Periodica Polytechnica Electrical Engineering and Computer Science, 62(4), pp. 107–116, 2018

Curvature Adaptive 3D Scanning Transformation Calculation

Adrián Mezei[1*], Tibor Kovács[1]

[1] Department of Automation and Applied Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, H-1521 Budapest, P.O.B. 91, Hungary
[*] Corresponding author, email: Mezei.Adrian@aut.bme.hu

Received: 24 October 2017, Accepted: 26 May 2018, Published online: 13 June 2018

Abstract
Three-dimensional objects can be scanned by 3D laser scanners that use active triangulation. These scanners create three-dimensional point clouds from the scanned objects. The laser line is identified in the images, which are captured at given transformations by the camera, and the point cloud can be calculated from these. The hardest challenge is to construct these transformations so that most of the surface can be captured. The result of a scanning may have missing parts because either not the best transformations were used or because some parts of the object cannot be scanned. Based on the results of the previous scans, a better transformation plan can be created, with which the next scan can be performed. In this paper, a method is proposed for transforming a special 3D scanner into a position from where the scanned point can be seen from an ideal angle. A method is described for estimating this transformation in real-time, so these can be calculated for every point of a previous scan to set up a next improved scan.

Keywords
laser scanner, 3D scanning, automation, active triangulation

1 Introduction

1.1 Context

A 3D scanner [1] 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 first steps of this optimization process are described in detail.

1.2 Related works

The development of active triangulation laser scanners has rapidly developed in the last 30 years [2]. Their main advantage is that they can scan the 3D space very precisely, which means that their error can even be a fraction of a millimetre. On the other hand, these scanners are usually slower than some other ones. This tradeoff must be considered for each use case separately; a good comparison can be found in [3]. The way how these scanners operate is based on a simple triangulation method. The scanner knows the exact position of its camera and its laser and it can calculate the positions of the scanned points from these. The principle is described in more details throughout the literature [4]. A similar scanner [5] has already been built with one rotational degree of freedom, where the scanning process was automated. In our case, we have an additional translational degree of freedom that we can take advantage of. With two degrees of freedom more complex objects can be scanned and a larger surface coverage can be achieved. This is a more complex system thus its configuration and automation are more complex as well.

There are several other techniques that can be used for 3D scannings like structured light scanners [6, 7], time of flight scanners or contact scanners. These are usually less precise, but they have other advantages. Structured light scanners can capture a much larger number of points per second, which means that they are faster. These devices are also capable of scanning moving objects [8]. Time of flight scanners [9] are capable of scanning large distances so they are sometimes better for outdoor usage.

1.3 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.

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 Fig. 1 has six very important components: the controller, the laser, the camera, the two step motors and the turntable.

2.2 Scanning workflow

First, 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.

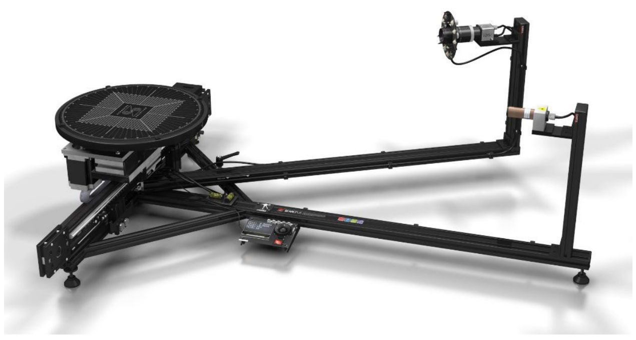
Fig. 1 The 3D scanner with the laser, the camera, the two motors, the controller and the turntable

2.3 Data structure

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 form a *cloud*.

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

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.

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.

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

Table 1 An example scanning plan

Index	Rotation (degree)	Translation (mm)
1	0	0
2	90	10
3	180	20
4	270	30

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.

Later, it will be a further requirement to create the scanning plan in such way, that the scanned stripes are equally distributed around the scanned object as far as possible. It is not the best 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.

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).

Algorithm 1 Execution of a transformation plan

```
Input: scanningPlan contains the transformations
Output: batch is the structure that contains the stripes
for all transformation in scanningPlan do
    turnTable.setRotation(transformation.rotation)
    turnTable.setTranslation(transformation.translation)
    image = camera.createImage()
    stripe = linetracker.track(image)
    batch.add(stripe)
end for
```

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 [10]. As an example, the error groups of a scanned object can be seen in Fig. 2.

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

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 visualized as a planar set of

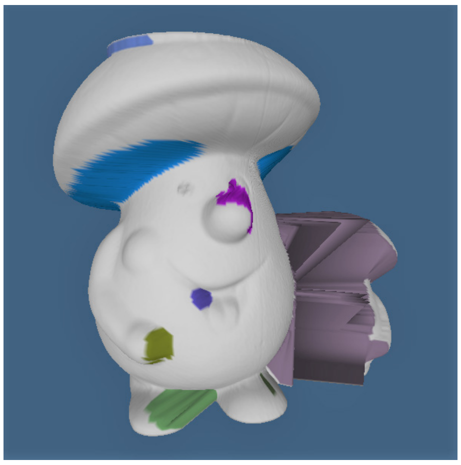
Fig. 2 The error groups of a scanned object

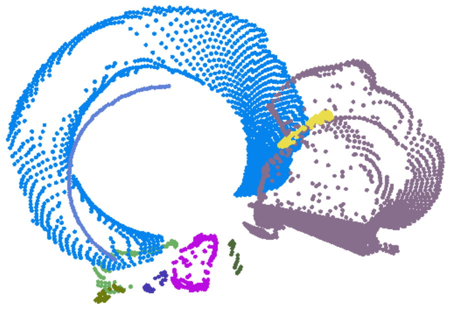

Fig. 3 The projected points of the error groups of Fig. 2

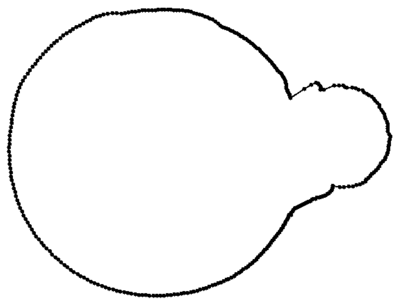

Fig. 4 The approximated curve of the projected points in Fig. 3

points. This can be seen in Fig. 3. 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 [11], moving least square, improved moving least squares [12] or the weighted least square methods. These are used in the iterative closest point algorithm [13]. This method always monotonically converges to the nearest local minimum of a mean-square distance metric.

This could be further improved by feature extraction [14-16], spline approximations [17, 18] or by identifying more 2D primitives. These features can be categorized 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. This can be seen in Fig. 4.

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.

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.

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 is to place the 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 Fig. 5 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.

A general point is given by its coordinates, such position can be seen in Fig. 6, which may be a calculated value from the parametric approximating curve. If the proper

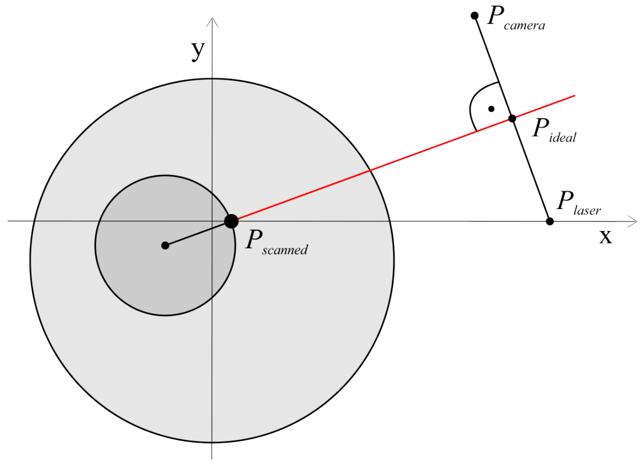

Fig. 5 Ideal position for scanning a point of the approximating curve

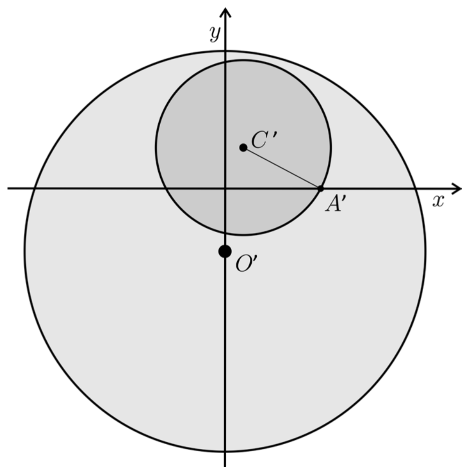

Fig. 7 The position after the first step

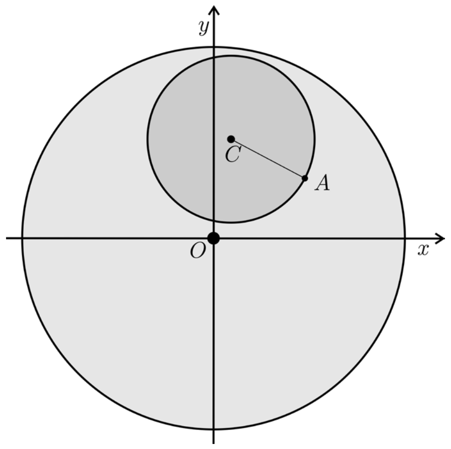

Fig. 6 General starting position

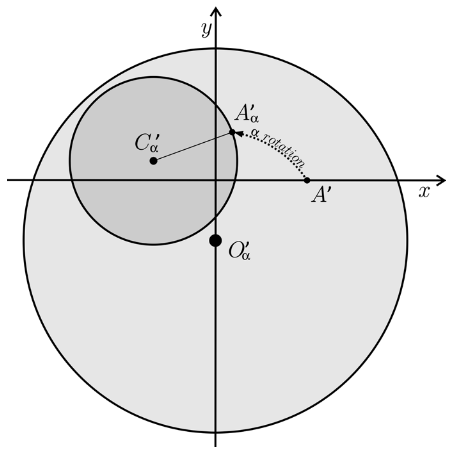

Fig. 8 The position after the calculated rotation

rotation of the required 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).

5.1 Equation to calculate the required rotation

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 other translated points are denoted by O' and C'. The result of this translation can be seen in Fig. 7.

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, where P_i is the ideal point. This can be seen in Fig. 8 and Fig. 9. 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

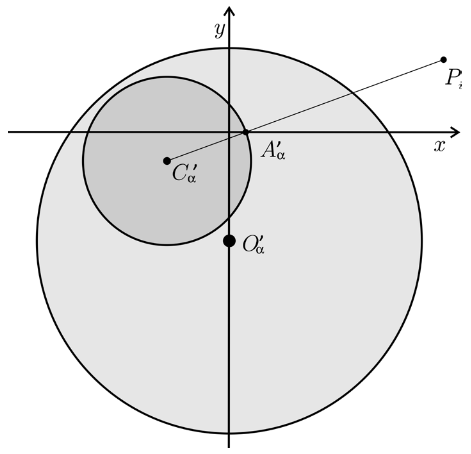

Fig. 9 The position after the final translation

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. (1)).

$$y - y_1 = m(x - x_1) \tag{1}$$

$O'_{\alpha,y}$ is calculated by rotating $A' - O'$ around the origin by α then the negative of the result's y-coordinate is taken (Eq. (2)).

$$\begin{aligned} O'_{\alpha,y} &= -\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} A'_x - O'_x \\ A'_y - O'_y \end{bmatrix} \\ &= O'_y \cos(\alpha) - A'_x \sin(\alpha) \end{aligned} \tag{2}$$

$A'_{\alpha,x}$ is obtained by taking the x-coordinate of the $A' - O'$ point after rotating it by α around the origin (Eq. (3)).

$$\begin{aligned} A'_{\alpha,x} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} A'_x - O'_x \\ A'_y - O'_y \end{bmatrix} \\ &= A'_x \cos(\alpha) + O'_y \sin(\alpha) \end{aligned} \tag{3}$$

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 $C' - O'$ around the origin by α and translating it by O' (Eq. (4)).

$$\begin{aligned} C'_\alpha &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} C'_x - O'_x \\ C'_y - O'_y \end{bmatrix} \\ &\quad + \begin{bmatrix} O'_x \\ O'_y \end{bmatrix} + \begin{bmatrix} O'_{\alpha,x} - O'_x \\ O'_{\alpha,y} - O'_y \end{bmatrix} \\ &= \begin{bmatrix} C'_x \cos(\alpha) - (C'_y - O'_y)\sin(\alpha) \\ C'_x \sin(\alpha) + (C'_y - O'_y)\cos(\alpha) + O'_{\alpha,y} \end{bmatrix} \end{aligned} \tag{4}$$

The slope m of the point-slope formula is expressed from points P_i and A'_α (Eq. (5)).

$$m = \frac{P_{i,y} - A'_{\alpha,y}}{P_{i,x} - A'_{\alpha,x}} = \frac{P_{i,y}}{P_{i,x} - (A'_x \cos(\alpha) + O'_y \sin(\alpha))} \tag{5}$$

With the calculated values $m, O'_\alpha, A'_\alpha, C'_\alpha$ and the ideal point P_i, the point-slope formula can be parametrized in the following way (Eq. (6)).

$$C'_{\alpha,y} - P_{i,y} = m(C'_{\alpha,x} - P_{i,x}) \tag{6}$$

After substituting the previously calculated values and some simplification, the equation contains only constants and α which is to be calculated (Eq. (7)).

$$\begin{aligned} &C'_x \sin(\alpha) + C'_y \cos(\alpha) - A'_x \sin(\alpha) - P_{i,y} \\ &= \frac{P_{i,y}\left(C'_x \cos(\alpha) - (C'_y - O'_y)\sin(\alpha) - P_{i,x}\right)}{P_{i,x} - (A'_x \cos(\alpha) + O'_y \sin(\alpha))} \end{aligned} \tag{7}$$

6 Rotation angle calculation

The calculated implicit equation (Eq. (7)) in Section 5.1 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. (7)). 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 Fig. 10.

A rotation by this estimated angular error usually moves point A' from the x-axis (Fig. 11), so in this case, the next step is to perform a translation so that A'_α gets to the x-axis again (from Fig. 11 to Fig. 12). The amount of this translation is simply the y coordinate of the rotated A'_α point. The estimated angular error for the next iteration can now be seen in Fig. 12.

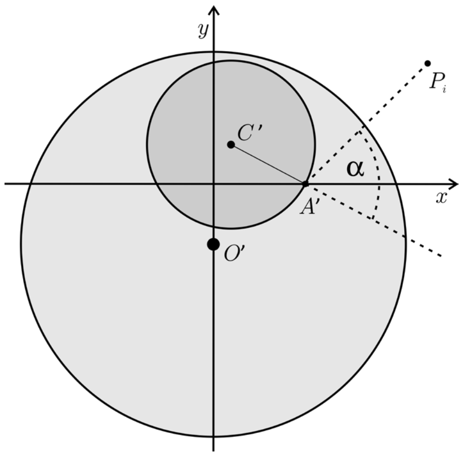

Fig. 10 Estimated angular error

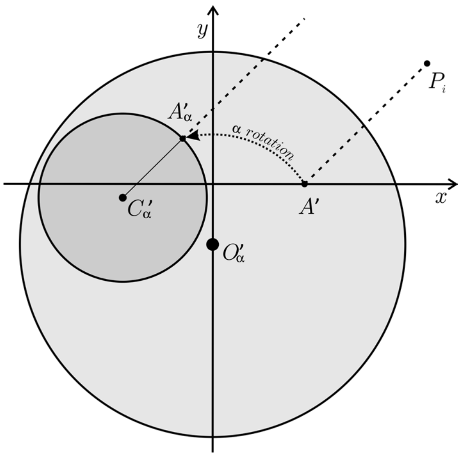

Fig. 11 Rotation by the angular error

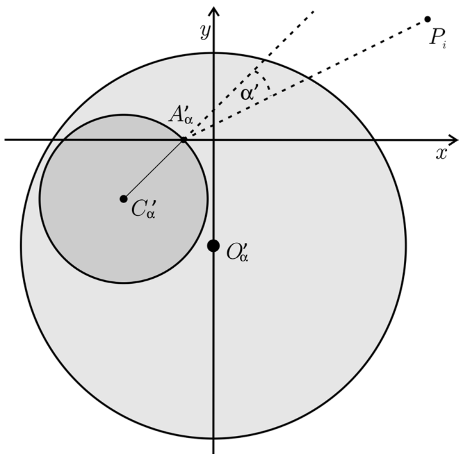

Fig. 12 Angular error after the first iteration

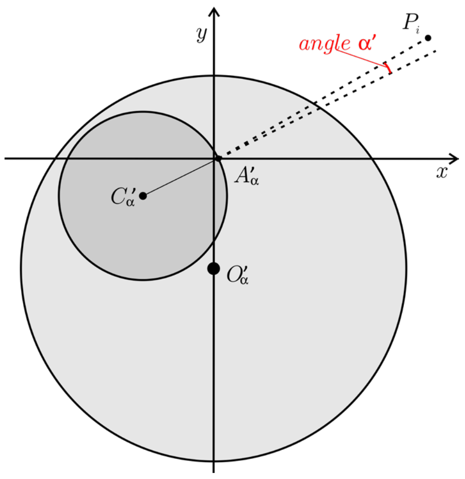

Fig. 13 Angular error after the second iteration

In the beginning, the estimated angular error is 72.473° in the example Fig. 10. After the first iteration, this angular error is only −18.233° in the example Fig. 12. After the second iteration, this angle is less than 4° in the example Fig. 13.

6.1 Convergence of the algorithm

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 Fig. 14.

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.

The provided Java class (Code 1) calculates the required angle for positioning a point to the ideal scanning position. This solves the implicit equation Eq. (7) presented in Section 5.1 with the method described in Section 6.

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 Fig. 7. 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.

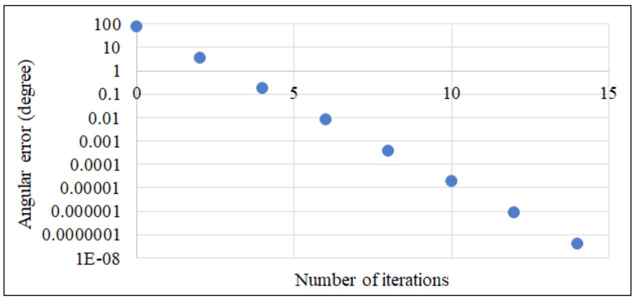

Fig. 14 The absolute values of the estimated angular errors represented on logarithmic scale

Table 2 The estimated angular errors after the iterations

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

7 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 6. The calculations were

Code 1 Java implementation of the rotation angle calculation

```
//RotationCalculator.java
public class RotationCalculator{
  public static double calculateRotationDeg(double Ax,
  double Cx, double Cy, double Oy, double Pix, double
  Piy, double maxAngleErrorDeg){
    double Oay, Aax, Cax, Cay;
    double errorAngle=Math.PI;
    double maxErrorAngle=maxAngleErrorDeg/180.0*Math.PI;
    double angle=0;

    while(Math.abs(errorAngle)>maxErrorAngle){
      //Calculated parameters
      Oay=Oy*Math.cos(angle)-Ax*Math.sin(angle);
      Aax=Ax*Math.cos(angle)+Oy*Math.sin(angle);
      Cax=Cx*Math.cos(angle)-(Cy-Oy)*Math.sin(angle);
      Cay=Cx*Math.sin(angle)+(Cy-Oy)*Math.cos(angle)+Oay;

      //Calculate the error by angle PiAa and AaCa
      double anglePiAa=Math.atan2(Piy,Pix-Aax);
      double angleAaCa=Math.atan2(-Cay,Aax-Cax);
      errorAngle=anglePiAa-angleAaCa;

      angle+=errorAngle;
    }
    return Math.toDegrees(angle);
  }
}
```

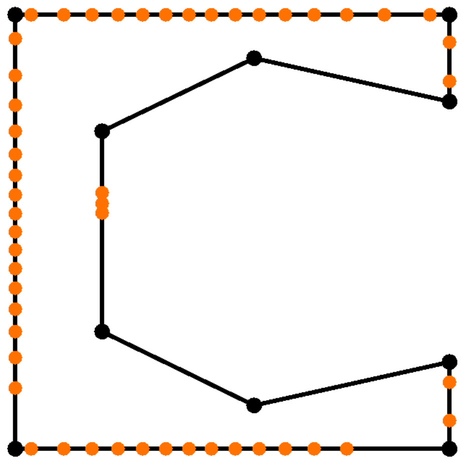

Fig. 15 The captured points after a simple rotational scan

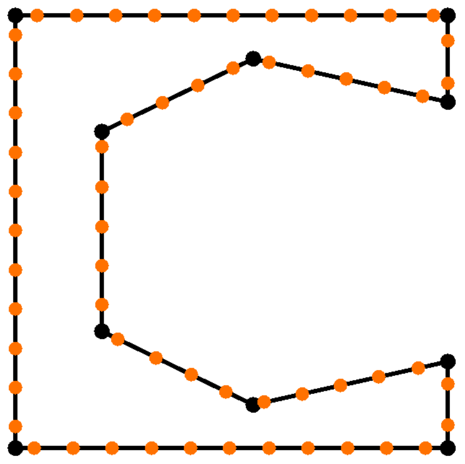

Fig. 16 The captured points after a few iterations of the ideal position configuration

implemented and tested in Java and these were also presented. This implementation calculates the required angle fast enough for real-time usage, which is an essential requirement in most cases.

As it can be seen in Fig. 15 and Fig. 16 the iterative method can calculate such transformations that can hardly be done by hand. With these new transformations, a larger surface coverage of the object could be achieved. In this particular case, every part of the object could be scanned. In some cases, there may be heavily concave parts, that cannot be scanned. These parts will simply by omitted and the rest of the object is scanned.

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 but a few of these can be handled in some way.

7.1 Future work

This method is perfect for transformation plan generation of simple objects. In more complex cases, calculations based on only the ideal angle used in this paper will possibly result in inadequate results. These transformations may sometimes hide the camera from seeing the required point or hide the laser beam from illuminating the required point. To avoid these cases, a more detailed analysis of the previous scans is needed so that only such transformations are generated that are adequate for scanning the required point (at least based on the previous scanning data).

This may probably require the introduction of a safety angle which would mean a reasonable distance between the laser beam and those parts of the object that are not currently scanned. The same is needed for the camera so that it can always see the laser beam illuminated point.

A further task is to perform these scans is such way, that the scanned points are equally distributed along the perimeter. This would make it easier to perform a scan with a given precision regardless of the size of the object. The current settings always make denser point clouds of smaller objects since they are closer to the centre of rotation.

Acknowledgement

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.

Emberi Erőforrások Minisztériuma

SUPPORTED BY THE ÚNKP-17-2-I NEW NATIONAL EXCELLENCE PROGRAM OF THE MINISTRY OF HUMAN CAPACITIES.

References

[1] Kovács, T. "Vonalkereső algoritmus vizsgálata zajos környezetben", (Analysis of a Line Tracking Algorithm in Noisy Environment), PhD Thesis, Budapest University of Technology and Economics, 2009. (in Hungarian)

[2] Blais, F. "Review of 20 years of range sensor development", Journal of Electronic Imaging, 13(1), pp. 231–244, 2004. https://doi.org/10.1117/1.1631921

[3] Jecić, S., Drvar, N. "The Assessment of Structured Light and Laser Scanning Methods in 3D Shape Measurements", In: 4th International Congress of Croatian Society of Mechanics, Bizovac, Croatia, 2003, pp. 237–244.

[4] Toedter, O., Koch, A. W. "A simple laser-based distance measuring device", Measurement, 20(2), pp. 121–128, 1997. https://doi.org/10.1016/S0263-2241(97)00024-9

[5] Fitzgibbon, A. W., Cross, G., Zisserman, A. "Automatic 3D Model Construction for Turn-Table Sequences", In: Koch, R., Van Gool, L. (eds.) 3D Structure from Multiple Images of Large-Scale Environments, SMILE 1998. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Germany, 1998, pp. 155–170. https://doi.org/10.1007/3-540-49437-5_11

[6] Geng, J. "Structured-light 3d surface imaging: a tutorial", Advances in Optics and Photonics, 3(2), pp. 128–160, 2011. https://doi.org/10.1364/AOP.3.000128

[7] Gupta, M., Agrawal, A., Veeraraghavan, A., Narasimhan, S. G. "Structured light 3D scanning in the presence of global illumination", In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, Rhode Island, USA, 2011, pp. 713–720. https://doi.org/10.1109/CVPR.2011.5995321

[8] Hall-Holt, O. Rusinkiewicz, S. "Stripe boundary codes for real-time structured-light range scanning of moving objects", In: Proceedings Eighth IEEE International Conference on Computer Vision, ICCV 2001, Vancouver, British Columbia, Canada, 2001, pp. 359–366. https://doi.org/10.1109/ICCV.2001.937648

[9] Cui, Y., Schuon, S., Chan, D., Thrun, S., Theobalt, C. "3D shape scanning with a time-of-flight camera", 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, California, USA, 2010, pp. 1173–1180. https://doi.org/10.1109/CVPR.2010.5540082

[10] Mezei, A. Zs. "Speciális 3D szkenner nagyméretű pontfelhőjéből generált háromszögháló javítása", (Improvement of Mesh Made of Big Point Cloud of a Special 3D Scanner), BSc Thesis, Budapest University of Technology and Economics, 2016. (in Hungarian)

[11] Gruen, A., Akca, D. "Least squares 3D surface and curve matching", ISPRS Journal of Photogrammetry and Remote Sensing, 59(3), pp. 151–174, 2005. https://doi.org/10.1016/j.isprsjprs.2005.02.006

[12] Lee, I.-K. "Curve reconstruction from unorganized points", Computer Aided Geometric Design, 17(2), pp. 161–177, 2000. https://doi.org/10.1016/S0167-8396(99)00044-8

[13] Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P. "The Trimmed Iterative Closest Point Algorithm", In: Proceedings 16th International Conference on Pattern Recognition, 2002, pp. 545–548. https://doi.org/10.1109/ICPR.2002.1047997

[14] Gumhold, S., Wang, X., MacLeod, R. "Feature Extraction From Point Clouds", In: Proceedings 10th International Meshing Roundtable, Newport Beach, California, USA, 2001, pp. 293–305.

[15] Vosselman, G., Gorte, B. G. H., Sithole, G., Rabbani, T. "Recognising Structure in Laser-Scanner Point Clouds", International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36(8), pp. 33–38, 2004.

[16] Lowe, D. G., "Local feature view clustering for 3D object recognition", In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR, Kauai, Hawaii, USA, 2001, pp. I-682–I-688. https://doi.org/10.1109/CVPR.2001.990541

[17] Wang, W., Pottmann, H., Liu, Y. "Fitting B-Spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization", ACM Transactions on Graphics (TOG), 25(2), pp. 214–238, 2006. https://doi.org/10.1145/1138450.1138453

[18] Hofer, M., Pottmann, H. "Energy-Minimizing Splines in Manifolds", ACM Transactions on Graphics (TOG), 23(3), pp. 284–293, 2004. https://doi.org/10.1145/1015706.101571