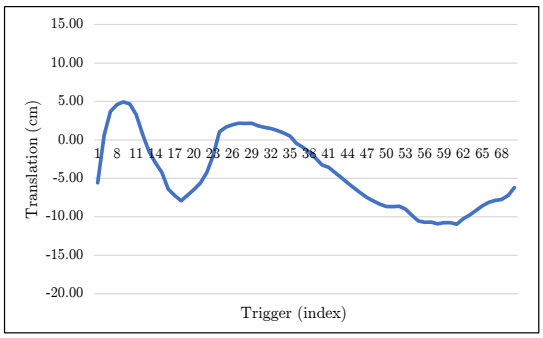


Figure 37: The scan result of the object



(a) The rotations of the scanning plan



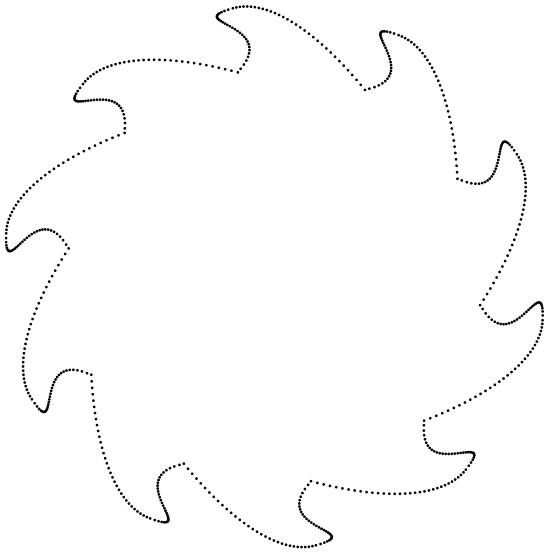
(b) The translations of the scanning plan

Figure 38: The rotation and translation functions of the scanning based on which the next scanning process can be done

8.4 Scanning results of a Kawasaki Chain Roller

The functionality of the implemented method was also tested on the constructed image of a special engineering component of a motorbike. This piece is a *Ufo Chain Roller Kawasaki Kxf 250-450 2013* which can be seen in Figure 39a and the point sequence created for representing it can be seen in Figure 39b.

(a) The Kawasaki Ufo Chain Roller



(b) The point sequence representing the chain roller

Figure 39: The chain roller and the point sequence created for its representation

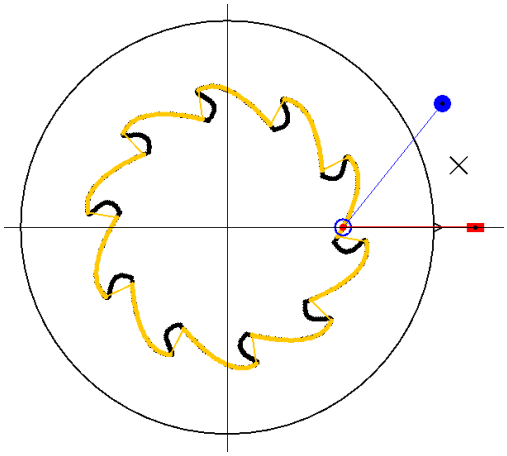


Figure 40: The first scan of the chain roller

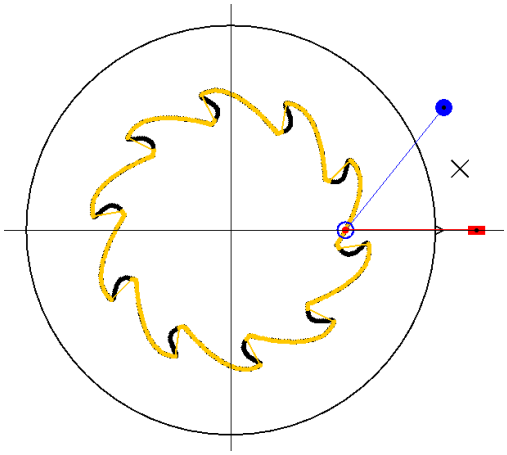


Figure 41: The second scan of the chain roller

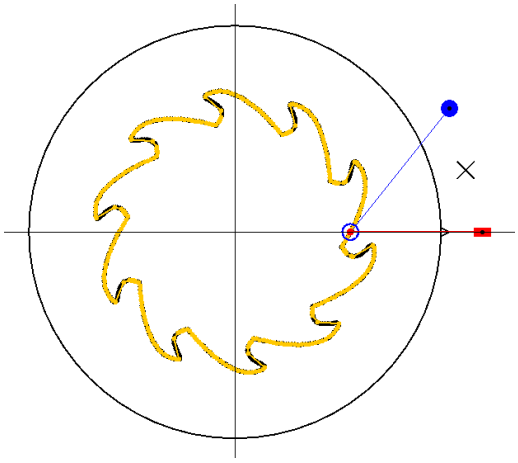


Figure 42: The third scan of the chain roller

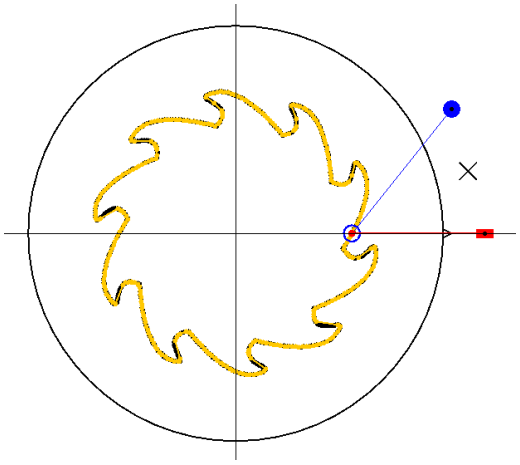
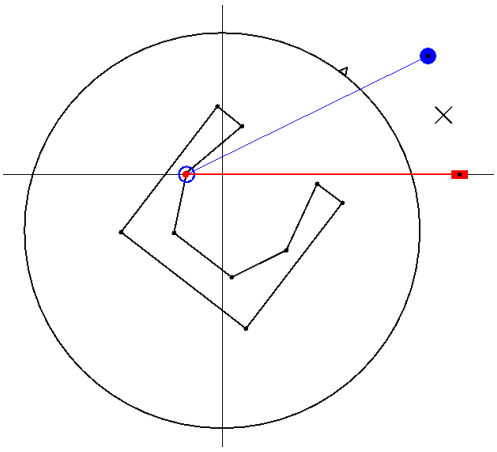


Figure 43: The sixth scan of the chain roller

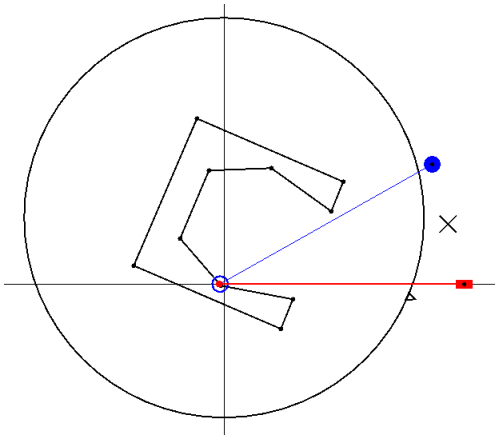
8.5 Observing the safety angle

The safety angel is very important. It makes sure that the shadow effect is avoided as far as possible. On the other hand, a large safety angle may reduce the probability that a point can be scanned. A few images from the scanning process of the previously presented *concave hole* (Section 8.2.2) visualizes this safety angle very well.

It can be seen in Figure 44 that the normal vector of the scanned point is as close to the ideal point as possible, but it keeps a safety distance from the other parts of the object. More images from the scanning of this hole can be found in Appendix C.



(a) Laser line is close to the object



(b) Camera line is close to the object

Figure 44: The camera line and the laser line are not allowed to get too close to other parts of the scanned object to avoid the shadow effect

9 Conclusions and future work

9.1 Conclusions

The main task was to somehow scan an object with the provided 3D laser scanner as precisely as possible. This is exquisitely solved for the one-dimensional case, which is the ideal scan of a point whose normal vector is given. A Java code snippet that calculates the required transformation was also provided. The two-dimensional case, which is the scanning of a polygon that is given by a sequence of points, is perfectly solved as well. A simulator was also implemented for this task with which several test results were presented.

The three-dimensional case can be solved by averaging and projecting the point cloud of the scanned object to two-dimensions. Examples were provided for this as well, in which the resulting point cloud of the scanning was projected onto the $y = 0$ horizontal plane, averaged for each stripe and then this was scanned as a two-dimensional point sequence.

The safety angle, as a further improvement was added to this method. This parameter can be adjusted before the scanning starts and was designed for avoiding the shadow effect problem. This parameter basically adjusts the "courage" of the algorithm which means that more concave parts can be scanned with a smaller safety angle, but this increases the possibility of the shadow effect phenomenon.

This means that the basics of the adaptive scanning method are presented in this paper. While the first scan is arbitrary, since there is no information about the scanned object, the further scans can be done based on the calculated scanning plans. This contains the required translation and rotation pairs for each trigger of the scans, which can be performed by a 3D scanner that has a rotational and a translational degree of freedom.

9.2 Future work

The currently proposed method can generate a scanning plan from a previous scan result, but it cannot decide whether it will be better than the previous one or not. This means that either someone tells the scanner that there is no need for further scans or a fixed number of scans are created each time. This can be solved by creating an adaptive termination condition that analyses the result of a scan and decides if that can be improved or not. The scanning details like precision, available time or the maximum number of scans could still be used as a parameter, but the

scanner must automatically decide when to finish. This is important because these parameters could be set in advance and then the whole scanning process would be automatic.

The resegmentation process is now based on a fixed number which means that the perimeter is going to be divided into this many equal parts. It would be better to resegment the perimeter in such a way that not the number of segments, but the length of the segments is given. In this case, the number of segments would be variable. An adaptive way may also be interesting, where more concave or more convex parts would be segmented into smaller pieces.

There are some objects whose cross-section cannot be approximated well enough because it may vary a lot. In this case, vertically dividing the object into parts and approximating their cross-sections may be a better solution. The two extremities of this idea are the not divided, which is the case now, and the totally divided case, where each division contains only one point. The first has too big error from averaging while the second is time-consuming. A solution between these may further improve the precision of the scans.

The scanning plans described in this paper can now only be executed by the simulator since the controller of the scanner is not yet updated for handling such plans. This means that the corresponding parts of the controller should be developed.

Another interesting point would be to specify the shape of the largest possible unit wide hole that can still be scanned. This would allow us to check if a certain part of an object can be scanned or not without performing any scans.

Acknowledgments

The author would like to express his thanks to Tibor Kovács for his support as a scientific adviser.

This work has been supported by the Department of Automation and Applied Informatics, Budapest University of Technology and Economics.

Supported by the ÚNKP-17-2-I New National Excellence Program of the Ministry of Human Capacities.

List of Figures

List of Tables

List of Algorithms

List of Codes

References

[1] Chetverikov, D., Svirko, D., Stepanov, D., and Krsek, P. The trimmed iterative closest point algorithm. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on* (2002), vol. 3, IEEE, pp. 545–548.

[2] Douros, I., and Buxton, B. F. Three-dimensional surface curvature estimation using quadric surface patches. *Scanning* (2002).

[3] Gruen, A., and Akca, D. Least squares 3d surface and curve matching. *ISPRS Journal of Photogrammetry and Remote Sensing 59*, 3 (2005), 151–174.

[4] Gumhold, S., Wang, X., and MacLeod, R. S. Feature extraction from point clouds. In *IMR* (2001).

[5] Hofer, M., and Pottmann, H. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 284–293.

[6] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph. 26*, 2 (July 1992), 71–78.

[7] Kovács, T. *Vonalkereső algoritmus vizsgálata zajos környezetben.* PhD thesis, 2009.

[8] Lee, I.-K. Curve reconstruction from unorganized points. *Computer aided geometric design 17*, 2 (2000), 161–177.

[9] Lee, R.-T., and Shiou, F.-J. Calculation of the unit normal vector using the cross-curve moving mask method for probe radius compensation of a freeform surface measurement. *Measurement 43*, 4 (2010), 469 – 478.

[10] Lowe, D. G. Local feature view clustering for 3d object recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (2001), vol. 1, IEEE, pp. I–I.

[11] Medioni, G., Lee, M.-S., and Tang, C.-K. *A computational framework for segmentation and grouping.* Elsevier, 2000.

[12] Mezei, A. Speciális 3d szkenner nagyméretű pontfelhőjéből generált háromszögháló javítása. Tech. rep., Budapest University of Technology and Economics, 2016.

[13] OuYang, D., and Feng, H.-Y. On the normal vector estimation for point cloud data from smooth surfaces. *Computer-Aided Design 37*, 10 (2005), 1071 – 1079.

[14] Park, J. C., Shin, H., and Choi, B. K. Elliptic gabriel graph for finding neighbors in a point set and its application to normal vector estimation. *Computer-Aided Design 38*, 6 (2006), 619 – 626.

[15] Pauly, M., Gross, M., and Kobbelt, L. P. Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 163–170.

[16] Vosselman, G., Gorte, B. G., Sithole, G., and Rabbani, T. Recognising structure in laser scanner point clouds. *International archives of photogrammetry, remote sensing and spatial information sciences 46*, 8 (2004), 33–38.

[17] Wang, W., Pottmann, H., and Liu, Y. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (ToG) 25*, 2 (2006), 214–238.

A Rectangle scanning plan details

Index	Scan 1	Scan 2	Scan 3
1	0.00	32.56	32.56
2	5.14	56.81	39.16
3	10.29	66.44	66.84
4	15.43	70.15	96.56
5	20.57	72.89	104.77
6	25.71	75.80	104.93
7	30.86	79.07	105.10
8	36.00	82.51	105.28
9	41.14	86.06	105.45
10	46.29	89.69	105.63
11	51.43	92.25	105.82
12	56.57	95.90	104.95
13	61.71	99.52	105.11
14	66.86	103.07	105.29
15	72.00	106.71	105.49
16	77.14	106.97	106.91
17	82.29	107.22	107.15
18	87.43	106.20	106.14
19	92.57	106.44	106.38
20	97.71	106.69	106.63
21	102.86	106.95	106.90
22	108.00	107.23	107.18
23	113.14	107.52	107.47
24	118.29	107.90	107.85
25	123.43	108.25	108.21
26	128.57	110.98	108.60
27	133.71	114.24	109.02
28	138.86	117.76	109.47
29	144.00	121.79	109.97
30	149.14	139.52	116.52
31	154.29	165.61	150.97
32	159.43	187.50	188.44
33	164.57	208.55	211.70
34	169.71	216.03	215.50
35	174.86	217.42	216.86

Index	Scan 1	Scan 2	Scan 3
36	180.00	218.95	218.37
37	185.14	239.37	224.73
38	190.29	251.64	254.63
39	195.43	254.24	279.53
40	200.57	255.84	285.17
41	205.71	259.19	285.33
42	210.86	262.86	285.50
43	216.00	266.59	285.68
44	221.14	270.46	285.87
45	226.29	274.40	286.07
46	231.43	278.36	286.28
47	236.57	282.24	287.69
48	241.71	287.15	287.91
49	246.86	288.24	288.20
50	252.00	288.55	288.51
51	257.14	288.87	288.84
52	262.29	289.21	289.18
53	267.43	289.57	289.55
54	272.57	288.49	288.48
55	277.71	288.85	288.84
56	282.86	289.23	289.23
57	288.00	289.65	289.66
58	293.14	290.08	290.10
59	298.29	290.66	290.70
60	303.43	293.13	293.19
61	308.57	293.93	294.02
62	313.71	295.96	294.88
63	318.86	298.73	295.99
64	324.00	302.99	297.51
65	329.14	323.05	302.40
66	334.29	347.58	339.07
67	339.43	363.44	382.35
68	344.57	383.36	388.89
69	349.71	390.64	390.61
70	354.86	391.41	391.39

Table: Rectangle scanning plan: rotation details (angles in degrees)

Index	Scan 1	Scan 2	Scan 3
1	0	-10.76	-10.76
2	0	-15.99	-11.56
3	0	-16.75	-17.00
4	0	-16.01	-18.71
5	0	-15.05	-17.31
6	0	-14.04	-15.97
7	0	-13.00	-14.63
8	0	-11.91	-13.29
9	0	-10.78	-11.96
10	0	-9.61	-10.63
11	0	-8.37	-9.30
12	0	-7.17	-7.96
13	0	-5.97	-6.63
14	0	-4.78	-5.30
15	0	-3.60	-3.97
16	0	-2.31	-2.70
17	0	-1.03	-1.39
18	0	0.32	-0.01
19	0	1.61	1.30
20	0	2.89	2.61
21	0	4.17	3.91
22	0	5.45	5.21
23	0	6.71	6.51
24	0	7.97	7.79
25	0	9.22	9.07
26	0	10.18	10.33
27	0	11.00	11.59
28	0	11.67	12.84
29	0	12.12	14.07
30	0	9.27	14.27
31	0	2.47	6.67
32	0	-4.40	-5.32
33	0	-9.96	-11.39
34	0	-11.03	-11.33
35	0	-10.36	-10.61

Index	Scan 1	Scan 2	Scan 3
36	0	-9.76	-9.97
37	0	-14.68	-10.88
38	0	-16.47	-17.29
39	0	-15.65	-19.02
40	0	-14.60	-17.87
41	0	-13.79	-16.55
42	0	-12.95	-15.22
43	0	-12.05	-13.90
44	0	-11.11	-12.57
45	0	-10.13	-11.26
46	0	-9.12	-9.94
47	0	-8.10	-8.70
48	0	-7.16	-7.40
49	0	-5.95	-6.11
50	0	-4.68	-4.82
51	0	-3.43	-3.54
52	0	-2.18	-2.26
53	0	-0.94	-1.00
54	0	0.46	0.43
55	0	1.71	1.69
56	0	2.94	2.95
57	0	4.16	4.20
58	0	5.38	5.43
59	0	6.56	6.64
60	0	7.42	7.51
61	0	8.52	8.63
62	0	9.37	9.73
63	0	10.01	10.76
64	0	10.21	11.66
65	0	5.80	11.69
66	0	-1.50	1.23
67	0	-6.61	-12.72
68	0	-11.66	-13.29
69	0	-12.53	-12.56
70	0	-11.58	-11.60

Table: Rectangle scanning plan: translation details (translation in centimeters)

B Scanning plan for the next scan of an object

Index	Rotation	Translation
1	49.61	-5.57
2	56.66	-5.09
3	62.57	-4.49
4	65.27	-3.64
5	85.55	-4.00
6	38.74	0.61
7	9.95	3.70
8	11.86	4.58
9	19.20	4.94
10	30.60	4.67
11	48.08	3.28
12	69.83	0.72
13	88.04	-1.47
14	102.11	-2.96
15	115.09	-4.21
16	135.36	-6.38
17	149.66	-7.22
18	166.39	-7.93
19	170.51	-7.19
20	174.73	-6.46
21	177.46	-5.63
22	173.54	-4.27
23	161.40	-2.04
24	142.40	1.05
25	145.81	1.66
26	151.42	1.96
27	157.12	2.18
28	164.09	2.14
29	170.22	2.16
30	177.99	1.83
31	184.51	1.64
32	190.46	1.49
33	196.59	1.23
34	202.69	0.91
35	208.84	0.50

Index	Rotation	Translation
36	217.07	-0.45
37	223.16	-0.97
38	229.51	-1.60
39	236.15	-2.37
40	243.13	-3.26
41	247.77	-3.53
42	253.89	-4.23
43	259.78	-4.88
44	265.96	-5.60
45	271.92	-6.26
46	277.84	-6.90
47	283.58	-7.47
48	289.02	-7.93
49	294.40	-8.35
50	299.42	-8.66
51	303.39	-8.68
52	307.08	-8.63
53	312.49	-8.99
54	319.87	-9.79
55	327.76	-10.53
56	333.38	-10.70
57	338.29	-10.67
58	344.75	-10.90
59	349.72	-10.76
60	355.69	-10.75
61	363.83	-10.96
62	366.42	-10.26
63	370.43	-9.76
64	373.97	-9.18
65	377.30	-8.56
66	382.06	-8.11
67	389.13	-7.87
68	398.28	-7.73
69	404.62	-7.22
70	404.68	-6.19

Table: Transformation details for configuring the scanner for the following scan

C Details of the scanning process of the hole